

UNIVERSIDAD AUTÓNOMA DE
NUEVO LEÓN

FACULTAD DE INGENIERÍA
MECÁNICA Y ELÉCTRICA

POSGRADO EN INGENIERÍA DE
SISTEMAS ...

PORAFOLIO DE EVIDENCIAS DE ANÁLISIS
ESTADÍSTICO MULTIVARIADO...

PROFESOR(A): DRA. SATU ELISA SCHAEFFER

AUTOR: LUIS ÁNGEL GUTIÉRREZ RODRÍGUEZ
MATRICULA: 1484412

Introducción

El siguiente documento es el portafolio de evidencias de la materia Análisis Estadístico Multivariado. El curso constó de 14 actividades, 13 proyectos en Jupyter Notebook y la redacción de un artículo de carácter científico en Latex. El cual fue co-evaluado por la Doctora Schaeffer y el grupo del curso.

Este portafolio consiste en anexar las actividades revisadas por la Dra. Schaeffer. Después detallar la experiencia adquirida al realizar la práctica. Finalmente, si es posible, se anexarán las mismas actividades pero corrigiendo los errores antes cometidos y que fueron señalados por la Dra. Schaeffer.

Práctica 1: Preparación de datos

Complicaciones

LA 6

CIENCIA_DE_DATOS / P1.ipynb

Reporte Práctica 1 Preparación de los datos

Origen de los datos

Los datos a analizar durante el curso fueron proporcionados por la Dra. Elisa y son de un festival de cine que se lleva a cabo anualmente en Colombia y se está empezando a difundir la idea en México. Se cuenta con la información de las boletas de inscripción al festival de los años 2015 al 2018. La información capturada varía y aumenta de acuerdo al año de su captura, es decir, la información de 2018 contiene más datos que la de 2015. A causa del crecimiento de la información cada año, la limpieza de los mismos me dejará con muchas celdas vacías.

Las preguntas con las pudieramos trabajar serían:

- ¿Cuáles son los teléfonos móviles que se utilizan más los productores de acuerdo a la categoría que pertenecen?
- ¿Cuál es el género que más producciones tiene?
- ¿Cuál fue el medio más efectivo por el que la mayoría de los concursantes se enteraron del festival?
- ¿Cuáles son los países extranjeros más activos en el festival, en qué categorías participan y qué géneros prefieren?

Datos disponibles

Como los datos disponibles varían con cada año, buscaré describir los datos por cada archivo y posteriormente buscaremos el común de estos.

Año 2015

El primer año del que se tiene registro. En este festival solo se contaba con los siguientes 8 datos:

- País: País de procedencia de la producción.
- Categoría: Si el video es producido por infantiles, profesionales ó aficionados.
- Nombre del corto.
- Medio de envío: El método por el cual la producción del corto hizo llegar el video al festival.
- Género: Categoría en la que el productor clasificó su cortometraje.
- Imágenes: Incomprensible que clase de información querían obtener. En la mayoría de los casos esta vacío este campo, en otros dice "video".
- Referencia Celular: el dispositivo móvil con el que se grabó el cortometraje.
- Sinopsis: Una breve sinopsis del cortometraje. Habrá que revisar si tiene un límite de caracteres.

- Sinopsis: Una breve sinopsis del cortometraje. Habrá que revisar si tiene un límite de caracteres.

Año 2016

En este año la información fue clasificada en diferentes hojas de cálculo de acuerdo a la categoría de los participantes, además se agregaron 2 categorías, Juvenil y SmarTIC, donde al parecer los de SmarTIC presentan algún tipo de discapacidad. Los datos capturados son los siguientes:

- ID: Supongo que el número de registro. No se cuentan con todos los ID consecutivos.
- Cómo se enteró
- Nombre del corto
- Género
- Sinopsis
- País
- Ciudad: Cuenta con muchos espacios vacíos, si el participante no es de Colombia se deja vacío.
- Departamento: Es como el concepto de estado en México.
- Referencia celular

Año 2017

En este año se agregó una categoría llamada "Reto al guión" la cual contiene muy poca información y no es compatible con la información proporcionada por las demás categorías, será descartada. A partir ahora obtenemos información del que registro el cortometraje, pero también se agrega una mini encuesta de SÍ/NO al final de la forma de inscripción. También se decidió categorizar los Celulares utilizados, agregando dos clasificaciones: "Tipo de dispositivo" y "marca". Los datos capturados son:

- ID.
- Cómo se enteró
- Sexo
- Edad
- Colombiano: Me imagino que es otra forma de decir el país de procedencia.
- Departamento: En la cual se capturó la ciudad.
- (Campo vacío): En el cual se capturó el departamento.
- Nombre del corto
- Género
- Sinopsis
- (Campo vacío): En el cual se repite el valor guardado en Departamento.
- (Campo vacío): En el cual se repite el departamento.
- Tipo de dispositivo
- Marca: del celular
- Referencia: cual celular usó.
- Campaña publicitaria. *A partir de aquí empieza la mini encuesta*
- making of: Si cuenta con detrás de camaras.
- documental: Es tipo documental?. Lo cual considero innecesario ya que Documental es un género de cortometrajes.
- infantil: Además de que es una categoría de participantes, también es un género de cortometrajes. No entiendo la existencia de este campo.
- afiche: Segundo google, es otra forma de decir cartel o poster. Me imagino que si tuvo campaña publicitaria debe tener poster.
- Cantidad de personas que realizaron el corto. Yo creo que es un campo muy ambiguo, ya que puede que la persona que realizó el corto sea solo una, pero en el participaron más de una. Como esta lleno este campo parece acertado que se refiera a las personas que participaron y no que lo realizaron.

Año 2018

En este año tenemos los datos de dos países, Colombia y México.

Colombia

Para este año tenemos los datos concentrados y no en diferentes hojas de cálculo. Los datos son los siguientes:

- Carpeta: Una cadena de caracteres, parecen cifrados.
- Categoría
- ¿Cómo se enteró?
- Edad Participante
- Colombiano
- País
- Departamento
- Ciudad
- Nombre Corto
- Género Corto
- Duración: Tal parece que en segundos, por la magnitud de las cifras, pero algunos son tan pequeños que pudieron haberlo puesto en minutos.
- Sinopsis
- País del corto: decidieron categorizar el cortometraje.
- Departamento del corto

- Ciudad corto
- Tipo de dispositivo
- Marca Dispositivo
- Referencia Dispositivo
- Otro dispositivo: Se abre la posibilidad de usar más de un dispositivo, antes se usaba una barra para dividir diferentes dispositivos pero ahora se pueden poner en dos campos diferentes.
- Corto Dias: tiempo de produccion.
- Corto Marcas: No tengo ni idea que signifique esto.
- Personas
- Discapacidad: Debido a que se concentraron los datos, en la misma lista están los de SmarTIC y ellos tienen discapacidad, los demás no.

México

Se realizó un festival en México con la misma mecánica que en Colombia y aquí se almacenaron los siguientes datos:

- Categoría: Parece ser que no se les explicó a los participantes que la categoría era donde iban a inscribir su cortometraje, no el género del mismo.
- Edad participante
- País participante
- Departamento participante
- Ciudad participante
- Nombre de Corto
- Género
- Sinopsis
- País
- Departamento
- Ciudad
- Tipo de dispositivo
- Marca de dispositivo
- Referencia dispositivo
- Dias
- Personas

Preprocesamiento

Los datos me fueron proporcionados en documentos .xlsx (extensión de Office Excel 2013 o superior), al momento de procesarlo en bash, la información proyectada era incomprendible. Googleando un poco, encontré el método para convertir el .xlsx a .csv y tratar la información.

Primero instalé el GNUmeric

```
In [ ]: $ sudo apt-get gnumeric
```

Después pasé a la conversión de los documentos.

```
In [ ]: $ ssconvert 2015.xlsx 2015.csv
$ ssconvert 2016.xlsx 2016.csv
$ ssconvert 2017.xlsx 2017.csv
$ ssconvert 2018.xlsx 2018.csv
$ ssconvert 2018mx.xlsx 2018mx.csv
```

Al final los archivos ya eran tratables en bash. Para trabajar con los archivos me topo con que estaban muy sucios los datos, debido a la conversión, por eso modifique los parámetros del convertidor y agregue la siguiente instrucción a cada uno.

```
In [ ]: --export-options="separator=i"
```

Decidí usar el separador ";" debido a que las comas, y otros símbolos eran utilizados en títulos de los cortos o en las sinopsis de los cortos. Consideré que era el signo que menos se utilizaba. Al final usé el comando awk para obtener la información de cada archivo. Además cambie la extensión de salida a .txt

Las columnas que utilicé y que consideré en común fueron:

- Año (Columna agregada)
- Categoría
- País
- Género
- ¿Cómo se enteró?
- Referencia celular

Herramienta

Los datos requieren limpiarse aún más, ya que al usar el comando:

```
In [ ]: awk -F '\t' '{ if(NF > 15) print $2}' 2015.txt | sort | uniq -c
```

8
309 Aficionado
1 Categoría
73 Infantil
143 Profesional

1, 2, holas a todos 31 4, 15

aún me arroja espacios vacios.

Conclusiones

En esta practica pude aprender a utilizar la plataforma de Jupyter Notebook, el bash de linux y a convertir archivos desde terminar. A pesar del preprocesamiento realizado, creo que tengo que buscar otro símbolo para realizar la separacion de columnas. La información obtenida en el año 2015 es prácticamente el cuello de botella de la información que podría procesar.

--28 de Enero 2019-- Luis Angel Gutierrez Rodriguez [1484412](#)

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 a few seconds ago

P01

June 6, 2019

1 Reporte de práctica 1: Preparación de los datos

1.1 Origen de los datos

Los datos a analizar durante el curso fueron proporcionados por la Dra. Elisa y son de un festival de cine que se lleva a cabo anualmente en Colombia y se está empezando a difundir la idea en México. Se cuenta con la información de las boletas de inscripción al festival de los años 2015 al 2018. La información capturada varía y aumenta de acuerdo al año de su captura, es decir, la información de 2018 contiene más datos que la de 2015. A causa del crecimiento de la información cada año, la limpieza de los mismos me dejará con muchas celdas vacías.

Las preguntas con las que se pudiera trabajar serían:

- ¿Cuáles son los teléfonos móviles que se utilizan más los productores de acuerdo a la categoría que pertenecen?
- ¿Cuál es el género que más producciones tiene?
- ¿Cuál fue el medio más efectivo por el que la mayoría de los concursantes se enteraron del festival?
- ¿Cuáles son los países extranjeros más activos en el festival, en qué categorías participan y qué géneros prefieren?

1.2 Datos disponibles

Como los datos disponibles varían con cada año, buscaré describir los datos por cada archivo y posteriormente buscaremos el común de estos.

1.2.1 Año 2015

El primer año del que se tiene registro. En este festival solo se contaba con los siguientes 8 datos:
* País: País de procedencia de la producción. * Categoría: Si el video es producido por infantiles, profesionales ó aficionados. * Nombre del corto. * Medio de envío: El método por el cual la producción del corto hizo llegar el video al festival. * Género: Categoría en la que el productor clasificó su cortometraje. * Imágenes: Incomprendible que clase de información querían obtener. En la mayoría de los casos está vacío este campo, en otros dice "video". * Referencia Celular: el dispositivo móvil con el que se grabó el cortometraje. * Sinopsis: Una breve sinopsis del cortometraje. Habrá que revisar si tiene un límite de caracteres.

1.2.2 Año 2016

En este año la información fue clasificada en diferentes hojas de cálculo de acuerdo a la categoría de los participantes, además se agregaron 2 categorías, Juvenil y SmarTIC, donde al parecer los de SmarTIC presentan algún tipo de discapacidad. Los datos capturados son los siguientes: * ID: Supongo que el número de registro. No se cuentan con todos los ID consecutivos. * Cómo se enteró * Nombre del corto * Género * Sinopsis * País * Ciudad: Cuenta con muchos espacios vacíos, si el participante no es de Colombia se deja vacío. * Departamento: Es como el concepto de estado en México. * Referencia celular

1.2.3 Año 2017

En este año se agregó una categoría llamada "Reto al guion" la cual contiene muy poca información y no es compatible con la información proporcionada por las demás categorías, será descartada. A partir ahora obtenemos información del que registro el cortometraje, pero también se agrega una mini encuesta de SI/NO al final de la forma de inscripción. También se decidió categorizar los Celulares utilizados, agregando dos elementos a la clasificación: "Tipo de dispositivo" y "marca". Los datos capturados son: * ID. * Cómo se enteró * Sexo * Edad * Colombiano: Me imagino que es otra forma de decir el país de procedencia. * Departamento: En la cual se capturó la ciudad. * (Campo vacío): En el cual se capturó el departamento.

* Nombre del corto * Género * Sinopsis * (Campo vacío): En el cual se repite el valor guardado en Departamento. * (Campo vacío): En el cual se repite el departamento. * Tipo de dispositivo * Marca: del celular * Referencia: cual celular usó.

- Campaña publicitaria. *A partir de aquí empieza la mini encuesta*
- making of: Si cuenta con detrás de cámaras.
- documental: Es tipo documental?. Lo cual considero innecesario ya que Documental es un género de cortometrajes.
- infantil: Además de que es una categoría de participantes, también es un género de cortometrajes. No entiendo la existencia de este campo.
- afiche: Según Google, es otra forma de decir cartel o póster. Me imagino que si tuvo campaña publicitaria debe tener póster.
- Cantidad de personas que realizaron el corto. Yo creo que es un campo muy ambiguo, ya que puede que la persona que realizó el corto sea solo una, pero en él participaron más de una. Como esta lleno este campo parece acertado que se refiera a las personas que participaron y no que lo realizaron.

1.2.4 Año 2018

En este año tenemos los datos de dos países, Colombia y México.

Colombia Para este año tenemos los datos concentrados y no en diferentes hojas de cálculo. Los datos son los siguientes: * Carpeta: Una cadena de caracteres, parecen cifrados. * Categoría * ¿Cómo se enteró? * Edad Participante * Colombiano * País * Departamento * Ciudad * Nombre Corto * Género Corto * Duración: Tal parece que en segundos, por la magnitud de las cifras, pero algunos son tan pequeños que pudieron haberlo puesto en minutos. * Sinopsis * País del corto: decidieron categorizar el cortometraje. * Departamento del corto * Ciudad corto * Tipo de dispositivo * Marca Dispositivo * Referencia Dispositivo * Otro dispositivo: Se abre la posibilidad de usar

más de un dispositivo, antes se usaba una barra para dividir diferentes dispositivos pero ahora se pueden poner en dos campos diferentes. * Corto Días: tiempo de producción. * Corto Marcas: No tengo ni idea que signifique esto. * Personas * Discapacidad: Debido a que se concentraron los datos, en la misma lista están los de SmarTIC y ellos tienen discapacidad, los demás no.

México Se realizó un festival en México con la misma mecánica que en Colombia y aquí se almacenaron los siguientes datos: * Categoría: Parece ser que no se les explicó a los participantes que la categoría era donde iban a inscribir su cortometraje, no el género del mismo. * Edad participante * País participante * Departamento participante * Ciudad participante * Nombre de Corto * Género * Sinopsis * País * Departamento * Ciudad * Tipo de dispositivo * Marca de dispositivo * Referencia dispositivo * Días * Personas

1.3 Preprocesamiento

Los datos me fueron proporcionados en documentos .xlsx (extensión de Office Excel 2013 o superior), al momento de procesarlo en bash, la información proyectada era incomprendible. Googleando un poco, encontré el método para convertir el .xlsx a .csv y tratar la información.

Primero instalé el GNUMeric

```
In [ ]: $ sudo apt-get gnumeric
```

Después pasé a la conversión de los documentos.

```
In [ ]: $ ssconvert 2015.xlsx 2015.csv  
$ ssconvert 2016.xlsx 2016.csv  
$ ssconvert 2017.xlsx 2017.csv  
$ ssconvert 2018.xlsx 2018.csv  
$ ssconvert 2018mx.xlsx 2018mx.csv
```

Al final los archivos ya eran tratables en bash. Para trabajar con los archivos me topé con que estaban muy sucios los datos, debido a la conversión, por eso modifique los parámetros del convertidor y agregué la siguiente instrucción a cada uno.

```
In [ ]: --export-options="separator=¤"
```

Decidí usar el separador "¤" debido a que las comas, y otros símbolos eran utilizados en títulos de los cortos o en las sinopsis de los cortos. Consideré que era el signo que menos se utilizaba. Al final usé el comando *awk* para obtener la información de cada archivo. Además cambie la extensión de salida a .txt

Las columnas que utilicé y que consideré en común fueron: * Año (Columna agregada) * Categoría * País * Género * ¿Cómo se enteró? * Referencia celular

Los datos requieren limpiarse aún más, ya que al usar el comando:

```
In [ ]: awk -F '¤' '{ if(NF > 15) print $2}' 2015.txt | sort | uniq -c  
8  
309 Aficionado  
1 Categoría  
73 Infantil  
143 Profesional
```

aún me arroja espacios vacíos.

1.4 Conclusiones

En esta práctica pude aprender a utilizar la plataforma de Jupyter Notebook, el bash de linux y a convertir archivos desde terminar. A pesar del preprocesamiento realizado, creo que tengo que buscar otro símbolo para realizar la separación de columnas. La información obtenida en el año 2015 es prácticamente el cuello de botella de la información que podría procesar.

--04 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 2: Lectura y manipulación de datos

Complicaciones

CIENCIA_DE_DATOS (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master)
/ .ipynb_checkpoints (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master/.ipynb_checkpoints)

Reporte de Práctica 2: Lectura y manipulación de datos

Objetivo

En esta práctica, se cargan los datos en marcos de datos (inglés: data frame) con la librería [pandas](https://pandas.pydata.org/) (<https://pandas.pydata.org/>) para poder iniciar su procesamiento.

Los objetivos de esta práctica son:

- Leer data frames de archivos
- Escribir data frames en CSV
- Agregar columnas en data frames
- Combinar dos o más data frames
- Filtrar renglones de un data frame En la práctica anterior decidí descargar un convertidor, ya que tengo archivos de excel y los convertí a .csv para tratamiento de datos, pero esa conversión me trajo muchos errores y tenía que guardar cada hoja individualmente, lo cual era muy tedioso. Después de investigar descubrí que la librería pandas también puede leer los archivos de excel, por lo que decidí volver a tomar los archivos originales para realizar esta prueba, esperando mejores resultados al tratar los archivos con pandas.

Lectura de datos

Para empezar a tratar los datos, use el orden por año, al utilizar los datos de '2015.xlsx', me lanzó este error ya que faltaban instalar librerías para el tratamiento de los archivos de excel:

```
In [ ]: Traceback (most recent call last):
      File "/usr/local/lib/python3.6/dist-packages/pandas/io/excel.py", line 391, in __init__
        import xlrd
ModuleNotFoundError: No module named 'xlrd'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.6/dist-packages/pandas/util/_decorators.py", line 188, in wrapper
    return func(*args, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/pandas/util/_decorators.py", line 188, in wrapper
    return func(*args, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/pandas/io/excel.py", line 350, in read_excel
    io = ExcelFile(io, engine=engine)
  File "/usr/local/lib/python3.6/dist-packages/pandas/io/excel.py", line 653, in __init__
    self._reader = self._engines[engine](self._io)
  File "/usr/local/lib/python3.6/dist-packages/pandas/io/excel.py", line 393, in __init__
    raise ImportError(err_msg)
ImportError: Install xlrd >= 1.0.0 for Excel support
```

Así que usé el siguiente comando, para instalar la librería que me recomendó pandas:

```
In [ ]:  sudo python3 -m pip install xlrd
```

Después de instalar todo correctamente usé un constructor específico para importar mis archivos a data frames.

```
In [ ]: import pandas as pd
df2015 = pd.read_excel('2015.xlsx', index_col=None, header=0, sheet_name=0)
df2016 = pd.read_excel('2016.xlsx', index_col=0, header=0, sheet_name=[0,1,2,3,4])
df2017 = pd.read_excel('2017.xlsx', index_col=0, header=0, sheet_name=[0,1,2,3,4])
df2018 = pd.read_excel('2018.xlsx', index_col=None, header=0, sheet_name=0)
df2018mx = pd.read_excel('2018mx.xlsx', index_col=None, header=0, sheet_name=0)
```

Los parámetros que modifique en cada instancia son:

- index_col: indica cual de las columnas tiene un identificador para las filas, si no el archivo no tiene se ponen None.
- header: indica si los archivos tienen fila de encabezados, si no tiene se pone None.
- sheet_name: indica cuales son las hojas del documento que deseó importar, por defecto se agrega la hoja 0 pero si se quieren agregar mas se inserta un arreglo con el nombre o número de hoja.

El parámetro **sheet_name** me ayudó a importar todas las hojas de los excel de los años 2016 y 2017 ya que en estos años se capturó la información en diferentes hojas. Al hacer esto, el objeto receptor se volvió un diccionario de data frames, por lo que tuve que combinar los data frames, lo cual lo explicaré más adelante.

Escribir data frames en CSV.

Una vez cargados los datos en data frames, busqué la forma de exportarlos a CSV. Debido a que los datos tienen un campo llamado "Sinópsis" el cual contiene muchos de los caracteres especiales para hacer la separación de los valores decidí usar el separador "\\" (slash invertido), se agregan 2 porque uno es el de fuga y el otro es el que se imprime.

Para exportar a CSV simplemente usamos el comando:

```
In [ ]: df2015.to_csv("2015.csv", sep='\\', index=False)
df2016[0].to_csv("2016-0.csv", sep='\\', index=False)
df2016[1].to_csv("2016-1.csv", sep='\\', index=False)
df2016[2].to_csv("2016-2.csv", sep='\\', index=False)
df2016[3].to_csv("2016-3.csv", sep='\\', index=False)
df2016[4].to_csv("2016-4.csv", sep='\\', index=False)
df2017[0].to_csv("2017-0.csv", sep='\\', index=False)
df2017[1].to_csv("2017-1.csv", sep='\\', index=False)
df2017[2].to_csv("2017-2.csv", sep='\\', index=False)
df2017[3].to_csv("2017-3.csv", sep='\\', index=False)
df2017[4].to_csv("2017-4.csv", sep='\\', index=False)
df2018.to_csv("2018.csv", sep='\\', index=False)
df2018mx.to_csv("2018mx.csv", sep='\\', index=False)
```

Omití el index ya que no es información relevante para tratar nuestros datos. No hay una continuidad de los Id, ni una herencia de id de años anteriores.

Agregar columnas

Debido a que en la práctica anterior definimos los datos a tomar en cuenta:

- Año (Columna agregada)
- Categoría
- País
- Género
- ¿Cómo se enteró?
- Referencia celular

Es necesario agregar la columna Año a todos los data frames para identificar el año de procedencia al unir todos los datos. Además de esto, se tomaron solo columnas que tenían la información necesaria y se descartaron las demás.

Para completar la información de los data frames usé los siguientes comandos:

```
In [ ]: df2015['Año']='2015'
df2016[0]['Año']='2016'
df2016[0]['Categoria']='Infantil'
df2016[1]['Año']='2016'
df2016[1]['Categoria']='Juvenil'
df2016[2]['Año']='2016'
df2016[2]['Categoria']='Aficionado'
df2016[3]['Año']='2016'
df2016[3]['Categoria']='Profesional'
df2016[4]['Año']='2016'
df2016[4]['Categoria']='SmarTIC'
df2017[0]['Año']='2017'
df2017[0]['Categoria']='Aficionado'
df2017[1]['Año']='2017'
df2017[1]['Categoria']='Juvenil'
df2017[2]['Año']='2017'
df2017[2]['Categoria']='Profesional'
df2017[3]['Año']='2017'
df2017[3]['Categoria']='SmarTIC'
df2017[4]['Año']='2017'
df2017[4]['Categoria']='Reto al guión'
df2018['Año']='2018'
df2018mx['Año']='2018'
```

Agregué las categorías a 2016 y 2017 ya que en esos años las categorías estaban separadas por hoja en la información original. Aproveché esta oportunidad para filtrar las columnas que necesitaba por data frame y sobreescribí los data frames, considerando que solo me iba a quedar con la información que necesitaba.

```
In [ ]: df2015 = df2015[['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2016[0] = df2016[0][['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2016[1] = df2016[1][['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2016[2] = df2016[2][['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2016[3] = df2016[3][['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2016[4] = df2016[4][['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2017[0] = df2017[0][['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2017[1] = df2017[1][['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2017[2] = df2017[2][['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2017[3] = df2017[3][['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2017[4] = df2017[4][['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2018 = df2018[['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
df2018mx = df2018mx[['Año','Categoria','País','Género','Como se enteró','Referencia Celular']]
```

Combinar 2 o más data frames

Dado que ya trate los datos y los filtré las columnas que solo quería llegó el momento de combinar los data frames más tediosos de toda la actividad, los años 2016 y 2017, cabe mencionar que hasta este punto fusioné los data frames porque además de estar resolviendo las actividades en el patrón sugerido, también hacia falta limpiar los datos, hasta este punto las columnas están ordenadas en el mismo orden y ya se agregaron y llenaron los campos que faltaban, como año y categoría.

Para combinar todos los data frames del año 2016 y 2017 con sus correspondientes años y además también combiné df2018 y 2018mx en df2018, usé los siguientes comandos:

```
In [ ]: df2016 = pd.concat([df2016[0],df2016[1],df2016[2],df2016[3],df2016[4]])
df2017 = pd.concat([df2017[0],df2017[1],df2017[2],df2017[3],df2017[4]])
df2018 = pd.concat([df2018,df2018mx])
```

Por último se concatenaron todas los data frames en uno solo donde ya estarían ordenados.

```
In [ ]: Concurso_cine = pd.concat([df2015,df2016,df2017,df2018])
Concurso_cine.to_csv('datosLimpiosCine.csv', sep='\\', index=False)
```

Filtrar renglones de data frame

Para filtrar renglones use pandas, la forma de hacerlo es muy sencilla, primero se introduce una función lógica entre corchetes al data frame y nos arroja los renglones que cumplen con el resultado verdadero de la función lógica. Por ejemplo, si queremos saber ¿Cuántos concursantes de Argentina participaron en 2016?, usamos el siguiente comando:

```
In [ ]: df2016[df2016['Pais'] == 'Argentina']
```

```
In [ ]:      Año    Categoría Cómo se enteró      País Referencia del celular
ID
45.0   2016    infantil           NaN  Argentina           NaN
105.0  2016    juvenil            redes Argentina        iPhone 5
108.0  2016    profesional         internet Argentina       iphone
```

Conclusiones

A pesar de que ya había trabajado antes con los archivos en la práctica anterior, ahora me ahorre mucho tiempo trabajando con las múltiples hojas de los archivos 2016 y 2017, ya conocía pandas, lo utilicé como un reemplazo de R studio ya que estaba más familiarizado con Python que con R.

Por fin pude unir todos los datos en un solo archivo, en la práctica anterior lo veía imposible y muy tedioso, pero esta herramienta me ayudó a tratar los datos solo variando el índice de la hoja del archivo original.

Aún y cuando en la práctica 2 que explica la Dra. Elisa, yo no usé bash en esta ocasión debido a que no fue necesario limpieza extra o tratamiento fuera de pandas, ya que es una herramienta muy completa.

--11 de Febrero 2019-- Luis Angel Gutierrez Rodriguez [1484412 \(tel:1484412\)](#)

P02

June 6, 2019

1 Reporte de práctica 2: Lectura y manipulación de datos

1.1 Objetivo

En esta práctica, se cargan los datos en marcos de datos (inglés: data frame) con la librería [pandas](#) para poder iniciar su procesamiento.

Los objetivos de esta práctica son:

- * Leer data frames de archivos
- * Escribir data frames en CSV
- * Agregar columnas en data frames
- * Combinar dos o más data frames
- * Filtrar renglones de un data frame

En la práctica anterior decidí descargar un convertidor, ya que tengo archivos de excel y los convertí a .csv para tratamiento de datos, pero esa conversión me trajo muchos errores y tenía que guardar cada hoja individualmente, lo cual era muy tedioso. Después de investigar descubrí que la librería pandas también puede leer los archivos de excel, por lo que decidí volver a tomar los archivos originales para realizar esta prueba, esperando mejores resultados al tratar los archivos con pandas.

1.2 Lectura de datos

Para empezar a tratar los datos, use el orden por año, al utilizar los datos de '2015.xlsx', me lanzó este error ya que faltaban instalar librerías para el tratamiento de los archivos de excel:

```
In [ ]: Traceback (most recent call last):
      File "/usr/local/lib/python3.6/dist-packages/pandas/io/excel.py", line 391, in __init__
        import xlrd
ModuleNotFoundError: No module named 'xlrd'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.6/dist-packages/pandas/util/_decorators.py", line 188,
    return func(*args, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/pandas/util/_decorators.py", line 188,
    return func(*args, **kwargs)
  File "/usr/local/lib/python3.6/dist-packages/pandas/io/excel.py", line 350, in read_
    io = ExcelFile(io, engine=engine)
  File "/usr/local/lib/python3.6/dist-packages/pandas/io/excel.py", line 653, in __init__
    self._reader = self._engines[engine](self._io)
```

```
File "/usr/local/lib/python3.6/dist-packages/pandas/io/excel.py", line 393, in __init__
    raise ImportError(err_msg)
ImportError: Install xlrd >= 1.0.0 for Excel support
```

Así que usé el siguiente comando, para instalar la librería que me recomendó pandas:

```
In [ ]: $ sudo python3 -m pip install xlrd
```

Después de instalar todo correctamente usé un constructor específico para importar mis archivos a data frames.

```
In [ ]: import pandas as pd
df2015 = pd.read_excel('2015.xlsx', index_col=None, header=0, sheet_name=0)
df2016 = pd.read_excel('2016.xlsx', index_col=0, header=0, sheet_name=[0,1,2,3,4])
df2017 = pd.read_excel('2017.xlsx', index_col=0, header=0, sheet_name=[0,1,2,3,4])
df2018 = pd.read_excel('2018.xlsx', index_col=None, header=0, sheet_name=0)
df2018mx = pd.read_excel('2018mx.xlsx', index_col=None, header=0, sheet_name=0)
```

Los parámetros que modifique en cada instancia son: * `index_col`: indica cual de las columnas tiene un identificador para las filas, si no el archivo no tiene se ponen None. * `header`: indica si los archivos tienen fila de encabezados, si no tiene se pone None. * `sheet_name`: indica cuales son las hojas del documento que deseó importar, por defecto se agrega la hoja 0 pero si se quieren agregar más se inserta un arreglo con el nombre o número de hoja.

El parámetro `sheet_name` me ayudó a importar todas las hojas de los excel de los años 2016 y 2017 ya que en estos años se capturó la información en diferentes hojas. Al hacer esto, el objeto receptor se volvió un diccionario de data frames, por lo que tuve que combinar los data frames, lo cual lo explicaré más adelante.

1.3 Escribir data frames en CSV.

Una vez cargados los datos en data frames, busqué la forma de exportarlos a CSV. Debido a que los datos tiene un campo llamado "Sinopsis" el cual contiene muchos de los caracteres especiales para hacer la separación de los valores decidí usar el separador "\\" (slash invertido), se agregan 2 porque uno es el de fuga y el otro es el que se imprime.

Para exportar a CSV simplemente usamos el comando:

```
In [ ]: df2015.to_csv("2015.csv", sep='\\', index=False)
df2016[0].to_csv("2016-0.csv", sep='\\', index=False)
df2016[1].to_csv("2016-1.csv", sep='\\', index=False)
df2016[2].to_csv("2016-2.csv", sep='\\', index=False)
df2016[3].to_csv("2016-3.csv", sep='\\', index=False)
df2016[4].to_csv("2016-4.csv", sep='\\', index=False)
df2017[0].to_csv("2017-0.csv", sep='\\', index=False)
df2017[1].to_csv("2017-1.csv", sep='\\', index=False)
df2017[2].to_csv("2017-2.csv", sep='\\', index=False)
df2017[3].to_csv("2017-3.csv", sep='\\', index=False)
df2017[4].to_csv("2017-4.csv", sep='\\', index=False)
df2018.to_csv("2018.csv", sep='\\', index=False)
df2018mx.to_csv("2018mx.csv", sep='\\', index=False)
```

Omití el `index` ya que no es información relevante para tratar nuestros datos. No hay una continuidad de los Id, ni una herencia de identificadores de años anteriores.

1.4 Agregar columnas

Debido a que en la práctica anterior definimos los datos a tomar en cuenta: * Año (Columna agregada) * Categoría * País * Género * fCómo se enteró? * Referencia celular

Es necesario agregar la columna Año a todos los data frames para identificar el año de procedencia al unir todos los datos. Además de esto, se tomaron solo columnas que tenían la información necesaria y se descartaron las demás.

Para completar la información de los data frames usé los siguientes comandos:

```
In [ ]: df2015['Año']='2015'
         df2016[0]['Año']='2016'
         df2016[0]['Categoría']='Infantil'
         df2016[1]['Año']='2016'
         df2016[1]['Categoría']='Juvenil'
         df2016[2]['Año']='2016'
         df2016[2]['Categoría']='Aficionado'
         df2016[3]['Año']='2016'
         df2016[3]['Categoría']='Profesional'
         df2016[4]['Año']='2016'
         df2016[4]['Categoría']='SmarTIC'
         df2017[0]['Año']='2017'
         df2017[0]['Categoría']='Aficionado'
         df2017[1]['Año']='2017'
         df2017[1]['Categoría']='Juvenil'
         df2017[2]['Año']='2017'
         df2017[2]['Categoría']='Profesional'
         df2017[3]['Año']='2017'
         df2017[3]['Categoría']='SmarTIC'
         df2017[4]['Año']='2017'
         df2017[4]['Categoría']='Reto al guión'
         df2018['Año']='2018'
         df2018mx['Año']='2018'
```

Agregué las categorías a 2016 y 2017 ya que en esos años las categorías estaban separadas por hoja en la información original. Aproveché esta oportunidad para filtrar las columnas que necesitaba por data frame y sobrescribí los data frames, considerando que solo me iba a quedar con la información que necesitaba.

```
In [ ]: df2015 = df2015[['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
         df2016[0] = df2016[0][['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
         df2016[1] = df2016[1][['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
         df2016[2] = df2016[2][['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
         df2016[3] = df2016[3][['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
         df2016[4] = df2016[4][['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
         df2017[0] = df2017[0][['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
         df2017[1] = df2017[1][['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
         df2017[2] = df2017[2][['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
         df2017[3] = df2017[3][['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
         df2017[4] = df2017[4][['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
```

```
df2018 = df2018[['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
df2018mx = df2018mx[['Año','Categoría','País','Género','Como se enteró','Referencia Celular']]
```

1.5 Combinar 2 o más data frames

Dado que ya trate los datos y los filtré las columnas que solo quería llegó el momento de combinar los data frames más tediosos de toda la actividad, los años 2016 y 2017, cabe mencionar que hasta este punto fusioné los data frames porque además de estar resolviendo las actividades en el patrón sugerido, también hacía falta limpiar los datos, hasta este punto las columnas están ordenadas en el mismo orden y ya se agregaron y llenaron los campos que faltaban, como año y categoría.

Para combinar todos los data frames del año 2016 y 2017 con sus correspondientes años y además también combiné df2018 y 2018mx en df2018, usé los siguientes comandos:

```
In [ ]: df2016 = pd.concat([df2016[0],df2016[1],df2016[2],df2016[3],df2016[4]])
df2017 = pd.concat([df2017[0],df2017[1],df2017[2],df2017[3],df2017[4]])
df2018 = pd.concat([df2018,df2018mx])
```

Por último se concatenaron todas los data frames en uno solo donde ya estarían ordenados.

```
In [ ]: Concurso_cine = pd.concat([df2015,df2016,df2017,df2018])
Concurso_cine.to_csv('datosLimpiosCine.csv', sep='\\', index=False)
```

1.6 Filtrar renglones de DataFrame

Para filtrar renglones use pandas, la forma de hacerlo es muy sencilla, primero se introduce una función lógica entre corchetes al DataFrame y nos arroja los renglones que cumplen con el resultado verdadero de la función lógica. Por ejemplo, si queremos saber ¿Cuántos concursantes de Argentina participaron en 2016?, usamos el siguiente comando:

```
In [ ]: df2016[df2016['País'] == 'Argentina']

In [ ]:      Año    Categoría Cómo se enteró      País Referencia del celular
ID
45.0   2016     infantil        NaN  Argentina           NaN
105.0   2016     juvenil       redes  Argentina      iPhone 5
108.0   2016   profesional     internet  Argentina      iphone
```

```
In [ ]:
```

1.7 Conclusiones

A pesar de que ya había trabajado antes con los archivos en la práctica anterior, ahora me ahorré mucho tiempo trabajando con las múltiples hojas de los archivos 2016 y 2017, ya conocía pandas, lo utilicé como un reemplazo de R studio ya que estaba más familiarizado con Python que con R.

Por fin pude unir todos los datos en un solo archivo, en la práctica anterior lo veía imposible y muy tedioso, pero esta herramienta me ayudó a tratar los datos solo variando el índice de la hoja del archivo original.

Aún y cuando en la práctica 2 que explica la Dra. Elisa, yo no usé bash en esta ocasión debido a que no fue necesario limpieza extra o tratamiento fuera de pandas, ya que es una herramienta muy completa.

--04 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 3: Estadística descriptiva básica

Complicaciones

CIENCIA_DE_DATOS (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master)
/ P3.ipynb (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master/P3.ipynb)

Reporte de Práctica 3: Estadística descriptiva básica

Objetivos

En esta práctica terminaremos de limpiar nuestros datos de la práctica anterior, también realizaremos categorizaciones de los datos algunos conteos, promedios y correlaciones.

Limpieza

Revisando los CSV de la práctica anterior, me di cuenta que las diferencias ortográficas en los nombres de las columnas me generaban columnas duplicadas y además muchos espacios vacíos, así que me di a la tarea de investigar cómo renombrar las columnas de un dataframe.

Primero necesitábamos obtener los nombres de las columnas cargadas al dataframe, y para eso ocupamos el código:

```
In [ ]: list(df2015)
list(df2016)
list(df2017)
list(df2018)
```

Cada línea nos arroja una lista de los nombres de las columnas del dataframe. Para no tener que unir las columnas, realicé los pasos de la práctica anterior hasta antes de la combinación de los datos, para asegurarse de que todas las columnas se llamen igual y evitar errores. En todos los dataframes tenemos 6 columnas de información que filtramos en la práctica 2, en todos los dataframes están en el mismo orden, gracias a esto, podemos usar la función set_axis.

```
In [ ]: df2015.set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2016[0].set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2016[1].set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2016[2].set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2016[3].set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2016[4].set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2017[0].set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2017[1].set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2017[2].set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2017[3].set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2018.set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
df2018mx.set_axis(['Año','Categoría','País','Género','¿Cómo se enteró?','Referencia Dispositivo'], axis='columns', inplace=True)
```

Ahora podemos proceder a combinar los dataframes sin tener columnas duplicadas y asegurandonos de que las columnas se llaman igual. Después de usar las funciones para combinar dataframes y exportar a csv, guardé toda la información en el archivo "datosLimpiosCine.csv", y ahora sabemos que tenemos 2735 registros en total.

Corrección de Nombres de Países

Al aplicar la función unique() en los nombres de los países, nos damos cuenta que escribieron los nombres en Inglés y Español, además de escribir con puntos y espacios donde no van. Como los datos son muy variados aplique el orden alfabético al array resultado y obtuve un error ya que en el año 2017 ponían en la columna País "Si" si era colombiano y "No" si no lo era pero como no especificaron el país, puse "Internacional", también a las colaboraciones con Colombia.

```
In [ ]: array(['Colombia', 'Irán', 'Georgia/Colombia', 'España', 'Venezuela',
'México', 'Ecuador', 'Francia', 'Argentina', 'Perú',
'Estados Unidos', nan, 'Canadá', 'Honduras', 'Brasil', 'Cuba',
'Uruguay', 'Alemania', 'COLOMBIA', 'colombia', 'bogota',
'colombia y mexico', 'Bogota', 'Colombia.', 'Marruecos',
'COLOMBIA', 'República de Colombia', 'colombia ',
'Colombia', 'Brasil y Panamá', 'República Dominicana', 'Colomba',
'À-Colombia', 'Venezuela/Colombia', 'francia', 'colomBIA',
'Brazil', 'Belgium', 'Canada', 'Colombia - China',
'Colombia - Estados Unidos', 'Antioquia,', 'Si', 'No', '1',
'Afganistán', 'Comoras', 'Afganistan', 'Spain', 'Mexico',
'United States of America', 'Comoros'], dtype=object)
```

Resultados después de la limpieza:

```
In [ ]: array(['Colombia', 'Irán', 'España', 'Venezuela', 'México', 'Ecuador',
'Francia', 'Argentina', 'Perú', 'Estados Unidos', nan, 'Canadá',
'Honduras', 'Brasil', 'Cuba', 'Uruguay', 'Alemania',
'Internacional', 'Marruecos', 'República Dominicana', 'Bélgica',
'Afganistán', 'Comoras'], dtype=object)
```

Todas estas modificaciones las realicé en una columna nueva basada en "País", llamada "Paises", para no arruinar la información y tener que volver a empezar si me equivocaba en algo. También limpiamos las categorías y pasamos de esto:

```
In [ ]: array(['Aficionado', 'Profesional', 'Infantil', nan, 'Juvenil', 'SmarTIC',
'AFICIONADO', 'CRONICAS', 'JUVENIL', 'FAMILIAR',
'SMARTIC INCLUYENTE', 'PROFESIONAL', 'HORROR', 'HUMOR'],
dtype=object)
```

Debido a que en México confundieron el concepto de categoría con género del corto, reemplacé los valores por una categoría llamada "nan" que es mejor que inventar una categoría. Y pasamos a esto:

```
In [ ]: array(['Aficionado', 'Profesional', 'Infantil', nan, 'Juvenil', 'SmarTIC'],
dtype=object)
```

Aprovechando que son los campos con menos errores, también limpiamos el campo de ¿Cómo se enteró? y pasamos de esto:

```
In [ ]: array(['Convocatoria', 'tv', 'internet', 'amigo', 'Amigo', 'Internet',
'redes', nan, 'prensa', 'Prensa', 'TV', 'Redes', 'radio', 'Radio',
'blanco', 'MSM', 'Redes sociales', 'Un(a) amigo(a) me contó',
'Televisión', 'TelevisiÃ³n', 'Un(a) amigo(a) me contÃ³',
'Mensaje de texto', 'Otra'], dtype=object)
```

A esto:

```
In [ ]: array(['Convocatoria', 'Televisión', 'Internet', 'Amigo',
'Redes Sociales', nan, 'Prensa', 'Radio', 'Otra',
'Mensaje de texto'], dtype=object)
```

Toda la información corregida la realice usando el comando replace con el siguiente formato:

```
In [ ]: dataframe['columnatemporal'] = dataframe.NombreColumna.replace('Ruido','DatoCoherente')
```

Como se podrá observar en los arreglos iniciales y como terminaron eliminando ruido, la cantidad de veces que ejecuté la función replace fueron muchas y no las alcancé a documentar, pero pueden darse una idea de como y cuantas veces lo utilice. Al final, después de asegurarse que los datos estaban correctos, reemplacé la columna original con los valores de la columna temporal.

También hice un segundo backup de los datos.

```
In [ ]: cine.to_csv('datosLimpiosCine2.csv', sep='\\', index=False)
```

Conteo y Promedio

Ahora la información que tenemos limpia es:

- Año
- País
- Categoría
- ¿Cómo se enteró?

Primero decidí instalar la librería "tabulate" de python para obtener resultados ordenados en tablas agradables a la vista. Después corrí el siguiente script para obtener la información de la participación de países divididos por año.

```
In [ ]: listaPaises = []
for contry in paises:
    listaPaises.append(contry,cine[(cine['País']==contry) & (cine['Año']==2015)].count()['Año'],cine[(cine['País']==contry) & (cine['Año']==2016)].count()['Año'],cine[(cine['País']==contry) & (cine['Año']==2017)].count()['Año'],cine[(cine['País']==contry) & (cine['Año']==2018)].count()['Año'],cine[(cine['País']==contry).count()['Año'], sep='\t')
print(tabulate(listaPaises, tablefmt="github"))
```

Utilicé el count()['Año'] debido a que es el único campo que no tiene NaN y me da un conteo exacto de los registros, pareciera que el contador no funcionó con los NaN por que me arrojó ceros y la cantidad de registros bajó a 2732. Obtuve estos resultados:

```
In [ ]: | País | 2015 | 2016 | 2017 | 2018 | Total |
|-----|-----|-----|-----|-----|-----|
| Colombia | 477 | 846 | 641 | 438 | 2402 |
| Irán | 1 | 0 | 0 | 0 | 1 |
| España | 26 | 17 | 1 | 6 | 50 |
| Venezuela | 5 | 1 | 1 | 0 | 7 |
| México | 8 | 3 | 2 | 171 | 184 |
| Ecuador | 2 | 3 | 0 | 1 | 6 |
| Francia | 2 | 1 | 0 | 0 | 3 |
| Argentina | 1 | 4 | 3 | 1 | 9 |
| Perú | 1 | 0 | 0 | 0 | 1 |
| Estados Unidos | 3 | 4 | 3 | 3 | 13 |
| nan | 0 | 2 | 0 | 1 | 3 |
| Canadá | 1 | 1 | 0 | 0 | 2 |
| Honduras | 1 | 0 | 0 | 0 | 1 |
| Brasil | 5 | 1 | 0 | 0 | 6 |
| Cuba | 1 | 0 | 0 | 0 | 1 |
| Uruguay | 2 | 1 | 0 | 0 | 3 |
| Alemania | 1 | 1 | 0 | 0 | 2 |
| Internacional | 0 | 5 | 11 | 0 | 16 |
| Marruecos | 0 | 1 | 0 | 0 | 1 |
| República Dominicana | 0 | 1 | 0 | 0 | 1 |
| Bélgica | 0 | 1 | 0 | 1 | 2 |
| Afganistán | 0 | 0 | 19 | 0 | 19 |
| Comoras | 0 | 0 | 1 | 1 | 2 |
|-----|-----|-----|-----|-----|
| Suma | 537 | 893 | 682 | 623 | 2735 |
```

Usé también un script para obtener los registros por Categoría y participación por Año. El código fué el siguiente:

```
In [ ]: listaCategoria = []
>>> for contry in categorias:
...     listaCategoria.append([contry,cine[(cine['Categoría']==contry) & (cine['Año']==2015)].count()
... ['Año'],cine[(cine['Categoría']==contry) & (cine['Año']==2016)].count()]['Año'],cine[(cine['Categoría'
... ]==contry) & (cine['Año']==2017)].count()]['Año'],cine[(cine['Categoría']==contry) & (cine['Año']==2
018)].count()]['Año'],cine[cine['Categoría']==contry].count()]['Año']])
...
>>> print(tabulate(listaCategoria, tablefmt="github"))
```

Me di cuenta que efectivamente el NaN no es contado por mi código, me doy cuenta de ésto, debido a que en 2017 tengo varios elementos que yo mismo convertí a NaN porque al momento de capturar la Categoría en México cometieron el error de escribir el Género del video, comprobé que en 2016 me hacen falta 2 video en NaN la tabla anterior y en ésta me hacen falta 165 registros en NaN. Obtuve los siguientes resultados:

Categoría	2015	2016	2017	2018	Total
Aficionado	313	522	377	241	1453
Profesional	147	151	85	53	436
Infantil	77	42	0	0	119
nan	0	0	0	165	0
Juvenil	0	123	132	83	338
SmarTIC	0	55	88	81	224
Suma	537	893	682	623	2735

Conclusiones

En ésta práctica me topé con que se pudo mejorar la limpieza de la práctica pasada. Aprendí a identificar escrituras únicas con la función unique() y a reemplazar string dentro de los dataframe con la función replace(), reemplacé los nombres de las columnas de todos los dataframes antes de unirlos y además me di cuenta que era innecesario el dataframe df2017[4] debido a que solo fué un evento de "Reto al Guión" no había referencia de videos ni de categorías, nada que lo relaciona a las otras tablas de registros.

A los datos les realicé 2 backups, uno después de la limpieza y el otro después de reemplazar el ruido en los campos.

Busqué la forma de sacar promedios y realizar correlaciones, pero en la práctica la Dra. Elisa dijo: "Todas las respuestas abiertas de texto quedan fuera del alcance de esta práctica." y en mis datos todo, salvo el año, son categorías de texto. Así que doy por concluída mi práctica.

--18 de Febrero 2019-- Luis Angel Gutierrez Rodriguez [1484412 \(tel:1484412\)](tel:1484412)

P03

June 6, 2019

1 Reporte de práctica 3: Estadística descriptiva básica

1.1 Objetivos

En esta práctica se terminan de limpiar los datos de la práctica anterior, también se realiza la categorización de los datos algunos conteos, promedios y correlaciones.

1.2 Limpieza

Revisando los CSV de la práctica anterior, se pudo observar que las diferencias ortográficas en los nombres de las columnas me generaban columnas duplicadas y además muchos espacios vacíos, se busca investigar cómo renombrar las columnas de un DataFrame.

Primero necesitábamos obtener los nombres de las columnas cargadas al DataFrame, y para eso ocupamos el código:

```
In [ ]: list(df2015)
        list(df2016)
        list(df2017)
        list(df2018)
```

Cada línea nos arroja una lista de los nombres de las columnas del DataFrame. Para no tener que unir las columnas, realicé los pasos de la práctica anterior hasta antes de la combinación de los datos, para asegurarse de que todas las columnas se llamen igual y evitar errores. En todos los DataFrame tenemos seis columnas de información que filtramos en la práctica dos, en todos los DataFrame están en el mismo orden, gracias a esto, podemos usar la función set_axis.

```
In [ ]: df2015.set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia Dispe
df2016[0].set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia D
df2016[1].set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia D
df2016[2].set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia D
df2016[3].set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia D
df2016[4].set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia D
df2017[0].set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia D
df2017[1].set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia D
df2017[2].set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia D
df2017[3].set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia D
df2018.set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia Dispe
df2018mx.set_axis(['Año', 'Categoría', 'País', 'Género', '¿Cómo se enteró?', 'Referencia Dispe
```

Ahora podemos proceder a combinar los conjuntos de datos sin tener columnas duplicadas y asegurándonos de que las columnas se llaman igual. Después de usar las funciones para combinarlos y exportar a csv, guardé toda la información en el archivo "datosLimpiosCine.csv", y ahora sabemos que tenemos 2735 registros en total.

1.2.1 Corrección de Nombres de Países

Al aplicar la función unique() en los nombres de los países, nos damos cuenta que escribieron los nombres en Inglés y Español, además de escribir con puntos y espacios donde no van. Como los datos son muy variados aplique el orden alfabético al array resultado y obtuve un error ya que en el año 2017 ponían en la columna País "Si" si era colombiano y "No" si no lo era pero como no especificaron el país, puse "Internacional", también a la colaboración con Colombia.

```
In [ ]: array(['Colombia', 'Irán', 'Georgia/Colombia', 'España', 'Venezuela',
    'México', 'Ecuador', 'Francia', 'Argentina', 'Perú',
    'Estados Unidos', nan, 'Canadá', 'Honduras', 'Brasil', 'Cuba',
    'Uruguay', 'Alemania', 'COLOMBIA', 'colombia', 'bogota',
    'colombia y mexico', 'Bogota', 'Colombia.', 'Marruecos',
    'COLO MBIA', 'República de Colombia', 'colombia ,',
    'Colombia, Brasil y Panamá', 'Republica Dominicana', 'Colomba',
    'ÁtColombia', 'Venezuela/Colombia', 'francia', 'colomBIA',
    'Brazil', 'Belgium', 'Canada', 'Colombia - China',
    'Colombia - Estados Unidos', 'Antioquia,', 'Si', 'No', '1',
    'Afganistán', 'Comoras', 'Afganistan', 'Spain', 'Mexico',
    'United States of America', 'Comoros'], dtype=object)
```

Resultados después de la limpieza:

```
In [ ]: array(['Colombia', 'Irán', 'España', 'Venezuela', 'México', 'Ecuador',
    'Francia', 'Argentina', 'Perú', 'Estados Unidos', nan, 'Canadá',
    'Honduras', 'Brasil', 'Cuba', 'Uruguay', 'Alemania',
    'Internacional', 'Marruecos', 'Republica Dominicana', 'Bélgica',
    'Afganistán', 'Comoras'], dtype=object)
```

Todas estas modificaciones las realicé en una columna nueva basada en "País", llamada "Países", para no arruinar la información y tener que volver a empezar si me equivocaba en algo. También limpiamos las categorías y pasamos de esto:

```
In [ ]: array(['Aficionado', 'Profesional', 'Infantil', nan, 'Juvenil', 'SmarTIC',
    'AFICIONADO', 'CRONICAS', 'JUVENIL', 'FAMILIAR',
    'SMARTIC INCLUYENTE', 'PROFESIONAL', 'HORROR', 'HUMOR'],
    dtype=object)
```

Debido a que en México confundieron el concepto de categoría con género del corto, reemplacé los valores por una categoría llamada "nan" que es mejor que inventar una categoría. Y pasamos a esto:

```
In [ ]: array(['Aficionado', 'Profesional', 'Infantil', nan, 'Juvenil', 'SmarTIC'],
    dtype=object)
```

Aprovechando que son los campos con menos errores, también limpiamos el campo de fCómo se enteró? y pasamos de esto:

```
In [ ]: array(['Convocatoria', 'tv', 'internet', 'amigo', 'Amigo', 'Internet',
   'redes', nan, 'prensa', 'Prensa', 'TV', 'Redes', 'radio', 'Radio',
   'blanco', 'MSM', 'Redes sociales', 'Un(a) amigo(a) me contó',
   'Televisión', 'TelevisiÃ±', 'Un(a) amigo(a) me contÃ§',
   'Mensaje de texto', 'Otra'], dtype=object)
```

A esto:

```
In [ ]: array(['Convocatoria', 'Televisión', 'Internet', 'Amigo',
   'Redes Sociales', nan, 'Prensa', 'Radio', 'Otra',
   'Mensaje de texto'], dtype=object)
```

Toda la información corregida la realice usando el comando replace con el siguiente formato:

```
In [ ]: dataframe['columnatemporal'] = dataframe.NombreColumna.replace('Ruido', 'DatoCoherente')
```

Como se podrá observar en los arreglos iniciales y como terminaron eliminando ruido, la cantidad de veces que ejecuté la función replace fueron muchas y no las alcancé a documentar, pero pueden darse una idea de como y cuantas veces lo utilice. Al final, después de asegurarse que los datos estaban correctos, reemplacé la columna original con los valores de la columna temporal.

También hice un segundo backup de los datos.

```
In [ ]: cine.to_csv('datosLimpiosCine2.csv', sep='\\', index=False)
```

1.3 Conteo y Promedio

Ahora la información que tenemos limpia es: * Año * País * Categoría * fCómo se enteró?

Primero decidí instalar la librería "tabulate" de Python para obtener resultados ordenados en tablas agradables a la vista. Después corrí el siguiente script para obtener la información de la participación de países divididos por año.

```
In [ ]: listaPaises = []
for country in paises:
    listaPaises.append(country,cine[(cine['País']==country) & (cine['Año']==2015)].count())
print(tabulate(listaPaises, tablefmt="github"))
```

Utilicé el count()['Año'] debido a que es el único campo que no tiene NaN y me da un conteo exacto de los registros, pareciera que el contador no funcionó con los NaN por que me arrojó ceros y la cantidad de registros bajó a 2732. Obtuve estos resultados:

```
In [ ]: | País           | 2015 | 2016 | 2017 | 2018 | Total |
|-----|-----|-----|-----|-----|-----|
| Colombia      | 477  | 846  | 641  | 438  | 2402 |
| Irán          | 1    | 0    | 0    | 0    | 1    |
| España         | 26   | 17   | 1    | 6    | 50   |
| Venezuela     | 5    | 1    | 1    | 0    | 7    |
| México         | 8    | 3    | 2    | 171  | 184  |
```

Ecuador	2	3	0	1	6
Francia	2	1	0	0	3
Argentina	1	4	3	1	9
Perú	1	0	0	0	1
Estados Unidos	3	4	3	3	13
nan	0	2	0	1	3
Canadá	1	1	0	0	2
Honduras	1	0	0	0	1
Brasil	5	1	0	0	6
Cuba	1	0	0	0	1
Uruguay	2	1	0	0	3
Alemania	1	1	0	0	2
Internacional	0	5	11	0	16
Marruecos	0	1	0	0	1
República Dominicana	0	1	0	0	1
Bélgica	0	1	0	1	2
Afganistán	0	0	19	0	19
Comoras	0	0	1	1	2
-----	-----	-----	-----	-----	-----
Suma	537	893	682	623	2735

Usé también un script para obtener los registros por Categoría y participación por Año. El código fue el siguiente:

```
In [ ]: listaCategoria = []
>>> for contry in categorias:
...     listaCategoria.append([contry,cine[(cine['Categoría']==contry) & (cine['Año']==2017)]])
...
>>> print(tabulate(listaCategoria, tablefmt="github"))
```

Me di cuenta que efectivamente el NaN no es contado por mi código, me doy cuenta de ésto, debido a que en 2017 tengo varios elementos que yo mismo convertí a NaN porque al momento de capturar la Categoría en México cometieron el error de escribir el Género del video, comprobé que en 2016 me hacen falta 2 video en NaN la tabla anterior y en ésta me hacen falta 165 registros en NaN. Obtuve los siguientes resultados:

Categoría	2015	2016	2017	2018	Total
Aficionado	313	522	377	241	1453
Profesional	147	151	85	53	436
Infantil	77	42	0	0	119
nan	0	0	0	165	0
Juvenil	0	123	132	83	338
SmarTIC	0	55	88	81	224
-----	-----	-----	-----	-----	-----
Suma	537	893	682	623	2735

1.4 Conclusiones

En esta práctica me topé con que se pudo mejorar la limpieza de la práctica pasada. Aprendí a identificar escrituras únicas con la función unique() y a reemplazar string dentro de los DataFrame con la función replace(), reemplacé los nombres de las columnas de todos los DataFrames antes de unirlos y además me di cuenta que era innecesario el DataFrames df2017[4] debido a que solo fue un evento de "Reto al Guion" no había referencia de videos ni de categorías, nada que lo relaciona a las otras tablas de registros.

A los datos les realicé 2 backups, uno después de la limpieza y el otro después de reemplazar el ruido en los campos.

Busqué la forma de sacar promedios y realizar correlaciones, pero en la práctica la Dra. Elisa dijo: "Todas las respuestas abiertas de texto quedan fuera del alcance de esta práctica." Y en mis datos todo, salvo el año, son categorías de texto. Así que doy por concluida mi práctica.

--04 de junio 2019-- Luis Angel Gutierrez Rodríguez 1484412

Práctica 4: Visualización de información

Complicaciones

Práctica no
realizada a
tiempo.

Reporte de práctica 4: Visualización de información con Plotly

Plotly, también conocida por su URL, [Plot.ly \(https://plot.ly/\)](https://plot.ly/), es una empresa de computación técnica con sede en Montreal, Quebec, que desarrolla herramientas de visualización y análisis de datos en línea. Plotly proporciona herramientas de gráficos, análisis y estadísticas en línea para individuos y colaboración, así como bibliotecas de gráficos científicos para Python, R, MATLAB, Perl, Julia, Arduino y REST.

Objetivos

- Incluir al menos tres diferentes tipos de gráficas usando Plotly e intentar concluir algo sobre los datos.

Los datos

Debido a que la base de datos con la que contamos se encuentra fragmentada en años, se utilizarán los datos que cuente con la información más completa.

Importando datos

```
In [1]: import pandas as pd  
df2018 = pd.read_excel('https://raw.githubusercontent.com/SamatarouKami/CIENCIA  
_DE_DATOS/master/2018.xlsx', index_col=None, header=0, sheet_name=0)  
df2018 = df2018[['Categoria', 'Edad', 'Pais', 'Titulo', 'Genero', 'Duracion', 'Mar  
ca', 'Referencia', 'Dias', 'Marcas', 'Personas']]  
  
print(len(df2018))
```

452

Gráficas

Una vez cargada la información, se procede a graficar y verificar si se puede concluir algo relevante.

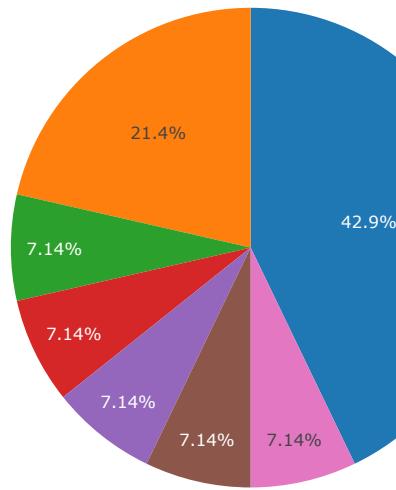
Extranjeros

Se busca saber cual es la proporción de extranjeros que participan en el festival de acuerdo a su país de procedencia. Se retiran a los Colombianos de la lista de participantes para obtener solo extranjeros. Se realiza una categorización por país, luego se hace un conteo y graficamos.

```
In [2]: import plotly.plotly as py
import plotly.figure_factory as ff
import statsmodels.api as sm
import numpy as np
import plotly.graph_objs as go
from numpy import isnan

cine = df2018
paises = cine['Pais'].unique()
in1 = np.argwhere(paises=='Colombia')
paises = np.delete(paises,in1)
listapaises = []
for country in paises:
    listapaises.append([country,cine[cine['Pais']==country].count()['Pais']])
dfpais = pd.DataFrame(data=listapaises)
dfpais.columns = ['Pais', 'Total']
trace = go.Pie(labels=dfpais.Pais, values=dfpais.Total)
py.iplot([trace], filename='P4_1')
```

Out[2]:



[EDIT CHART](#)

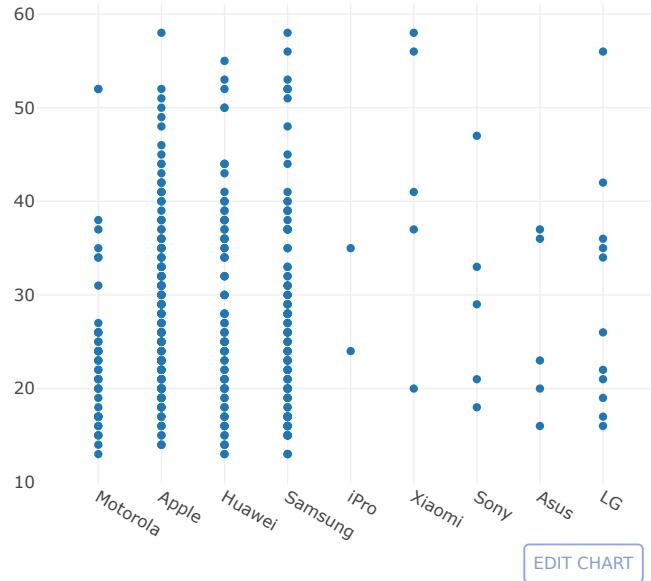
Se puede concluir que en el año 2018, la mayor parte de extranjeros procedía de España.

Marca de smartphones

Ahora se comprueba los rangos de edades donde los participantes utilizan cierta marca de celular.

```
In [3]: datos = [go.Scatter(x = df2018.Marca, y = df2018.Edad, mode = 'markers')]
py.iplot(datos, filename='P4_2')
```

Out[3]:



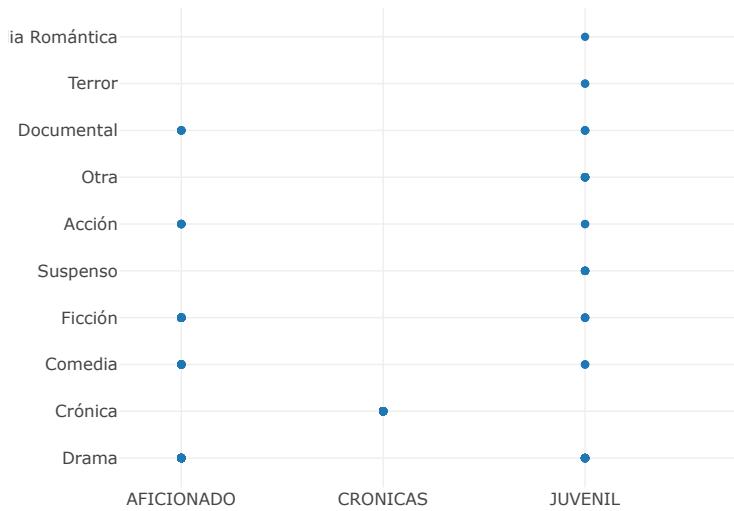
Se puede concluir que la mayoría de los concursantes utiliza smartphones de la marca Apple, Samsung y Huawei.

Género por categoría

Ahora se verifica si hay alguna relación entre los géneros seleccionados y la categoría de participación.

```
In [4]: datos = [go.Scatter(x = df2018.Categoría, y = df2018.Género, mode = 'markers')]
py.iplot(datos, filename='P4_3')
```

Out[4]:



[EDIT CHART](#)

Se puede concluir en los géneros de Ficción, Comedia, Drama, Acción y Documental existen participantes de todas las categorías, excepto Crónicas que parece ser un género únicamente relacionado con la categoría.

Conclusión

En esta práctica pude aprender a utilizar la plataforma de Plotly. Se pudo determinar tres cualidades sobre nuestros datos simplemente graficando la información.

--04 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 5: Pruebas estadísticas

Complicaciones

```
CIENCIA_DE_DATOS (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master)
/ P5.ipynb (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master/P5.ipynb)
```

Reporte de Práctica 5: Pruebas estadísticas

En ésta práctica se realizarán 3 análisis estadísticos a los datos del cine. Primero realizaremos histogramas de la información para observar su comportamiento si es normalizado o no.

Como se harán gráficas de los datos, se instaló la librería Plotly de python3 para hacer esta tarea, debido a que es una librería de pago, se creó una cuenta gratuita en la cual estamos limitados a 25 gráficas.

Se usaron las siguientes instrucciones para configurar el Plotly.

```
In [16]: !pip3 install plotly

Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (3.6.1)
Requirement already satisfied: nbformat>=4.2 in /usr/local/lib/python3.6/dist-packages (from plotly) (4.4.0)
Requirement already satisfied: pytz in /usr/lib/python3/dist-packages (from plotly) (2018.3)
Requirement already satisfied: six in /home/samataroukami/.local/lib/python3.6/site-packages (from plotly) (1.11.0)
Requirement already satisfied: decorator>=4.0.6 in /usr/local/lib/python3.6/dist-packages (from plotly) (4.3.0)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (from plotly) (1.3.3)
Requirement already satisfied: requests in /home/samataroukami/.local/lib/python3.6/site-packages (from plotly) (2.20.1)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-packages (from nbformat>=4.2->plotly) (0.2.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/python3.6/dist-packages (from nbformat>=4.2->plotly) (2.6.0)
Requirement already satisfied: traitlets>=4.1 in /usr/local/lib/python3.6/dist-packages (from nbformat>=4.2->plotly) (4.3.2)
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.6/dist-packages (from nbformat>=4.2->plotly) (4.4.0)
Requirement already satisfied: idna<2.8,>=2.5 in /home/samataroukami/.local/lib/python3.6/site-packages (from requests->plotly) (2.7)
Requirement already satisfied: certifi>=2017.4.17 in /home/samataroukami/.local/lib/python3.6/site-packages (from requests->plotly) (2018.10.15)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /home/samataroukami/.local/lib/python3.6/site-packages (from requests->plotly) (3.0.4)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /home/samataroukami/.local/lib/python3.6/site-packages (from requests->plotly) (1.24.1)
```

```
In [21]: !pip install --user --upgrade pip

Cache entry deserialization failed, entry ignored
Collecting pip
  Cache entry deserialization failed, entry ignored
    Downloading https://files.pythonhosted.org/packages/d8/f3/413bab4ff08e1fc4828dfc59996d721917df8e
8583ea85385d51125dcfff/pip-19.0.3-py2.py3-none-any.whl (1.4MB)
      100% |████████████████████████████████| 1.4MB 581kB/s ta 0:00:01
Installing collected packages: pip
  Found existing installation: pip 9.0.3
    Uninstalling pip-9.0.3:
      Successfully uninstalled pip-9.0.3
Successfully installed pip-19.0.3
```

```
In [41]: !python -m pip install plotly --user

Collecting plotly
  Using cached https://files.pythonhosted.org/packages/fd/db/003b5cfbc710f4d4982440451185b952269e4
  080a57ae7e760a2ceb8ce0c/plotly-3.6.1-py2.py3-none-any.whl
Requirement already satisfied: six in /home/samataroukami/.local/lib/python2.7/site-packages (from
plotly) (1.11.0)
Requirement already satisfied: decorator>=4.0.6 in /home/samataroukami/.local/lib/python2.7/site-p
ackages (from plotly) (4.3.2)
Requirement already satisfied: requests in /home/samataroukami/.local/lib/python2.7/site-packages
(from plotly) (2.20.1)
Requirement already satisfied: nbformat>=4.2 in /home/samataroukami/.local/lib/python2.7/site-p
ackages (from plotly) (4.4.0)
Collecting retrying>=1.3.3 (from plotly)
  Using cached https://files.pythonhosted.org/packages/44/ef/beae4b4ef80902f22e3af073397f079c96969
  c69b2c7d52a57ea9ae61c9d/retrying-1.3.3.tar.gz
Requirement already satisfied: pytz in /home/samataroukami/.local/lib/python2.7/site-packages (fro
m plotly) (2018.9)
Requirement already satisfied: idna<2.8,>=2.5 in /home/samataroukami/.local/lib/python2.7/site-pac
kages (from requests->plotly) (2.7)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /home/samataroukami/.local/lib/python2.7/s
ite-packages (from requests->plotly) (1.24.1)
Requirement already satisfied: certifi>=2017.4.17 in /home/samataroukami/.local/lib/python2.7/site
-packages (from requests->plotly) (2018.10.15)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /home/samataroukami/.local/lib/python2.7/s
ite-packages (from requests->plotly) (3.0.4)
Requirement already satisfied: traitlets>=4.1 in /home/samataroukami/.local/lib/python2.7/site-pac
kages (from nbformat>4.2->plotly) (4.3.2)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /home/samataroukami/.local/lib/python2.7
/site-packages (from nbformat>4.2->plotly) (3.0.0)
Requirement already satisfied: ipython-genutils in /home/samataroukami/.local/lib/python2.7/site-p
ackages (from nbformat>4.2->plotly) (0.2.0)
Requirement already satisfied: jupyter-core in /home/samataroukami/.local/lib/python2.7/site-packa
ges (from nbformat>4.2->plotly) (4.4.0)
Requirement already satisfied: enum34; python_version == "2.7" in /home/samataroukami/.local/lib/p
ython2.7/site-packages (from traitlets>=4.1->nbformat>=4.2->plotly) (1.1.6)
Requirement already satisfied: pyrsistent>=0.14.0 in /home/samataroukami/.local/lib/python2.7/site
-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2->plotly) (0.14.11)
Requirement already satisfied: functools32; python_version < "3" in /home/samataroukami/.local/lib
/python2.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2->plotly) (3.2.3.post2)
Requirement already satisfied: attrs>=17.4.0 in /home/samataroukami/.local/lib/python2.7/site-p
ackages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2->plotly) (18.2.0)
Requirement already satisfied: setuptools in /home/samataroukami/.local/lib/python2.7/site-package
s (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2->plotly) (40.6.2)
Building wheels for collected packages: retrying
  Running setup.py bdist_wheel for retrying ... done
  Stored in directory: /home/samataroukami/.cache/pip/wheels/d7/a9/33/acc7b709e2a35caa7d4cae442f6f
  e6fbf2c43f80823d46460c
Successfully built retrying
Installing collected packages: retrying, plotly
Successfully installed plotly-3.6.1 retrying-1.3.3
You are using pip version 18.1, however version 19.0.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

```
In [42]: !python -m pip install plotly --user --upgrade pip

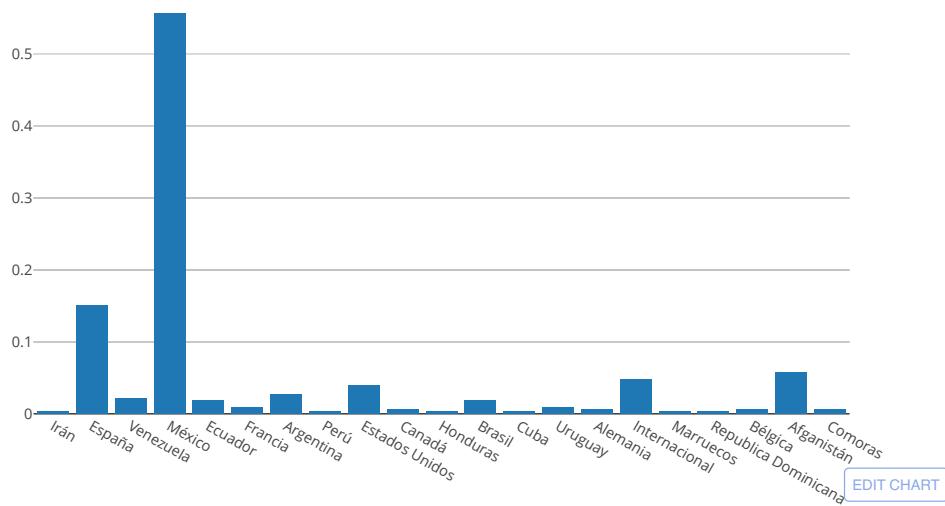
Requirement already up-to-date: plotly in /home/samataroukami/.local/lib/python2.7/site-packages (3.6.1)
Collecting pip
  Using cached https://files.pythonhosted.org/packages/d8/f3/413bab4ff08e1fc4828dfc59996d721917df8e8583ea85385d51125dceff/pip-19.0.3-py2.py3-none-any.whl
Requirement already satisfied, skipping upgrade: six in /home/samataroukami/.local/lib/python2.7/site-packages (from plotly) (1.11.0)
Requirement already satisfied, skipping upgrade: decorator>=4.0.6 in /home/samataroukami/.local/lib/python2.7/site-packages (from plotly) (4.3.2)
Requirement already satisfied, skipping upgrade: requests in /home/samataroukami/.local/lib/python2.7/site-packages (from plotly) (2.20.1)
Requirement already satisfied, skipping upgrade: nbformat>=4.2 in /home/samataroukami/.local/lib/python2.7/site-packages (from plotly) (4.4.0)
Requirement already satisfied, skipping upgrade: retrying>=1.3.3 in /home/samataroukami/.local/lib/python2.7/site-packages (from plotly) (1.3.3)
Requirement already satisfied, skipping upgrade: pytz in /home/samataroukami/.local/lib/python2.7/site-packages (from plotly) (2018.9)
Requirement already satisfied, skipping upgrade: idna<2.8,>=2.5 in /home/samataroukami/.local/lib/python2.7/site-packages (from requests->plotly) (2.7)
Requirement already satisfied, skipping upgrade: urllib3<1.25,>=1.21.1 in /home/samataroukami/.local/lib/python2.7/site-packages (from requests->plotly) (1.24.1)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in /home/samataroukami/.local/lib/python2.7/site-packages (from requests->plotly) (2018.10.15)
Requirement already satisfied, skipping upgrade: chardet<3.1.0,>=3.0.2 in /home/samataroukami/.local/lib/python2.7/site-packages (from requests->plotly) (3.0.4)
Requirement already satisfied, skipping upgrade: traitlets>=4.1 in /home/samataroukami/.local/lib/python2.7/site-packages (from nbformat>=4.2->plotly) (4.3.2)
Requirement already satisfied, skipping upgrade: jsonschema!=2.5.0,>=2.4 in /home/samataroukami/.local/lib/python2.7/site-packages (from nbformat>=4.2->plotly) (3.0.0)
Requirement already satisfied, skipping upgrade: ipython-genutils in /home/samataroukami/.local/lib/python2.7/site-packages (from nbformat>=4.2->plotly) (0.2.0)
Requirement already satisfied, skipping upgrade: jupyter-core in /home/samataroukami/.local/lib/python2.7/site-packages (from nbformat>=4.2->plotly) (4.4.0)
Requirement already satisfied, skipping upgrade: enum34; python_version == "2.7" in /home/samataroukami/.local/lib/python2.7/site-packages (from traitlets>=4.1->nbformat>=4.2->plotly) (1.1.6)
Requirement already satisfied, skipping upgrade: pyrsistent>=0.14.0 in /home/samataroukami/.local/lib/python2.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2->plotly) (0.14.11)
Requirement already satisfied, skipping upgrade: functools32; python_version < "3" in /home/samataroukami/.local/lib/python2.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2->plotly) (3.2.3.post2)
Requirement already satisfied, skipping upgrade: attrs>=17.4.0 in /home/samataroukami/.local/lib/python2.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2->plotly) (18.2.0)
Requirement already satisfied, skipping upgrade: setuptools in /home/samataroukami/.local/lib/python2.7/site-packages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2->plotly) (40.6.2)
Installing collected packages: pip
  Found existing installation: pip 18.1
    Uninstalling pip-18.1:
      Successfully uninstalled pip-18.1
Successfully installed pip-19.0.3
```

Despues de esto importamos el .csv donde guardamos la informacion de la práctica 3 y tuve muchos problemas para instalar plotly en jupyter notebook.

```
In [43]: import plotly
plotly.tools.set_credentials_file(username='SamatarouKami', api_key='X0OGa3VP5G6DWcfBivYa')
```

```
In [2]: import plotly.plotly as py
import plotly.graph_objs as go
import pandas as pd
cine = pd.read_csv('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/datosLimpiosCine2.csv', sep='\\')
ext = cine.Pais[cine.Pais != 'Colombia']
print((py.iplot({'data': [go.Histogram(x = ext, histnorm='probability')]}), filename='extranjeros')).embed_code)
```

<iframe id="igraph" scrolling="no" style="border:none;" seamless="seamless" src="https://plot.ly/~SamatarouKami/2.embed" height="525px" width="100%">></iframe>



Se supone que aquí arriba hay un gráfico. Así generamos el histograma de la cantidad de países extranjeros que participaron.

```
In [5]: !python -m pip install statsmodels --user

DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7.
Collecting statsmodels
  Downloading https://files.pythonhosted.org/packages/ed/56/ald32debdaba5cc7986aeeeb79e33d74ae3394eabfd008b0113368b91981/statsmodels-0.9.0-cp27-cp27mu-manylinux1_x86_64.whl (7.4MB)
    100% |████████████████████████████████| 7.4MB 1.1MB/s ta 0:00:01
Requirement already satisfied: pandas in /home/samataroukami/.local/lib/python2.7/site-packages (from statsmodels) (0.24.1)
Collecting patsy (from statsmodels)
  Using cached https://files.pythonhosted.org/packages/ea/0c/5f61f1a3d4385d6bf83b83ea495068857ff8dfb89e74824c6e9eb63286d8/patsy-0.5.1-py2.py3-none-any.whl
Requirement already satisfied: numpy>=1.12.0 in /home/samataroukami/.local/lib/python2.7/site-packages (from pandas->statsmodels) (1.16.1)
Requirement already satisfied: pytz>=2011k in /home/samataroukami/.local/lib/python2.7/site-packages (from pandas->statsmodels) (2018.9)
Requirement already satisfied: python-dateutil>=2.5.0 in /home/samataroukami/.local/lib/python2.7/site-packages (from pandas->statsmodels) (2.8.0)
Requirement already satisfied: six in /home/samataroukami/.local/lib/python2.7/site-packages (from patsy->statsmodels) (1.11.0)
Installing collected packages: patsy, statsmodels
Successfully installed patsy-0.5.1 statsmodels-0.9.0
```

```

In [7]: import pandas as pd
from statsmodels.graphics.gofplots import qqplot
import matplotlib as plt

cine = pd.read_csv('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/datosLimpiosCine2.csv', sep='\\')
ext = cine.Pais[cine.Pais != 'Colombia']

py.iplot({'data': [go.Histogram(x = ext, histnorm='probability')]}), filename='extranjeros')

f = qqplot(ext, line='s')
plt.savefig(f, "qq_cfep.png")
-----
```

```

TypeError                                 Traceback (most recent call last)
<ipython-input-7-3e0b50d9d610> in <module>()
      9
     10
--> 11 f = qqplot(ext, line='s')
     12 plt.savefig(f, "qq_cfep.png")

/home/samataroukami/.local/lib/python2.7/site-packages/statsmodels/graphics/gofplots.py in qqplot
(data, dist, distargs, a, loc, scale, fit, line, ax, **plotkwargs)
  504     """
  505     probplot = ProbPlot(data, dist=dist, distargs=distargs,
--> 506                         fit=fit, a=a, loc=loc, scale=scale)
  507     fig = probplot.qqplot(ax=ax, line=line, **plotkwargs)
  508     return fig

/home/samataroukami/.local/lib/python2.7/site-packages/statsmodels/graphics/gofplots.py in __init__
(self, data, dist, fit, distargs, a, loc, scale)
  135         dist = getattr(stats, dist)
  136
--> 137         self.fit_params = dist.fit(data)
  138         if fit:
  139             self.loc = self.fit_params[-2]

/home/samataroukami/.local/lib/python2.7/site-packages/scipy/stats/_continuous_distns.py in fit(s
elf, data, **kwds)
  266
  267     if floc is None:
--> 268         loc = data.mean()
  269     else:
  270         loc = floc

/home/samataroukami/.local/lib/python2.7/site-packages/numpy/core/_methods.py in _mean(a, axis, d
type, out, keepdims)
    73         is_float16_result = True
    74
--> 75     ret = umr_sum(arr, axis, dtype, out, keepdims)
    76     if isinstance(ret, mu.ndarray):
    77         ret = um.true_divide(
```

TypeError: cannot concatenate 'str' and 'float' objects

Conclusiones

No pude hacer que se vieran en la práctica los gráficos, por lo menos no se ven en mi computadora, espero que mañana se vean, los análisis estadísticos requieren que de alguna forma yo convierta mis datos a números ya que manejo muchas cadenas de caracteres y eso no me sirve para hacer análisis estadísticos, tal vez el objetivo de esta práctica era encontrar o buscar la manera de hacer eso, pero no encontré forma de hacerlo, solo puedo hacer conteos de la información. Tengo que categorizar.

--04 de Marzo 2019-- Luis Angel Gutierrez [1484412 \(tel:1484412\)](#)

Reporte de práctica 5: Pruebas estadísticas

Objetivo

- Realizar tres análisis estadísticos a los datos del cine. Primero realizaremos histogramas de la información para observar su comportamiento si es normalizado o no.

Pruebas estadísticas

Existen dos grandes grupos de pruebas de significación estadística, las paramétricas y las no paramétricas

Pruebas paramétricas

Las pruebas paramétricas son para datos numéricos (escalas de intervalos o razones) y, por lo general, están basadas en las propiedades de la distribución normal o gaussiana, para la variable dependiente. Los requisitos para poder usar estas pruebas son:

- Las observaciones deben ser independientes entre sí.
- Las poblaciones deben hacerse en poblaciones distribuidas normalmente.
- Estas poblaciones deben tener la misma varianza.
- Las variables deben haberse medido en una escala de intervalo de manera que sea posible utilizar las operaciones aritméticas.

Pruebas no paramétricas

Las pruebas no paramétricas son utilizadas con variables nominales y ordinales, no asumen un tipo particular de distribución, se aceptan distribuciones no normales, la exigencia en cuanto al tamaño de la muestra es menor que en el caso de las paramétricas.

Las pruebas no paramétricas son necesarias cuando:

- Los tamaños de las muestras son tan pequeñas como N=6
- La investigación aporta resultados que solo se puedan referir a un comportamiento de los sujetos en mayor o menor grado de ciertas características, pero sin especificar cantidad.

Lectura de Datos

Como se harán gráficas de los datos, se instaló la librería Plotly de python3 para hacer esta tarea, debido a que es una librería de pago, se creó una cuenta gratuita en la cual estamos limitados a veinticinco gráficas.

```
In [1]: import plotly.plotly as py
import plotly.graph_objs as go
import pandas as pd
import pandas as pd
cine = pd.read_excel('2018.xlsx', index_col=None, header=0, sheet_name=0)
cine = cine[['Categoria', 'Edad', 'Pais', 'Titulo', 'Genero', 'Duracion', 'Marca',
'Referencia', 'Dias', 'Marcas', 'Personas']]
```

Sincronizar con Plotly

Agregaremos los datos de usuario y la llave api para poder graficar con Plotly.

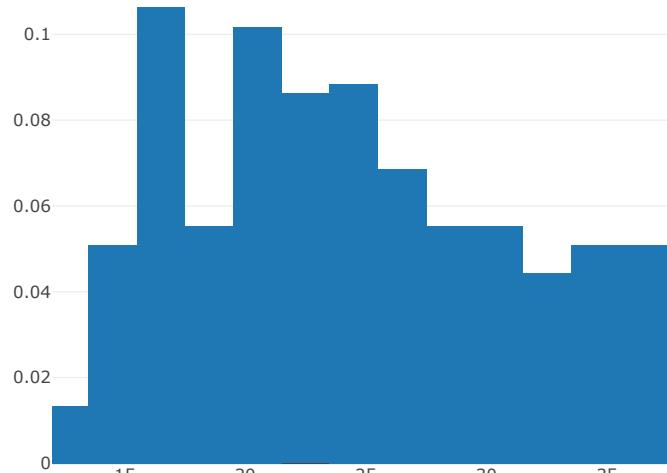
```
In [2]: import plotly  
plotly.tools.set_credentials_file(username='SamatarouKami', api_key='X00Ga3VP5G  
6DWcfBivYa')
```

Características de los datos

Se quiere comprobar si los datos de Edad y Personas que participan en un proyecto están normalmente distribuidos. Se determina que clase de pruebas se deben realizar sobre estos. Primero que nada haremos pruebas de normalidad, graficaremos el histograma de los datos.

```
In [3]: layout = {'xaxis': {'range': [12, 59]}}  
py.iplot({'data': [go.Histogram(x = cine.Edad, histnorm='probability')], 'layout': layout}, filename='P5_1')
```

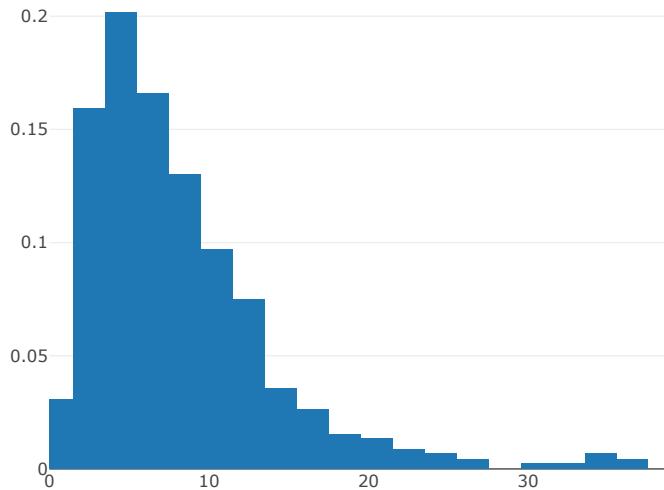
Out[3]:



[EDIT CHART](#)

```
In [4]: layout = {'xaxis': {'range': [0, 71]}}
py.iplot({'data': [go.Histogram(x = cine.Personas, histnorm='probability')]}, 'layout', filename='P5_2')
```

Out[4]:

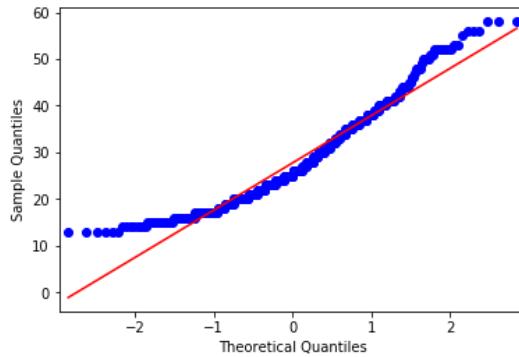


[EDIT CHART](#)

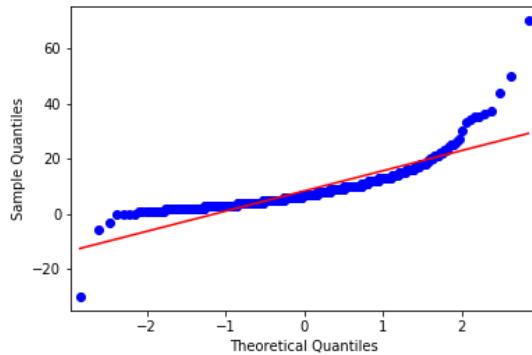
Como se observa en el histograma, la edad parece estar normalmente distribuida y las personas por film se encuentran en sesgo, se puede identificar una asimetría a la derecha.

Ahora se prueba con QQplot si los datos son normalmente distribuidos.

```
In [12]: import pandas as pd
from statsmodels.graphics.gofplots import qqplot
import matplotlib.pyplot as plt
g = qqplot(cine.Edad, line='s')
```



```
In [6]: y = qqplot(cine.Personas, line='s')
```



Se tiene que los histogramas no son suficientes para determinar la normalidad de las Edades. En el grafico de Personas sabemos que no está normalizado. Los resultados de QQplot son curiosos porque pareciera que ambas distribuciones están normalizadas.

Para confirmar la naturaleza de los datos aplicaremos la prueba de normalidad Shapiro-Wilks.

```
In [7]: from numpy import concatenate, isnan

datos = {'Edad': cine.Edad , \
          'Participantes': cine.Personas}
from scipy.stats import shapiro
for alpha in [0.05, 0.01]:
    for data in datos:
        crudos = datos[data]
        s, p = shapiro(crudos[~isnan(crudos)]) # es MUY importante quitar los N
aN
        print('{:s} {:.2f} {:.3f}'.format(data, s, p))
        if p > alpha:
            print('aceptablemente normal con nivel de significancia', alpha)
        else:
            print('no parece ser normal con nivel de significancia', alpha)

Edad 0.94 0.000
('no parece ser normal con nivel de significancia', 0.05)
Participantes 0.76 0.000
('no parece ser normal con nivel de significancia', 0.05)
Edad 0.94 0.000
('no parece ser normal con nivel de significancia', 0.01)
Participantes 0.76 0.000
('no parece ser normal con nivel de significancia', 0.01)
```

Naturaleza de los datos

Con el histograma y la prueba Shapiro-Wilks, determinamos que la naturaleza de los datos es no paramétrica. Por esto se aplican pruebas de este tipo a los datos.

Pruebas estadísticas no paramétricas

Las pruebas que se realizan sobre los datos son:

Mann-Whitney
Wilcoxon
Kruskal-Wallis

Se propone la hipótesis de que los jóvenes forman grupos para poder grabar los filmes y se registran como grupo. Se realizan las tres pruebas sobre esta misma hipótesis. Se preparan las funciones para realizar las pruebas.

```
In [8]: from numpy import isnan
from scipy.stats import mannwhitneyu
from scipy.stats import wilcoxon
from scipy.stats import kruskal

def MannWhitney(c1, c2):
    d1 = c1[~isnan(c1)]
    d2 = c2[~isnan(c2)]
    n1 = len(d1)
    n2 = len(d2)
    if min(n1, n2) < 20:
        print('hay muy pocos datos como para obtener un resultado confiable')
        return
    for alpha in [0.05, 0.01]:
        s, p = mannwhitneyu(d1, d2)
        print('{:d} {:d} {:.2f} {:.3f}'.format(n1, n2, s, p))
        if p > alpha:
            print('son igualmente distribuidos con nivel de significancia', alpha)
        else:
            print('se ven differentemente distribuidos con nivel de significancia', alpha)

def Wilcoxon(c1, c2):
    d1 = c1[~isnan(c1)]
    d2 = c2[~isnan(c2)]
    n1 = len(d1)
    n2 = len(d2)
    if min(n1, n2) < 20:
        print('hay muy pocos datos como para obtener un resultado confiable')
        return
    for alpha in [0.05, 0.01]:
        s, p = wilcoxon(d1, d2)
        print('{:d} {:d} {:.2f} {:.3f}'.format(n1, n2, s, p))
        if p > alpha:
            print('son igualmente distribuidos con nivel de significancia', alpha)
        else:
            print('se ven differentemente distribuidos con nivel de significancia', alpha)

def KruskalWallis(c1, c2):
    d1 = c1[~isnan(c1)]
    d2 = c2[~isnan(c2)]
    n1 = len(d1)
    n2 = len(d2)
    if min(n1, n2) < 20:
        print('hay muy pocos datos como para obtener un resultado confiable')
        return
    for alpha in [0.05, 0.01]:
        s, p = kruskal(d1, d2)
        print('{:d} {:d} {:.2f} {:.3f}'.format(n1, n2, s, p))
        if p > alpha:
            print('son igualmente distribuidos con nivel de significancia', alpha)
        else:
            print('se ven differentemente distribuidos con nivel de significancia', alpha)
```

```
In [9]: MannWhitney(cine.Edad, cine.Personas)
```

```
452 452 7679.00 0.000
('se ven differentemente distribuidos con nivel de significancia', 0.05)
452 452 7679.00 0.000
('se ven differentemente distribuidos con nivel de significancia', 0.01)
```

```
In [10]: Wilcoxon(cine.Edad, cine.Personas)
```

```
452 452 1949.50 0.000
('se ven differentemente distribuidos con nivel de significancia', 0.05)
452 452 1949.50 0.000
('se ven differentemente distribuidos con nivel de significancia', 0.01)
```

```
In [11]: KruskalWallis(cine.Edad, cine.Personas)
```

```
452 452 579.81 0.000
('se ven differentemente distribuidos con nivel de significancia', 0.05)
452 452 579.81 0.000
('se ven differentemente distribuidos con nivel de significancia', 0.01)
```

Conclusiones

A pesar de que los datos de las edades parecen distribuidos normalmente, al realizar pruebas de normalidad se pudo comprobar que los datos no están normalizados. Por este motivo se realizaron pruebas no paramétricas para comprobar la relación de los datos. Se buscó demostrar la hipótesis de que los jóvenes se agrupan para participar. Quedando demostrado por tres diferentes pruebas que no hay relación.

--04 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 6: Modelos lineales

Complicaciones

CIENCIA_DE_DATOS (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master)
/ P6.ipynb (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master/P6.ipynb)

Reporte de práctica 6: Modelos lineales con scipy.stats

En esta práctica tratamos de buscar al menos dos modelos lineales en los datos del cine. Iniciar los datos

Cargar datos de csv

Queremos probar si la categoría eligieron para su clip, está relacionado de alguna forma con el país de procedencia y por año de participación.

Primero importamos el archivo que esta en github que tiene información limpia de prácticas pasadas.

```
In [105]: import pandas as pd
from numpy import isnan
from statsmodels.graphics.gofplots import qqplot
import matplotlib.pyplot as plt
from scipy.stats import shapiro
from pandas.compat import u

cine = pd.read_csv('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/datosLimpiosCine2.csv', sep='\\')
print(len(cine))
cine = cine.dropna()
print(len(cine))
```

2735
2277

Para evitar errores de cálculos, usando la función dropna() retiramos todos los registros que tengan algún campo vacío y nos deshacemos de 458 registros que no tienen género o tipo de celular y nos quedamos con 2277 registros válidos.

Categorizaciones

Como tenemos muchas cadenas de texto en nuestros datos, es importante hacer categorizaciones ya que hacemos conteos de la información que tenemos disponible y así poder hacer cálculos estadísticos. Categorías por país

Categorías por país

Aplicamos una categorización por país, pero se removieron los datos que involucran a México y Colombia que son los países anfitriones del concurso y por ende son los que nos sesgan la información, entonces esta información será relevante a los países extranjeros que participan.

```
In [106]: paises = cine['País'].unique()

listapaises = []
for country in paises:
    listapaises.append([country,cine[(cine['País']==country) & (cine['Año']==2015)].count()['Año'],cine[(cine['País']==country) & (cine['Año']==2016)].count()['Año'],cine[(cine['País']==country) & (cine['Año']==2017)].count()['Año'],cine[(cine['País']==country) & (cine['Año']==2018)].count()['Año'],cine[cine['País']==country].count()['Año']])
#print(listapaises)
listapaises.pop(0)
listapaises.pop(1)

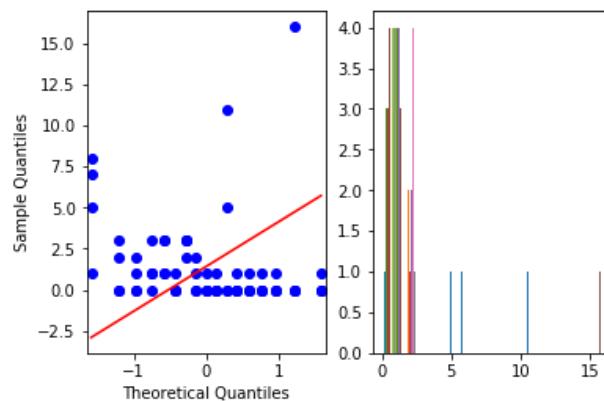
dfpais = pd.DataFrame(data=listapaises)
dfpais.columns = ['Pais', '2015', '2016', '2017', '2018', 'Total']
print(dfpais)
```

	Pais	2015	2016	2017	2018	Total
0	España	7	8	1	5	21
1	Ecuador	2	3	0	0	5
2	Francia	2	1	0	0	3
3	Venezuela	3	1	1	0	5
4	Argentina	1	3	3	1	8
5	Perú	1	0	0	0	1
6	Estados Unidos	3	3	3	2	11
7	Brasil	2	1	0	0	3
8	Cuba	1	0	0	0	1
9	Alemania	0	1	0	0	1
10	Internacional	0	5	11	0	16
11	Marruecos	0	1	0	0	1
12	Uruguay	0	1	0	0	1
13	República Dominicana	0	1	0	0	1
14	Canadá	0	1	0	0	1
15	Afganistán	0	0	16	0	16
16	Comoras	0	0	1	0	1

Al aplicar pruebas de normalidad a los datos, podemos observar que no está presente la normalidad.

```
In [107]: f, ax = plt.subplots(1, 2)
dfpais2=dfpais[['2015','2016','2017','2018']].copy()
qqplot(dfpais2, line='s', ax = ax[0])
ax[1] = plt.hist(dfpais2)
for a in [0.05, 0.01]:
    s, p = shapiro(dfpais2)
    print(s, p, a, "normal" if p > a else "no normal")
plt.show()
```

```
(0.5631282329559326, 6.444122037201072e-13, 0.05, 'no normal')
(0.5631282329559326, 6.444122037201072e-13, 0.01, 'no normal')
```



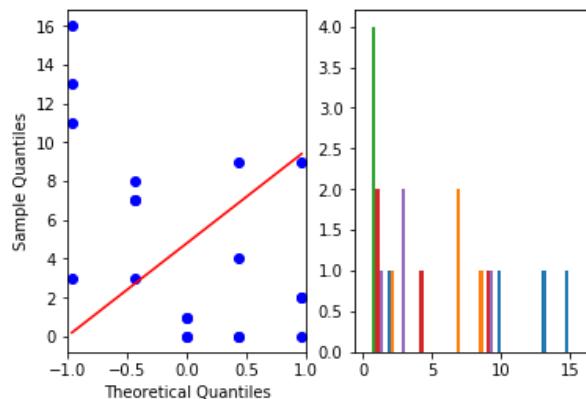
También categorizamos la Categoría del corto, y se clasificó por año.

```
In [108]: categorias = cine['Categoría'].unique()
listaCategoria = []
for cat in categorias:
    listaCategoria.append([cat,cine[(cine['Categoría']==cat) & (cine['Año']==2015) & (cine['País']!='Colombia') & (cine['País']!='México')].count()['Año'],cine[(cine['Categoría']==cat) & (cine['Año']==2016) & (cine['País']!='Colombia') & (cine['País']!='México')].count()['Año'],cine[(cine['Categoría']==cat) & (cine['Año']==2017) & (cine['País']!='Colombia') & (cine['País']!='México')].count()['Año'],cine[(cine['Categoría']==cat) & (cine['Año']==2018) & (cine['País']!='Colombia') & (cine['País']!='México')].count()['Año'],cine[(cine['Categoría']==cat) & (cine['País']!='Colombia') & (cine['País']!='México')].count()['Año']])
dfcat = pd.DataFrame(data=listaCategoria)
dfcat.columns = ['Categoria', '2015', '2016', '2017', '2018', 'Total']
print(dfcat)
```

	Categoría	2015	2016	2017	2018	Total
0	Aficionado	13	16	11	3	43
1	Profesional	8	7	7	3	25
2	Infantil	1	1	0	0	2
3	Juvenil	0	4	9	0	13
4	SmarTIC	0	2	9	2	13

```
In [109]: f, ax = plt.subplots(1, 2)
dfcat2=dfcat[['2015','2016','2017','2018']].copy()
qqplot(dfcat2, line='s', ax = ax[0])
ax[1] = plt.hist(dfcat2)
for a in [0.05, 0.01]:
    s, p = shapiro(dfcat2)
    print(s, p, a, "normal" if p > a else "no normal")
plt.show()
```

```
(0.8778810501098633, 0.01620960235595703, 0.05, 'no normal')
(0.8778810501098633, 0.01620960235595703, 0.01, 'normal')
```



Podemos observar que los datos de la categoría tampoco estar normales.

Modelo lineal

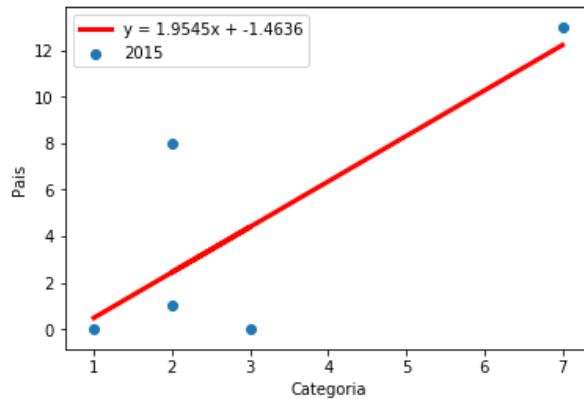
Empezamos con la modelación lineal, suponiendo que la categoría es una consecuencia del país de los creadores del clip. Se probó la relación lineal por año, y también la total.

```
In [110]: import matplotlib.pyplot as plt
from scipy import stats

y = dfcat['2015']
x = dfpais['2015']
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)

y = f(x) = 1.9545 x + -1.4636
('error', 0.8974251505426212)
('valor p', 0.11757517937281314)
pendiente no significativo
('R^2', 0.6125761993108931)
```

```
In [111]: plt.plot(x, (a * x + b), label = 'y = {:.4f}x + {:.4f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Categoria")
plt.ylabel("Pais")
plt.show()
```

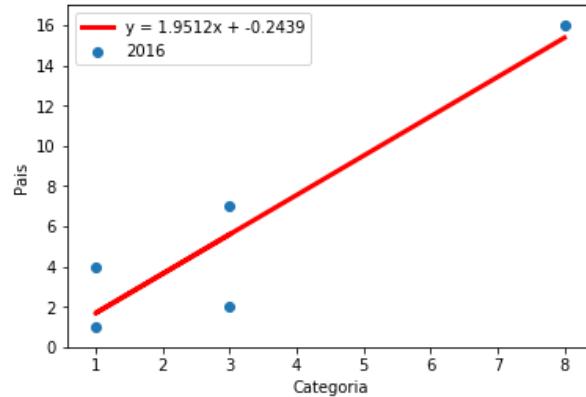


```
In [112]: import matplotlib.pyplot as plt
from scipy import stats

y = dfcat['2016']
x = dfpais['2016']
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)

y = f(x) = 1.9512 x + -0.2439
('error', 0.4633076470424925)
('valor p', 0.024454336793225805)
pendiente significativo
('R^2', 0.8553291012362179)
```

```
In [113]: plt.plot(x, (a * x + b), label = 'y = {:.4f}x + {:.4f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Categoria")
plt.ylabel("Pais")
plt.show()
```

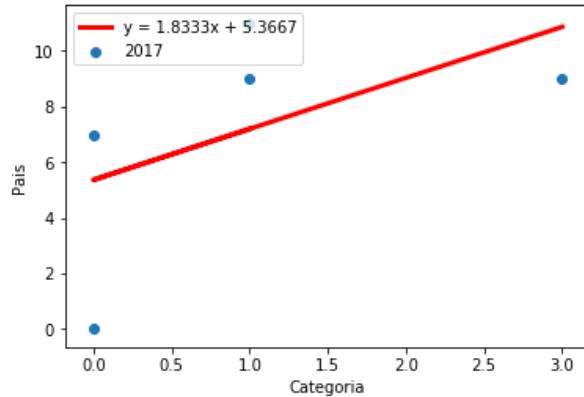


```
In [114]: import matplotlib.pyplot as plt
from scipy import stats

y = dfcat['2017']
x = dfpais['2017']
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)

y = f(x) = 1.8333 x + 5.3667
('error', 1.7099924193031015)
('valor p', 0.36223895879468526)
pendiente no significativo
('R^2', 0.277014652014652)
```

```
In [115]: plt.plot(x, (a * x + b), label = 'y = {:.4f}x + {:.4f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Categoria")
plt.ylabel("Pais")
plt.show()
```

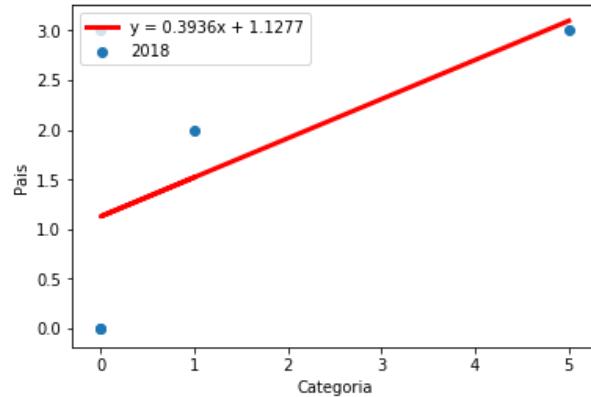


```
In [116]: import matplotlib.pyplot as plt
from scipy import stats

y = dfcat['2018']
x = dfpais['2018']
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)

y = f(x) = 0.3936 x + 1.1277
('error', 0.3338798899250046)
('valor p', 0.32341836282824227)
pendiente no significativo
('R^2', 0.316604995374653)
```

```
In [117]: plt.plot(x, (a * x + b), label = 'y = {:.4f}x + {:.4f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Categoria")
plt.ylabel("Pais")
plt.show()
```

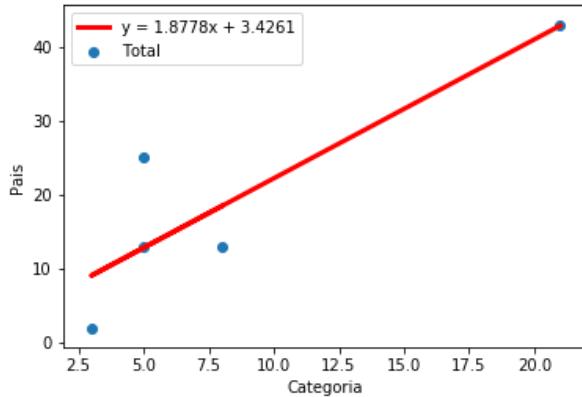


```
In [118]: import matplotlib.pyplot as plt
from scipy import stats

y = dfcat['Total']
x = dfpais['Total']
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)

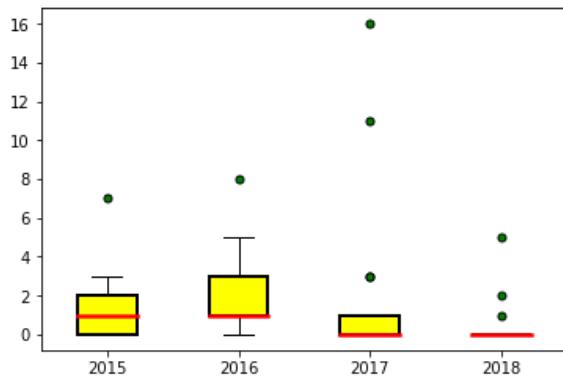
y = f(x) = 1.8778 x + 3.4261
('error', 0.5999372539983482)
('valor p', 0.052063036755892314)
pendiente no significativo
('R^2', 0.7655753541791265)
```

```
In [119]: plt.plot(x, (a * x + b), label = 'y = {:.4f}x + {:.4f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Categoria")
plt.ylabel("Pais")
plt.show()
```



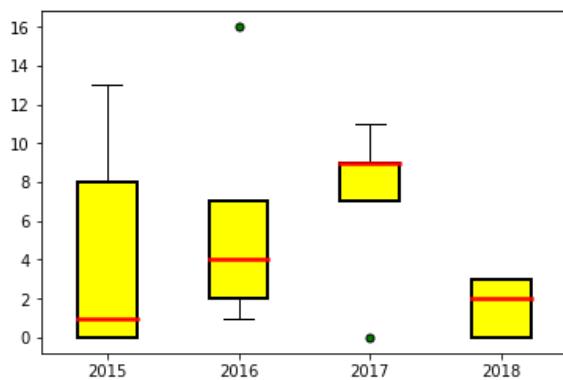
La pendiente fue no significativa a excepción del año 2016, esto fue determinante para predecir el total. Entonces como no hubo una fuerte significancia determinaremos lo opuesto, es decir, la categoría no está relacionada con el país de procedencia.

```
In [120]: bp = dict(linestyle='-', linewidth = 2, color='black', facecolor='yellow')
fp = dict(marker = 'o', markerfacecolor = 'green', markersize = 5, linestyle = 'none')
mp = dict(linestyle = '-', linewidth = 2.5, color = 'red')
plt.boxplot([dfpais['2015'], dfpais['2016'],dfpais['2017'],dfpais['2018']])
, labels=["2015", "2016", "2017", "2018"], \
    boxprops = bp, flierprops = fp, medianprops = mp, patch_artist
= True)
plt.show()
```



En esta gráfica podemos determinar que en 2017 fue el año en el que más gente extranjera de un solo país participó en el concurso.

```
In [121]: bp = dict(linestyle='-', linewidth = 2, color='black', facecolor='yellow')
fp = dict(marker = 'o', markerfacecolor = 'green', markersize = 5, linestyle = 'none')
mp = dict(linestyle = '-', linewidth = 2.5, color = 'red')
plt.boxplot([dfcat['2015'], dfcat['2016'],dfcat['2017'],dfcat['2018']], la
bels=["2015", "2016", "2017", "2018"], \
    boxprops = bp, flierprops = fp, medianprops = mp, patch_artist
= True)
plt.show()
```



En esta gráfica podemos determinar que en 2017 fue el año en el que hubo más distribución de participantes entre todas las categorías

Conclusiones

Fue interesante aprender a usar la librería matplotlib, tuve menos problemas para trabajar con ésta librería que con la de plotly. También logré hacer categorizaciones de los datos para poder tratarlos estadísticamente.

De los datos, pudimos determinar que en los años 2015, 2017, 2018 la categoría elegida no era significativa según el país de procedencia, pero en 2016 ocurrió que si era significativo. Al hacer los cálculos de todos los años juntos perdió la significancia. En el año 2017 hubo más participación extranjera de un solo país y además donde se participó con mayor uniformidad en las categorías disponibles.

--11 de Marzo 2019-- Luis Angel Gutierrez Rodriguez [1484412 \(tel:1484412\)](#)

Reporte de práctica 6: Modelos lineales con scipy.stats

En esta práctica tratamos de buscar al menos dos modelos lineales en los datos del cine. Iniciar los datos

Cargar datos de csv

Queremos probar si la categoría eligieron para su clip, está relacionado de alguna forma con el país de procedencia y por año de participación.

Primero importamos el archivo que está en GitHub que tiene información limpia de prácticas pasadas.

```
In [1]: import pandas as pd
from numpy import isnan
from statsmodels.graphics.gofplots import qqplot
import matplotlib.pyplot as plt
from scipy.stats import shapiro
from pandas.compat import u

cine = pd.read_csv('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_
DATOS/master/old/datosLimpiosCine2.csv', sep='\\')
print(len(cine))
cine = cine.dropna()
print(len(cine))

print(cine.columns)
```

2735
2277
Index([u'Anio', u'Categoría', u'Pais', u'Genero', u'ComoSeEnteró',
 u'ReferenciaCelular'],
 dtype='object')

Para evitar errores de cálculos, usando la función dropna() retiramos todos los registros que tengan algún campo vacío y nos deshacemos de 458 registros que no tienen género o tipo de celular y nos quedamos con 2277 registros válidos.

Categorización

Como tenemos muchas cadenas de texto en nuestros datos, es importante hacer categorización ya que hacemos conteos de la información que tenemos disponible y así poder hacer cálculos estadísticos. Categorías por país

Categorías por país

Aplicamos una categorización por país, pero se removieron los datos que involucran a México y Colombia que son los países anfitriones del concurso y por ende son los que nos sesgan la información, entonces esta información será relevante a los países extranjeros que participan.

```
In [2]: paises = cine.Pais.unique()

listaPaises = []
for country in paises:
    listaPaises.append([country,cine[(cine['Pais']==country) & (cine['Anio']=='2015')].count()['Anio'],cine[(cine['Pais']==country) & (cine['Anio']=='2016')].count()['Anio'],cine[(cine['Pais']==country) & (cine['Anio']=='2017')].count()['Anio'],cine[(cine['Pais']==country) & (cine['Anio']=='2018')].count()['Anio'],cine[cine['Pais']==country].count()['Anio']])
#print(listaPaises)
listaPaises.pop(0)
listaPaises.pop(1)

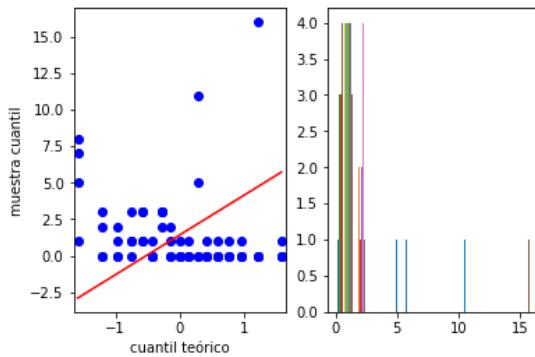
dfpais = pd.DataFrame(data=listaPaises)
dfpais.columns = ['Pais', '2015', '2016', '2017', '2018', 'Total']
print(dfpais)
```

	Pais	2015	2016	2017	2018	Total
0	España	7	8	1	5	21
1	Ecuador	2	3	0	0	5
2	Francia	2	1	0	0	3
3	Venezuela	3	1	1	0	5
4	Argentina	1	3	3	1	8
5	Perú	1	0	0	0	1
6	Estados Unidos	3	3	3	2	11
7	Brasil	2	1	0	0	3
8	Cuba	1	0	0	0	1
9	Alemania	0	1	0	0	1
10	Internacional	0	5	11	0	16
11	Marruecos	0	1	0	0	1
12	Uruguay	0	1	0	0	1
13	República Dominicana	0	1	0	0	1
14	Canadá	0	1	0	0	1
15	Afganistán	0	0	16	0	16
16	Comoras	0	0	1	0	1

Al aplicar pruebas de normalidad a los datos, podemos observar que no está presente la normalidad.

```
In [3]: f, ax = plt.subplots(1, 2)
dfpais2=dfpais[['2015', '2016', '2017', '2018']].copy()
qqplot(dfpais2, line='s', ax = ax[0], xlabel=u"cuantil teórico", ylabel="muestra cuantil")
ax[1] = plt.hist(dfpais2)
for a in [0.05, 0.01]:
    s, p = shapiro(dfpais2)
    print(s, p, a, "normal" if p > a else "no normal")
plt.show()
```

```
(0.5631282329559326, 6.444122037201072e-13, 0.05, 'no normal')
(0.5631282329559326, 6.444122037201072e-13, 0.01, 'no normal')
```



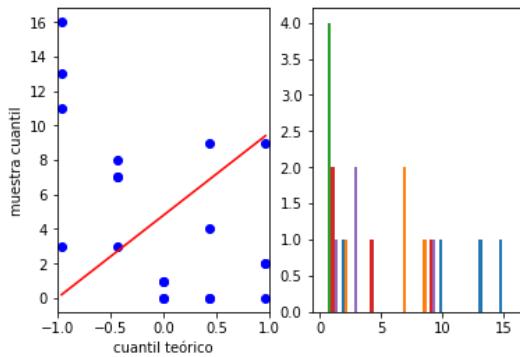
También categorizamos la Categoría del corto, y se clasificó por año.

```
In [4]: categorias = cine.Categoría.unique()
listaCategoria = []
for cat in categorias:
    listaCategoria.append([cat,cine[(cine['Categoría']==cat) & (cine['Anio']==2015) & (cine['Pais']!='Colombia')& (cine['Pais']!='México')].count()['Anio'],cine[(cine['Categoría']==cat) & (cine['Anio']==2016)& (cine['Pais']!='Colombia')& (cine['Pais']!='México')].count()['Anio'],cine[(cine['Categoría']==cat) & (cine['Anio']==2017)& (cine['Pais']!='Colombia')& (cine['Pais']!='México')].count()['Anio'],cine[(cine['Categoría']==cat) & (cine['Anio']==2018)& (cine['Pais']!='Colombia')& (cine['Pais']!='México')].count()['Anio'],cine[(cine['Categoría']==cat) & (cine['Pais']!='Colombia')& (cine['Pais']!='México')].count()['Anio']])
dfcat = pd.DataFrame(data=listaCategoria)
dfcat.columns = ['Categoría', '2015', '2016', '2017', '2018', 'Total']
print(dfcat)
```

	Categoría	2015	2016	2017	2018	Total
0	Aficionado	13	16	11	3	43
1	Profesional	8	7	7	3	25
2	Infantil	1	1	0	0	2
3	Juvenil	0	4	9	0	13
4	SmartIC	0	2	9	2	13

```
In [5]: f, ax = plt.subplots(1, 2)
dfcat2=dfcat[['2015','2016','2017','2018']].copy()
qqplot(dfcat2, line='s', ax = ax[0], xlabel=u"cuantil teórico", ylabel="muestra c
uantil")
ax[1] = plt.hist(dfcat2)
for a in [0.05, 0.01]:
    s, p = shapiro(dfcat2)
    print(s, p, a, "normal" if p > a else "no normal")
plt.show()

(0.8778810501098633, 0.01620960235595703, 0.05, 'no normal')
(0.8778810501098633, 0.01620960235595703, 0.01, 'normal')
```



Podemos observar que los datos de la categoría tampoco estar normales.

Modelo lineal

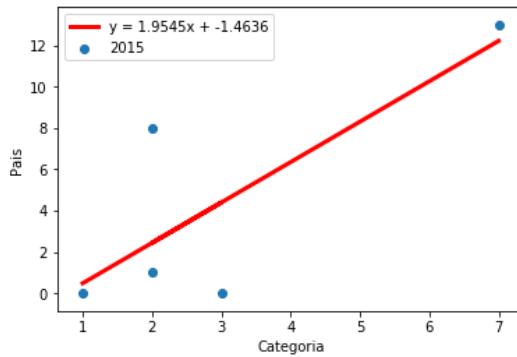
Empezamos con la modelación lineal, suponiendo que la categoría es una consecuencia del país de los creadores del clip. Se probó la relación lineal por año, y también la relación total.

```
In [6]: import matplotlib.pyplot as plt
from scipy import stats

y = dfcat['2015']
x = dfpais['2015']
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)

y = f(x) = 1.9545 x + -1.4636
('error', 0.8974251505426212)
('valor p', 0.11757517937281314)
pendiente no significativo
('R^2', 0.6125761993108931)
```

```
In [7]: plt.plot(x, (a * x + b), label = 'y = {:.4f}x + {:.4f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Categoria")
plt.ylabel("Pais")
plt.show()
```

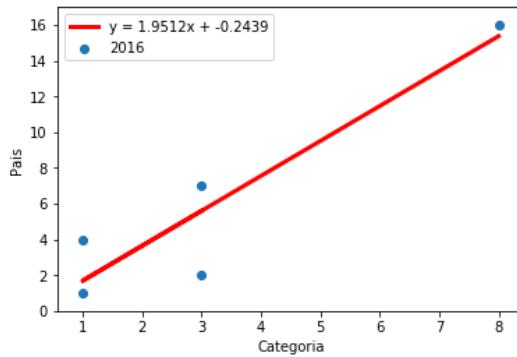


```
In [8]: import matplotlib.pyplot as plt
from scipy import stats

y = dfcat['2016']
x = dfpais['2016']
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)

y = f(x) = 1.9512 x + -0.2439
('error', 0.4633076470424925)
('valor p', 0.024454336793225805)
pendiente significativo
('R^2', 0.8553291012362179)
```

```
In [9]: plt.plot(x, (a * x + b), label = 'y = {:.4f}x + {:.4f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Categoria")
plt.ylabel("Pais")
plt.show()
```

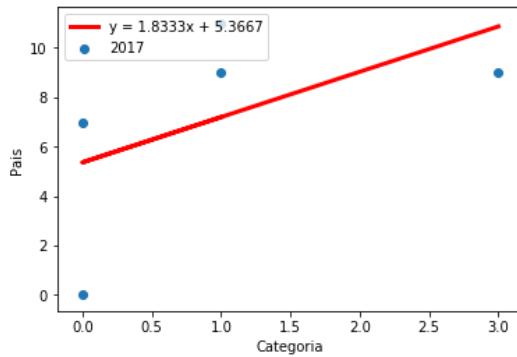


```
In [10]: import matplotlib.pyplot as plt
from scipy import stats

y = dfcat['2017']
x = dfpais['2017']
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)

y = f(x) = 1.8333 x + 5.3667
('error', 1.7099924193031015)
('valor p', 0.36223895879468526)
pendiente no significativo
('R^2', 0.277014652014652)
```

```
In [11]: plt.plot(x, (a * x + b), label = 'y = {:.4f}x + {:.4f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Categoria")
plt.ylabel("Pais")
plt.show()
```

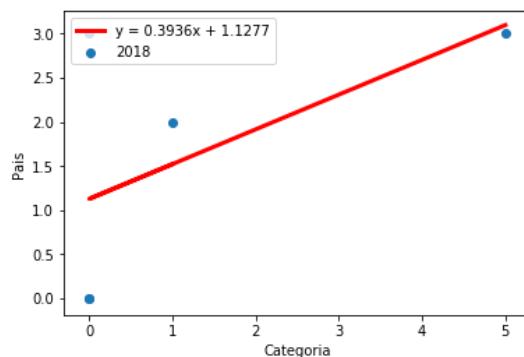


```
In [12]: import matplotlib.pyplot as plt
from scipy import stats

y = dfcat['2018']
x = dfpais['2018']
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)

y = f(x) = 0.3936 x + 1.1277
('error', 0.3338798899250046)
('valor p', 0.32341836282824227)
pendiente no significativo
('R^2', 0.316604995374653)
```

```
In [13]: plt.plot(x, (a * x + b), label = 'y = {:.4f}x + {:.4f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Categoria")
plt.ylabel("Pais")
plt.show()
```

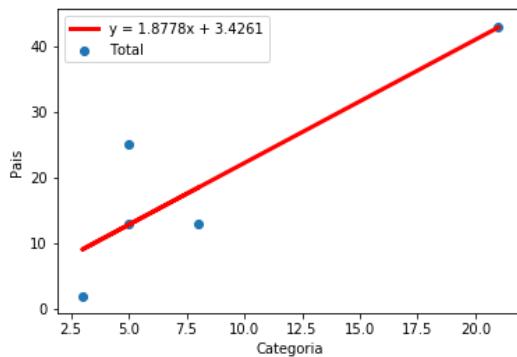


```
In [14]: import matplotlib.pyplot as plt
from scipy import stats

y = dfcat['Total']
x = dfpais['Total']
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)

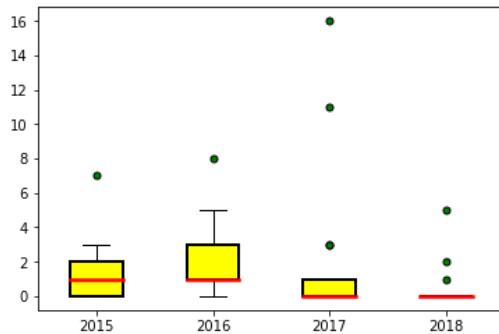
y = f(x) = 1.8778 x + 3.4261
('error', 0.5999372539983482)
('valor p', 0.052063036755892314)
pendiente no significativo
('R^2', 0.7655753541791265)
```

```
In [15]: plt.plot(x, (a * x + b), label = 'y = {:.4f}x + {:.4f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Categoria")
plt.ylabel("País")
plt.show()
```



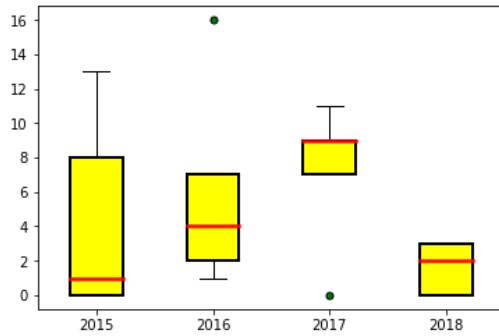
La pendiente fue no significativa a excepción del año 2016, esto fue determinante para predecir el total. Entonces como no hubo una fuerte significación, se determina lo opuesto, es decir, la categoría no está relacionada con el país de procedencia.

```
In [16]: bp = dict(linestyle='-', linewidth = 2, color='black', facecolor='yellow')
fp = dict(marker = 'o', markerfacecolor = 'green', markersize = 5, linestyle = 'none')
mp = dict(linestyle = '-', linewidth = 2.5, color ='red')
plt.boxplot([dfpais['2015'], dfpais['2016'],dfpais['2017'],dfpais['2018']], labels=["2015", "2016", "2017", "2018"], boxprops = bp, flierprops = fp, medianprops = mp, patch_artist = True)
plt.show()
```



En esta gráfica podemos determinar que en 2017 fue el año en el que más gente extranjera de un solo país participó en el concurso.

```
In [17]: bp = dict(linestyle='-', linewidth = 2, color='black', facecolor='yellow')
fp = dict(marker = 'o', markerfacecolor = 'green', markersize = 5, linestyle =
'none')
mp = dict(linestyle = '-', linewidth = 2.5, color ='red')
plt.boxplot([dfcat['2015'], dfcat['2016'],dfcat['2017'],dfcat['2018']], label
s=["2015", "2016", "2017", "2018"], \
boxprops = bp, flierprops = fp, medianprops = mp, patch_artist = Tr
ue)
plt.show()
```



En esta gráfica podemos determinar que en 2017 fue el año en el que hubo más distribución de participantes entre todas las categorías

Conclusiones

Fue interesante aprender a usar la librería matplotlib, tuve menos problemas para trabajar con esta librería que con la de Plotly. También logré hacer categorización de los datos para poder tratarlos de forma estadística.

De los datos, pudimos determinar que en los años 2015, 2017, 2018 la categoría elegida no era significativa según el país de procedencia, pero en 2016 ocurrió que si era significativo. Al hacer los cálculos de todos los años juntos perdió la significación. En el año 2017 hubo más participación extranjera de un solo país y además donde se participó con mayor uniformidad en las categorías disponibles.

--05 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 7: Regresión múltiple

Complicaciones

CIENCIA_DE_DATOS (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master)
/ P7.ipynb (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master/P7.ipynb)

Reporte de práctica 7: Regresión múltiple con statsmodels

En ésta práctica es parecida a la anterior, solo que ahora los modelos lineales tendrán mas de un factor.

Objetivo

Incluir por lo menos un modelo de regresión múltiple e intenta usar sus resultados como un clasificador de alguna variable de interés para el proyecto.

Cargar datos de csv

Queremos probar si la categoría eligieron para su clip, está relacionado de alguna forma con el país de procedencia y por año de participación.

Primero importamos el archivo que esta en github que tiene información limpia de prácticas pasadas.

```
In [6]: import statsmodels.api as sm
from numpy import isnan
import pandas as pd

cine = pd.read_csv('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/cineWoNA
N.csv')
#print(cine)
n=len(cine)
```

Categorizaciones

Como tenemos muchas cadenas de texto en nuestros datos, es importante hacer categorizaciones ya que hacemos conteos de la información que tenemos disponible y así poder hacer cálculos estadísticos.

Categorías por país

Aplicamos una categorización por país, pero se removieron los datos que involucran a México y Colombia que son los países anfitriones del concurso y por ende son los que nos sesgan la información, entonces esta información será relevante a los países extranjeros que participan.

```
In [27]: paises = cine['País'].unique()

listaPaises = []
for country in paises:
    listaPaises.append([country,cine[(cine['País']==country) & (cine['Año']==2015)].count()['Año'],cine[(cine['País']==country) & (cine['Año']==2016)].count()['Año'],cine[(cine['País']==country) & (cine['Año']==2017)].count()['Año'],cine[(cine['País']==country) & (cine['Año']==2018)].count()['Año'],cine[cine['País']==country].count()['Año']])
#print(listaPaises)
#listaPaises.pop(0)## quita Colombia
#listaPaises.pop(1)## quita México

dfpais = pd.DataFrame(data=listaPaises)
dfpais.columns = ['País', '2015', '2016', '2017', '2018', 'Total']
print(dfpais)
```

	País	2015	2016	2017	2018	Total
0	Colombia	276	837	614	357	2084
1	España	7	8	1	5	21
2	México	5	2	1	89	97
3	Ecuador	2	3	0	0	5
4	Francia	2	1	0	0	3
5	Venezuela	3	1	1	0	5
6	Argentina	1	3	3	1	8
7	Perú	1	0	0	0	1
8	Estados Unidos	3	3	3	2	11
9	Brasil	2	1	0	0	3
10	Cuba	1	0	0	0	1
11	Alemania	0	1	0	0	1
12	Internacional	0	5	11	0	16
13	Marruecos	0	1	0	0	1
14	Uruguay	0	1	0	0	1
15	República Dominicana	0	1	0	0	1
16	Canadá	0	1	0	0	1
17	Afganistán	0	0	16	0	16
18	Comoras	0	0	1	0	1

Categorías por "Categoría"

También categorizamos la Categoría del corto, y se clasificó por año.

```
In [28]: categorias = cine['Categoría'].unique()
listaCategoria = []
for cat in categorias:
    listaCategoria.append([cat,cine[(cine['Categoría']==cat) & (cine['Año']==2015) & (cine['País']!='Colombia') & (cine['País']!='México')].count()['Año'],cine[(cine['Categoría']==cat) & (cine['Año']==2016) & (cine['País']!='Colombia') & (cine['País']!='México')].count()['Año'],cine[(cine['Categoría']==cat) & (cine['Año']==2017) & (cine['País']!='Colombia') & (cine['País']!='México')].count()['Año'],cine[(cine['Categoría']==cat) & (cine['Año']==2018) & (cine['País']!='Colombia') & (cine['País']!='México')].count()['Año'],cine[(cine['Categoría']==cat) & (cine['País']!='Colombia') & (cine['País']!='México')].count()['Año']])

dfcat = pd.DataFrame(data=listaCategoria)
dfcat.columns = ['Categoria', '2015', '2016', '2017', '2018', 'Total']
print(dfcat)
```

	Categoría	2015	2016	2017	2018	Total
0	Aficionado	13	16	11	3	43
1	Profesional	8	7	7	3	25
2	Infantil	1	1	0	0	2
3	Juvenil	0	4	9	0	13
4	SmarTIC	0	2	9	2	13

Categorías por Género

También categorizamos género del corto, y se clasificó por año.

```
In [16]: #generos = cine['Género'].unique()
#print(generos)
#listaGenero = []
#for gen in generos:
#    listaGenero.append([gen,cine[(cine['Género']==gen) & (cine['Año']==2015)].count()['Año'],cine[(cine['Género']==gen) & (cine['Año']==2016)].count()['Año'],cine[(cine['Género']==gen) & (cine['Año']==2017)].count()['Año'],cine[(cine['Género']==gen) & (cine['Año']==2018)].count()['Año'],cine[(cine['Género']==gen)].count()['Año']])
#
#dfgen = pd.DataFrame(data=listaGenero)
#dfgen.columns = ['Género', '2015', '2016', '2017', '2018', 'Total']
#print(dfgen)
```

```
In [25]: if n >= 8: # no se puede con menos de ocho
    y = dfpais["Total"]
    x = dfpais[["2015", "2016", "2017", "2018"]]
    x = sm.add_constant(x) # para contar con la b en nuestra f()
    m = sm.OLS(y, x).fit()
    #print(datos.profe.unique()[0])
    print(m.summary())
```

OLS Regression Results

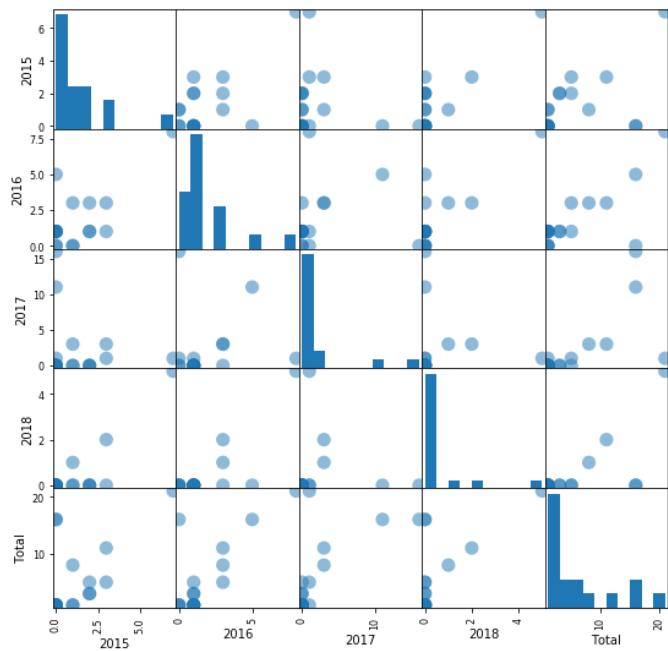
Dep. Variable:	Total	R-squared:	1.000			
Model:	OLS	Adj. R-squared:	1.000			
Method:	Least Squares	F-statistic:	5.196e+30			
Date:	Mon, 25 Mar 2019	Prob (F-statistic):	2.59e-181			
Time:	11:21:47	Log-Likelihood:	536.45			
No. Observations:	17	AIC:	-1063.			
Df Residuals:	12	BIC:	-1059.			
Df Model:	4					
Covariance Type:	nonrobust					
const	-9.992e-16	2.35e-15	-0.426	0.678	-6.11e-15	4.12e-15
2015	1.0000	1.53e-15	6.55e+14	0.000	1.000	1.000
2016	1.0000	1.19e-15	8.43e+14	0.000	1.000	1.000
2017	1.0000	3.45e-16	2.9e+15	0.000	1.000	1.000
2018	1.0000	2.57e-15	3.89e+14	0.000	1.000	1.000
Omnibus:	6.797	Durbin-Watson:	1.502			
Prob(Omnibus):	0.033	Jarque-Bera (JB):	5.453			
Skew:	-0.354	Prob(JB):	0.0655			
Kurtosis:	5.683	Cond. No.	12.2			

Warnings:

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

```
/home/samataroukami/.local/lib/python2.7/site-packages/scipy/stats/stats.py:1416: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=17
  "anyway, n=%i" % int(n))
```

```
In [26]: import matplotlib.pyplot as plt  
tmp = pd.plotting.scatter_matrix(dfpais, figsize = (9, 9), s = 500)
```



Haremos lo mismo con las categorías, pero no espero obtener nada ya que la cantidad de registros es menor a ocho.

```
In [29]: if n >= 8: # no se puede con menos de ocho
    y = dfcat["Total"]
    x = dfcat[["2015", "2016", "2017", "2018"]]
    x = sm.add_constant(x) # para contar con la b en nuestra f()
    m = sm.OLS(y, x).fit()
    #print(datos.profe.unique()[0])
    print(m.summary())

OLS Regression Results
=====
Dep. Variable:                 Total   R-squared:                   1.000
Model:                          OLS   Adj. R-squared:                  nan
Method:                         Least Squares   F-statistic:                0.000
Date: Mon, 25 Mar 2019   Prob (F-statistic):                  nan
Time: 11:38:44   Log-Likelihood:                153.74
No. Observations:                  5   AIC:                     -297.5
Df Residuals:                      0   BIC:                     -299.4
Df Model:                           4
Covariance Type:            nonrobust
=====
              coef    std err      t      P>|t|      [0.025      0.975]
-----  

const    -1.155e-14        inf     -0       nan       nan       nan
2015        1.0000        inf      0       nan       nan       nan
2016        1.0000        inf      0       nan       nan       nan
2017        1.0000        inf      0       nan       nan       nan
2018        1.0000        inf      0       nan       nan       nan
=====
Omnibus:                      nan   Durbin-Watson:                1.364
Prob(Omnibus):                  nan   Jarque-Bera (JB):             0.934
Skew:                           1.035   Prob(JB):                  0.627
Kurtosis:                        2.558   Cond. No.                  54.6
=====

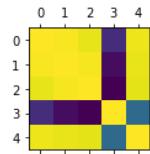
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

/home/samataroukami/.local/lib/python2.7/site-packages/statsmodels/stats/stattools.py:72: ValueWarning: "omni_normtest is not valid with less than 8 observations; 5 samples were given.
"samples were given." % int(n), ValueWarning)
/home/samataroukami/.local/lib/python2.7/site-packages/statsmodels/regression/linear_model.py:1549 : RuntimeWarning: divide by zero encountered in divide
    return 1 - (np.divide(self.nobs - self.k_constant, self.df_resid)
/home/samataroukami/.local/lib/python2.7/site-packages/statsmodels/regression/linear_model.py:1550 : RuntimeWarning: invalid value encountered in double_scalars
    * (1 - self.rsquared))
/home/samataroukami/.local/lib/python2.7/site-packages/statsmodels/regression/linear_model.py:1558 : RuntimeWarning: divide by zero encountered in double_scalars
    return self.ssr/self.df_resid
/home/samataroukami/.local/lib/python2.7/site-packages/statsmodels/regression/linear_model.py:1510 : RuntimeWarning: divide by zero encountered in double_scalars
    return np.dot(wresid, wresid) / self.df_resid
```

No funcionó por que tengo menos de 8 categorías.

```
In [31]: print(dfpais.corr())
f = plt.figure()
sf = f.add_subplot(212)
tmp = sf.matshow(dfpais.corr())
```

	2015	2016	2017	2018	Total
2015	1.000000	0.999679	0.998901	0.972890	0.999269
2016	0.999679	1.000000	0.999529	0.969860	0.999011
2017	0.998901	0.999529	1.000000	0.968822	0.998677
2018	0.972890	0.969860	0.968822	1.000000	0.979544
Total	0.999269	0.999011	0.998677	0.979544	1.000000



Observación

Entonces descubrí que la mejor manera de tratar los datos es reemplazando la categorización en el dataframe original, es decir, cambiar texto por numeros que indican el país o la categoría.

Conclusión

Esta práctica no la concluí a tiempo, tendré que realizarla completa y cumplir con los objetivos para poder agregarla al portafolio al final del semestre.

--25 de Marzo 2019-- Luis Angel Gutierrez Rodriguez [1484412](#) ([tel:1484412](#))

P07

June 6, 2019

1 Reporte de práctica 7: Regresión múltiple con statsmodels

En esta práctica es parecida a la anterior, solo que ahora los modelos lineales tendrán más de un factor.

1.1 Objetivo

- Incluir por lo menos un modelo de regresión múltiple
- Intentar usar sus resultados como un clasificador de alguna variable de interés para el proyecto.

1.1.1 Cargar datos de csv

Primero importamos el archivo que está en GitHub que tiene información limpia de prácticas pasadas.

```
In [1]: import statsmodels.api as sm
        from numpy import isnan
        import pandas as pd
        cine = pd.read_excel('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/main/cine.xlsx')
        cine = cine[['Categoria','Edad','Pais', 'Titulo','Genero', 'Duracion', 'Marca','Referencia']]
        cine = cine.dropna()

        print(len(cine))
```

266

Se quiere probar si la cantidad de personas que participaron en el filme esta relacionada con la Duracion de este, la categoría a la que pertenece y a los Dias de grabacion. Se categoriza la categoría del filme utilizando la librería pandas.Categorical().

```
In [2]: cat = pd.Categorical(cine.Categoria)
        cine['cat'] = cat.codes
```

1.2 Mínimos cuadrados ordinarios

Los mínimos cuadrados ordinarios sirven para encontrar los parámetros poblacionales en un modelo de regresión lineal. Este método minimiza la suma de las distancias verticales entre las respuestas observadas en la muestra y las respuestas del modelo. El parámetro resultante puede expresarse a través de una fórmula sencilla, especialmente en el caso de un único factor.

Para hacer la regresión se confirma que la cantidad de datos que se tienen sea mayor a ocho.

```
In [3]: n = len(cine)
```

```
if n >= 8:  
    y = cine["Personas"]  
    x = cine[["Duracion", "cat", "Dias"]]  
    m = sm.OLS(y, x).fit()  
    print(m.summary())
```

OLS Regression Results

```
=====
```

Dep. Variable:	Personas	R-squared:	0.502			
Model:	OLS	Adj. R-squared:	0.496			
Method:	Least Squares	F-statistic:	88.31			
Date:	Wed, 05 Jun 2019	Prob (F-statistic):	1.48e-39			
Time:	21:26:28	Log-Likelihood:	-952.98			
No. Observations:	266	AIC:	1912.			
Df Residuals:	263	BIC:	1923.			
Df Model:	3					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Duracion	0.0012	0.000	3.733	0.000	0.001	0.002
cat	2.3984	0.259	9.245	0.000	1.888	2.909
Dias	0.0824	0.013	6.282	0.000	0.057	0.108

```
=====
```

Omnibus:	74.119	Durbin-Watson:	1.738
Prob(Omnibus):	0.000	Jarque-Bera (JB):	566.260
Skew:	0.870	Prob(JB):	1.09e-123
Kurtosis:	9.933	Cond. No.	824.

```
=====
```

Warnings:

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

Se verifica si los campos se encuentran correlacionados entre ellos. Por que afectaría si en el modelo trabajamos con columnas que estén fuertemente correlacionados.

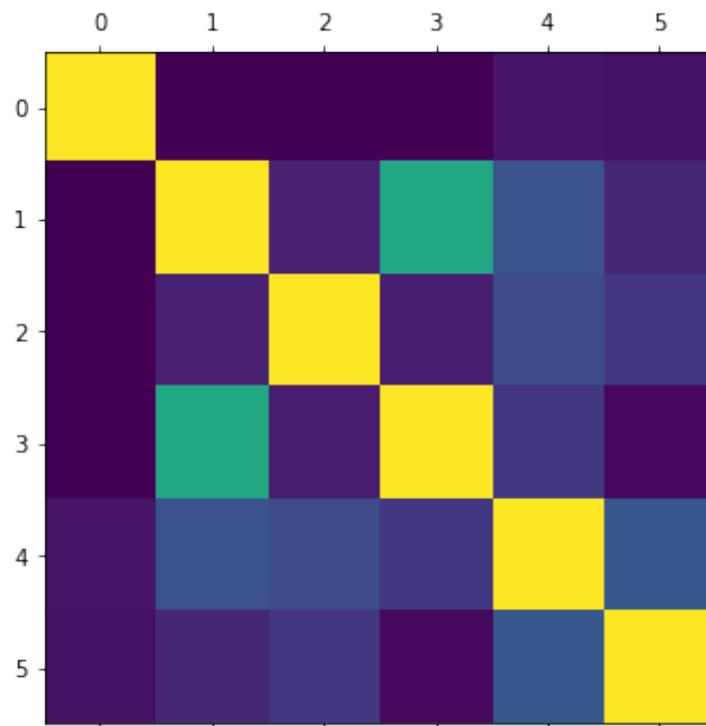
```
In [4]: import matplotlib.pyplot as plt  
f = plt.figure(figsize = (12, 12))  
print(cine.corr())
```

```

sf = f.add_subplot(211)
tmp = sf.matshow(cine.corr())

      Edad Duracion      Dias   Marcas Personas       cat
Edad    1.000000 -0.067706 -0.067153 -0.069770 -0.012014 -0.022563
Duracion -0.067706  1.000000  0.028244  0.571155  0.204051  0.044962
Dias     -0.067153  0.028244  1.000000  0.012944  0.179026  0.102243
Marcas   -0.069770  0.571155  0.012944  1.000000  0.102514 -0.046332
Personas -0.012014  0.204051  0.179026  0.102514  1.000000  0.224517
cat      -0.022563  0.044962  0.102243 -0.046332  0.224517  1.000000

```

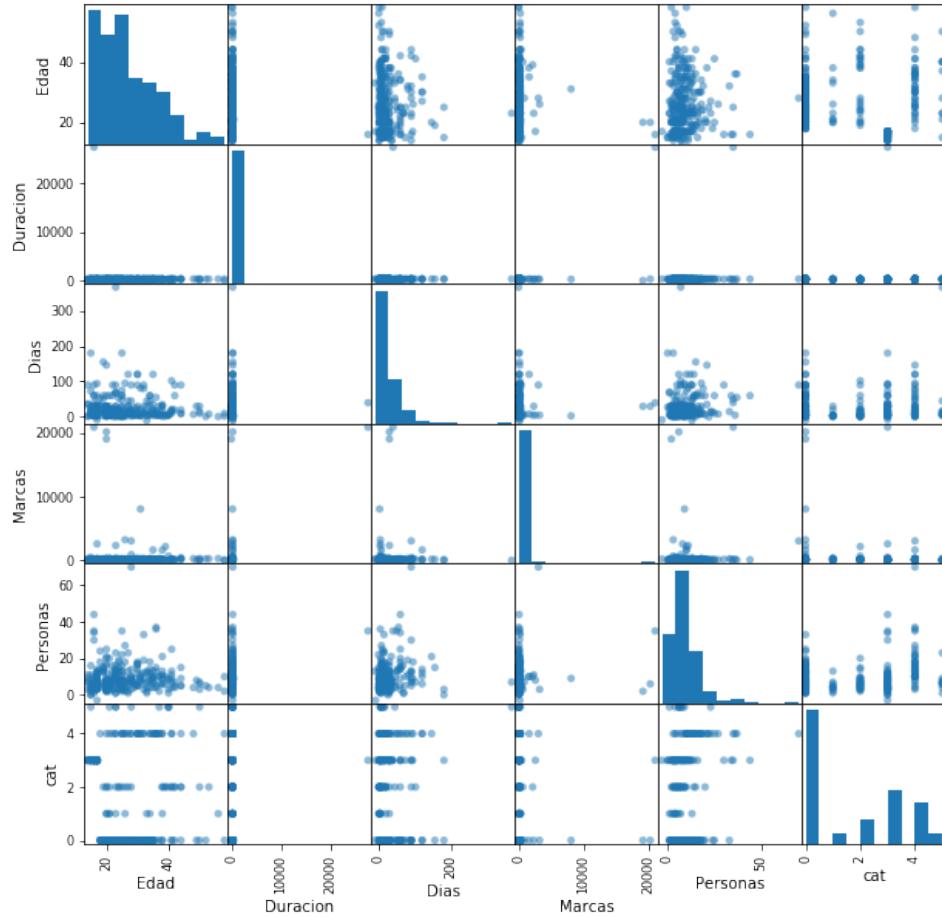


Se puede ver que ningún parámetro se encuentra fuertemente correlacionado. Esto indica que todos los parámetros son independientes.

Se realiza una grafica de dispersión para determinar que variables nos conviene estudiar.

```
In [5]: print(len(cine))
tmp = pd.plotting.scatter_matrix(cine, figsize = (10, 10), s =100)
```

266



En este caso, los datos se encuentran dispersos. Se puede observar que no están correlacionados. En los histogramas de la diagonal principal se observa que la distribución de los datos no es normalmente distribuida.

1.3 Usar los resultados como clasificador

A pesar de que no se obtuvieron los resultados deseados, se intenta predecir el tamaño de la sinopsis en función del género, categoría y país del filme y la marca del smartphone con el cual se grabó el filme. Este tamaño no deberá ser mayor a 140 caracteres, como límite de Twitter.

```
In [6]: y = cine["Personas"]
x = cine[["Duracion", "cat", "Dias"]]
m = sm.OLS(y, x).fit()
comp = pd.DataFrame(cine, columns = ["Personas"])
comp['pron'] = m.predict(x)
```

```

comp['error'] = comp.Personas - comp.pron
comp['absE'] = pd.DataFrame.abs(comp['error'])
orden = comp.sort_values(by = ['absE'])
mejores = orden.head(10)
mejores.insert(0, 'tipo', 'mejores')
peores = orden.tail(10)
peores.insert(0, 'tipo', 'peores')
pd.concat([mejores, peores])
n = len(comp)
for e in [1, 5, 10]:
    k = sum(comp.absE < e)
    print(k, "de", n, "= {:.2f}% estuvieron dentro de".format(100 * k / n), e, "% de"
26 de 266 = 9.77% estuvieron dentro de 1 % de error del valor real
136 de 266 = 51.13% estuvieron dentro de 5 % de error del valor real
230 de 266 = 86.47% estuvieron dentro de 10 % de error del valor real

```

Se vera qué tanto cambia si no usamos el estimado del la marca del smartphone. Para esto se considerarán las siguientes variables.

fp = falsos positivos: según el modelo pasan, pero en realidad no pasan.
 fn = falsos negativos: según el modelo reprobaban, pero en realidad pasan.
 tp = verdaderos positivos: pasan en la realidad y también según el modelo.
 tn = verdaderos negativos: reprobaban en la realidad y también según el modelo.

Teniendo $tt = tp + tn + fp + fn$ podemos calcular

La sensibilidad: $tp / (tp + fn)$ que mide qué tan bien se modela los que aprueban.
La especificidad: $tn / (tn + fp)$ que mide qué tan bien se modela los que reprobaban.
La precisión: $(tp + tn) / tt$ que mide el desempeño total de La asignación correcta de la clasificación.

```

In [7]: y = cine["Personas"]
x = cine[["Duracion", "cat", "Dias"]]
m = sm.OLS(y, x).fit()
comp = pd.DataFrame(cine, columns = ["Personas"])
comp['pron'] = m.predict(x)
comp['error'] = comp.Personas - comp.pron
comp['absE'] = pd.DataFrame.abs(comp['error'])
orden = comp.sort_values(by = ['absE'])
mejores = orden.head(10)
mejores.insert(0, 'tipo', 'mejores')
peores = orden.tail(10)
peores.insert(0, 'tipo', 'peores')
pd.concat([mejores, peores])
n = len(comp)
for e in [1, 5, 10]:
    k = sum(comp.absE < e)
    print(k, "de", n, "= {:.2f}% estuvieron dentro de".format(100 * k / n), e, \

```

```

    "% de error del valor real")
fp = sum((comp.Personas < 9) & (comp.pron >= 9))
fn = sum((comp.Personas >= 9) & (comp.pron < 9))
tp = sum((comp.Personas >= 9) & (comp.pron >= 9))
tn = sum((comp.Personas < 9) & (comp.pron < 9))
print('Clasificación')
print(fp, fn, tp, tn)
tt = tp + tn + fp + fn
print('sensibilidad', tp / (tp + fn))
print('especificidad', tn / (tn + fp))
print('precisión', (tp + tn) / tt)

26 de 266 = 9.77% estuvieron dentro de 1 % de error del valor real
136 de 266 = 51.13% estuvieron dentro de 5 % de error del valor real
230 de 266 = 86.47% estuvieron dentro de 10 % de error del valor real
Clasificación
33 69 48 116
sensibilidad 0.41025641025641024
especificidad 0.7785234899328859
precisión 0.6165413533834586

```

A pesar de que esta versión tiene mejores resultados, parece que este modelo es incapaz de predecir o clasificar, falsos negativos y verdaderos negativos ya que la función con la que predice no es muy buena.

1.4 Conclusión

Se trabajó con parámetros que eran cadenas de caracteres y se utilizó la función Categorical para convertirlos a números y poder hacer la regresión tomando en cuenta estos parámetros no numéricos. Al hacer la regresión de mínimos cuadrados ordinarios se obtuvo un valor de R2 de 0.502. Se intentó clasificar por el conjunto de personas. Donde se filtraban los elementos que fueran menores o iguales a 9 integrantes, para poder clasificarlos. Con lo cual se pudo acercar al 86.47% de clasificación de los filmes. Posteriormente se realizó una prueba considerando falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos.

--05 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 8: Análisis de varianza y de componentes principales

Complicaciones

Práctica no
realizada a
tiempo.

P08

June 6, 2019

1 Reporte de práctica 8: Análisis de varianza y de componentes principales

En esta práctica haremos análisis de varianza(ANOVA) a los datos de las prácticas anteriores.

1.1 Objetivo

- Realizar un análisis de varianza (ANOVA)
- Realizar un análisis de componentes principales (PCA)

1.1.1 Lectura de datos

Se procede a abrir los datos de la práctica anterior.

```
In [1]: import statsmodels.api as sm
from numpy import isnan
import pandas as pd
cine = pd.read_excel('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS,
cine = cine[['Categoria','Edad','Pais', 'Titulo','Genero', 'Duracion', 'Marca','Referencia']]
cine = cine.dropna()

print(len(cine))
```

266

1.2 ANOVA

El Análisis de varianza (ANOVA) es una herramienta para cuantificar sí o no una variable o factor tiene un efecto estadísticamente significativo en una variable de interés. Se realiza a partir de la salida del modelo de regresión que se obtiene con otra versión de ols.

Hay tres tipos de ANOVA, cuyo uso depende de sí o no se espera contar con interacciones entre las variables de entrada y sí o lo los datos son balanceados.

La hipótesis nula en ANOVA es que no haya diferencias; si el valor p de una variable es menor a la significación establecida (para nosotros 0.05 está bien; con muchos datos muy precisos es mejor usar 0.01), entonces se rechaza la hipótesis nula, concluyendo que esa variable sí tiene un efecto estadísticamente significativo.

Se analiza si las personas están relacionadas con la categoría, género y los días de grabación del filme.

```
In [2]: from statsmodels.formula.api import ols
import statsmodels.api as sm

m = ols('Personas ~ Categoria + Genero + Dias', data = cine).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 es significativo"))
    print("sum_sq      df          F          PR(>F)\n")
    print(a)

sum_sq      df          F          PR(>F)
Categoria   2281.950763  5.0  8.659926  1.366205e-07
Genero      320.198519   8.0  0.759464  6.388620e-01
Dias         205.450311   1.0  3.898385  4.942439e-02
Residual    13280.751254 252.0       NaN        NaN
Categoria es significativo
Genero NO es significativo
Dias es significativo
Residual NO es significativo
```

Podemos ver que los días y categoría si son significativos, pero el residual es muy grande. Agreguemos el factor "Duración" para ver su interacción con estos factores.

```
In [3]: from statsmodels.formula.api import ols
import statsmodels.api as sm

m = ols('Personas ~ Categoria * Duracion + Genero * Duracion + Dias * Duracion ', data = cine)
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 es significativo"))
    print("sum_sq      df          F          PR(>F)\n")
    print(a)

sum_sq      df          F          PR(>F)
Categoria   2204.147359  5.0  8.696252  1.371812e-07
Genero      292.599339   8.0  0.721514  6.724490e-01
Duracion    703.101072   1.0  13.870089  2.446383e-04
Categoria:Duracion  203.155195  5.0  0.801529  5.495267e-01
Genero:Duracion  364.989594   8.0  0.900020  5.171006e-01
Dias         237.026122   1.0  4.675819  3.158689e-02
Dias:Duracion  33.485104   1.0  0.660561  4.171741e-01
Residual    12064.670548 238.0       NaN        NaN
Categoria es significativo
```

```

Genero NO es significativo
Duracion es significativo
Categoria:Duracion NO es significativo
Genero:Duracion NO es significativo
Dias es significativo
Dias:Duracion NO es significativo
Residual NO es significativo

```

Ahora la Duración tomó significación. Lo único significativo, sin interacciones, siguen siendo Categoría y Días. Revisemos por las interacciones.

```

In [4]: m = ols('Personas ~ Categoria * Dias + Dias * Duracion + Duracion * Categoria', data)
         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 es significativo"))

          sum_sq      df        F      PR(>F)
Categoria    2611.588301  5.0  10.456200  4.079372e-09
Dias        270.898309  1.0   5.423073  2.067991e-02
Categoria:Dias  319.177406  5.0   1.277913  2.739514e-01
Duracion     684.351795  1.0  13.699938  2.644666e-04
Duracion:Categoria  135.880382  5.0   0.544034  7.428013e-01
Dias:Duracion   0.926056  1.0   0.018539  8.918085e-01
Residual     12338.369436 247.0       NaN        NaN

```

Categoría es significativo
Días es significativo
Categoría:Días NO es significativo
Duración es significativo
Duración:Categoría NO es significativo
Días:Duración NO es significativo
Residual NO es significativo

Se puede observar que a pesar de ser significativas independientemente, al interactuar no significan nada.

1.3 PCA

El **análisis de componentes principales (PCA)** sirve para reducir múltiples variable de entrada en una menor cantidad de variables para modelar la variable de interés. Funciona con datos numéricos que primero se normalizan y luego se proyectan a una dimensión deseada.

Se aplica un PCA sobre los datos,

- * Primero nos deshacemos de las columnas que no conviene codificar con números.
- * Luego categorizamos las que sí convienen.

- * Las reemplazamos con los números de las categorías.
- * Estandarizamos rangos.
- * Probamos reducción proyectando los datos a dos dimensiones.
- * Visualizamos el resultado.

```
In [5]: from sklearn.decomposition import PCA
        from sklearn.preprocessing import StandardScaler

        print('Qué tenemos')
        print(cine.columns)

xVars = ['Edad', 'Duracion', 'Marcas', 'Categoria']

cat = pd.Categorical(cine.Categoria)
cine['Categoria'] = cat.codes

#print(cine.Personas.mean())
#print(cine.Dias.mean())

#d.CF2op = d.CF2op.replace(nan, 0) # para no perder a los que pasaron en primera
#d = d.dropna() # podemos únicamente usar los renglones que contienen todas las respues
d = cine
d = d.dropna()
x = d.loc[:, xVars].values
x = StandardScaler().fit_transform(x)
y1 = d.loc[:, ['Personas']].values
y2 = d.loc[:, ['Dias']].values
k = 2 # dimensiones deseadas
pca = PCA(n_components = k)
cd = pd.DataFrame(data = pca.fit_transform(x), columns = ['comp_{}'.format(i) for i in range(k)])
cd['Personas'] = y1
cd['Dias'] = y2
ordenado = pd.DataFrame.sort_values(cd, ['Personas'], ascending = False)
display(cd.head(10))
```

Qué tenemos

```
Index(['Categoria', 'Edad', 'Pais', 'Titulo', 'Genero', 'Duracion', 'Marca',
       'Referencia', 'Dias', 'Marcas', 'Personas', 'Sinopsis'],
      dtype='object')
```

	comp_0	comp_1	Personas	Dias
0	-0.093040	0.634029	4	7
1	-0.330257	1.187852	7	8
2	-0.383991	1.564923	11	16
3	0.038034	-1.231784	5	3
4	-0.086769	0.520188	13	90
5	-0.262758	1.181875	12	30

```

6 -0.269580  0.902898      6      5
7 -0.035894 -1.288132      1      7
8  0.061712 -1.343304     16     10
9 -0.078685  0.582270      9     20

```

In [6]: `import matplotlib.pyplot as plt`

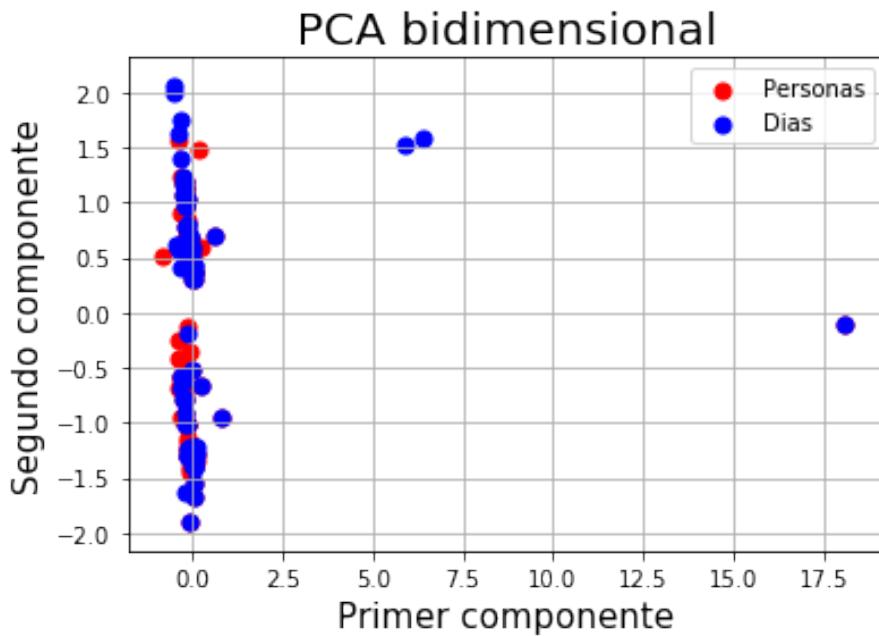
```

d = cd
n = len(d)
pri = d.Personas >= 10
seg = d.Dias >= 28
#ter = ~ (pri / seg)
n == sum(pri) + sum(seg) #+ sum(ter)

#print(sum(pri), sum(seg), sum(ter))

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 = 'r', 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(['Personas', 'Dias'])
plt.grid()

```



1.4 Conclusión

En ANOVA se pudo observar que aunque individualmente los datos sean significativos, cuando interactúan no lo son. En el PCA se modeló en para las variables Personas y Días en función de la Edad del participante, la duración y categoría del filme y las marcas de grabación. Con esta información disponible, en dos dimensiones, no se pueden separar los la cantidad de personas con los días de grabación.

--05 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 9: Pronósticos

Complicaciones

CIENCIA_DE_DATOS (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master)
/ P9.ipynb (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master/P9.ipynb)

Reporte de Práctica 9: Pronósticos con statsmodels

En esta práctica trabajamos con valores distintos a los de las demás debido a que no se concretó la actualización de las demás prácticas y por ende, no tenemos un conjunto de datos validos para pronosticar. Así que siguiendo el ejemplo de la Dra. Elisa, busque otro conjunto de datos con suficientes registros para poder realizar un pronostico decente.

En ésta ocasión el conjunto de datos que decidí utilizar es el histórico del valor del Dolar Americano en relación al Nuevo Peso Mexicano(MXP), hago el enfasis en el Nuevo Peso ya que antes del 1° de Enero de 1993 el Peso Mexicano(MXN) tenía mil veces el valor que representaba, es decir, $1 \text{ MXP} = 1000 \text{ MXN}$, debido a una devaluación acordada por la administración ejecutiva de México entre los años 1988 y 1994. El país atravesó una inflación muy alta heredada de los malos manejos de las anteriores administraciones. Para mitigar los costos excesivos de los productos, se acordó la eliminación de los ultimos tres ceros de cualquier número que representara dinero. Virtualmente los productos pasaron de costar millones a costar miles.

Debido a que quiero hacer un pronostico acertado, busque todos los resgitros desde el 1° de Enero de 1993 al 8 de Abril de 2019, por la cantidad de datos que manejo decidí descargar la información en tres archivos, cada uno representa una década, 1990's, 2000's y 2010's. Los datos fueron obtenidos del sitio web [mx.investing.com](https://mx.investing.com/currencies/usd-mxn-historical-data) (<https://mx.investing.com/currencies/usd-mxn-historical-data>)

Objetivo

Pronosticar el precio del dolar para el 9 de Abril de 2019

Preparación de los datos

Una vez cargados los datos en nuestro repositorio, procedemos a importarlos a python3 y los ingresamos a un dataframe de pandas. reemplazamos los puntos de las fechas por espacios vacios para poder formatear la entrada de la fecha. Luego ordenamos las fechas de forma creciente.

```
In [247]: import pandas as pd
df90s = pd.read_csv('Pronosticos/DHUSD_MXN1990.csv')
df90s['Fecha'] = df90s['Fecha'].apply(lambda x: x.replace('.',''))
df90s['Fecha'] = pd.to_datetime(df90s['Fecha'], format='%d%m%Y')
df90s['% var.'] = df90s['% var.'].str.rstrip('%').astype('float') / 100.0
df90s = df90s.sort_values(['Fecha'])
print("90s = ",len(df90s))

#print(df90s)

df00s = pd.read_csv('Pronosticos/DHUSD_MXN2000.csv')
df00s['Fecha'] = df00s['Fecha'].apply(lambda x: x.replace('.',''))
df00s['Fecha'] = pd.to_datetime(df00s['Fecha'], format='%d%m%Y')
df00s['% var.'] = df00s['% var.'].str.rstrip('%').astype('float') / 100.0
df00s = df00s.sort_values(['Fecha'])
print("00s = ",len(df00s))

#print(df00s)

df10s = pd.read_csv('Pronosticos/DHUSD_MXN2010.csv')
df10s['Fecha'] = df10s['Fecha'].apply(lambda x: x.replace('.',''))
df10s['Fecha'] = pd.to_datetime(df10s['Fecha'], format='%d%m%Y')
df10s['% var.'] = df10s['% var.'].str.rstrip('%').astype('float') / 100.0
df10s = df10s.sort_values(['Fecha'])
print("10s = ",len(df10s))

#print(df10s)

('90s = ', 1821)
('00s = ', 2606)
('10s = ', 2463)
```

Una vez ordenados los datos como los necesitamos, los fusionamos en un solo dataframe, y trasponemos esta matriz de datos, reemplazamos el nombre de las columnas con la fecha del valor, quitamos las fechas del dataframe y guardamos todos en un .csv para respaldar la información procesada. Imprimimos los tipos de dato antes de la transposición para asegurarnos de que todo quedó en el tipo de dato correcto.

```
In [248]: historico = pd.concat([df90s,df00s,df10s], ignore_index=True)

print(historico.dtypes)

#historico = historico.transpose()
#historico.columns = historico.iloc[0]
#historico = historico[1:]

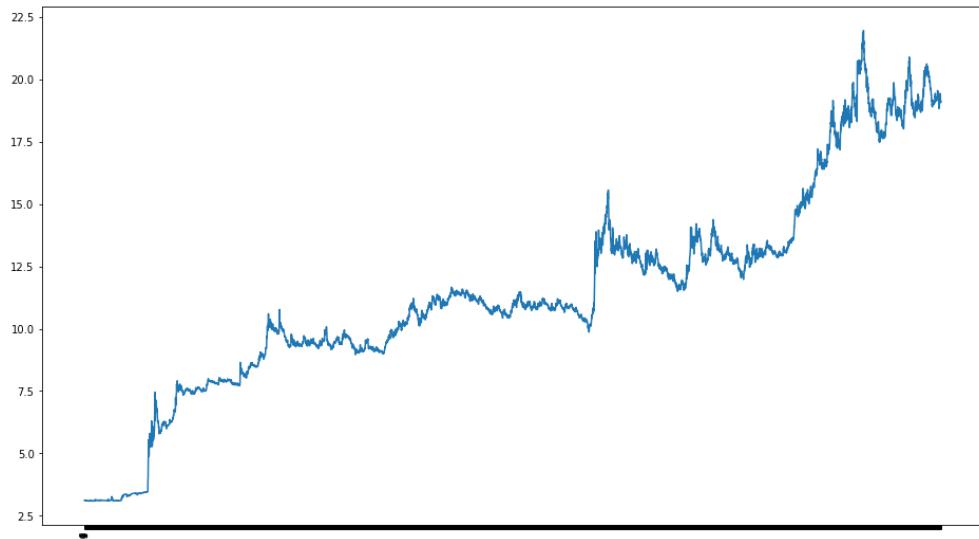
print(len(historico),len(historico.columns))

historico.to_csv('historicoUSD_MXN.csv', index=False)

Fecha        datetime64[ns]
Cierre       float64
Apertura    float64
Máximo       float64
Mínimo       float64
% var.       float64
dtype: object
(6890, 6)
```

Una vez ordenada la información como la necesitamos, empezamos a observar el comportamiento de los datos graficando.

```
In [249]: import matplotlib.pyplot as plt
from numpy import asarray
ejemplo = historico['Cierre']
#print(ejemplo)
x = range(len(ejemplo))
plt.plot(x, asarray(ejemplo))
plt.xticks(x, "04/01/1993", rotation='vertical', fontsize=8)
plt.show()
```



Ahora aplicaremos el pronóstico de un paso de Holt

```
In [180]: from statsmodels.tsa.api import Holt

#historico = historico.transpose()
#historico.columns = historico.iloc[0]
#historico = historico[1:]

#print(historico)
lbls = historico['Cierre']
#print(lbls)
x = range(len(lbls))
#print(x)
pronosticos = []
for i in range(len(historico)):
    y = asarray(historico['Cierre'].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('CF 1 op verdadero', fontsize = 15)
plt.ylabel('Pronóstico', fontsize = 15)
plt.scatter(d.CFira, pronosticos, c = 'g', s = 50)
plt.show()
```

```
-----
KeyboardInterrupt                                     Traceback (most recent call last)
<ipython-input-180-910304bee12a> in <module>()
  13 for i in range(len(historico)):
  14     y = asarray(historico['Cierre'].loc[i:])
```

```

--> 15     f = Holt(asarray(y)).fit(smoothing_level = 0.1)
16     pronosticos.append(f.forecast(1))
17 plt.title('Pronóstico de un paso con el método de Holt', fontsize = 20)

/home/samataroukami/.local/lib/python2.7/site-packages/statsmodels/tsa/holtwinters.py in fit
(self, smoothing_level, smoothing_slope, damping_slope, optimized)
    887         return super(Holt, self).fit(smoothing_level=smoothing_level,
    888                                         smoothing_slope=smoothing_slope, damping_slope=d
amping_slope,
--> 889                                         optimized=optimized)

/home/samataroukami/.local/lib/python2.7/site-packages/statsmodels/tsa/holtwinters.py in fit
(self, smoothing_level, smoothing_slope, smoothing_seasonal, damping_slope, optimized, use_bo
xcox, remove_bias, use_basinhopping)
    584         # solution to parameters
    585         res = minimize(func, p[xi], args=
--> 586             xi, p, y, l, b, s, m, self.nobs, max_seen), bounds=bounds[xi])

    587         p[xi] = res.x
    588         [alpha, beta, gamma, 10, b0, phi] = p[:6]

/home/samataroukami/.local/lib/python2.7/site-packages/scipy/optimize/_minimize.py in minimi
ze(fun, x0, args, method, jac, hess, hessp, bounds, constraints, tol, callback, options)
    599         elif meth == 'l-bfgs-b':
    600             return _minimize_lbfqsb(fun, x0, args, jac, bounds,
--> 601                                         callback=callback, **options)
    602         elif meth == 'tnc':
    603             return _minimize_tnc(fun, x0, args, jac, bounds, callback=callback,

/home/samataroukami/.local/lib/python2.7/site-packages/scipy/optimize/lbfqsb.py in _minimize_
_lbfqsb(fun, x0, args, jac, bounds, disp, maxcor, ftol, gtol, eps, maxfun, maxiter, iprint, c
allback, maxls, **unknown_options)
    333             # until the completion of the current minimization iteration.
    334             # Overwrite f and g:
--> 335             f, g = func_and_grad(x)
    336             elif task_str.startswith(b'NEW_X'):
    337                 # new iteration

/home/samataroukami/.local/lib/python2.7/site-packages/scipy/optimize/lbfqsb.py in func_and_
grad(x)
    278     if jac is None:
    279         def func_and_grad(x):
--> 280             f = fun(x, *args)
    281             g = _approx_fprime_helper(x, fun, epsilon, args=args, f0=f)
    282             return f, g

/home/samataroukami/.local/lib/python2.7/site-packages/scipy/optimize/optimize.py in functio
n_wrapper(*wrapper_args)
    298     def function_wrapper(*wrapper_args):
    299         ncalls[0] += 1
--> 300         return function(*wrapper_args + args)
    301
    302     return ncalls, function_wrapper

/home/samataroukami/.local/lib/python2.7/site-packages/statsmodels/tsa/holtwinters.py in _ho
lt_add_dam(x, xi, p, y, l, b, s, m, n, max_seen)
    85         for i in range(1, n):
    86             l[i] = (y_alpha[i - 1]) + (alphac * (l[i - 1] + phi * b[i - 1]))
--> 87             b[i] = (beta * (l[i] - l[i - 1])) + (betac * phi * b[i - 1])
    88         return sqeuclidean(l + phi * b, y)
    89

KeyboardInterrupt:

```

Como tarda mucho en procesar la información pasaremos a las demás pruebas

```
In [250]: historico = historico.transpose()
historico.columns = historico.iloc[0]
historico = historico[1:]
historico = historico.transpose()

In [271]: from statsmodels.tsa.stattools import adfuller

# rutina de https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/
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='BIC')
    #dfoutput = adfuller(ts)
    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 ({:s})'.format(key)] = value
    return '\n\nResults of Dickey-Fuller Test:\n' + '\n'.join(['{:s}\t{:,.3f}'.format(k, v) for (k, v) in dfoutput.items()])

plt.rcParams["figure.figsize"] = [16, 10]
f = plt.figure()
i = 1
lvls = historico.columns
r = len(lvls)
t = ''
w = 6889
for c in lvls:
    #print(historico[c])
    t += test_stationarity(historico[c], w, r, i)
    i += 1
plt.plot()
print(t)

Results of Dickey-Fuller Test:
Test Statistic -1.068
p-value 0.728
#Lags Used 1.000
Number of Observations Used 6888.000
Critical Value (5%) -2.862
Critical Value (1%) -3.431
Critical Value (10%) -2.567

Results of Dickey-Fuller Test:
Test Statistic -1.070
p-value 0.727
#Lags Used 1.000
Number of Observations Used 6888.000
Critical Value (5%) -2.862
Critical Value (1%) -3.431
Critical Value (10%) -2.567

Results of Dickey-Fuller Test:
Test Statistic -1.101
p-value 0.715
#Lags Used 3.000
Number of Observations Used 6886.000
Critical Value (5%) -2.862
```

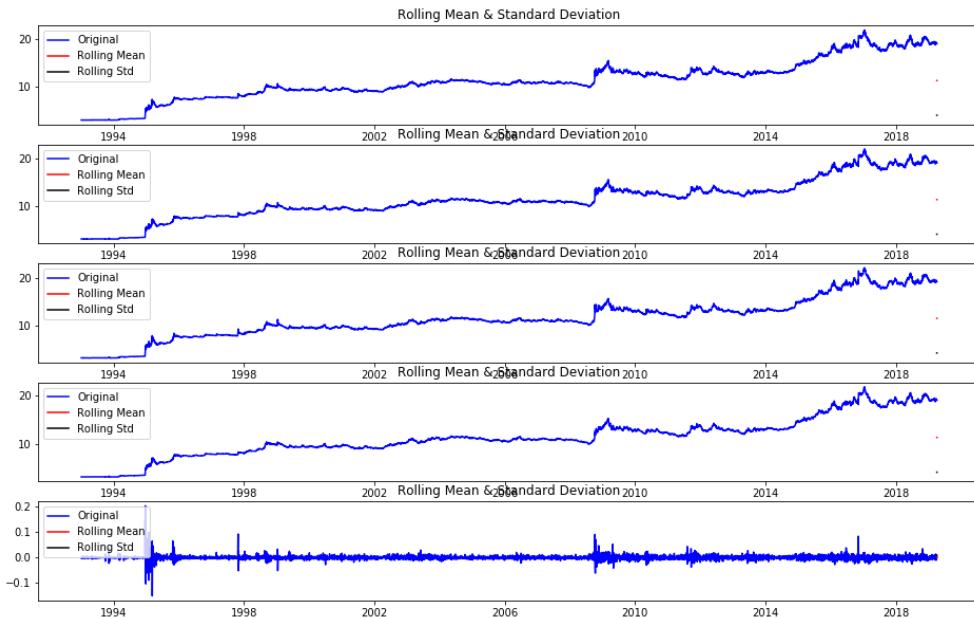
```

Critical Value (1%)      -3.431
Critical Value (10%)     -2.567

Results of Dickey-Fuller Test:
Test Statistic   -1.013
p-value 0.749
#Lags Used      8.000
Number of Observations Used    6881.000
Critical Value (5%)      -2.862
Critical Value (1%)       -3.431
Critical Value (10%)      -2.567

Results of Dickey-Fuller Test:
Test Statistic   -19.643
p-value 0.000
#Lags Used      12.000
Number of Observations Used    6877.000
Critical Value (5%)      -2.862
Critical Value (1%)       -3.431
Critical Value (10%)      -2.567

```



Conclusión

Demasiados datos para trabajar, falta obtener p y q para pronosticar. --08 de Abril 2019-- Luis Angel Gutierrez Rodriguez [1484412](#)
(tel:1484412)

P09

June 6, 2019

1 Reporte de práctica 9: Pronósticos con statsmodels

En esta práctica trabajamos con valores distintos a los de las demás debido a que no se concretó la actualización de las demás prácticas y por ende, no tenemos un conjunto de datos validos para pronosticar. Así que siguiendo el ejemplo de la Dra. Elisa, busque otro conjunto de datos con suficientes registros para poder realizar un pronostico decente.

En esta ocasión el conjunto de datos que decidí utilizar es el histórico del valor del Dólar Americano en relación con el Nuevo Peso Mexicano(MXP), hago el énfasis en el Nuevo Peso ya que antes del 1º de enero de 1993 el Peso Mexicano(MXN) tenía mil veces el valor que representaba, es decir, 1 MXP = 1000 MXN, debido a una devaluación acordada por la administración ejecutiva de México entre los años 1988 y 1994. El país atravesó una inflación muy alta heredada de los malos manejos de las anteriores administración. Para mitigar los costos excesivos de los productos, se acordó la eliminación de los últimos tres ceros de cualquier número que representara dinero. Virtualmente los productos pasaron de costar millones a costar miles.

Debido a que quiero hacer un pronostico acertado, busque todos los registros desde el 1º de enero de 1993 al 7 de abril de 2019, por la cantidad de datos que manejo decidí descargar la información en tres archivos, cada uno representa una década, 1990's, 2000's y 2010's. Los datos fueron obtenidos del sitio web mx.investing.com

1.1 Objetivo

- Pronosticar el precio del dólar para el 8 de abril de 2019

1.2 Preparación de los datos

Una vez cargados los datos en nuestro repositorio, procedemos a importarlos a python3 y los ingresamos a un dataframe de pandas. reemplazamos los puntos de las fechas por espacios vacios para poder formatear la entrada de la fecha. Luego ordenamos las fechas de forma creciente.

```
In [1]: import pandas as pd
df90s = pd.read_csv('old/Pronosticos/DHUSD_MXN1990.csv')
df90s['Fecha'] = df90s['Fecha'].apply(lambda x: x.replace('.', ' '))
df90s['Fecha'] = pd.to_datetime(df90s['Fecha'], format='%d%m%Y')
df90s['% var.'] = df90s['% var.'].str.rstrip('%').astype('float') / 100.0
df90s = df90s.sort_values(['Fecha'])
print("90s = ", len(df90s))

#print(df90s)
```

```

df00s = pd.read_csv('old/Pronosticos/DHUSD_MXN2000.csv')
df00s['Fecha'] = df00s['Fecha'].apply(lambda x: x.replace('.', ''))
df00s['Fecha'] = pd.to_datetime(df00s['Fecha'], format='%d%m%Y')
df00s['% var.'] = df00s['% var.'].str.rstrip('%').astype('float') / 100.0
df00s = df00s.sort_values(['Fecha'])
print("00s = ", len(df00s))

#print(df00s)

df10s = pd.read_csv('old/Pronosticos/DHUSD_MXN2010.csv')
df10s['Fecha'] = df10s['Fecha'].apply(lambda x: x.replace('.', ''))
df10s['Fecha'] = pd.to_datetime(df10s['Fecha'], format='%d%m%Y')
df10s['% var.'] = df10s['% var.'].str.rstrip('%').astype('float') / 100.0
df10s = df10s.sort_values(['Fecha'])
print("10s = ", len(df10s))

#print(df10s)

90s = 1821
00s = 2606
10s = 2463

```

Una vez ordenados los datos como los necesitamos, los fusionamos en un solo DataFrame, y transponemos esta matriz de datos, reemplazamos el nombre de las columnas con la fecha del valor, quitamos las fechas del DataFrame y guardamos todos en un .csv para respaldar la información procesada. Imprimimos los tipos de dato antes de la transposición para asegurarnos de que todo quedo en el tipo de dato correcto.

```

In [2]: historico = pd.concat([df90s,df00s,df10s], ignore_index=True)

print(historico.dtypes)

print(len(historico),len(historico.columns))

historico.to_csv('old/historicoUSD_MXN.csv', index=False)

    Fecha      datetime64[ns]
    Cierre      float64
    Apertura    float64
    Máximo      float64
    Mínimo      float64
    % var.      float64
dtype: object

```

6890 6

Una vez ordenada la información como la necesitamos, empezamos a observar el comportamiento de los datos graficando.

```
In [5]: import matplotlib.pyplot as plt
        from numpy import asarray
        ejemplo = historico['Cierre']
        #print(ejemplo)
        x = range(len(ejemplo))
        plt.plot(x, asarray(ejemplo))
        plt.xticks(x, "04/01/1993", rotation='vertical', fontsize=1)
        plt.title('Variación del precio del dolar', fontsize = 20)
        plt.xlabel('Días cotizados', fontsize = 20)
        plt.ylabel('Precio del dolar', fontsize = 20)

Out[5]: Text(0, 0.5, 'Precio del dolar')
```



1.3 Pronóstico de Holt-Winters

El modelo Holt-Winters incorpora un conjunto de procedimientos que conforman el núcleo de la familia de series temporales de suavizado exponencial. A diferencia de muchas otras técnicas, el modelo Holt-Winters puede adaptarse fácilmente a cambios y tendencias, así como a patrones

estacionales. En comparación con otras técnicas, como ARIMA, el tiempo necesario para calcular el pronóstico es considerablemente más rápido. Más allá de sus características técnicas, su aplicación en entornos de negocio es muy común. De hecho, Holt-Winters se utiliza habitualmente por muchas compañías para pronosticar la demanda a corto plazo cuando los datos de venta contienen tendencias y patrones estacionales de un modo subyacente.

Ahora se aplica el pronóstico de un paso de Holt

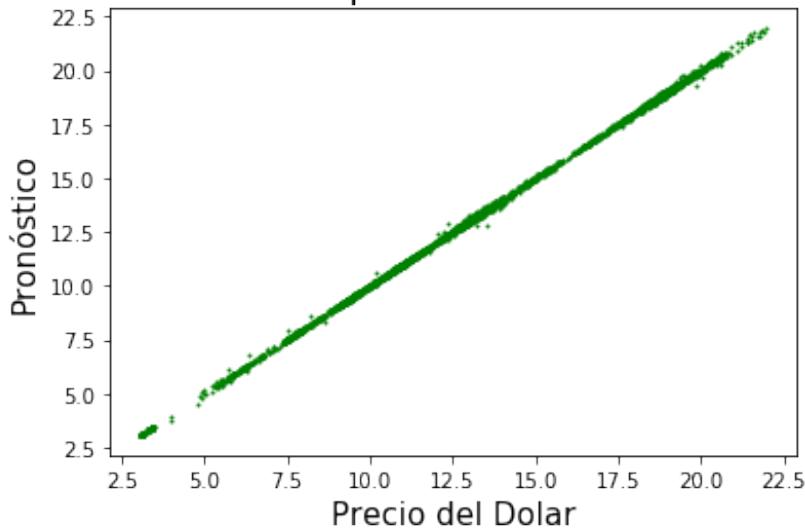
```
In [7]: from statsmodels.tsa.api import Holt

historico = historico[['Cierre', 'Apertura', 'Máximo', 'Mínimo']]

d = historico
print(d.columns)
lbls = historico.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('Precio del Dolar', fontsize = 15)
plt.ylabel('Pronóstico', fontsize = 15)
plt.scatter(historico.Cierre, pronosticos, c = 'g', s = 1)
plt.show()

Index(['Cierre', 'Apertura', 'Máximo', 'Mínimo'], dtype='object')
```

Pronóstico de un paso con el método de Holt



Parece que la información está en una línea recta de pendiente igual a uno, por la cantidad de los datos parece que el pronostico acierta al valor real, y además están fuertemente correlacionados.

1.4 Prueba de Dickey-Fuller aumentada

La prueba de Dickey-Fuller aumentada (ADF) es una prueba de raíz unitaria para una muestra de una serie de tiempo. Es una versión aumentada de la prueba Dickey-Fuller para un conjunto más amplio y más complejo de modelos de series de tiempo. La estadística Dickey-Fuller Aumentada (ADF), utilizada en la prueba, es un número negativo. Cuanto más negativo es, más fuerte es el rechazo de la hipótesis nula de que existe una raíz unitaria para un cierto nivel de confianza.

Ahora se aplica la prueba de Dickey-Fuller aumentada

In [8]: `from statsmodels.tsa.stattools import adfuller`

```
# rutina de https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes
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')
    dftest = adfuller(ts, autolag='BIC')
    #dftest = adfuller(ts)
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used',''])
    for key,value in dftest[4].items():
        dfoutput['Critical Value {:.3f}'.format(key)] = value
    return '\n\nResults of Dickey-Fuller Test:\n' + '\n'.join(['{:s}\t{:.3f}'.format(k,v) for k,v in dfoutput.items()])
plt.rcParams["figure.figsize"] = [16, 10]
f = plt.figure()
i = 1
lvls = historico.columns
r = len(lvls)
print(r)
t = ''
w = 26
for c in lvls:
    #print(historico[c])
    t += test_stationarity(historico[c], w, r, i)
    i += 1
plt.plot()
print(t)
```

Results of Dickey-Fuller Test:

Test Statistic -1.068
p-value 0.728
#Lags Used 1.000
Number of Observations Used 6888.000
Critical Value (1%) -3.431
Critical Value (5%) -2.862
Critical Value (10%) -2.567

Results of Dickey-Fuller Test:

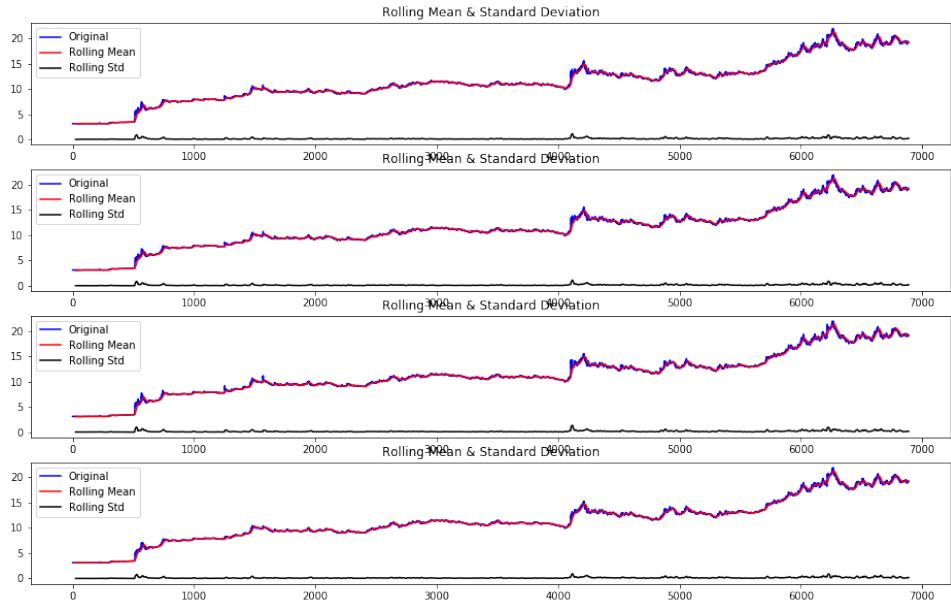
Test Statistic -1.070
p-value 0.727
#Lags Used 1.000
Number of Observations Used 6888.000
Critical Value (1%) -3.431
Critical Value (5%) -2.862
Critical Value (10%) -2.567

Results of Dickey-Fuller Test:

Test Statistic -1.101
p-value 0.715
#Lags Used 3.000
Number of Observations Used 6886.000
Critical Value (1%) -3.431
Critical Value (5%) -2.862
Critical Value (10%) -2.567

Results of Dickey-Fuller Test:

Test Statistic -1.013
p-value 0.749
#Lags Used 8.000
Number of Observations Used 6881.000
Critical Value (1%) -3.431
Critical Value (5%) -2.862
Critical Value (10%) -2.567



Se intenta restar de cada valor el valor correspondiente un mes antes (hace 30 días)

```
In [9]: from pandas import Series
def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return Series(diff)

f = plt.figure()
i = 1
for c in ['Cierre', 'Apertura', 'Máximo', 'Mínimo']:
    ts = difference(asarray(d[c]), interval = 30)
    ts = ts.dropna()
    plt.subplot(4, 1, i)
    plt.title(c)
    plt.plot(ts)
    i += 1
result = adfuller(ts)
print('{:s}\nADF Stat{:3f}\nvalue{:3f}\n'.format(c, result[0], result[1]))
for key, value in result[4].items():
    print('\t{:s}: {:.3f}'.format(key, value))
plt.tight_layout()
```

Cierre

ADF Stat -9.539
p-value 0.000

1%: -3.431
5%: -2.862
10%: -2.567

Apertura

ADF Stat -9.549
p-value 0.000

1%: -3.431
5%: -2.862
10%: -2.567

Máximo

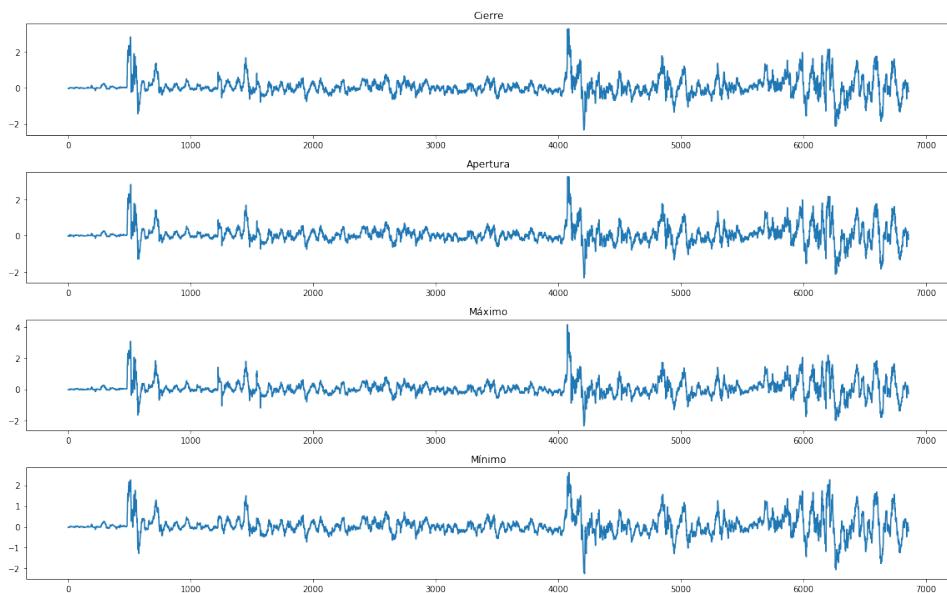
ADF Stat -9.324
p-value 0.000

1%: -3.431
5%: -2.862
10%: -2.567

Mínimo

ADF Stat -9.413
p-value 0.000

1%: -3.431
5%: -2.862
10%: -2.567



Según la prueba estadística los cuatro factores ya quedaron listos para poder pronosticar. Todos tienen el valor de p=0, entonces todos tienen valor q=1.

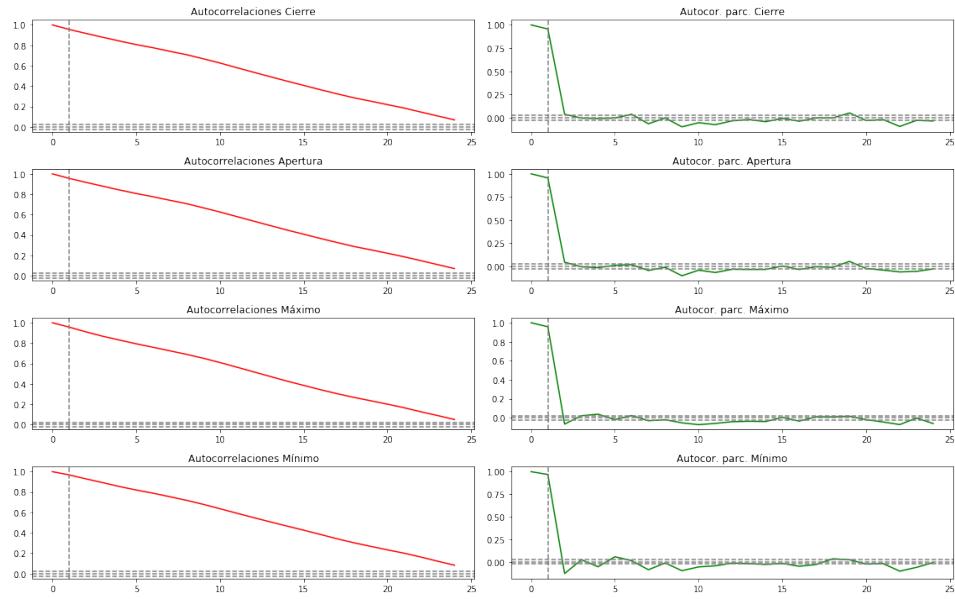
1.5 ARIMA

Un modelo autorregresivo integrado de promedio móvil o ARIMA (acrónimo del inglés autoregressive integrated moving average) es un modelo estadístico que utiliza variaciones y regresiones de datos estadísticos con el fin de encontrar patrones para una predicción hacia el futuro. Se trata de un modelo dinámico de series temporales, es decir, las estimaciones futuras vienen explicadas por los datos del pasado y no por variables independientes.

Se aplica un ARIMA sobre los datos.

```
In [10]: from statsmodels.tsa.stattools import acf, pacf
         from math import sqrt

         f = plt.figure()
         i = 1
         n = len(d)
         q = {'Cierre': 1, 'Apertura': 1, 'Máximo': 1, 'Mínimo': 1}
         p = {'Cierre': 1, 'Apertura': 1, 'Máximo': 1, 'Mínimo': 1}
         for c in ['Cierre', 'Apertura', 'Máximo', 'Mínimo']:
             ts = difference(asarray(d[c]), interval = 26).dropna()
             # manera del primer tutorial
             plt.subplot(4, 2, 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, 2, 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)
         plt.tight_layout()
```



Entonces, siguiendo el procedimiento del primer [tutorial](#), buscando la coordenada x del primer cruce con el intervalo de confianza superior, se obtiene $q = \{'Cierre': 1, 'Apertura': 1, 'Máximo': 1, 'Mínimo': 1\}$ y $p = \{'Cierre': 1, 'Apertura': 1, 'Máximo': 1, 'Mínimo': 1\}$

1.6 Pronóstico

Aplicando la información obtenida podemos pronosticar el valor del dólar el 8 de abril de 2019.

```
In [11]: from statsmodels.tsa.arima_model import ARIMA
        import pandas as pd
        import ssl

# https://machinelearningmastery.com/time-series-forecast-study-python-monthly-sales-
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

guardar = d.iloc[-1:, :]
d = d.iloc[:-1, :]
n = len(d)
```

```

q = {'Cierre': 1, 'Apertura': 1, 'Máximo': 1, 'Mínimo': 1}
p = {'Cierre': 1, 'Apertura': 1, 'Máximo': 1, 'Mínimo': 1}
prons = []
for c in ['Cierre', 'Apertura', 'Máximo', 'Mínimo']:
    ts = difference(asarray(d[c]), interval = 30).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 - 1])))
        print('{:s}\treal\t'.format(c), guardar[c].iloc[0])
    except:
        print(c, "no se pudo pronosticar con esos parámetros")
    print('...')

Cierre      pron.      19
Cierre      real     19.1021
...
Apertura    pron.      19
Apertura    real     19.0733
...
Máximo      pron.      19
Máximo      real     19.1108
...
Mínimo      pron.      19
Mínimo      real     19.0716
...

```

In [3]: `from IPython.display import Image
Image(filename='Dolar8Abril2019.jpeg')`

Out [3] :



Podemos observar en esta imagen que el modelo de pronostico tuvo un GAP por factor correspondiente a la siguiente tabla.

Parámetro	Modelo real	Realidad	GAP
Cierre	19.1021	18.9675	-0.7%
Apertura	19.0733	19.0755	+0.01%
Máximo	19.1108	19.1367	+0.13%
Mínimo	19.0716	18.9135	-0.83%

2 Conclusión

La cantidad información obtenida nos permitió hacer un pronostico acertado. Al aplicar correctamente los métodos de pronósticos, llegamos a valores muy cercanos a la realidad.

--05 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 10: Clasificación de datos

Complicaciones

CIENCIA_DE_DATOS (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master)
/ P10.ipynb (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master/P10.ipynb)

Reporte de Práctica 10: Clasificación de datos con sklearn

Para esta práctica solamente trabajamos con los datos del año 2017 para clasificar la categoría a la que pertenecen los cortos tomando como base la edad y sexo del concursante y el género del corto. Utilizaremos los [clasificadores de scikit-learn](https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html) (https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html) y la distribución de los datos que vamos a usar serán 60% para entrenar y 40% para validar.

Objetivos

- Utiliza por lo menos tres distintos métodos de clasificación
- Por lo menos una división de interés en tus datos

Preparación de los datos

Primero tomamos los archivos originales y los procesamos fuera de la nube, producto de esta limpieza se generó el archivo "clasificacion2017.csv"

Para poder trabajar importaremos las librerías necesarias y cargaremos el documento .csv

```
In [1]: import pandas as pd
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.datasets import make_moons, make_circles, make_classification
from sklearn.neural_network import MLPClassifier
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
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

df = pd.read_csv("https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/clasificacion2017.csv")

/usr/lib/python2.7/dist-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning:
numpy.core.umath_tests is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
    from numpy.core.umath_tests import inner1d
```

Categorización de los campos

Como tenemos campos con cadenas de caracteres utilizaremos la Categorización por defecto de pandas, y generaremos una columna con la etiqueta que utilizaremos para la clasificación.

```
In [11]: gen = pd.Categorical(df['Género'])
df['Género'] = gen.codes

pai = pd.Categorical(df['País'])
df['País'] = pai.codes

sex = pd.Categorical(df.Sexo)
df.Sexo = sex.codes

df['etiquetas'] = [1 if df['Categoría'][i] == 'Juvenil' else 0 for i in df['Categoría'].keys()]# Clasificar Categorías
print(df.etiquetas.value_counts())

File "<ipython-input-11-dbf01bc04159>", line 1
    gen = pd.Categorical(df.Género)
                           ^
SyntaxError: invalid syntax
```

Preparamos las variables que necesitamos para preparar el clasificador y además aplicamos un PCA, como debió haberse hecho en la práctica 8.

```
In [5]: y = df.etiquetas

xVars = ['Edad', 'Sexo', 'Género']
x = df.loc[:, xVars].values
x = StandardScaler().fit_transform(x)
pca = PCA(n_components = 2) # pedimos uno bidimensional
X = pca.fit_transform(x)

-----
AttributeError                                     Traceback (most recent call last)
<ipython-input-5-d266a2929f96> in <module>()
----> 1 y = df.etiquetas
      2
      3 xVars = ['Edad', 'Sexo', 'Género']
      4 x = df.loc[:, xVars].values
      5 x = StandardScaler().fit_transform(x)

/home/samataroukami/.local/lib/python2.7/site-packages/pandas/core/generic.py in __getattr__
(self, name)
  5065         if self._info_axis._can_hold_identifiers_and_holds_name(name):
  5066             return self[name]
-> 5067         return object.__getattribute__(self, name)
  5068
  5069     def __setattr__(self, name, value):
```

AttributeError: 'DataFrame' object has no attribute 'etiquetas'

```
In [7]: xVars = ['Edad', 'Sexo', 'Género']
x = df.loc[:, xVars].values

pca = PCA(n_components = 2) # pedimos uno bidimensional
X = pca.fit_transform(x)
from math import ceil, sqrt
from numpy import isnan, nan, arange, meshgrid, c_
import matplotlib.pyplot as plt
h=0.2
# código de https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
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) # 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()])
    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()

-----
ValueError Traceback (most recent call last)
<ipython-input-7-e060a1c9522c> in <module>()
      4
      5 pca = PCA(n_components = 2) # pedimos uno bidimensional
--> 6 X = pca.fit_transform(x)
      7 from math import ceil, sqrt
      8 from numpy import isnan, nan, arange, meshgrid, c_
/usr/lib/python2.7/dist-packages/scikit-learn/decomposition/pca.py in fit_transform(self, X, y)
  346
  347      """
--> 348      U, S, V = self._fit(X)
  349      U = U[:, :self.n_components_]
  350

/usr/lib/python2.7/dist-packages/scikit-learn/decomposition/pca.py in _fit(self, X)
  368
  369      X = check_array(X, dtype=[np.float64, np.float32], ensure_2d=True,
--> 370                  copy=self.copy)
  371
  372      # Handle n_components==None

/usr/lib/python2.7/dist-packages/scikit-learn/utils/validation.py in check_array(array, accept_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, warn_on_dtype, estimator)

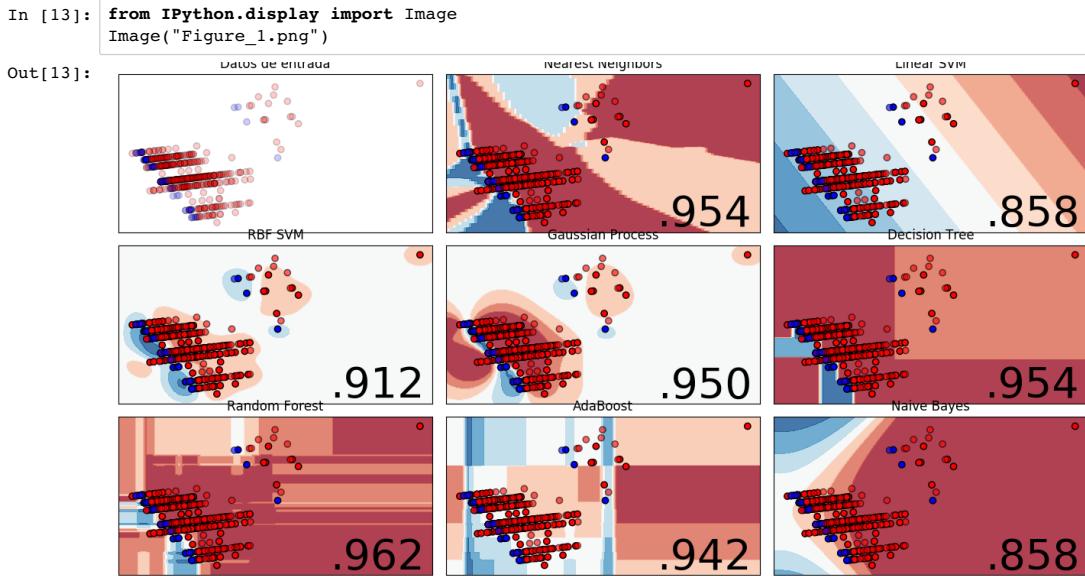
```

```

431                     force_all_finite)
432     else:
--> 433         array = np.array(array, dtype=dtype, order=order, copy=copy)
434
435     if ensure_2d:
ValueError: invalid literal for float(): 47

```

Como se puede ver aqui en el sklearn no funcionó con jupyter notebook asi que decidí hacer el codigo en un block de notas y corrí todo en la terminal de Ubuntu 18 obteniendo los siguientes graficos con diferentes metodos



Ahora calcularemos las matrices de confusión. Este fue el código que utilicé:

```

In [ ]: # código de https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
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=.4, 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)

```

Obteniendo los siguientes resultados

```
/home/samataroukami/.local/lib/python3.6/site-packages/sklearn/externals/joblib/externals/cloudpickle/cloudpickle.py:47:  
DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses  
import imp 0 521 1 128 Name: etiquetas, dtype: int64 /home/samataroukami/.local/lib/python3.6/site-  
packages/sklearn/utils/validation.py:590: DataConversionWarning: Data with input dtype object was converted to float64 by  
StandardScaler. warnings.warn(msg, DataConversionWarning) /home/samataroukami/.local/lib/python3.6/site-  
packages/sklearn/utils/validation.py:590: DataConversionWarning: Data with input dtype object was converted to float64 by  
StandardScaler. warnings.warn(msg, DataConversionWarning) Nearest Neighbors 0.9538461538461539 precision recall f1-score  
support
```

0	0.99	0.96	0.97	223
1	0.79	0.92	0.85	37

micro avg 0.95 0.95 0.95 260 macro avg 0.89 0.94 0.91 260 weighted avg 0.96 0.95 0.96 260

[[214 9]

[3 34]]

```
Linear SVM 0.8576923076923076 /home/samataroukami/.local/lib/python3.6/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') precision recall f1-score support
```

0	0.86	1.00	0.92	223
1	0.00	0.00	0.00	37

micro avg 0.86 0.86 0.86 260 macro avg 0.43 0.50 0.46 260 weighted avg 0.74 0.86 0.79 260

[[223 0]

[37 0]]

RBF SVM 0.9115384615384615 precision recall f1-score support

0	0.99	0.91	0.95	223
1	0.63	0.92	0.75	37

micro avg 0.91 0.91 0.91 260 macro avg 0.81 0.91 0.85 260 weighted avg 0.93 0.91 0.92 260

[[203 20]

[3 34]]

```
/home/samataroukami/.local/lib/python3.6/site-packages/sklearn/gaussian_process/gpc.py:434: ConvergenceWarning:  
fmin_l_bfgs_b terminated abnormally with the state: {'grad': array([-1.45960984, 8.20994214]), 'task':  
b'ABNORMAL_TERMINATION_IN_LNSRCH', 'funcalls': 184, 'nit': 14, 'warnflag': 2} ConvergenceWarning) Gaussian Process  
0.9615384615384616 precision recall f1-score support
```

0	1.00	0.96	0.98	223
1	0.79	1.00	0.88	37

micro avg 0.96 0.96 0.96 260 macro avg 0.89 0.98 0.93 260 weighted avg 0.97 0.96 0.96 260

[[213 10]

[0 37]]

Decision Tree 0.9538461538461539 precision recall f1-score support

0	0.99	0.96	0.97	223
1	0.79	0.92	0.85	37

micro avg 0.95 0.95 0.95 260 macro avg 0.89 0.94 0.91 260 weighted avg 0.96 0.95 0.96 260

[[214 9]

[3 34]]

Random Forest 0.9615384615384616 precision recall f1-score support

0	1.00	0.96	0.98	223
1	0.80	0.97	0.88	37

micro avg 0.96 0.96 0.96 260 macro avg 0.90 0.97 0.93 260 weighted avg 0.97 0.96 0.96 260

[[214 9]

[1 36]]

AdaBoost 0.9423076923076923 precision recall f1-score support

0	0.97	0.96	0.97	223
1	0.78	0.84	0.81	37

micro avg 0.94 0.94 0.94 260 macro avg 0.87 0.90 0.89 260 weighted avg 0.94 0.94 0.94 260

[[214 9]

[6 31]]

Naive Bayes 0.8576923076923076 /home/samataroukami/.local/lib/python3.6/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) precision recall f1-score support

0	0.86	1.00	0.92	223
1	0.00	0.00	0.00	37

micro avg 0.86 0.86 0.86 260 macro avg 0.43 0.50 0.46 260 weighted avg 0.74 0.86 0.79 260

[[223 0]

[37 0]]

Conclusión

Como se puede apreciar en la gráfica y en las matrices de confusión los 3 mejores métodos para clasificar los datos son

- Random Forest
- Nearest Neighbors
- Gaussian Process

obteniendo valores por encima del 95% de precisión.

--29 de Abril 2019-- Luis Angel Gutierrez Rodriguez [1484412](#) (tel:1484412)

P10

June 6, 2019

1 Reporte de práctica 10: Clasificación de datos con sklearn

Para esta práctica solamente trabajamos con los datos del año 2017 para clasificar la categoría a la que pertenecen los cortos tomando como base la edad y sexo del concursante y el género del corto. Utilizaremos los [clasificadores de scikit-learn](#) y la distribución de los datos que vamos a usar serán 60% para entrenar y 40% para validar.

1.1 Objetivos

- Utiliza por lo menos tres distintos métodos de clasificación
- Por lo menos una división de interés en tus datos

1.2 Preparación de los datos

Primero tomamos los archivos originales y los procesamos fuera de la nube, producto de esta limpieza se generó el archivo "clasificacion2017.csv"

Para poder trabajar importaremos la librería necesaria y cargaremos el documento .csv

```
In [7]: import pandas as pd
        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.datasets import make_moons, make_circles, make_classification
        from sklearn.neural_network import MLPClassifier
        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
        from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

df = pd.read_csv("https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/clasificacion2017.csv")
```

```
df = df.dropna()
print(len(df))
```

649

1.3 Categorización de los campos

Como tenemos campos con cadenas de caracteres utilizaremos la Categorización por defecto de pandas, y generaremos una columna con la etiqueta que utilizaremos para la clasificación.

```
In [8]: gen = pd.Categorical(df.Genero)
df.Genero = gen.codes

pai = pd.Categorical(df.Pais)
df.Pais = pai.codes

sex = pd.Categorical(df.Sexo)
df.Sexo = sex.codes

#cat = pd.Categorical(df.Categoría)
#df['CategoríaCat'] = cat.codes

df['etiquetas1'] = [1 if df['Categoría'][i] == 'Aficionado' else 0 for i in df['Categoría']]
df['etiquetas2'] = [1 if df['Categoría'][i] == 'Juvenil' else 0 for i in df['Categoría']]
df['etiquetas3'] = [1 if df['Categoría'][i] == 'Profesional' else 0 for i in df['Categoría']]
df['etiquetas4'] = [1 if df['Categoría'][i] == 'SmarTIC' else 0 for i in df['Categoría']]

#df['etiquetas'] = df.CategoríaCat
print(df.etiquetas1.value_counts())
print(df.etiquetas2.value_counts())
print(df.etiquetas3.value_counts())
print(df.etiquetas4.value_counts())

1    362
0    287
Name: etiquetas1, dtype: int64
0    521
1    128
Name: etiquetas2, dtype: int64
0    573
1     76
Name: etiquetas3, dtype: int64
0    566
1     83
Name: etiquetas4, dtype: int64
```

1.4 Procedimiento

Se preparan las variables que necesitamos para preparar el clasificador y además aplicamos un PCA. Se separan los datos y se pone el 60% para entrenamiento del clasificador y el 40% para pruebas. Se busca clasificar la categoría de participación del filme, donde cada categoría es clasificada individualmente, y no en grupo, para buscar particularidades en los datos.

Después se muestran las tablas de confusión para entender mejor porque les otorgaron esas precisiones a los diferentes métodos de clasificación.

```
In [12]: for indexEtiqueta in range(1,5):
    print("-----Iniciando Proceso con Etiqueta {:d}-----".format(indexEtiqueta))
    y = df['etiquetas'+str(indexEtiqueta)]

    xVars = ['Edad', 'Sexo', 'Genero']
    x = df.loc[:, xVars].values
    #x = StandardScaler().fit_transform(x)
    pca = PCA(n_components = 2) # pedimos uno bidimensional
    X = pca.fit_transform(x)

    from math import ceil, sqrt
    from numpy import isnan, nan, arange, meshgrid, c_
    import matplotlib.pyplot as plt
    h=0.2
    # código de https://scikit-learn.org/stable/auto_examples/classification/plot_cla
    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_es\
                   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_st
    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, edg
    ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, alpha=0.2, edgeco
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    i = 2
```

```

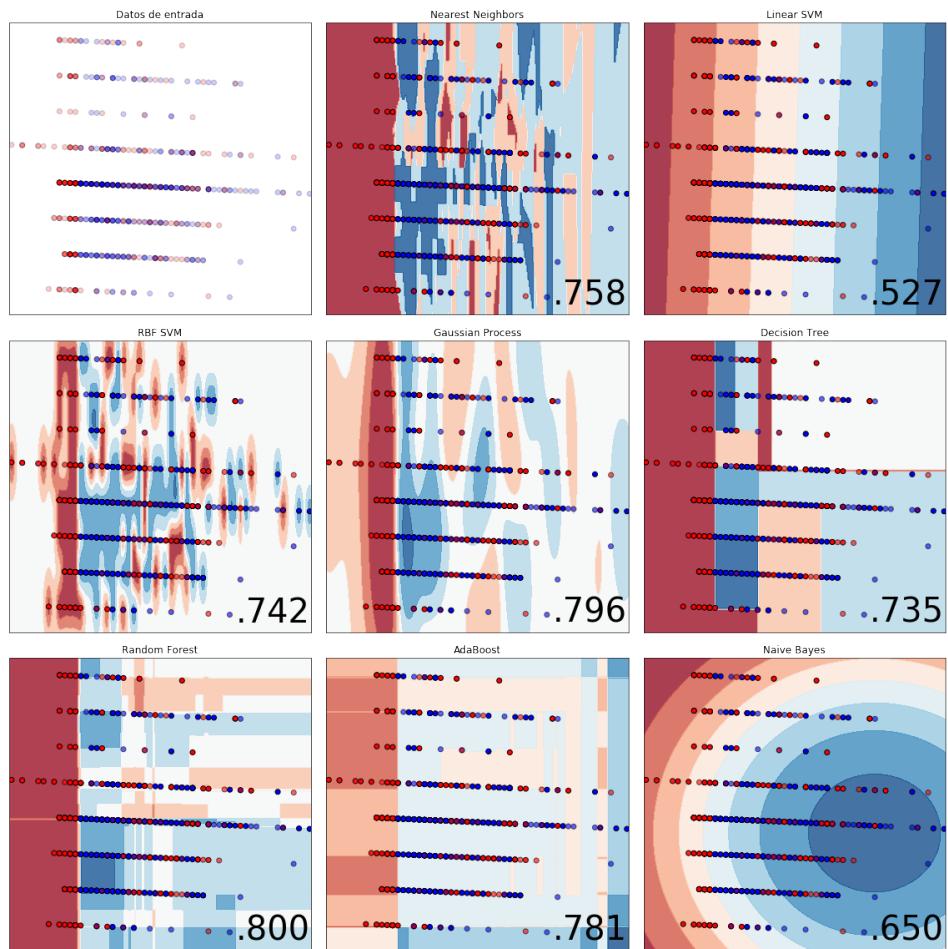
for name, clf in zip(names, classifiers):
    ax = plt.subplot(k, k, i)
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    if hasattr(clf, "decision_function"):
        Z = clf.decision_function(c_[xx.ravel(), yy.ravel()])
    else:
        Z = clf.predict_proba(c_[xx.ravel(), yy.ravel()])[:, 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')
    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', verticalalignment='bottom')
    i += 1
plt.tight_layout()
plt.show()

print("Ahora calcularemos las matrices de confusión para la etiqueta{:d}.".format(i))

# código de https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
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), \
               AdaBoostClassifier(), GaussianNB()]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.4, 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)
print("-----Iniciando Proceso con Etiqueta 1-----")

```



Ahora calcularemos las matrices de confusión para la etiqueta 1.
 ('Nearest Neighbors', 0.7576923076923077)

	precision	recall	f1-score	support
0	0.68	0.66	0.67	96
1	0.80	0.82	0.81	164
avg / total	0.76	0.76	0.76	260

```
[[ 63  33]
 [ 30 134]]
```

('Linear SVM', 0.5269230769230769)

	precision	recall	f1-score	support
0	0.41	0.61	0.49	96
1	0.68	0.48	0.56	164
avg / total	0.58	0.53	0.53	260

[[59 37]
[86 78]]

	precision	recall	f1-score	support
0	0.67	0.60	0.63	96
1	0.78	0.82	0.80	164
avg / total	0.74	0.74	0.74	260

[[58 38]
[29 135]]

	precision	recall	f1-score	support
0	0.80	0.59	0.68	96
1	0.79	0.91	0.85	164
avg / total	0.80	0.80	0.79	260

[[57 39]
[14 150]]

	precision	recall	f1-score	support
0	0.64	0.65	0.64	96
1	0.79	0.79	0.79	164
avg / total	0.74	0.73	0.73	260

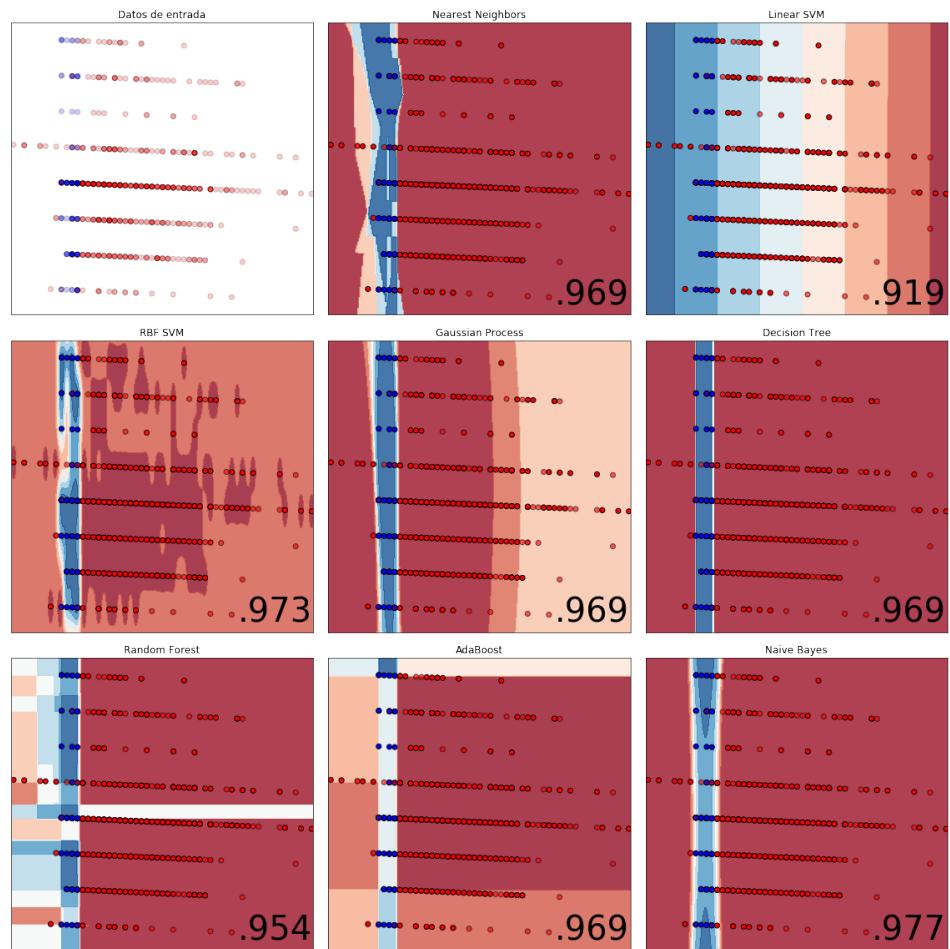
[[62 34]
[35 129]]

	precision	recall	f1-score	support
0	0.76	0.59	0.67	96
1	0.79	0.89	0.84	164

```

avg / total      0.78      0.78      0.77      260
[[ 57  39]
 [ 18 146]]
-----
('AdaBoost', 0.7807692307692308)
      precision    recall   f1-score   support
          0       0.75      0.61      0.67       96
          1       0.80      0.88      0.83      164
avg / total      0.78      0.78      0.78      260
[[ 59  37]
 [ 20 144]]
-----
('Naive Bayes', 0.65)
      precision    recall   f1-score   support
          0       0.52      0.60      0.56       96
          1       0.74      0.68      0.71      164
avg / total      0.66      0.65      0.65      260
[[ 58  38]
 [ 53 111]]
-----
%%%%%%%%%%%%%
-----Iniciando Proceso con Etiqueta 2-----

```



Ahora calcularemos las matrices de confusión para la etiqueta2.
 ('Nearest Neighbors', 0.9692307692307692)

	precision	recall	f1-score	support
0	1.00	0.96	0.98	223
1	0.82	1.00	0.90	37
avg / total	0.97	0.97	0.97	260

```
[[215  8]
 [ 0  37]]
```

('Linear SVM', 0.9192307692307692)

	precision	recall	f1-score	support
0	1.00	0.91	0.95	223
1	0.64	1.00	0.78	37
avg / total	0.95	0.92	0.93	260

[[202 21]
[0 37]]

	precision	recall	f1-score	support
0	0.99	0.98	0.98	223
1	0.88	0.95	0.91	37
avg / total	0.97	0.97	0.97	260

[[218 5]
[2 35]]

	precision	recall	f1-score	support
0	0.99	0.97	0.98	223
1	0.85	0.95	0.90	37
avg / total	0.97	0.97	0.97	260

[[217 6]
[2 35]]

	precision	recall	f1-score	support
0	0.99	0.97	0.98	223
1	0.85	0.95	0.90	37
avg / total	0.97	0.97	0.97	260

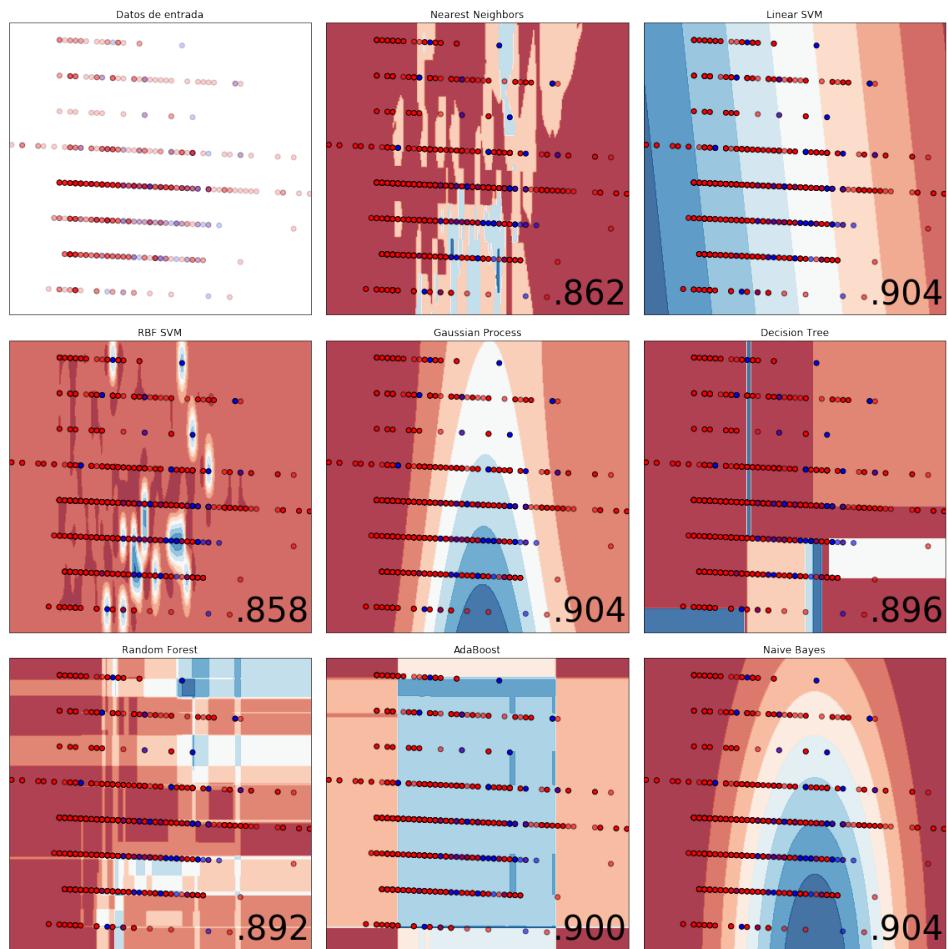
[[217 6]
[2 35]]

	precision	recall	f1-score	support
0	0.99	0.97	0.98	223
1	0.85	0.95	0.90	37

```

avg / total      0.97      0.97      0.97      260
[[217   6]
 [ 2  35]]
-----
('AdaBoost', 0.9692307692307692)
      precision    recall   f1-score   support
          0       0.99      0.97      0.98      223
          1       0.85      0.95      0.90       37
avg / total      0.97      0.97      0.97      260
[[217   6]
 [ 2  35]]
-----
('Naive Bayes', 0.9769230769230769)
      precision    recall   f1-score   support
          0       1.00      0.97      0.99      223
          1       0.86      1.00      0.92       37
avg / total      0.98      0.98      0.98      260
[[217   6]
 [ 0  37]]
-----
%%%%%%%%%%%%%
-----Iniciando Proceso con Etiqueta 3-----

```



Ahora calcularemos las matrices de confusión para la etiqueta3.
 ('Nearest Neighbors', 0.8615384615384616)

	precision	recall	f1-score	support
0	0.91	0.94	0.92	235
1	0.21	0.16	0.18	25
avg / total	0.85	0.86	0.85	260

```
[ [220 15]
 [ 21  4]]
```

('Linear SVM', 0.9038461538461539)

	precision	recall	f1-score	support
0	0.90	1.00	0.95	235
1	0.00	0.00	0.00	25
avg / total	0.82	0.90	0.86	260

[[235 0]
[25 0]]

	precision	recall	f1-score	support
0	0.90	0.94	0.92	235
1	0.07	0.04	0.05	25
avg / total	0.82	0.86	0.84	260

[[222 13]
[24 1]]

	precision	recall	f1-score	support
0	0.90	1.00	0.95	235
1	0.00	0.00	0.00	25
avg / total	0.82	0.90	0.86	260

[[235 0]
[25 0]]

	precision	recall	f1-score	support
0	0.91	0.98	0.94	235
1	0.33	0.08	0.13	25
avg / total	0.85	0.90	0.87	260

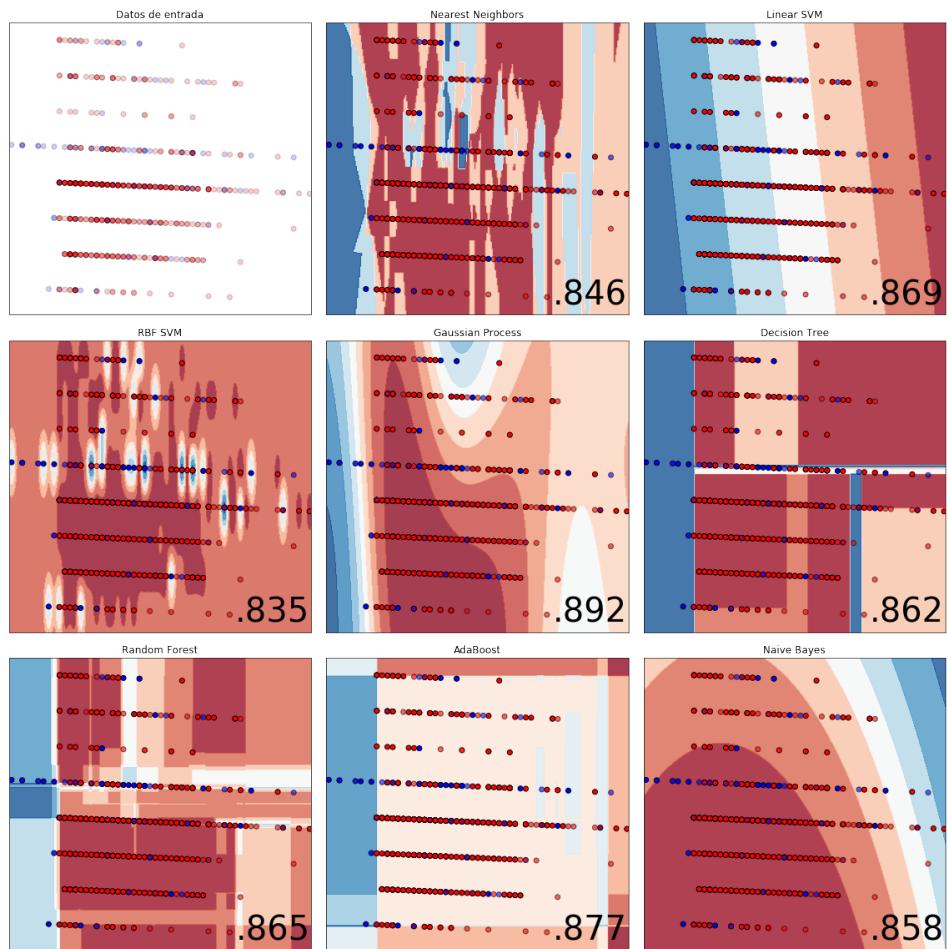
[[231 4]
[23 2]]

	precision	recall	f1-score	support
0	0.91	0.95	0.93	235
1	0.21	0.12	0.15	25

```

avg / total      0.84      0.87      0.86      260
[[224  11]
 [ 22   3]]
-----
('AdaBoost', 0.9)
      precision    recall   f1-score   support
          0       0.90     1.00      0.95      235
          1       0.00     0.00      0.00       25
avg / total      0.82      0.90      0.86      260
[[234   1]
 [ 25   0]]
-----
('Naive Bayes', 0.9038461538461539)
      precision    recall   f1-score   support
          0       0.90     1.00      0.95      235
          1       0.00     0.00      0.00       25
avg / total      0.82      0.90      0.86      260
[[235   0]
 [ 25   0]]
-----
%%%%%%%%%%%%%
-----Iniciando Proceso con Etiqueta 4-----

```



Ahora calcularemos las matrices de confusión para la etiqueta4.
 ('Nearest Neighbors', 0.8461538461538461)

	precision	recall	f1-score	support
0	0.89	0.94	0.91	226
1	0.36	0.24	0.29	34
avg / total	0.82	0.85	0.83	260

```
[ [212 14]
 [ 26  8]]
```

('Linear SVM', 0.8692307692307693)

	precision	recall	f1-score	support
0	0.87	1.00	0.93	226
1	0.00	0.00	0.00	34
avg / total	0.76	0.87	0.81	260
	[[226 0]			
	[34 0]]			

	precision	recall	f1-score	support
0	0.87	0.95	0.91	226
1	0.20	0.09	0.12	34
avg / total	0.79	0.83	0.81	260
	[[214 12]			
	[31 3]]			

	precision	recall	f1-score	support
0	0.89	1.00	0.94	226
1	1.00	0.18	0.30	34
avg / total	0.90	0.89	0.86	260
	[[226 0]			
	[28 6]]			

	precision	recall	f1-score	support
0	0.90	0.95	0.92	226
1	0.45	0.29	0.36	34
avg / total	0.84	0.86	0.85	260
	[[214 12]			
	[24 10]]			

	precision	recall	f1-score	support
0	0.89	0.96	0.92	226
1	0.43	0.18	0.25	34

```

avg / total      0.83      0.86      0.84      260
[[218  8]
 [ 28  6]]
-----
('AdaBoost', 0.8769230769230769)
      precision    recall   f1-score   support
          0       0.90      0.97      0.93      226
          1       0.56      0.26      0.36       34
avg / total      0.85      0.88      0.86      260
[[219  7]
 [ 25  9]]
-----
('Naive Bayes', 0.8576923076923076)
      precision    recall   f1-score   support
          0       0.87      0.98      0.92      226
          1       0.20      0.03      0.05       34
avg / total      0.78      0.86      0.81      260
[[222  4]
 [ 33  1]]
-----
%%%%%%%%%%%%%

```

1.5 Conclusión

Después de probar los diferentes métodos y buscar clasificar todas las categorías de participación individualmente, se obtuvieron los siguientes resultados.

Método de Clasificación

Etiqueta1

Etiqueta2

Etiqueta3

Etiqueta4

Nearest Neighbors

.758

.969

.862

.846

Linear SVM

.527

.919

.904
.869
RBF SVM
.742
.973
.858
.835
Gaussian Process
.796
.969
.904
.892
Decision Tree
.735
.969
.896
.862
Random Forest
.8
.954
.892
.865
AdaBoost
.781
.969
.9
.877
Naive Bayes
.650
.977
.904
.858

Se puede concluir que con los datos de edad y sexo del participante y el género del Filme, se puede precisar si la participación del concursante es en la categoría Juvenil. En cambio, para identificar las otras categorías, los datos proporcionados otorgaron al clasificador una precisión a lo más del 79.6% para las demás categorías.

--05 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 11: Agrupamiento de datos

Complicaciones

CIENCIA_DE_DATOS (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master)
/ P11.ipynb (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master/P11.ipynb)

Reporte de Práctica 11: Agrupamiento de datos

Para esta práctica solamente trabajamos con los datos del año 2017, los mismos que utilizamos la práctica pasada, salvo que se encontró el error “47”, era un string con muchos tabuladores y ocasionó error de conversión a float. En ésta práctica realizaremos clusters con de la información, no proporcionaremos etiquetas para clasificar, dejaremos que los datos por su propia naturaleza y aplicando distintos algoritmos se agrupen ellos mismos.

Objetivos

- Utiliza por dos algoritmos de agrupamiento.
- Reportar los hallazgos con gráficas y con medidas de calidad.

Algoritmos de agrupamiento.

Utilizaremos los algoritmos de agrupamiento mostrados en [el tutorial de agrupamiento de scikit-learn \(https://scikit-learn.org/stable/modules/clustering.html\)](https://scikit-learn.org/stable/modules/clustering.html) y seleccionamos dos para aplicar. Usamos los siguientes:

- K-Means
- Affinity Propagation

Preparación de los datos

Primero tomamos los archivos originales y los procesamos fuera de la nube, producto de esta limpieza se generó el archivo "clasificacion2017.csv" en la práctica pasada.

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/clasificacion2017.csv")
print(len(df))
```

649

Algoritmo K-Means

Este algoritmo agrupa los datos al tratar de separar muestras en n grupos de igual varianza, minimizando un criterio conocido como la inercia o la suma de cuadrados dentro del grupo. Este algoritmo requiere que se especifique la cantidad de grupos. Se adapta bien a un gran número de muestras y se ha utilizado en una amplia gama de áreas de aplicación en muchos campos diferentes.

Aplicación de K-Means

```
In [2]: from sklearn import metrics
from numpy.random import seed
from sklearn.cluster import KMeans
from numpy import isnan, nan, take, where

d = pd.read_csv("https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/clasificacion2017.csv")

cat = pd.Categorical(d.Categoría)
d.Categoría = cat.codes

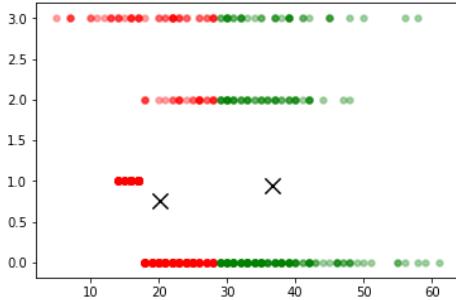
gen = pd.Categorical(d.Género)
d.Género = gen.codes

pais = pd.Categorical(d.Pais)
d.Pais = pais.codes

sex = pd.Categorical(d.Sexo)
d.Sexo = sex.codes

keep = ['Edad', 'Categoría', 'Sexo']
d = d.loc[:, keep]
d = d.dropna()
x = d.values
k = 2
m = KMeans(init = 'random', n_clusters = k, n_init = 10)
m.fit(x)
centroides = m.cluster_centers_
grupos = m.predict(x)
plt.figure(1)
plt.scatter(centroides[:, 0], centroides[:, 1], marker='x', s=150, linewidths=3, color='black', zorder=10)
colores = ['r', 'g', 'b']
for g in range(k):
    incl = where(grupos == g)[0]
    print(len(incl), "integrantes en grupo", g)
    grupo = take(x, incl, 0)
    plt.scatter(grupo[:, 0], grupo[:, 1], marker='o', s=20, linewidths=2, color=colores[g], alpha = 0.3, zorder=5)
plt.show()
print(metrics.silhouette_score(x, grupos, metric='euclidean'))
```

(428, 'integrantes en grupo', 0)
(221, 'integrantes en grupo', 1)



0.6081740309385313

Probemos con diferentes valores de k para ver cuál da el mejor valor de la coeficiente de silueta

```
In [3]: import pandas as pd
from sklearn import metrics
from numpy.random import seed
from numpy import isnan, nan, take, where
from sklearn.cluster import KMeans

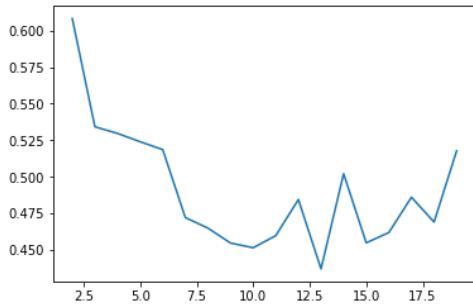
d = pd.read_csv("https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/clasificacion2017.csv")

cat = pd.Categorical(d.Categoría)
d.Categoría = cat.codes

sex = pd.Categorical(d.Sexo)
d.Sexo = sex.codes

keep = ['Edad', 'Categoría', 'Sexo']

d = d.loc[:, keep]
d = d.dropna()
x = d.values
ks = [k for k in range(2, 20)]
sil = []
for k in ks:
    m = KMeans(init = 'random', n_clusters = k, n_init = 10)
    m.fit(x)
    sil.append(metrics.silhouette_score(x, m.predict(x), metric='euclidean'))
plt.figure(1)
plt.plot(ks, sil)
plt.show()
```



En definitiva, la gráfica nos muestra que el hacer el agrupamiento en dos grupos es el que obtiene la mejor calidad de agrupamiento, ya que no vuelve a crecer la calidad si hacemos la división en más grupos.

Algoritmo AffinityPropagation

Este algoritmo crea clústeres enviando mensajes entre pares de muestras hasta la convergencia. Luego se describe un conjunto de datos utilizando un pequeño número de ejemplares, que se identifican como los más representativos de otras muestras. Los mensajes enviados entre pares representan la idoneidad para que una muestra sea el ejemplar de la otra, que se actualiza en respuesta a los valores de otros pares. Esta actualización ocurre de manera iterativa hasta la convergencia, momento en el que se eligen los ejemplares finales y, por lo tanto, se proporciona la agrupación final.

Aplicación de AffinityPropagation

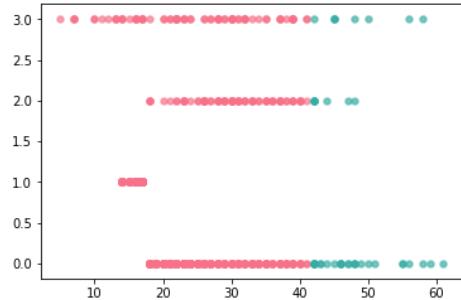
```
In [9]: import seaborn as sns
from numpy import take, where, unique, concatenate
from sklearn.cluster import AffinityPropagation
m = AffinityPropagation(damping = 0.9, convergence_iter = 30)

xVars = ['Edad', 'Categoria', 'Sexo']

d = d.loc[:, xVars]
x = d.values

c = m.fit(x)
grupos = c.labels_
plt.clf()
plt.figure(1)
k = len(unique(grupos))
colores = sns.color_palette("husl", k)
for g in range(k):
    incl = where(grupos == g)[0]
    print(len(incl), "integrantes en grupo", g)
    grupo = take(x, incl, 0)
    plt.scatter(grupo[:, 0], grupo[:, 1], marker='o', s=20, linewidths=2, color=colores[g], alpha = 0.6, zorder=5)
plt.show()
print(metrics.silhouette_score(x, grupos, metric='euclidean'))
```

(606, 'integrantes en grupo', 0)
(43, 'integrantes en grupo', 1)



0.5498546010295401

Conclusión

Al aplicar estos dos algoritmos de agrupamiento se puede notar que la mejor forma de agrupar estos datos es en dos grupos, aunque el K-Means tuvo una mejor calidad de solución al obtener 0.6081740309385313 y el AffinityPropagation obtuvo 0.5498546010295401. En K-Means se agrupan las edades en los de 29 años o menor, y los de 30 o mayor y en AffinityPropagation se agrupan en los de 41 años o menor y los de 42 o mayor.

-- de Mayo 2019-- Luis Angel Gutierrez Rodriguez [1484412](#) (tel:1484412)

P11

June 6, 2019

1 Reporte de práctica 11: Agrupamiento de datos

Para esta práctica solamente trabajamos con los datos del año 2017, los mismos que utilizamos la práctica pasada, salvo que se encontró el error "47", era un string con muchos tabuladores y ocasionó error de conversión a float. En esta práctica realizaremos clústeres con de la información, no proporcionaremos etiquetas para clasificar, dejaremos que los datos por su propia naturaleza y aplicando distintos algoritmos se agrupen ellos mismos.

1.1 Objetivos

- Utiliza por dos algoritmos de agrupamiento.
- Reportar los hallazgos con gráficas y con medidas de calidad.

1.2 Algoritmos de agrupamiento.

Utilizaremos los algoritmos de agrupamiento mostrados en [el tutorial de agrupamiento de scikit-learn](#) y seleccionamos dos para aplicar. Usamos los siguientes:

- K-Means
- Affinity Propagation

1.3 Preparación de los datos

Primero tomamos los archivos originales y los procesamos fuera de la nube, producto de esta limpieza se generó el archivo "clasificacion2017.csv" en la práctica pasada.

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt

        df0 = pd.read_excel('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/Clasificacion2017.xlsx')
        df0 = df0[['Categoria', 'Edad', 'Pais', 'Titulo', 'Genero', 'Duracion', 'Marca', 'Referencia', 'Año']]
        df0 = df0.dropna()
        print(len(df0))
```

266

1.4 Algoritmo K-Means

Este algoritmo agrupa los datos al tratar de separar muestras en n grupos de igual varianza, minimizando un criterio conocido como la inercia o la suma de cuadrados dentro del grupo. Este algoritmo requiere que se especifique la cantidad de grupos. Se adapta bien a un gran número de muestras y se ha utilizado en una amplia gama de áreas de aplicación en muchos campos diferentes.

1.4.1 Aplicación de K-Means

```
In [2]: from sklearn import metrics
        from numpy.random import seed
        from sklearn.cluster import KMeans
        from numpy import isnan, nan, take, where

df1 = pd.read_excel('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/1
df1 = df1[['Categoria', 'Edad', 'Pais', 'Titulo', 'Genero', 'Duracion', 'Marca', 'Referencia
df1 = df1.dropna()
print(len(df1))
d = df1

cat = pd.Categorical(d.Categoria)
d.Categoria = cat.codes

gen = pd.Categorical(d.Genero)
d.Genero = gen.codes

pais = pd.Categorical(d.Pais)
d.Pais = pais.codes

mar = pd.Categorical(d.Marca)
d.Marca = mar.codes

d['sin'] = d["Sinopsis"].str.len()

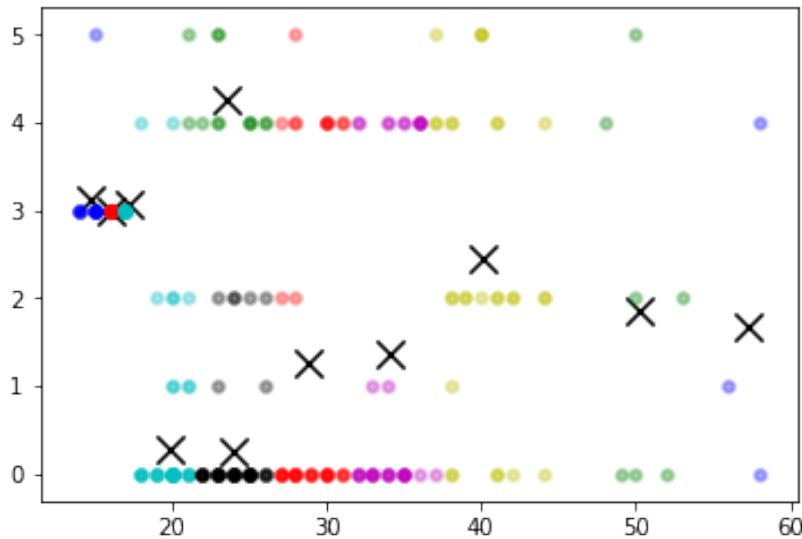
keep = ['Edad', 'Categoria']
d = d.loc[:, keep]
d = d.dropna()
x = d.values
k = 11
m = KMeans(init = 'random', n_clusters = k, n_init = 1)
m.fit(x)
centroides = m.cluster_centers_
grupos = m.predict(x)
plt.figure(1)
plt.scatter(centroides[:, 0], centroides[:, 1], marker='x', s=150, linewidths=3, color=co
colores = 3*['r', 'g', 'b', 'c', 'm', 'y', 'k']
for g in range(k):
```

```

        incl = where(grupos == g)[0]
        print(len(incl), "integrantes en grupo", g)
        grupo = take(x, incl, 0)
        plt.scatter(grupo[:, 0], grupo[:, 1], marker='o', s=20, linewidths=2, color=colores)
        plt.show()
        print(metrics.silhouette_score(x, grupos, metric='euclidean'))

266
(36, 'integrantes en grupo', 0)
(12, 'integrantes en grupo', 1)
(16, 'integrantes en grupo', 2)
(42, 'integrantes en grupo', 3)
(34, 'integrantes en grupo', 4)
(29, 'integrantes en grupo', 5)
(48, 'integrantes en grupo', 6)
(15, 'integrantes en grupo', 7)
(7, 'integrantes en grupo', 8)
(3, 'integrantes en grupo', 9)
(24, 'integrantes en grupo', 10)

```



0.5220206737202812

Probemos con diferentes valores de k para ver cuál da el mejor valor del coeficiente de silueta.

```

In [3]: import pandas as pd
        from sklearn import metrics
        from numpy.random import seed
        from numpy import isnan, nan, take, where
        from sklearn.cluster import KMeans

df2 = pd.read_excel('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/CIENCIA_DE_DATOS.xlsx')
df2 = df2[['Categoria','Edad','Pais', 'Titulo','Genero', 'Duracion', 'Marca','Referencia']]
df2 = df2.dropna()
print(len(df2))
d = df2

cat = pd.Categorical(d.Categoria)
d.Categoria = cat.codes

gen = pd.Categorical(d.Genero)
d.Genero = gen.codes

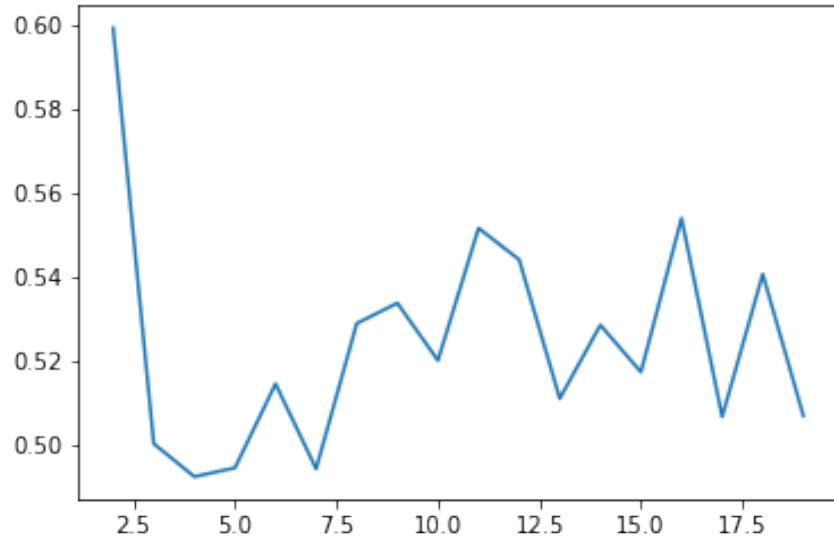
pai = pd.Categorical(d.Pais)
d.Pais = pai.codes

mar = pd.Categorical(d.Marca)
d.Marca = mar.codes

keep = ['Edad', 'Categoria']
d = d.loc[:, keep]
d = d.dropna()
x = d.values
ks = [k for k in range(2, 20)]
sil = []
for k in ks:
    m = KMeans(init = 'random', n_clusters = k, n_init = 10)
    m.fit(x)
    sil.append(metrics.silhouette_score(x, m.predict(x), metric='euclidean'))
plt.figure(1)
plt.plot(ks, sil)
plt.show()

```

266



En definitiva, la gráfica nos muestra que el hacer el agrupamiento en dos grupos es el que obtiene la mejor calidad de agrupamiento, tiene un ligero incremento al hacer separaciones de ocho grupos. Después ya que no vuelve a crecer la calidad si hacemos la división en más grupos.

1.5 Algoritmo Affinity Propagation

Este algoritmo crea clústeres enviando mensajes entre pares de muestras hasta la convergencia. Luego se describe un conjunto de datos utilizando un pequeño número de ejemplares, que se identifican como los más representativos de otras muestras. Los mensajes enviados entre pares representan la idoneidad para que una muestra sea el ejemplar de la otra, que se actualiza en respuesta a los valores de otros pares. Esta actualización ocurre de manera iterativa hasta la convergencia, momento en el que se eligen los ejemplares finales y, por lo tanto, se proporciona la agrupación final.

1.5.1 Aplicación de Affinity Propagation

```
In [4]: import seaborn as sns
        from numpy import take, where, unique, concatenate
        from sklearn.cluster import AffinityPropagation
        m = AffinityPropagation(damping = 0.9, convergence_iter = 30)

        xVars = ['Edad', 'Categoria']

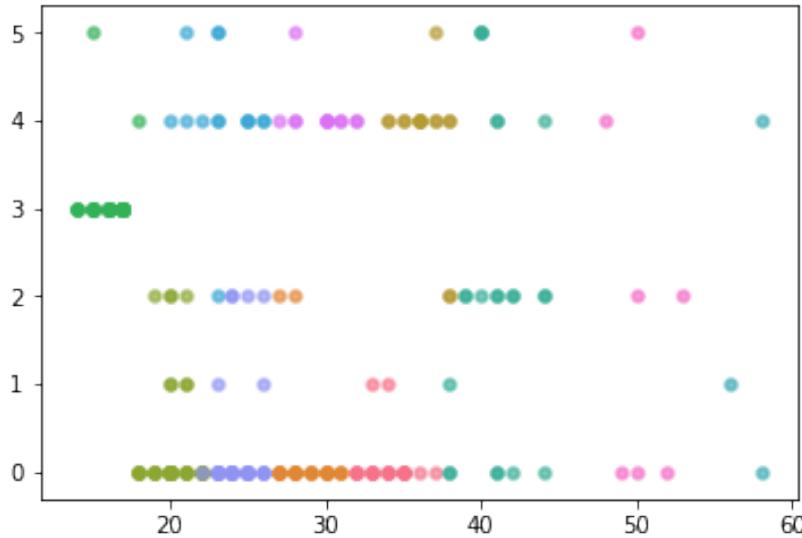
        d = d.loc[:, xVars]
        x = d.values
```

```

c = m.fit(x)
grupos = c.labels_
plt.clf()
plt.figure(1)
k = len(unique(grupos))
colores = sns.color_palette("husl", k)
for g in range(k):
    incl = where(grupos == g)[0]
    print(len(incl), "integrantes en grupo", g)
    grupo = take(x, incl, 0)
    plt.scatter(grupo[:, 0], grupo[:, 1], marker='o', s=20, linewidths=2, color=colores[g])
plt.show()
print(metrics.silhouette_score(x, grupos, metric='euclidean'))

(23, 'integrantes en grupo', 0)
(26, 'integrantes en grupo', 1)
(16, 'integrantes en grupo', 2)
(48, 'integrantes en grupo', 3)
(54, 'integrantes en grupo', 4)
(22, 'integrantes en grupo', 5)
(3, 'integrantes en grupo', 6)
(14, 'integrantes en grupo', 7)
(41, 'integrantes en grupo', 8)
(12, 'integrantes en grupo', 9)
(7, 'integrantes en grupo', 10)

```



0.564122635978908

1.6 Conclusión

Al aplicar estos dos algoritmos de agrupamiento se puede notar que la mejor forma de agrupar estos datos es en dos grupos, aunque el K-Means tuvo una mejor calidad de solución al obtener 0.5220206737202812 y el AffinityPropagation obtuvo 0.564122635978908. En K-Means se agrupan las edades en once grupos y en AffinityPropagation a pesar de que tiene la misma cantidad de grupos, tiene una mejor precisión que el K-Means para este caso.

--06 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 12: Análisis de texto

Complicaciones

Práctica no
realizada a
tiempo.

P12

June 6, 2019

1 Reporte de práctica 12: Análisis de texto con nltk y wordcloud

En esta práctica trataremos con cadenas de caracteres. En la base de datos se cuenta con las sinopsis de los videos que se registraron en el concurso.

1.1 Objetivo

- Usar nltk o sklearn para hacer algún tipo de análisis de texto
- Apoyarse en las herramientas de bash para preprocesamiento.

1.2 Lectura de datos

Se aplican comandos del bash de sistema para poder saber cuales son las columnas de los archivos .csv y para poder obtener un ejemplo de los datos de cada archivo.

1.2.1 Columnas de los csv

In [8]: !head -n 1 2016.csv | tr , '\n' | grep -v '^\$' | nl -v 2

```
2      Sinopsis
3      limpios
```

In [10]: !head -n 1 2017.csv | tr , '\n' | grep -v '^\$' | nl -v 2

```
2      Sinopsis
3      limpios
```

In [11]: !head -n 1 2018.csv | tr , '\n' | grep -v '^\$' | nl -v 2

```
2      Categoria
3      Edad
4      Pais
5      Titulo
6      Genero
7      Duracion
8      Marca
9      Referencia
```

```
10      Dias
11      Marcas
12      Personas
13      Sinopsis
14      limpios
```

```
In [12]: !head -n 1 2018mx.csv | tr , '\n' | grep -v '^$' | nl -v 2
2      Sinopsis
3      limpios
```

Se consulta los primeros cinco registros para confirmar que la información que tenemos es la que se necesita.

1.2.2 Ejemplos

```
In [13]: !head -n 5 2016.csv
```

```
Sinopsis,limpios
"Valentina España, una niña preocupada por que sus amigas ya no juegan si no que se la pasan en
"sammy es una pequeña soñadora que espera con ansias un día especial en la playa para jugar y
es una niña que tiene un sueño lo cual ella se preocupa por lo que soñó así que decide investigar
"Es un documental en donde se investiga el punto de vista de una niña de cinco años ante el mu
```

```
In [14]: !head -n 5 2017.csv
```

```
Sinopsis,limpios
"ESTA HISTORIA INICIA CON CADA UNO DE LOS PERSONAJES CAMINANDO POR LA CALLE, ELLOS SE ENCUENTRAN
"La historia tratará de un convencional joven, rutinario, que a pesar de estar metido en su vida
"Se quiere resaltar Bogotá y sus alrededores capturando paisajes y su cotidianidad desde los te
"A lo largo del día se muestra la rutina de Martín, un joven aparentemente comízón y corriente
```

```
In [16]: !head -n 5 2018.csv
```

```
Categoría,Edad,Pais,Título,Genero,Duración,Marca,Referencia,Días,Marcas,Personas,Sinopsis,limpios
AFICIONADO,24,Colombia,0.5,Drama,300,Motorola,Moto G 2,7,130.0,4,"Jimena, mujer rígidamente es
CRÓNICAS,26,Bélgica,12,Crónica,180,Apple,ad,,1,<ksdvnc,ksdvnc
AFICIONADO,34,Colombia,#ATuRitmo,Comedia,65,Huawei,P10 lite,8,245.0,7,"Un día cualquiera en el
AFICIONADO,41,Colombia,#EstamosConPipe,Ficción,300,Apple,Ipad Mini 2 Modelo: MD528E7/A,16,153.0
```

```
In [17]: !head -n 5 2018mx.csv
```

```
Sinopsis,limpios
"Se trata de una niña que pensaba que la última palabra la tenía el culo, pero comprueba que no
una pequeña probada de lo que tenemos planeado para una serie de nuestro pequeño set de producc
"Luzma es una maestra jubilada que recuerda con melancolía sus años de enseñanza, pero una tar
"TRATA SOBRE CUANDO ALGUIEN VIAJA DE NOCHE POR LAS CARRETERAS, SE CUENTAS DIVERSAS LEYENDAS UR
```

1.3 La librería nltk

El kit de herramientas de lenguaje natural, o más comúnmente NLTK, es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural (PLN) simbólico y estadísticos para el lenguaje de programación Python. NLTK incluye demostraciones gráficas y datos de muestra. Se importa la librería al programa.

```
In [1]: import nltk
```

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('vader_lexicon')
from nltk.corpus import stopwords
print(stopwords.words("spanish")[:10])
from nltk.sentiment.vader import SentimentIntensityAnalyzer
s = SentimentIntensityAnalyzer() # en inglés hasta podemos distinguir entre palabras p
print(s.polarity_scores('useless'))
print(s.polarity_scores('marvelous'))
```



```
[u'de', u'la', u'que', u'el', u'en', u'y', u'a', u'los', u'del', u'se']
{'neg': 1.0, 'neu': 0.0, 'pos': 0.0, 'compound': -0.4215}
{'neg': 0.0, 'neu': 0.0, 'pos': 1.0, 'compound': 0.5994}
```



```
[nltk_data] Downloading package punkt to
[nltk_data]      /home/samataroukami/nltk_data...
[nltk_data]      Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]      /home/samataroukami/nltk_data...
[nltk_data]      Package stopwords is already up-to-date!
[nltk_data] Downloading package vader_lexicon to
[nltk_data]      /home/samataroukami/nltk_data...
[nltk_data]      Package vader_lexicon is already up-to-date!
```

Ahora se importa un documento para procesar en nltk, el 2018.xlsx. Se cuenta con la librería necesaria para poder trabajar con archivos tipo Excel desde Python.

```
In [18]: import pandas as pd
```

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from nltk.tokenize import RegexpTokenizer
```



```
d = pd.read_excel('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/master/2018.xlsx')
d = d[['Categoria', 'Edad', 'Pais', 'Titulo', 'Genero', 'Duracion', 'Marca', 'Referencia']]
n = len(d)
```

```

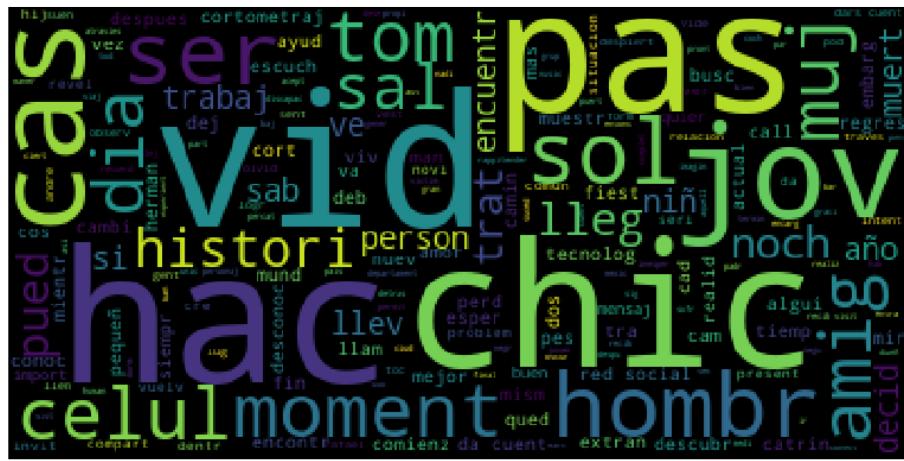
spa = stopwords.words("spanish")
stemmer = SnowballStemmer('spanish')
tokenizer = RegexpTokenizer(r'\w+') # para eliminar puntuación
reemplazos = []
for r in range(n):
    original = d.Sinopsis[r]
    reemplazo = ''
    if original != 'SIN_DESCR':
        quedar = [stemmer.stem(p) for p in tokenizer.tokenize(original) if p.lower() != 'sin_descr']
        reemplazo = ' '.join(quedar)
    reemplazos.append(reemplazo)
d['limpios'] = reemplazos
texto = ' '.join(reemplazos)
nube = WordCloud().generate(texto)
plt.rcParams["figure.figsize"] = [15, 7]
plt.imshow(nube)
plt.axis("off")
plt.show()
d.to_csv("2018.csv", index=False, encoding="utf-8")

```



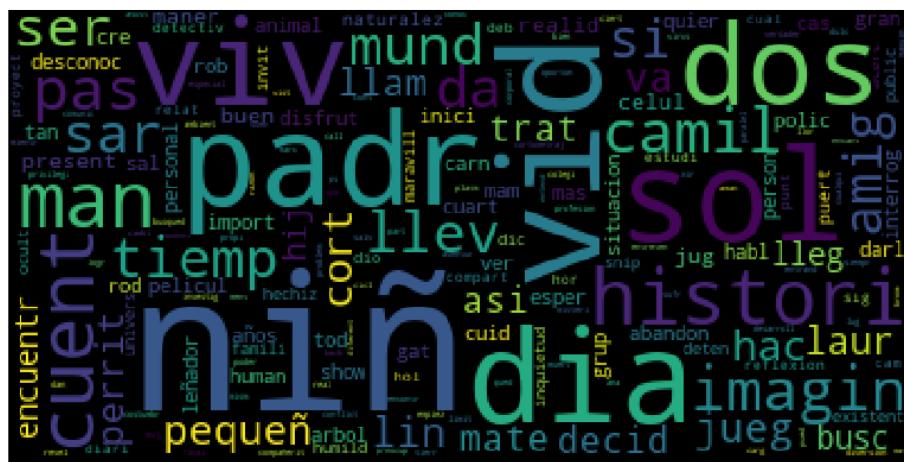
```
In [3]: d = pd.read_excel('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/main/datos/iris.xlsx')
d = d[['Sinopsis']]
n = len(d)
spa = stopwords.words("spanish")
stemmer = SnowballStemmer('spanish')
tokenizer = RegexpTokenizer(r'\w+') # para eliminar puntuación
reemplazos = []
for r in range(n):
```

```
original = d.Sinopsis[r]
reemplazo = ''
if original != 'SIN_DESCR':
    quedar = [stemmer.stem(p) for p in tokenizer.tokenize(original) if p.lower() not in stopWords]
    reemplazo = ' '.join(quedar)
reemplazos.append(reemplazo)
d['limpios'] = reemplazos
texto = ' '.join(reemplazos)
nube = WordCloud().generate(texto)
plt.rcParams["figure.figsize"] = [15, 7]
plt.imshow(nube)
plt.axis("off")
plt.show()
d.to_csv("2018mx.csv", index=False, encoding="utf-8")
```



```
In [4]: d = pd.read_excel('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/main/lexico.xlsx')
d = d[['Sinopsis']]
n = len(d)
spa = stopwords.words("spanish")
stemmer = SnowballStemmer('spanish')
tokenizer = RegexpTokenizer(r'\w+') # para eliminar puntuación
reemplazos = []
for r in range(n):
    original = d.Sinopsis[r]
    reemplazo = ''
    if original != 'SIN_DESCR':
        quedar = [stemmer.stem(p) for p in tokenizer.tokenize(original) if p.lower() not in stop]
        reemplazo = ' '.join(quedar)
```

```
reemplazos.append(reemplazo)
d['limpios'] = reemplazos
texto = ' '.join(reemplazos)
nube = WordCloud().generate(texto)
plt.rcParams["figure.figsize"] = [15, 7]
plt.imshow(nube)
plt.axis("off")
plt.show()
d.to_csv("2016.csv", index=False, encoding="utf-8")
```



```
In [5]: d = pd.read_excel('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DATOS/main/lexico.xlsx')
d = d[['Sinopsis']]
n = len(d)
spa = stopwords.words("spanish")
stemmer = SnowballStemmer('spanish')
tokenizer = RegexpTokenizer(r'\w+') # para eliminar puntuación
reemplazos = []
for r in range(n):
    original = d.Sinopsis[r]
    reemplazo = ''
    if original != 'SIN_DESCR':
        quedar = [stemmer.stem(p) for p in tokenizer.tokenize(original) if p.lower() not in stop_words]
        reemplazo = ' '.join(quedar)
    reemplazos.append(reemplazo)
d['limpios'] = reemplazos
texto = ' '.join(reemplazos)
nube = WordCloud().generate(texto)
plt.rcParams["figure.figsize"] = [15, 7]
```

```
plt.imshow(nube)
plt.axis("off")
plt.show()
d.to_csv("2017.csv", index=False, encoding="utf-8")
```



1.4 Conclusión

Al usar la librería nltk se obtuvieron imágenes con las palabras más utilizadas en las sinopsis de los videos. Las palabras que se ven presentes en todos gráficos son:

- vid
- dia
- viv
- histori
- ser

La librería es muy útil ya que se puede corregir los errores ortográficos desde código Python.
--06 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Práctica 13: Análisis de imágenes

Complicaciones

CIENCIA_DE_DATOS (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master)
/ P13.ipynb (/github/SamatarouKami/CIENCIA_DE_DATOS/tree/master/P13.ipynb)

Reporte de práctica 13: Análisis de imágenes

Para ésta práctica se tomarán en cuenta los ganadores de las seis categorías disponibles para el concurso de Colombia en el año 2018. Contamos con los videos ganadores en la plataforma de Youtube, todos en alta calidad. Se necesitan hacer análisis sobre las imágenes que obtendrémos de los fotogramas de los videos.

Objetivo

- Aplicar algún tipo de procesamiento de imágenes.

Datos

"Base de datos completa" de los registros al concurso de SmartFilm.

En este caso, es necesario limpiar los datos, seleccionaremos los registros que concuerden con el nombre de los videos ganadores por categoría.

De la Base de datos tomaremos los siguientes campos para trabajar:

- Categoría
- Edad
- País
- Título del Corto
- Género
- Duración
- Marca
- Referencia Celular
- Días de rodaje
- Marcas del rodaje
- Personas

Contamos con 6 videos en resolución 1920 x 1080 pixeles sin audio formato mp4.

Los videos ganadores son los siguientes:

Categoría	Título
Aficionado	EZEQUIEL 18:27
Crónica	Mil colores para mi pueblo, Arte para la paz.
Familiar	Ellos
Juvenil	La Otra Cara de Karla.
Profesional	Una última vez
SmarTIC	Sin Ataduras 1812

De cada video tomaremos todos los fotogramas para procesarlos.

Hipótesis

Para Imágenes

Para separar las transiciones relevantes de posibles cambios de iluminación en el video, haremos un clasificador sencillo, si la transición está por encima de la media de todas las transiciones, quiere decir no es un simple cambio de iluminación.

Imágenes contra datos

La cantidad de transiciones está relacionada con la cantidad de días de rodaje.

Método

La forma en la que analizaremos los videos será tomando un fotograma (fotograma base) y comparándolo con los dos contiguos (prueba 1 y 2). Utilizaremos [OpenCV](https://opencv.org/) para el tratamiento de las imágenes, compararemos histogramas. Convertiremos los colores de la imagen de RGB a HSV para trabajar tal como lo muestra el [tutorial](https://docs.opencv.org/master/d8/dc8/tutorial_histogram_comparison.html). Utilizamos el método de comparación Chi cuadrada ya que si uno de los fotogramas de prueba es completamente diferente al fotograma base los valores que arroja son significativamente altos, indicando una transición de escena en el video.

Compararemos la cantidad de transiciones se relaciona con los días de rodaje.

Se crearon dos clases en python para procesar los videos y después los fotogramas.

- Extract_frames
- Hist_comparison

Se muestran a continuación.

Clase Extract_frames.py

```
In [ ]: import cv2
import os
from console_progressbar import ProgressBar

class Extract_frames:

    def __init__(self,filename,segundos=300):
        self.vidcap = cv2.VideoCapture(filename)
        success,image = self.vidcap.read()
        self.success = success
        self.image = image
        self.filename = filename[:-4]
        self.segundos = segundos
        fps = self.vidcap.get(cv2.CAP_PROP_FPS)
        self.Numframes = segundos*fps
        self.getFrames()

    def getFrames(self):
        try:
            os.stat(self.filename)
        except:
            os.mkdir(self.filename)
        print('\n'+ "Leyendo: " + self.filename)
        pb = ProgressBar(total=self.Numframes, prefix='Progreso:', suffix = 'frames', decimals=2, length=50)
        count = 0
        while self.success:
            cv2.imwrite(self.filename+"/frame%d.jpg" % count, self.image)
            success,image = self.vidcap.read()
            self.success = success
            self.image = image
            count += 1
            pb.print_progress_bar(count)
        pb.print_progress_bar(self.Numframes)

aficionado = Extract_frames('aficionado2018.mp4')
cronica = Extract_frames('cronica2018.mp4')
familiar = Extract_frames('familiar2018.mp4')
juvenil = Extract_frames('juvenil2018.mp4')
profesional = Extract_frames('profesional2018.mp4')
smarTIC = Extract_frames('smartic2018.mp4')
```

Clase Hist_comparison.py

```
In [ ]: import cv2 as cv
import numpy as np
```

```

class Hist_comparison:

    def __init__(self,base,t1,t2):
        self.base = base
        src_base = cv.imread(self.base)
        src_test1 = cv.imread(t1)
        src_test2 = cv.imread(t2)
        if src_base is None or src_test1 is None or src_test2 is None:
            print('Could not open or find the images!')
            exit(0)
        self.hsv_base = cv.cvtColor(src_base, cv.COLOR_BGR2HSV)
        self.hsv_test1 = cv.cvtColor(src_test1, cv.COLOR_BGR2HSV)
        self.hsv_test2 = cv.cvtColor(src_test2, cv.COLOR_BGR2HSV)
        h_bins = 50
        s_bins = 60
        self.histSize = [h_bins, s_bins]
        # hue varies from 0 to 179, saturation from 0 to 255
        h_ranges = [0, 180]
        s_ranges = [0, 256]
        self.ranges = h_ranges + s_ranges # concat lists
        # Use the 0-th and 1-st channels
        self.channels = [0, 1]
        self.base_base = None
        self.base_test1 = None
        self.base_test2 = None
        self.doComparison()

    def doComparison(self):
        hist_base = cv.calcHist([self.hsv_base], self.channels, None, self.histSize, self.ranges, accumulate=False)
        cv.normalize(hist_base, hist_base, alpha=0, beta=1, norm_type=cv.NORM_MINMAX)
        hist_test1 = cv.calcHist([self.hsv_test1], self.channels, None, self.histSize, self.ranges, accumulate=False)
        cv.normalize(hist_test1, hist_test1, alpha=0, beta=1, norm_type=cv.NORM_MINMAX)
        hist_test2 = cv.calcHist([self.hsv_test2], self.channels, None, self.histSize, self.ranges, accumulate=False)
        cv.normalize(hist_test2, hist_test2, alpha=0, beta=1, norm_type=cv.NORM_MINMAX)
        compare_method = 1
        self.base_test1 = cv.compareHist(hist_base, hist_test1, compare_method)
        self.base_test2 = cv.compareHist(hist_base, hist_test2, compare_method)

    def getValues(self,filename):
        print(self.base[5:-4],self.base_test1,self.base_test2, file=open(filename, "a"))

```

```

from console_progressbar import ProgressBar
import glob

totImages=len(glob.glob("*.jpg"))

print('Procesando: ' + str(totImages) + ' frames')
pb1 = ProgressBar(total=totImages, prefix='Progreso:', suffix = 'frames',
', decimals=2, length=50)
for i in range(0,totImages-2):
    HC = Hist_comparison('frame'+str(i)+'.jpg','frame'+str(i+1)+'.jpg',
'frame'+str(i+2)+'.jpg')
    HC.getValues("frames.csv")
    pb1.print_progress_bar(i)
pb1.print_progress_bar(totImages)

print('Procesando: ' + str(300) + ' segundos')
pb24 = ProgressBar(total=totImages, prefix='Progreso:', suffix = 'segundos',
decimals=2, length=50)
step=int(totImages/300)
for i in range(0,totImages-2,step):
    HC = Hist_comparison('frame'+str(i)+'.jpg','frame'+str(i+int(step/2
))+'.jpg','frame'+str(i+step)+'.jpg')
    HC.getValues("seconds.csv")
    pb24.print_progress_bar(i)
pb24.print_progress_bar(totImages)

```

Preparación

Primero preparamos el datafame de la base de datos.

```
In [1]: import pandas as pd
df2018 = pd.read_excel('2018.xlsx', index_col=None, header=0, sheet_name=0)
df2018 = df2018[['Categoria', 'Edad', 'Pais', 'Titulo', 'Genero', 'Duracion', 'Marca', 'Referencia', 'Dias', 'Marcas', 'Personas']]
#print(df2018.columns)

ganadores = ['EZEQUIEL 18:27', 'Mil colores para mi pueblo, Arte para la paz.', 'Ellos', 'La Otra Cara de Karla', 'Una última vez', 'Sin Ataduras 1812']

df2018 = df2018.loc[df2018['Titulo'].isin(ganadores)]
df2018 = df2018.sort_values(by='Categoria').reset_index()

print(df2018)
```

	index	Categoría	Edad	Pais	\
0	175	AFICIONADO	19	Colombia	
1	294	CRONICAS	31	Colombia	
2	156	FAMILIAR	21	Colombia	
3	234	JUVENIL	16	Colombia	
4	433	PROFESIONAL	25	Colombia	
5	381	SMARTIC INCLUYENTE	29	Colombia	

	Marca \	Titulo	Genero	Duracion
0	Apple	EZEQUIEL 18:27	Drama	299
1	Samsung	Mil colores para mi pueblo, Arte para la paz.	Crónica	300 Sa
2	Apple	Ellos	Drama	290
3	Samsung	La Otra Cara de Karla	Otra	27423 Sa
4	Apple	Una última vez	Drama	299
5	Samsung	Sin Ataduras 1812	Drama	298 Sa

	Referencia	Dias	Marcas	Personas
0	iPhone 7	155	154.0	15
1	Galaxy s6 Edge	30	NaN	6
2	iPhone8	60	172.0	20
3	S9+	40	20899.0	35
4	iphone x + rig para lentes	50	52.0	35
5	Galaxy A8	90	NaN	6

Como la extracción de frames de los videos agranda el volumen de archivos con los cuales trabajar, se aplicó el método en los videos fuera de linea. Se obtuvieron doce archivos, emparejados representan un video, el primero es la comparación de todos los frames, y el segundo es la comparación de los frames revisando cada cierta cantidad de frames, dada por el "fps rate".

	Video	Archivos
EZEQUIEL 18:27	framesA.csv	secondsA.csv
Mil colores para mi pueblo, Arte para la paz.	framesC.csv	secondsC.csv
Ellos	framesF.csv	secondsF.csv
La Otra Cara de Karla	framesJ.csv	secondsJ.csv
Una última vez	framesP.csv	secondsP.csv
Sin Ataduras 1812	framesS.csv	secondsS.csv

Ahora que tenemos los datos del análisis de imágenes podemos calcular sus transiciones.

```
In [2]: import numpy as np

framesA = pd.read_csv('P13csv/framesA.csv', sep=' ', index_col =0)
framesA.set_axis(['Test1','Test2'], axis='columns', inplace=True)
framesA['Diferencia'] = framesA['Test2']-framesA['Test1']
framesA['Transicion'] = np.where(framesA["Diferencia"]>framesA["Diferencia"].mean(), 1, 0 )

secondsA = pd.read_csv('P13csv/secondsA.csv', sep=' ', index_col =0)
secondsA.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsA['Diferencia'] = secondsA['Test2']-secondsA['Test1']
secondsA['Transicion'] = np.where(secondsA["Diferencia"]>secondsA["Diferencia"].mean(), 1, 0 )

framesC = pd.read_csv('P13csv/framesC.csv', sep=' ', index_col =0)
framesC.set_axis(['Test1','Test2'], axis='columns', inplace=True)
framesC['Diferencia'] = framesC['Test2']-framesC['Test1']
framesC['Transicion'] = np.where(framesC["Diferencia"]>framesC["Diferencia"].mean(), 1, 0 )

secondsC = pd.read_csv('P13csv/secondsC.csv', sep=' ', index_col =0)
secondsC.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsC['Diferencia'] = secondsC['Test2']-secondsC['Test1']
secondsC['Transicion'] = np.where(secondsC["Diferencia"]>secondsC["Diferencia"].mean(), 1, 0 )

framesF = pd.read_csv('P13csv/framesF.csv', sep=' ', index_col =0)
framesF.set_axis(['Test1','Test2'], axis='columns', inplace=True)
```

```

framesF['Diferencia'] = framesF['Test2']-framesF['Test1']
framesF['Transicion'] = np.where(framesF["Diferencia"]>framesF["Diferencia"].mean(), 1, 0 )

secondsF = pd.read_csv('P13csv/secondsF.csv', sep=' ', index_col =0)
secondsF.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsF['Diferencia'] = secondsF['Test2']-secondsF['Test1']
secondsF['Transicion'] = np.where(secondsF["Diferencia"]>secondsF["Diferencia"].mean(), 1, 0 )

framesJ = pd.read_csv('P13csv/framesJ.csv', sep=' ', index_col =0)
framesJ.set_axis(['Test1','Test2'], axis='columns', inplace=True)
framesJ['Diferencia'] = framesJ['Test2']-framesJ['Test1']
framesJ['Transicion'] = np.where(framesJ["Diferencia"]>framesJ["Diferencia"].mean(), 1, 0 )

secondsJ = pd.read_csv('P13csv/secondsJ.csv', sep=' ', index_col =0)
secondsJ.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsJ['Diferencia'] = secondsJ['Test2']-secondsJ['Test1']
secondsJ['Transicion'] = np.where(secondsJ["Diferencia"]>secondsJ["Diferencia"].mean(), 1, 0 )

framesP = pd.read_csv('P13csv/framesP.csv', sep=' ', index_col =0)
framesP.set_axis(['Test1','Test2'], axis='columns', inplace=True)
framesP['Diferencia'] = framesP['Test2']-framesP['Test1']
framesP['Transicion'] = np.where(framesP["Diferencia"]>framesP["Diferencia"].mean(), 1, 0 )

secondsP = pd.read_csv('P13csv/secondsP.csv', sep=' ', index_col =0)
secondsP.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsP['Diferencia'] = secondsP['Test2']-secondsP['Test1']
secondsP['Transicion'] = np.where(secondsP["Diferencia"]>secondsP["Diferencia"].mean(), 1, 0 )

framesS = pd.read_csv('P13csv/framesS.csv', sep=' ', index_col =0)
framesS.set_axis(['Test1','Test2'], axis='columns', inplace=True)
framesS['Diferencia'] = framesS['Test2']-framesS['Test1']
framesS['Transicion'] = np.where(framesS["Diferencia"]>framesS["Diferencia"].mean(), 1, 0 )

secondsS = pd.read_csv('P13csv/secondsS.csv', sep=' ', index_col =0)
secondsS.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsS['Diferencia'] = secondsS['Test2']-secondsS['Test1']
secondsS['Transicion'] = np.where(secondsS["Diferencia"]>secondsS["Diferencia"].mean(), 1, 0 )

trans=[]

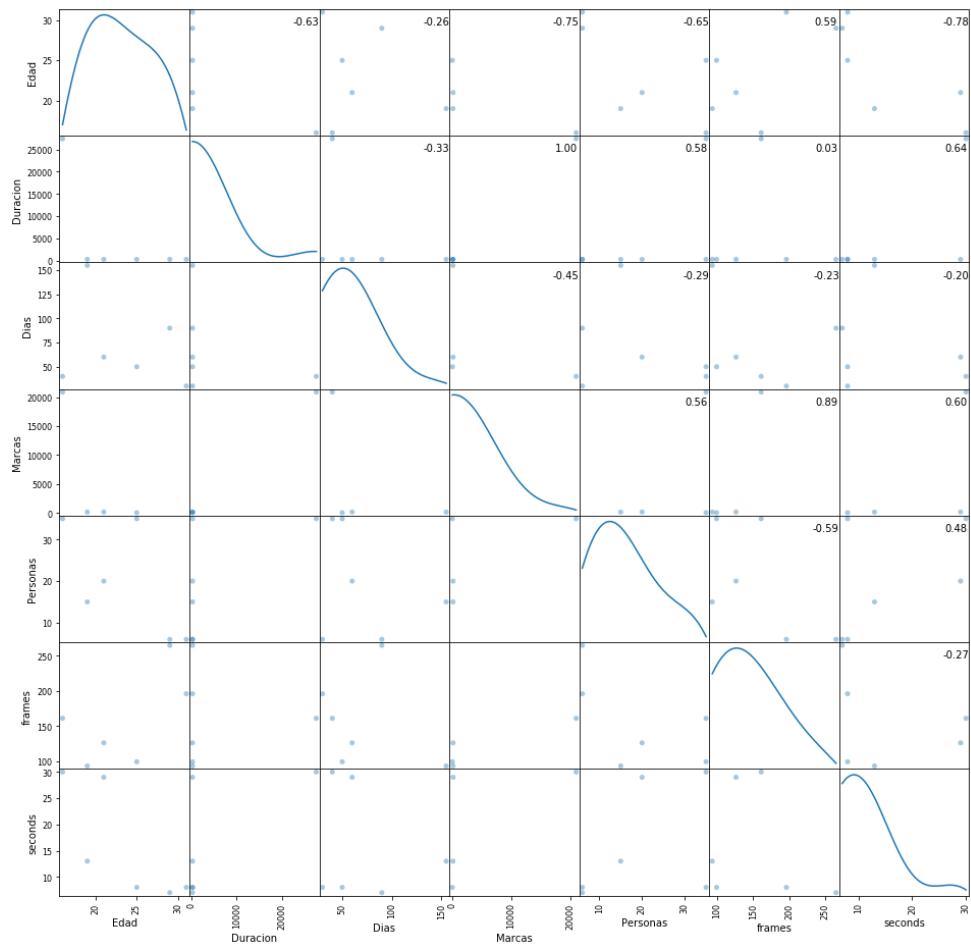
trans.append([framesA['Transicion'].sum(),secondsA['Transicion'].sum()])
trans.append([framesC['Transicion'].sum(),secondsC['Transicion'].sum()])
trans.append([framesF['Transicion'].sum(),secondsF['Transicion'].sum()])

```

```
)  
trans.append([framesJ['Transicion'].sum(),secondsJ['Transicion'].sum()])  
)  
trans.append([framesP['Transicion'].sum(),secondsP['Transicion'].sum()])  
)  
trans.append([framesS['Transicion'].sum(),secondsS['Transicion'].sum()])  
)  
  
#print(trans)  
  
df = pd.DataFrame(data=trans)  
  
df.set_axis(['frames','seconds'], axis='columns', inplace=True)  
  
  
  
df2018['id']=df2018.index  
df['id']=df.index  
  
dfFinal = pd.merge(df2018, df)  
  
dfFinal = dfFinal.drop(['id','index'], axis=1)  
  
#print(dfFinal)  
e = dfFinal
```

```
In [4]: import pandas as pd
from numpy import NaN
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

ax = scatter_matrix(e, alpha = 0.4, figsize = (16, 16), diagonal = 'kde',
', s = 100)
c = e.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()
```



Conclusión

Son muy pocos datos para obtener una gráfica con más puntos, pero revisando los coeficientes, podemos comprobar que la duracion y las marcas estan relacionadas, y tambien las marcas con la cantidad de transiciones. Asi que damos por probada la hipótesis de Imagen contra datos.

--20 de Mayo 2019-- Luis Angel Gutierrez Rodriguez 1484412 (tel:1484412)

Reporte de práctica 13: Análisis de imágenes

Para esta práctica se tomarán en cuenta los ganadores de las seis categorías disponibles para el concurso de Colombia en el año 2018. Contamos con los videos ganadores en la plataforma de YouTube, todos en alta calidad. Se necesitan hacer análisis sobre las imágenes que obtendremos de los fotogramas de los videos.

Objetivo

- Aplicar algún tipo de procesamiento de imágenes.

Datos

"Base de datos completa" de los registros al concurso de SmartFilm.

En este caso, es necesario limpiar los datos, seleccionaremos los registros que concuerden con el nombre de los videos ganadores por categoría.

De la base de datos tomaremos los siguientes campos para trabajar:

- Categoría
- Edad
- País
- Título del Corto
- Género
- Duración
- Marca
- Referencia Celular
- Días de rodaje
- Marcas del rodaje
- Personas

Contamos con 6 videos en resolución 1920 x 1080 pixeles sin audio formato mp4.

Los videos ganadores son los siguientes:

Categoría	Título
Aficionado	EZEQUIEL 18:27 (https://www.youtube.com/watch?v=gx8wACV3Wyc)
Crónica	Mil colores para mi pueblo, Arte para la paz. (https://www.youtube.com/watch?v=SdYrUJ1vo_I)
Familiar	Ellos (https://www.youtube.com/watch?v=MBtZc9HOFsM)
Juvenil	La Otra Cara de Karla (https://www.youtube.com/watch?v=H_4QW3xdI5w)
Profesional	Una última vez (https://www.youtube.com/watch?v=_67Y-K7ASDk)
SmarTIC	Sin Ataduras 1812 (https://www.youtube.com/watch?v=FLofpshTn8k)

De cada video tomaremos todos los fotogramas para procesarlos.

Hipótesis

Para Imágenes

```
In [ ]: import cv2
import os
from console_progressbar import ProgressBar

class Extract_frames:

    def __init__(self,filename,segundos=300):
        self.vidcap = cv2.VideoCapture(filename)
        success,image = self.vidcap.read()
        self.success = success
        self.image = image
        self.filename = filename[:-4]
        self.segundos = segundos
        fps = self.vidcap.get(cv2.CAP_PROP_FPS)
        self.Numframes = segundos*fps
        self.getFrames()

    def getFrames(self):
        try:
            os.stat(self.filename)
        except:
            os.mkdir(self.filename)
        print('\n'+ "Leyendo: " + self.filename)
        pb = ProgressBar(total=self.Numframes, prefix='Progreso:', suffix = 'frames', decimals=2, length=50)
        count = 0
        while self.success:
            cv2.imwrite(self.filename+"/frame%d.jpg" % count, self.image)
            success,image = self.vidcap.read()
            self.success = success
            self.image = image
            count += 1
            pb.print_progress_bar(count)
        pb.print_progress_bar(self.Numframes)

aficionado = Extract_frames('aficionado2018.mp4')
cronica = Extract_frames('cronica2018.mp4')
familiar = Extract_frames('familiar2018.mp4')
juvenil = Extract_frames('juvenil2018.mp4')
profesional = Extract_frames('profesional2018.mp4')
smartIC = Extract_frames('smartic2018.mp4')
```

Clase Hist_comparison.py

```
In [ ]: import cv2 as cv
import numpy as np

class Hist_comparison:

    def __init__(self,base,t1,t2):
        self.base = base
        src_base = cv.imread(self.base)
        src_test1 = cv.imread(t1)
        src_test2 = cv.imread(t2)
        if src_base is None or src_test1 is None or src_test2 is None:
            print('Could not open or find the images!')
            exit(0)
        self.hsv_base = cv.cvtColor(src_base, cv.COLOR_BGR2HSV)
        self.hsv_test1 = cv.cvtColor(src_test1, cv.COLOR_BGR2HSV)
        self.hsv_test2 = cv.cvtColor(src_test2, cv.COLOR_BGR2HSV)
        h_bins = 50
        s_bins = 60
        self.histSize = [h_bins, s_bins]
        # hue varies from 0 to 179, saturation from 0 to 255
        h_ranges = [0, 180]
        s_ranges = [0, 256]
        self.ranges = h_ranges + s_ranges # concat lists
        # Use the 0-th and 1-st channels
        self.channels = [0, 1]
        self.base_base = None
        self.base_test1 = None
        self.base_test2 = None
        self.doComparison()

    def doComparison(self):
        hist_base = cv.calcHist([self.hsv_base], self.channels, None, self.histSize, self.ranges, accumulate=False)
        cv.normalize(hist_base, hist_base, alpha=0, beta=1, norm_type=cv.NORM_MINMAX)
        hist_test1 = cv.calcHist([self.hsv_test1], self.channels, None, self.histSize, self.ranges, accumulate=False)
        cv.normalize(hist_test1, hist_test1, alpha=0, beta=1, norm_type=cv.NORM_MINMAX)
        hist_test2 = cv.calcHist([self.hsv_test2], self.channels, None, self.histSize, self.ranges, accumulate=False)
        cv.normalize(hist_test2, hist_test2, alpha=0, beta=1, norm_type=cv.NORM_MINMAX)
        compare_method = 1
        self.base_test1 = cv.compareHist(hist_base, hist_test1, compare_method)
        self.base_test2 = cv.compareHist(hist_base, hist_test2, compare_method)

    def getValues(self,filename):
        print(self.base[5:-4],self.base_test1,self.base_test2, file=open(filename, "a"))

from console_progressbar import ProgressBar
import glob

totImages=len(glob.glob("*.jpg"))

print('Procesando: ' + str(totImages) + ' frames')
pbl = ProgressBar(total=totImages, prefix='Progreso:', suffix = 'frames', decimals=2, length=50)
for i in range(0,totImages-1):
```

Grafica de diferencias

Se utiliza Plotly para graficar los .csv de referencia que fueron producidos para verificar visualmente las transiciones.

```
In [1]: import plotly.plotly as py
import plotly.graph_objs as go
import plotly.figure_factory as FF

import numpy as np
import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_DA
TOS/master/old/P13csv/framesP.csv',sep=' ')

df.columns = ['frame','test1','test2']

df['diferencia'] = df.test2 - df.test1

df.diferencia = df.diferencia.apply(np.sqrt).apply(np.sqrt)

df = df.dropna()

df2 = pd.read_csv('https://raw.githubusercontent.com/SamatarouKami/CIENCIA_DE_D
ATOS/master/old/P13csv/secondsP.csv',sep=' ')

df2.columns = ['frame','test1','test2']

df2['diferencia'] = df2.test2 - df2.test1

df2.diferencia = df2.diferencia.apply(np.sqrt).apply(np.sqrt)

df2 = df2.dropna()

trace1 = go.Scatter(x = df['frame'], y = df['diferencia'],
                     name='Variaciones por Frame')

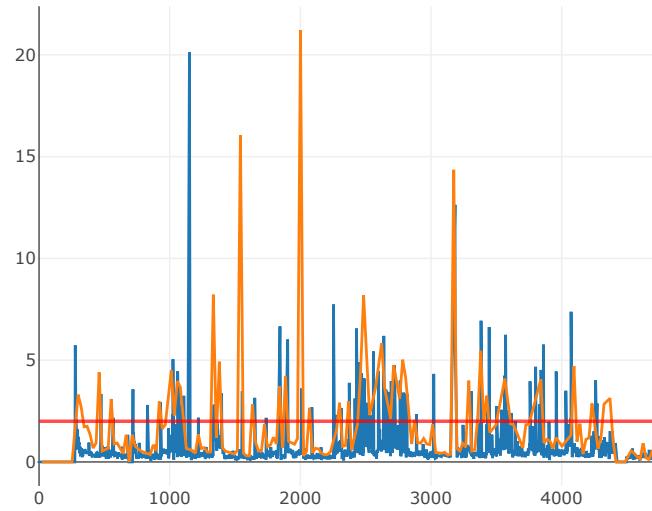
trace2 = go.Scatter(x = df2['frame'], y = df2['diferencia'],
                     name='Variaciones por segundo')

layout = {
    'title': "Transiciones medidas por variación de Histogramas",
    'shapes': [
        {
            'type': 'line',
            'x0': 1,
            'y0': 2,
            'x1': 7200,
            'y1': 2,
            'opacity': 0.7,
            'line': {
                'color': 'red',
                'width': 2.5,
            },
        },
    ],
}
fig = go.Figure(data=[trace1,trace2], layout=layout)

py.iplot(fig, filename='P13')
```

Out[1]:

Transiciones medidas por variá



[EDIT CHART](#)

En esta gráfica se busca demostrar como se sabia el número de transiciones de los videos ya que donde se encuentran los picos más altos, significa un cambio radical en la escena. La línea roja representa el umbral de que debe pasar una transición para no ser considerada cambio de brillo.

Preparación

Primero preparamos el DataFrame de la base de datos.

```
In [6]: import pandas as pd
df2018 = pd.read_excel('2018.xlsx', index_col=None, header=0, sheet_name=0)
df2018 = df2018[['Categoria', 'Edad', 'Pais', 'Titulo', 'Genero', 'Duracion', 'Marca', 'Referencia', 'Dias', 'Marcas', 'Personas']]
#print(df2018.columns)

ganadores = ['EZEQUIEL 18:27', 'Mil colores para mi pueblo, Arte para la paz.', 'Ellos', 'La Otra Cara de Karla', 'Una última vez', 'Sin Ataduras 1812']

df2018 = df2018.loc[df2018['Titulo'].isin(ganadores)]
df2018 = df2018.sort_values(by='Categoria').reset_index()

print(df2018)
```

	index	Categoría	Edad	Pais	Titulo	Genero	Duracion	Marca
0	175	AFICIONADO	19	Colombia	EZEQUIEL 18:27	Drama	299	Apple
1	294	CRONICAS	31	Colombia	Mil colores para mi pueblo, Arte para la paz.	Crónica	300	Samsung
2	156	FAMILIAR	21	Colombia	Ellos	Drama	290	Apple
3	234	JUVENIL	16	Colombia	La Otra Cara de Karla	Otra	27423	Samsung
4	381	SMARTIC INCLUYENTE	29	Colombia	Sin Ataduras 1812	Drama	298	Samsung

	Referencia	Dias	Marcas	Personas
0	iPhone 7	155	154.0	15
1	Galaxy s6 Edge	30	Nan	6
2	iPhone8	60	172.0	20
3	S9+	40	20899.0	35
4	Galaxy A8	90	Nan	6

Como la extracción de frames de los videos agranda el volumen de archivos con los cuales trabajar, se aplicó el método en los videos fuera de linea. Se obtuvieron doce archivos, emparejados representan un video, el primero es la comparación de todos los frames, y el segundo es la comparación de los frames revisando cada cierta cantidad de frames, dada por el "fps rate".

Video	Archivos
EZEQUIEL 18:27	framesA.csv secondsA.csv
Mil colores para mi pueblo, Arte para la paz.	framesC.csv secondsC.csv
Ellos	framesF.csv secondsF.csv
La Otra Cara de Karla	framesJ.csv secondsJ.csv
Una última vez	framesP.csv secondsP.csv
Sin Ataduras 1812	framesS.csv secondsS.csv

Ahora que tenemos los datos del análisis de imágenes podemos calcular sus transiciones.

```
In [3]: import numpy as np

framesA = pd.read_csv('old/P13csv/framesA.csv', sep=' ', index_col =0)
framesA.set_axis(['Test1','Test2'], axis='columns', inplace=True)
framesA['Diferencia'] = framesA['Test2']-framesA['Test1']
framesA['Transicion'] = np.where(framesA["Diferencia"]>framesA["Diferencia"].mean(), 1, 0 )

secondsA = pd.read_csv('old/P13csv/secondsA.csv', sep=' ', index_col =0)
secondsA.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsA['Diferencia'] = secondsA['Test2']-secondsA['Test1']
secondsA['Transicion'] = np.where(secondsA["Diferencia"]>secondsA["Diferencia"].mean(), 1, 0 )

framesC = pd.read_csv('old/P13csv/framesC.csv', sep=' ', index_col =0)
framesC.set_axis(['Test1','Test2'], axis='columns', inplace=True)
framesC['Diferencia'] = framesC['Test2']-framesC['Test1']
framesC['Transicion'] = np.where(framesC["Diferencia"]>framesC["Diferencia"].mean(), 1, 0 )

secondsC = pd.read_csv('old/P13csv/secondsC.csv', sep=' ', index_col =0)
secondsC.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsC['Diferencia'] = secondsC['Test2']-secondsC['Test1']
secondsC['Transicion'] = np.where(secondsC["Diferencia"]>secondsC["Diferencia"].mean(), 1, 0 )

framesF = pd.read_csv('old/P13csv/framesF.csv', sep=' ', index_col =0)
framesF.set_axis(['Test1','Test2'], axis='columns', inplace=True)
framesF['Diferencia'] = framesF['Test2']-framesF['Test1']
framesF['Transicion'] = np.where(framesF["Diferencia"]>framesF["Diferencia"].mean(), 1, 0 )

secondsF = pd.read_csv('old/P13csv/secondsF.csv', sep=' ', index_col =0)
secondsF.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsF['Diferencia'] = secondsF['Test2']-secondsF['Test1']
secondsF['Transicion'] = np.where(secondsF["Diferencia"]>secondsF["Diferencia"].mean(), 1, 0 )

framesJ = pd.read_csv('old/P13csv/framesJ.csv', sep=' ', index_col =0)
framesJ.set_axis(['Test1','Test2'], axis='columns', inplace=True)
framesJ['Diferencia'] = framesJ['Test2']-framesJ['Test1']
framesJ['Transicion'] = np.where(framesJ["Diferencia"]>framesJ["Diferencia"].mean(), 1, 0 )

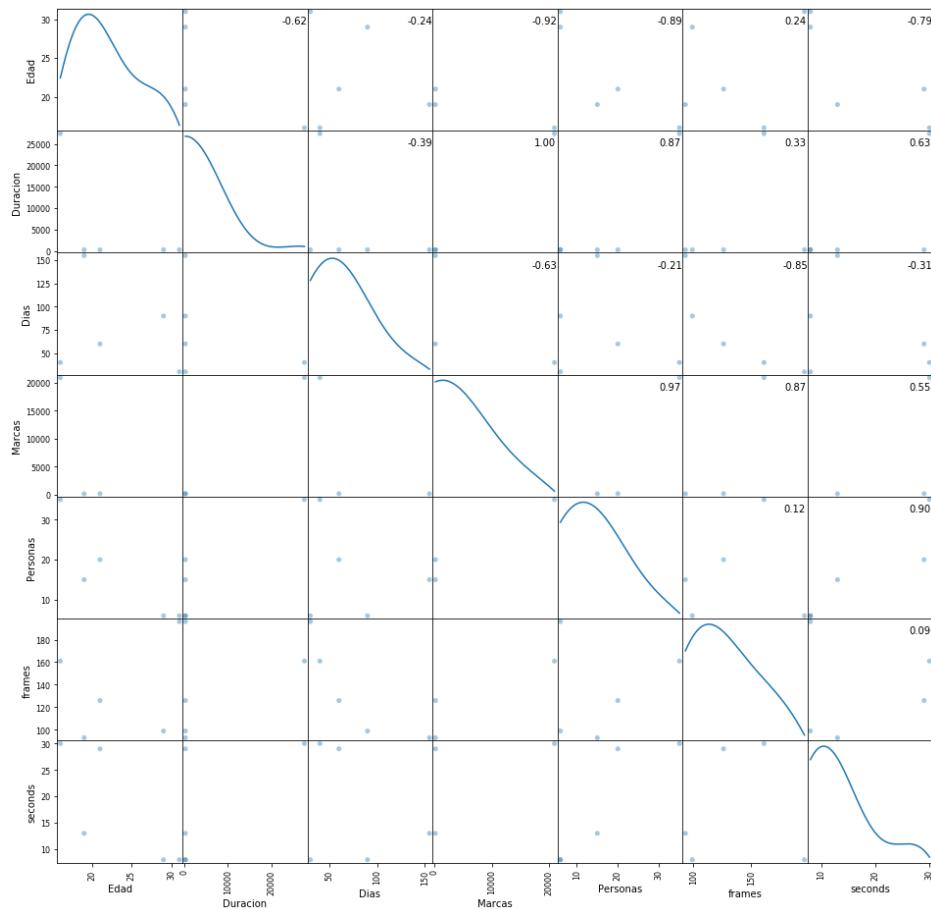
secondsJ = pd.read_csv('old/P13csv/secondsJ.csv', sep=' ', index_col =0)
secondsJ.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsJ['Diferencia'] = secondsJ['Test2']-secondsJ['Test1']
secondsJ['Transicion'] = np.where(secondsJ["Diferencia"]>secondsJ["Diferencia"].mean(), 1, 0 )

framesP = pd.read_csv('old/P13csv/framesP.csv', sep=' ', index_col =0)
framesP.set_axis(['Test1','Test2'], axis='columns', inplace=True)
framesP['Diferencia'] = framesP['Test2']-framesP['Test1']
framesP['Transicion'] = np.where(framesP["Diferencia"]>framesP["Diferencia"].mean(), 1, 0 )

secondsP = pd.read_csv('old/P13csv/secondsP.csv', sep=' ', index_col =0)
secondsP.set_axis(['Test1','Test2'], axis='columns', inplace=True)
secondsP['Diferencia'] = secondsP['Test2']-secondsP['Test1']
secondsP['Transicion'] = np.where(secondsP["Diferencia"]>secondsP["Diferencia"].mean(), 1, 0 )
```

```
In [5]: import pandas as pd
from numpy import NaN
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

ax = scatter_matrix(e, alpha = 0.4, figsize = (16, 16), diagonal = 'kde', s =
100)
c = e.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()
```



Conclusión

Son muy pocos datos para obtener una gráfica con más puntos, pero revisando los coeficientes, se puede comprobar que la duración y las marcas están relacionadas y también las marcas con la cantidad de transiciones. Así que damos por probada la hipótesis de Imagen contra datos.

--06 de junio 2019-- Luis Angel Gutiérrez Rodríguez 1484412

Preparación de artículo

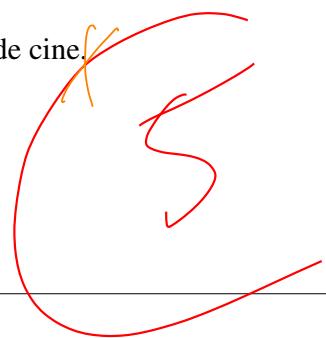
Complicaciones

Análisis de datos

~~Datos interesantes detrás de un concurso de cine~~

L.A. Gutierrez-Rodriguez

*Posgrado de Ingeniería en Sistemas,
Facultad de Ingeniería Mecánica y Eléctrica,
Universidad Autónoma de Nuevo León*



Abstract Resumen

Contamos con los registros de un concurso de cine, cuyos filmes han sido grabados con dispositivos smartphone. Se busca identificar el alcance del certamen, ayudar a definir las mejores técnicas de difusión, caracterizar las categorías para equilibrar la cantidad de participantes, mejorar la forma en que se registraron los participantes y pronosticar la participación de los próximos años.

La información se encuentra almacenada en cinco documentos de hoja de cálculo. Cada documento cuenta con los registros del concurso de cada año, del 2015 al 2018. Cuatro de ellos son del festival con sede en Colombia y el último del primer festival en México.

Se utiliza el lenguaje de programación Python y las librerías Numpy, Pandas, Scipy y Matplotlib para realizar análisis estadístico sobre los datos y poder graficar los resultados. Se aplican modelos de regresión lineal y múltiple. Se analiza la varianza y las componentes principales de los datos. Se pronostica la cantidad de participantes de futuros años. Se implementa la librería SKLearn para clasificar y agrupar los datos. Y se realiza análisis de texto de las sinopsis e imágenes de los videos ganadores. Al aplicar técnicas de estadística descriptiva se puede determinar que la mayoría de extranjeros que participan en el concurso son de México. Por este motivo se decidió en 2018 abrir un festival en ese país. ?

Introducción

En la actualidad contamos con una cantidad inmensa de datos debido a que almacenamos toda clase de información. Se busca darle un sentido útil a estos datos para poder comprenderlos. La ciencia de datos es el estudio de la extracción generalizable de conocimiento a partir de datos [6]. En este trabajo se busca aplicar la ciencia de datos ayudar en la toma de decisiones de dónde y cuándo se deben organizar los eventos, la mejor forma de organizar las categorías y en qué otros países se puede expandir el festival.

Se sabe que la mayoría de los concursantes son de regiones cercanas a la capital de Colombia, que estos concursantes usan smartphones para grabar sus videos, que proporcionan una sinopsis, y cada participante ubica su filme en un género. Además, no se tiene identificación de los participantes, y por esto no se puede trazar su participación a lo largo de los festivales.

En antecedentes hablaremos de todo lo relacionado con el festival de cine. En la literatura relacionada, veremos que problemas ya han sido tratados y como la obtención de información ayudó a la toma de decisiones. Además, se tratarán temas que están relacionadas con el tipo de información que probamos sobre los datos y las herramientas utilizadas. En la sección de los resultados podremos observar cómo se comportaron los

datos con los análisis que probamos. Y en conclusión determinamos el sentido de nuestras hipótesis.

Antecedentes

SmartFilms se ha convertido en el festival de cine más importante del momento en Colombia, gracias a los diferentes escenarios que les permiten a los participantes exhibir todo tipo de contenidos, usando los valores y conceptos de las marcas patrocinadoras de las diferentes categorías por medio del storytelling, product placement y branded content, de esta manera, marcar un nuevo camino hacia el crecimiento de las industrias, utilizando las empresas privadas como aliado en la producción cinematográfica. Cinco

Los participantes deben realizar un cortometraje de máximo 5 minutos, incluyendo créditos y este debe ser grabado en su totalidad con un celular o dispositivo móvil, adjuntar un making of (detrás de cámaras), adjuntar cartel de propaganda y enviar el cortometraje antes del cierre de la convocatoria.

El festival es anual. En cada proceso se realizan actividades pedagógicas y de activación para incentivar a las personas a que participen y fomenten la industria cinematográfica

Literatura relacionada

A través del tiempo, el cine ha sido considerado el séptimo arte, la octava maravilla y por ello se han creado certámenes para apreciar el trabajo que conlleva realizar un film. El certamen más importante en el mundo del cine son los premios

Email address: luis.gutierrezrd@uanl.edu.mx (L.A. Gutierrez-Rodriguez)

Oscar. Depende de la academia determinar cual film es ganador de ciertas categorías como iluminación, guión, director, actores, entre otros. Las tecnologías de la información son un rubro tan amplio que se extiende al cine. Se puede analizar la composición de imagen de un film no solo con el ojo de un experto, si no también con visión computacional. En el 2013 se publicó un artículo [7] donde se hizo un análisis predictivo y se buscaba determinar quiénes serían los ganadores de las diversas categorías. Esta aplicación también ilustró cómo Data Science podría implementarse en las industrias de medios y entretenimiento.

En el libro *Movie analytics: a hollywood introduction to big data* [8] se aplicaron la minería de datos, minería de textos y análisis de redes sociales para aprender a analizar los datos de las películas. Se buscó una relación entre la información obtenida del análisis y la cantidad de estrellas otorgadas por el público en la IMDB. Además compararon lo aprendido con datos reales obtenidos de una película Francesa. En el capítulo *Oscar Prediction and Prediction Markets* [9] se examinó el papel de los mercados de predicción en la evaluación de la probabilidad de que una película nominada reciba un premio de la Academia.

Metodología

En esta sección veremos la información particular de los datos, y que herramientas utilizamos y con que fin. Se aplicó un proceso de limpieza de datos ya que estos presentaban espacios vacíos, errores de tipo de dato o se encontraban formas en las que se llenó un campo, siendo varias de ellas diferentes representaciones de la misma información. Finalmente, de acuerdo a que hipótesis se estaba probando, se seleccionaba solo algunos campos de los registros.

Datos

Contamos con cinco documentos en formato de hoja de cálculo, los cuales son los registros al festival de cine. Estos datos representan los registros desde el año 2015 al 2018.

Herramientas

Utilizamos las librerías Numpy[2] y Pandas[3] para la captura y almacenamiento de los datos limpios y poder trabajar con ellos.

Importamos la librería Scipy[4] para poder realizar análisis estadísticos y Matplotlib[1] para graficar los resultados.

Se implementó SKLearn[5] para realizar clasificaciones y agrupamientos de los datos en base a su comportamiento.

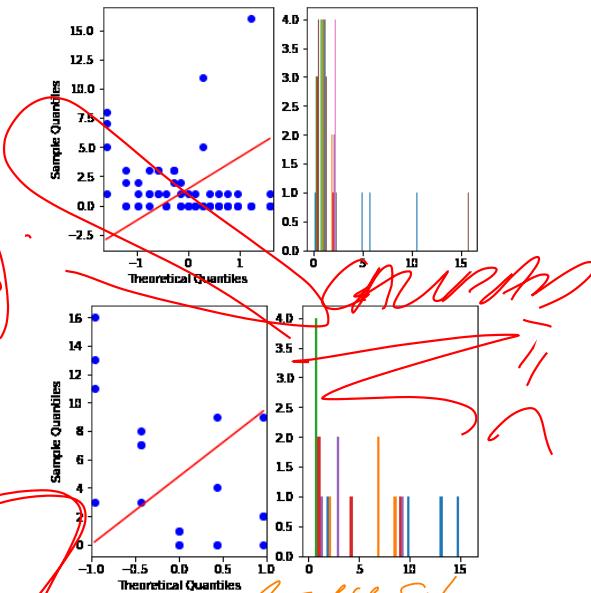
Resultados

Se graficaron los datos para ver si se podía concluir algo sobre éstos. Después se seleccionaron los participantes extranjeros de todos los cuatro años, y se ordenaron por el país de procedencia.

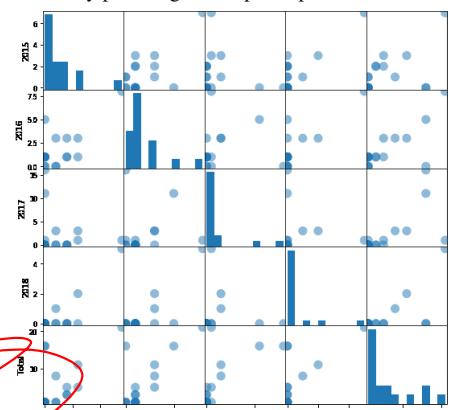
Categorizaciones

Como tenemos muchas cadenas de texto en nuestros datos, es importante hacer categorizaciones ya que hacemos conteos de la información que tenemos disponible y así poder hacer cálculos estadísticos. Aplicamos una categorización por país, pero se removieron los datos que involucran a México y Colombia que son los países anfitriones del concurso y por ende son los que nos sesgan la información, entonces esta información será relevante a los países extranjeros que participan.

Con estas categorías pudimos buscar modelos lineales y regresión múltiple.



Primeros probamos la normalidad de los datos por país de procedencia y por categoría de participación.

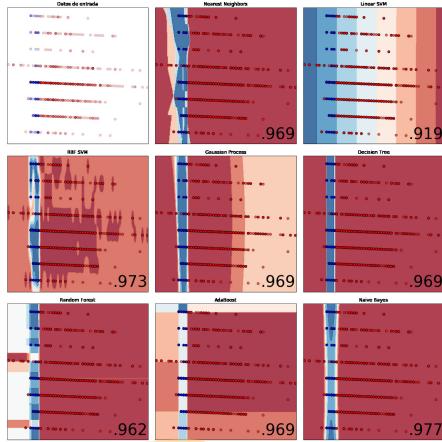


Se hizo una regresión lineal múltiple entre los años de participación, para saber si existía correlación.

Lectura (Resumen)

Machine Learning

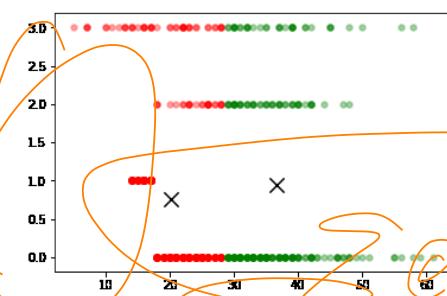
También se usó la librería `sklearn` para ver si las clasificaciones de las categorías ayudaban a entrenar a una red neuronal para que cuando se le entregaran otro conjunto de datos los clasificara correctamente.



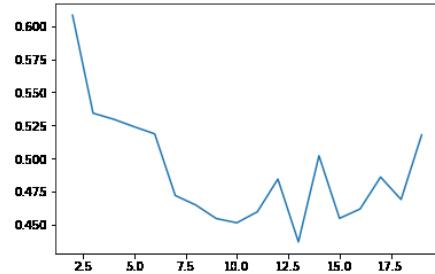
Posteriormente, seguimos utilizando la librería para aplicar algoritmos de agrupamiento, para que los datos se ordenaran de acuerdo a sus propias características.

Algoritmo K-Means

Este algoritmo agrupa los datos al tratar de separar muestras en n grupos de igual varianza, minimizando un criterio conocido como la inercia o la suma de cuadrados dentro del grupo. Este algoritmo requiere que se especifique la cantidad de grupos. Se adapta bien a un gran número de muestras y se ha utilizado en una amplia gama de áreas de aplicación en muchos campos diferentes.

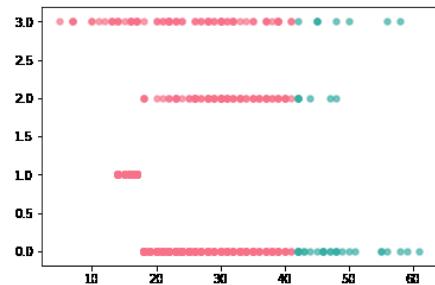


Al aplicar el algoritmo de K-medias, el valor de K usado fue 2. También evaluó cambiar el valor de K, y obtuvimos la siguiente gráfica.



Algoritmo Affinity Propagation

Este algoritmo crea clústeres enviando mensajes entre pares de muestras hasta la convergencia. Luego se describe un conjunto de datos utilizando un pequeño número de ejemplares, que se identifican como los más representativos de otras muestras. Los mensajes enviados entre pares representan la idoneidad para que una muestra sea el ejemplar de la otra, que se actualiza en respuesta a los valores de otros pares. Esta actualización ocurre de manera iterativa hasta la convergencia, momento en el que se eligen los ejemplares finales y, por lo tanto, se proporciona la agrupación final.



Se aplicó el algoritmo Affinity Propagation sobre los mismos datos que el K-medias, donde también se dividió en 2 grupos, pero la división de esos grupos se movió a mayor edad.

Conclusiones

La información proporcionada que se procesó fue insuficiente. Contábamos con aproximadamente 4600 registros de los cuales solo el 10 % tenía la información completa.

Debido a que fue el primer año en registrarse el año 2015 fue el cuello de botella para procesar el resto de la información.

El año 2016, se tuvo mayor participación en las categorías infantil y juvenil.

En el año 2017, la cantidad de participantes extranjeros aumentó.

En 2018, debido a que se contemplaron los dos festivales, el Colombiano y el Mexicano, la cantidad de extranjeros que participaron disminuyó, porque se decidió iniciar un festival nuevo en el país que más extranjeros aportaba al festival.

Se deberían agregar limitaciones a los campos de categoría de este festival, como los géneros de los filmes, las Marcas de

celulares y limitar la sinopsis a una cantidad precisa de caracteres.

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por haberme financiado una beca sin la cual no podría haber analizado estos datos.

Agradezco al Juan Beltran, Director creativo de Valencia Producciones y al festival SmartFilm, por haberme proporcionado los datos para su análisis.

Referencias

- [1] Matplotlib. URL: <https://matplotlib.org/>.
- [2] Numpy. URL: <https://www.numpy.org/>.
- [3] Python data analysis library. URL: <https://pandas.pydata.org/>.
- [4] Scipy.org. URL: <https://www.scipy.org/>.
- [5] Sklearn. URL: <https://scikit-learn.org/stable/>.
- [6] Vasant Dhar. Data science and prediction. 2012.
- [7] Michael Gold, Ryan McClaren, and Conor Gaughan. The lessons oscar taught us: Data science and media & entertainment. *Big Data*, 1(2):105–109, 2013.
- [8] Dominique Haughton, Mark-David McLaughlin, Kevin Mentzer, and Changan Zhang. *Movie analytics: a hollywood introduction to big data*. Springer, 2015
- [9] Dominique Haughton, Mark-David McLaughlin, Kevin Mentzer, and Changan Zhang. *Oscar Prediction and Prediction Markets*, pages 37–39. 01 2015.

(@)

Análisis de datos de un concurso de cine

L.A. Gutierrez-Rodriguez

*Posgrado de Ingeniería en Sistemas,
Facultad de Ingeniería Mecánica y Eléctrica,
Universidad Autónoma de Nuevo León*

Resumen

Contamos con los registros de un concurso de cine, cuyos filmes han sido grabados con dispositivos smartphone. Se busca identificar el alcance del certamen, ayudar a definir las mejores técnicas de difusión, caracterizar las categorías para equilibrar la cantidad de participantes, mejorar la forma en que se registraron los participantes y pronosticar la participación de los próximos años. La información se encuentra almacenada en cinco documentos de hoja de cálculo. Cada documento cuenta con los registros del concurso de cada año, del 2015 al 2018. Cuatro de ellos son del festival con sede en Colombia y el último del primer festival en México.

Se utiliza el lenguaje de programación Python y las librerías Numpy, Pandas, Scipy y Matplotlib para realizar análisis estadístico sobre los datos y poder graficar los resultados. Se aplican modelos de regresión lineal y múltiple. Se analiza la varianza y las componentes principales de los datos. Se pronostica la cantidad de participantes de futuros años. Se implementa la librería SKLearn para clasificar y agrupar los datos. Se realiza análisis de texto de las sinopsis e imágenes de los videos ganadores. Al aplicar técnicas de estadística descriptiva se puede determinar que la mayoría de extranjeros que participan en el concurso son de México.

Introducción

En la actualidad contamos con una cantidad inmensa de datos debido a que almacenamos toda clase de información. Se busca darle un sentido útil a estos datos para poder comprenderlos. La ciencia de datos es el estudio de la extracción generalizable de conocimiento a partir de datos [6]. En este trabajo se busca aplicar la ciencia de datos ayudar en la toma de decisiones de dónde y cuándo se deben organizar los eventos, la mejor forma de organizar las categorías y en qué otros países se puede expandir el festival.

Se sabe que la mayoría de los concursantes son de regiones cercanas a la capital de Colombia, que estos concursantes usan smartphones para grabar sus videos, que proporcionan una sinopsis, y cada participante ubica su filme en un género. Además, no se tiene identificación de los participantes, y por esto no se puede trazar su participación a lo largo de los festivales.

En antecedentes hablaremos de todo lo relacionado con el festival de cine. En la literatura relacionada, se vera que problemas ya han sido tratados y como la obtención de información ayudó a la toma de decisiones. Además, se tratan temas que están relacionadas con el tipo de información que buscamos. En la metodología aplicada se mencionan las hipótesis que probamos sobre los datos y las herramientas utilizadas. En la sección de los resultados, se puede observar cómo se comportaron los datos con los análisis que probamos. Y en conclusión determinamos el sentido de nuestras hipótesis.

Email address: luis.gutierrezrd@uanl.edu.mx (L.A. Gutierrez-Rodriguez)

Antecedentes

SmartFilms [11] se ha convertido en el festival de cine más importante del momento en Colombia, gracias a los diferentes escenarios que les permiten a los participantes exhibir todo tipo de contenidos, usando los valores y conceptos de las marcas patrocinadoras de las diferentes categorías por medio del storytelling, product placement y branded content, de esta manera, marcar un nuevo camino hacia el crecimiento de las industrias, utilizando las empresas privadas como aliado en la producción cinematográfica.

Los participantes deben realizar un cortometraje de máximo cinco minutos, incluyendo créditos y este debe ser grabado en su totalidad con un celular o dispositivo móvil, adjuntar un making of (detrás de cámaras), adjuntar cartel de propaganda y enviar el cortometraje antes del cierre de la convocatoria.

El festival es anual. En cada proceso se realizan actividades pedagógicas y de activación para incentivar a las personas a que participen y fomenten la industria cinematográfica

Literatura relacionada

A través del tiempo, el cine ha sido considerado el séptimo arte, la octava maravilla y por ello se han creado certámenes para apreciar el trabajo que conlleva realizar un film. El certamen más importante en el mundo del cine son los premios Oscar. Depende de la academia determinar cual film es ganador de ciertas categorías como iluminación, guión, director, actores, entre otros. Las tecnologías de la información son un rubro tan amplio que se extiende al cine. Se puede analizar la composición de imagen de un film no solo con el ojo de un experto, si

no también con visión computacional. En [7] se hizo un análisis predictivo y se buscaba determinar quienes serían los ganadores de las diversas categorías. Esta aplicación también ilustró cómo la Ciencia de Datos podría implementarse en las industrias de medios y entretenimiento.

En [10] se aplicó la minería de datos, minería de textos y análisis de redes sociales para aprender a analizar los datos de las películas. Se buscó una relación entre la información obtenida del análisis y la cantidad de estrellas otorgadas por el público en la IMDB. Además compararon lo aprendido con datos reales obtenidos de una película Francesa. En [9] se examinó el papel de los mercados de predicción en la evaluación de la probabilidad de que una película nominada reciba un premio de la Academia.

Metodología

En esta sección veremos la información particular de los datos, y que herramientas utilizamos y con qué fin. Se aplicó un proceso de limpieza de datos ya que estos presentaban espacios vacíos, errores de tipo de dato o se encontraban formas en las que se llenó un campo, siendo varias de ellas diferentes representaciones de la misma información. Finalmente, de acuerdo a que hipótesis se estaba probando, se seleccionaba solo algunos campos de los registros.

Datos

Contamos con cinco documentos en formato de hoja de cálculo, los cuales son los registros al festival de cine. Estos datos representan los registros desde el año 2015 al 2018.

Herramientas

Utilizamos las librerías Numpy [2] y Pandas [3] para la captura y almacenamiento de los datos limpios y poder trabajar con ellos.

Importamos la librería Scipy [4] para poder realizar análisis estadísticos y Matplotlib [1] para graficar los resultados.

Se implementó SKLearn [5] para realizar clasificaciones y agrupamientos de los datos en base a su comportamiento.

Resultados

Se graficaron los datos para ver si se podía concluir algo sobre éstos. Después se seleccionaron los participantes extranjeros de todos los cuatro años, y se ordenaron por el país de procedencia.

Categorizaciones

Como se tienen muchas cadenas de texto en nuestros datos, es importante hacer categorizaciones ya que se hacen conteos de la información que se tiene disponible y así poder hacer cálculos estadísticos. Se aplica una categorización por país, pero se removieron los datos que involucran a México y Colombia que son los países anfitriones del concurso y por ende son los que sesgan la información, entonces esta información será relevante a los países extranjeros que participan.

Regresión Lineal Simple

La regresión lineal simple es un modelo de regresión lineal con una sola variable explicativa. Es decir, se trata de puntos de muestra bidimensionales con una variable independiente y una variable dependiente y encuentra una función lineal (una línea recta no vertical) que, con la misma precisión que posible, predice los valores de las variables dependientes en función de las variables independientes. El adjetivo simple se refiere al hecho de que la variable de resultado está relacionada con un solo predictor.

Con estas categorías se pudo buscar Modelos lineales y regresión múltiple.

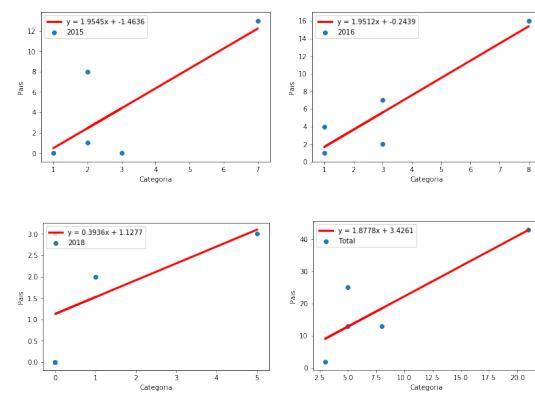


Figura 1: Modelos de regresión lineal

En la Figura 1 se observa como las rectas no quedaron significativas. En la Figura 2 se muestran las distribuciones de participantes por año.

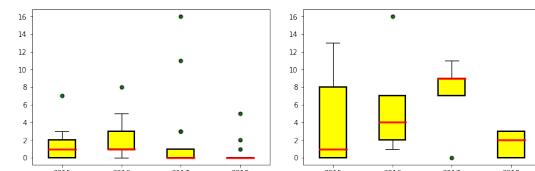


Figura 2: Distribución de participantes por año

Primero se probó la regresión de los datos por país de procedencia y por categoría de participación. Después se hizo un gráfico de bigote para entender la distribución de los datos.

Regresión Lineal Múltiple

La regresión lineal permite trabajar con una variable a nivel de intervalo o razón. De la misma manera, es posible analizar la relación entre dos o más variables a través de ecuaciones, lo que se denomina regresión múltiple o regresión lineal múltiple. Constantemente en la práctica de la investigación estadística, se encuentran variables que de alguna manera están relacionadas entre sí, por lo que es posible que una de las variables puedan

relacionarse matemáticamente en función de otra u otras variables.

Los mínimos cuadrados ordinarios (OLS por sus siglas en inglés) es un método para encontrar los parámetros poblacionales en un modelo de regresión lineal. Este método minimiza la suma de las distancias verticales entre las respuestas observadas en la muestra y las respuestas del modelo. El parámetro resultante puede expresarse a través de una fórmula sencilla, especialmente en el caso de un único regresionador.

En el Anexo A , se puede observar la prueba OLS que se le hizo a los parámetros. En esa prueba quedaron varios factores como No significativos, por eso se realiza una gráfica de dispersión de los datos para comprender su naturaleza, como se puede ver en la Figura 3.

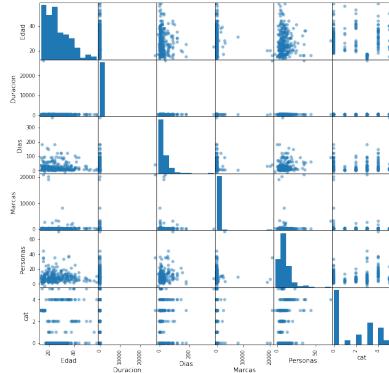


Figura 3: Modelos de regresión lineal múltiples

Se hizo una regresión lineal múltiple entre los años de participación, para saber si existía correlación.

Aprendizaje Maquina

También se usa la librería *sklearn* para ver si las clasificaciones de las categorías ayudaban a entrenar a una red neuronal para que cuando se le entregaran otro conjunto de datos los clasificara correctamente.

Posteriormente, seguimos utilizando la librería para aplicar algoritmos de agrupamiento, para que los datos se ordenaran de acuerdo a sus propias características.

Algoritmo K-Means

Este algoritmo [8] agrupa los datos al tratar de separar muestras en n grupos de igual varianza, minimizando un criterio conocido como la inercia o la suma de cuadrados dentro del grupo. Este algoritmo requiere que se especifique la cantidad de grupos. Se adapta bien a un gran número de muestras y se ha utilizado en una amplia gama de áreas de aplicación en muchos campos diferentes. En la Figura 4 en los incisos (a) se puede observar como se realizaron las K-medias, en el (b) se grafica el comportamiento de la precisión de la clasificación al variar la k.

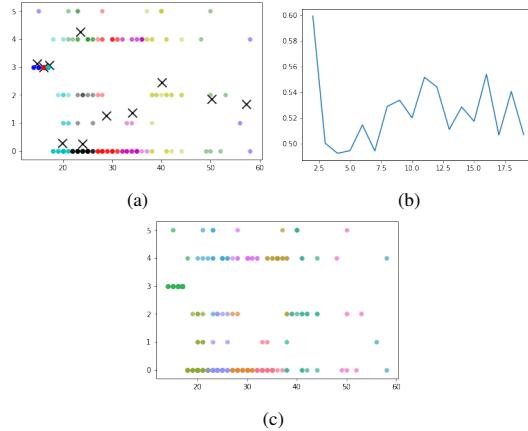


Figura 4: Resultados de los métodos de aprendizaje maquina

Al aplicar el algoritmo de K-medias, el valor de K usado fue dos. También evaluó cambiar el valor de K, y se obtuvo la siguiente gráfica.

Algoritmo Propagación de Afinidad

Este algoritmo crea clústeres [12] enviando mensajes entre pares de muestras hasta la convergencia. Luego se describe un conjunto de datos utilizando un pequeño número de ejemplos, que se identifican como los más representativos de otras muestras. Los mensajes enviados entre pares representan la idoneidad para que una muestra sea el ejemplar de la otra, que se actualiza en respuesta a los valores de otros pares. Esta actualización ocurre de manera iterativa hasta la convergencia, momento en el que se eligen los ejemplares finales y, por lo tanto, se proporciona la agrupación final. En la Figura 4 en el inciso (c), se muestra como el Algoritmo determinó que también deberían ser once grupos, pero diferentes al de K-medias.

Se aplica el algoritmo Propagación de Afinidad sobre los mismos datos que el K-medias, donde también se dividió en dos grupos, pero la división de esos grupos se movió a mayor edad. En la Figura 7 se pueden ver las precisiones obtenidas en cada método probado sobre el mismo conjunto de datos. En el Anexo B se pueden ver las clasificaciones que hicieron los métodos.

Método de Clasificación	Etiqueta1	Etiqueta2	Etiqueta3	Etiqueta4
Nearest Neighbors	.758	.969	.862	.846
Linear SVM	.527	.919	.904	.869
RBF SVM	.742	.973	.858	.835
Gaussian Process	.796	.969	.904	.892
Decision Tree	.735	.969	.896	.862
Random Forest	.8	.954	.892	.865
AdaBoost	.781	.969	.9	.877
Naive Bayes	.650	.977	.904	.858

Figura 5: Resultados de los métodos de aprendizaje maquina

NLTK

El kit de herramientas de lenguaje natural, o más comúnmente NLTK, es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural (PLN) simbólico y estadísticos para el lenguaje de programación Python. NLTK incluye demostraciones gráficas y datos de muestra.

Al usar NLTK para hacer análisis de texto, se obtuvo que en las sinopsis de los filmes cuentan con palabras que son muy frecuentemente usadas. Al hacer un mosaico con estas palabras como se ve en la Figura 6.



Figura 6: Mosaicos NLTK

Análisis de Imágenes

Se realiza un análisis de imágenes de los filmes ganadores de cada categoría. Se utilizó YouTube para obtener los vídeos, y OpenCV para poder hacer el procesamiento.

OpenCV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos.

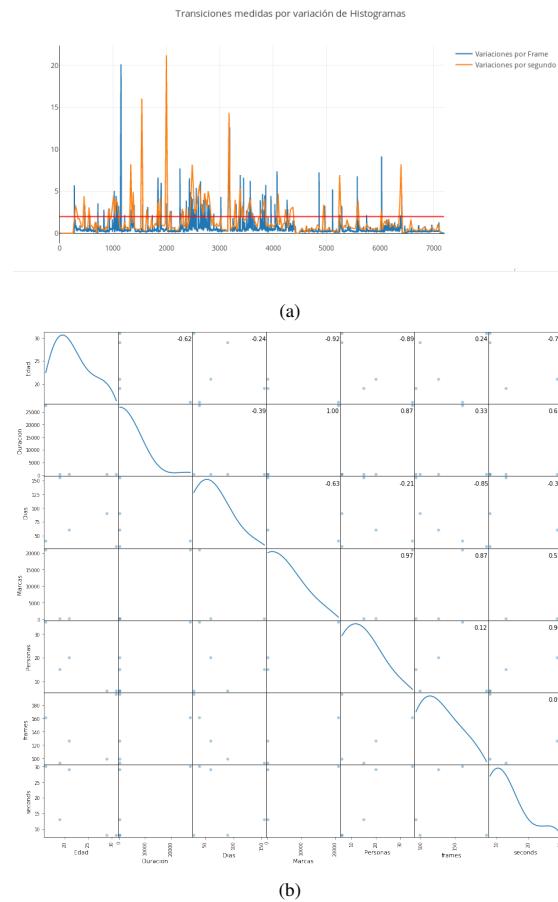


Figura 7: Resultados de los metodos de aprendizaje maquina

Conclusiones

La información proporcionada que se proceso fue insuficiente. Contábamos con aproximadamente 4600 registros de los cuales solo el 10 % tenía la información completa. Debido a que fue el primer año en registrarse el año 2015 fue el cuello de botella para procesar el resto de la información. El año 2016, se tuvo mayor participación en las categorías infantil y juvenil. En el año 2017, la cantidad de participantes extranjeros aumentó. En 2018, debido a que se contemplaron los dos festivales, el Colombiano y el Mexicano, la cantidad de extranjeros que participaron disminuyó, porque se decidió iniciar un festival nuevo en el país que más extranjeros aportaba al festival.

Se deberían agregar limitaciones a los campos de categoría de este festival, como los géneros de los filmes, las marcas de celulares y limitar la sinopsis a una cantidad precisa de caracteres.

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por haberme financiado una beca sin la cual no podría haber analizado estos datos.

Agradezco a Juan Beltrán, director creativo de Valencia Producciones y al festival SmartFilms, por haber proporcionado los datos para su análisis.

Referencias

- [1] Matplotlib. URL <https://matplotlib.org/>.
- [2] Numpy. URL <https://www.numpy.org/>.
- [3] Python data analysis library. URL <https://pandas.pydata.org/>.
- [4] Scipy.org. URL <https://www.scipy.org/>.
- [5] Sklearn. URL <https://scikit-learn.org/stable/>.
- [6] Vasant Dhar. Data science and prediction. 2012.
- [7] Michael Gold, Ryan McClaren, and Conor Gaughan. The lessons oscar taught us: Data science and media & entertainment. *Big Data*, 1(2):105–109, 2013.
- [8] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.
- [9] Dominique Haughton, Mark-David McLaughlin, Kevin Mentzer, and Changan Zhang. *Oscar Prediction and Prediction Markets*, pages 37–39. 01 2015.
- [10] Dominique Haughton, Mark-David McLaughlin, Kevin Mentzer, and Changan Zhang. *Movie analytics: a hollywood introduction to big data*. Springer, 2015.
- [11] Valencia Producciones. Smartfilms, 2019. URL <https://smartfilms.com.co/smartfilms>.
- [12] Kaijun Wang, Junying Zhang, Dan Li, Xinha Zhang, and Tao Guo. Adaptive affinity propagation clustering. *arXiv preprint arXiv:0805.1096*, 2008.

Anexo A

OLS Regression Results							
Dep. Variable:	Personas	R-squared:	0.502	Model:	OLS	Adj. R-squared:	0.496
Method:	Least Squares	F-statistic:	88.31	Date:	Wed, 05 Jun 2019	Prob (F-statistic):	1.48e-39
Time:	21:26:28	Log-Likelihood:	-952.98	No. Observations:	266	AIC:	1912.
Df Residuals:	263	BIC:	1923.	Df Model:	3	Covariance Type:	nonrobust
	coef	std err	t	P> t	[0.025	0.975]	
Duracion	0.0012	0.000	3.733	0.000	0.001	0.002	
cat	2.3984	0.259	9.245	0.000	1.888	2.999	
Dias	0.0824	0.013	6.282	0.000	0.057	0.108	
Omnibus:		74.119	Durbin-Watson:		1.738		
Prob(Omnibus):		0.000	Jarque-Bera (JB):		566.260		
Skew:		0.870	Prob(JB):		1.09e-123		
Kurtosis:		9.933	Cond. No.		824.		

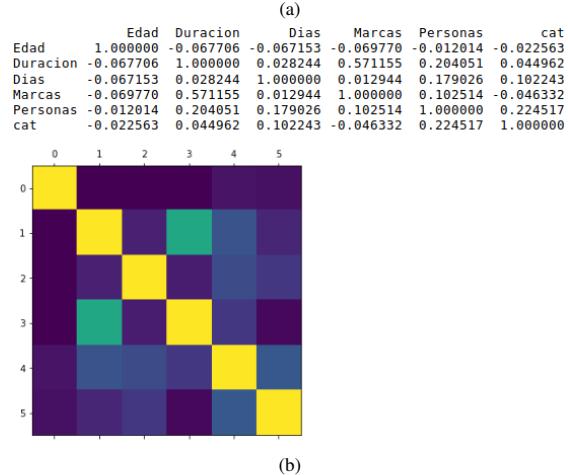
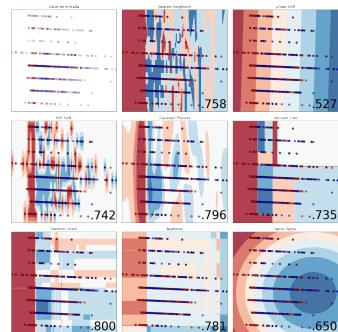
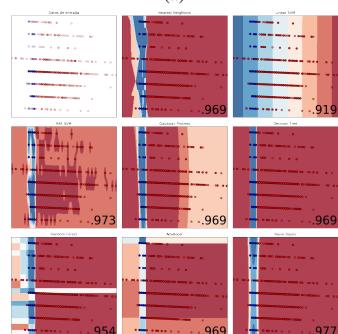


Figura 8: Modelos de regresión lineal múltiples

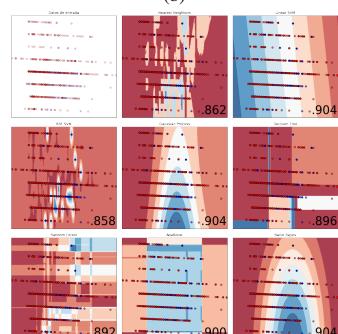
Anexo B



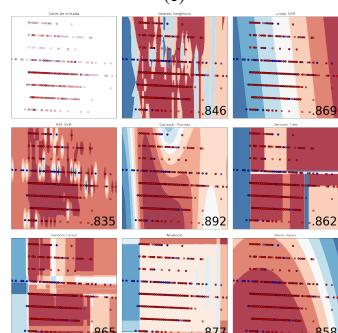
(a)



(b)



(c)



(d)

Figura 9: Gráficos de los métodos de aprendizaje maquina