prog_datasci_8_vis_entrega

May 31, 2021

1 Fundamentos de Programación

1.1 PAC 8 - Visualitzación de datos en Python

En este Notebook se encontraréis el conjunto de actividades evaluables como PEC de la asignatura. Veréis que cada una de ellas tiene asociada una puntuación, que indica el peso que tiene la actividad sobre la nota final de la PEC. Adicionalmente, hay un ejercicio opcional, que no tiene puntuación dentro de la PEC, pero que se valora al final del semestre de cara a conceder las matrículas de honor y redondear las notas finales. Podréis sacar la máxima nota de la PEC sin necesidad de hacer este ejercicio. El objetivo de este ejercicio es que sirva como pequeño reto para los estudiantes que quieran profundizar en el contenido de la asignatura.

Veréis que todas las actividades de la PEC tienen una etiqueta, que indica los recursos necesarios para llevarla a cabo. Hay tres posibles etiquetas:

- NM **Sólo materiales**: las herramientas necesarias para realizar la actividad se pueden encontrar en los materiales de la asignatura.
- EG **Consulta externa guiada**: la actividad puede requerir hacer uso de herramientas que no se encuentran en los materiales de la asignatura, pero el enunciado contiene indicaciones de dónde o cómo encontrar la información adicional necesaria para resolver la actividad.
- EI **Consulta externa independiente**: la actividad puede requerir hacer uso de herramientas que no se encuentran en los materiales de la asignatura, y el enunciado puede no incluir la descripción de dónde o cómo encontrar esta información adicional. Será necesario que el estudiante busque esta información utilizando los recursos que se han explicado en la asignatura.

Es importante notar que estas etiquetas no indican el nivel de dificultad del ejercicio, sino únicamente la necesidad de consulta de documentación externa para su resolución. Además, recordad que las **etiquetas son informativas**, pero podréis consultar referencias externas en cualquier momento (aunque no se indique explícitamente) o puede ser que podáis hacer una actividad sin consultar ningún tipo de documentación. Por ejemplo, para resolver una actividad que sólo requiera los materiales de la asignatura, puedéis consultar referencias externas si queréis, ya sea tanto para ayudaros en la resolución como para ampliar el conocimiento!

En cuanto a la consulta de documentación externa en la resolución de los ejercicios, recordad **citar siempre la bibliografía utilizada** para resolver cada actividad.

1.1.1 Ejercicio 1 (Seaborn)

A partir de una web de referencia de datos de sanidad (a https://github.com/datadista/datasets/tree/master/COVID%2019) hemos bajado datos históricos de casos de infección del virus SARS-cov-2 y su severidad, disponibles en el archivo el archivo `provincias_covid19_datos_sanidad_nueva_serie.csv`, donde verás los casos, hospitalizaciones, ingresos en UCI y defunciones, por provincia y fecha.

a) Carga los datos en un dataframe de pandas. Agrega (suma) casos, hospitalizaciones, ingresos en UCI por cada provincia, y utilizando Seaborn, representa:

- Un histograma de las defunciones (con `displot`)
- Un histograma de las hospitalizaciones (con `displot`)
- Un diagrama de puntos cruzando defunciones contra hospitalizaciones (con `jointplot`)

¿Qué representa un punto en la última gráfica?

EG **(0.5 puntos)**

[9]:
```
# Respuesta
```

b) Define una nueva variable llamada `sever` que contenga una string de entre`low`, `midlow`,`midhigh`, `high`, en función del *cuartil* de cada provincia respecto al número defunciones.

Vuelve a generar el último gráfico del apartado anterior, mostrando ahora una relación entre los casos de UCI y las hospitalizaciones (usando `scatterplot`).

Utiliza diferentes colores para mostrar el grado de severidad definido por la variable `sever`.

Sobre los datos mostrados, muestra una recta de regresión de todos los datos utilizando `regplot`. Qué podemos decir después de ver la gráfica?

EG
(1 punto)

[10]:
```
# Respuesta
```

c) Existe una clara correlación entre los cuatro índices que contiene la base de datos. Usa un mapa de calor o `heatmap` para visualizar las correlaciones entre los atributos numéricos de este último dataframe.

NM

(0.5 puntos)

[11]:
```
# Respuesta
```

d) Representa un plot de parejas mediante la función `pairplot`, marca el color de cada provincia en función de la severidad (variable `sever`).

NM **(0.5 puntos)**

[12]:
```
# Respuesta
```

1.1.2 Ejercicio 2 (NETWORKX)

Con el mismo archivo del ejercicio 1 podemos intentar calcular la correlación de casos entre provincias:

a) Calcula la correlación de casos entre todas las provincias a lo largo del tiempo. Visualiza la matriz de correlación de forma clusterizada con la función `clustermap` de Seaborn. Visualmente, qué provincias aparecen menos correlacionadas con el resto?

EG **(0.5 puntos)**

[13]:
```
# Respuesta
```

b) Construye un grafo con los elementos de la matriz de correlación anterior de forma que dos provincias estén conectadas cuando presenten una correlación mayor de 0.8. Genera una nueva visualización del grafo. Muestra las etiquetas de las provincias en cada nodo.

EI **(1 punto)**

[14]:
```
# Respuesta
```

c) Encuentra el componente conexo más grande del grafo (esto es, el subgrafo más grande). Representa sólo el subgrafo más grande de forma que los nodos tengan un tamaño proporcional al máximo de casos de cada provincia y el color indique de forma proporcional la mortalidad (usando por ejemplo un mapa de color secuencia, de forma que el color esté indicando el valor de la variable de interés, ver aquí).

EI **(1 punto)**

[15]:
```
# Respuesta
```

d) Calcula utilizando las herramientas de `networkx` la ruta de propagación del virus más probable a partir del grafo anterior de correlación de casos entre Zaragoza y Baleares. Muestra la primera ruta.

NOTA Puedes consultar https://networkx.org/documentation/networkx-1.10/reference/generated/networkx.algorithms.simple_paths.shortest_simple_paths.html para ver cómo buscar caminos entre nodos.

EG **(0.5 puntos)**

[16]:
```
# Respuesta
```

1.1.3 Ejercicio 3 (GEOPLOTLIB)

Intentaremos mostrar las provincias con un comportamiento relevante en un mapa mediante la librería `geoplotlib`.

a) Nos faltará una tabla con las coordenadas de longitud y latitud asociadas a cada provincia.

- Usar `geocode` degeopy para encontrar la latitud y longitud asociada a cada provincia. Añade esta información en un nuevo dataframe que tenga variables `provincia`,`lon` y `lat` (por `longitude`, `latitude`).

- Crear un tercer dataframe, con las siguientes columnas para cada uno de los datos de evolución de los que disponemos: `[Fecha, cod_ine, provincia, Casos, Fallecidos, Hospitalizados, UCI,lat,lon]`

NM **(1.5 puntos)**

[17]:
```
# Respuesta
```

b) Marca en un mapa las provincias que presentaron más de 10 casos en fecha 2020-05-01, utilizando el dataframe generado anteriormente y la librería geoplotlib.

NM **(1 punto)**

[18]:
```
# Respuesta
```

1.1.4 Ejercicio 4 (matplotlib)

En el último ejercicio representaremos la evolución a lo largo del tiempo de las variables del conjunto de datos usando `matplotlib`.

a) Construye un plot con 4 subplots diferentes (2x2, es decir, dos filas y dos columnas) mostrando la evolución temporal de cada índice para los casos de Madrid y Barcelona, indicando las fechas en el eje `x`:

```
---------------------------------- | Casos              | Defunciones |
----------------------------------| | Hospitalizaciones | UCI         |
----------------------------------
```

Recuerda etiquetar las figuras, ejes e incluir leyendas.

Nota: Puedes usar el módulo `datetime` para procesar la columna de fechas.

NM **(1 punto)**

[19]:
```
# Respuesta
```

b) Vemos que las series temporales tienen fuertes oscilaciones:

- Calcula la media en ventana deslizante a 14 días para cada uno de los índices a fin de reducir la oscilación de cada curva.
- Representa en dos plots:
  1. Los datos de Defunciones y UCI de Barcelona (en el eje `y`) contra las de Madrid (en el ejex), representando una evolución respecto de la otra.
  2. Los datos de casos y Hospitalizaciones de Barcelona (en el eje `y`) contra las de Madrid (en el ejex).

Recuerda etiquetar los plots, ejes e incluir leyendas.

Nota: puedes usar la función `pandas.rolling` para calcular las medias en ventanas deslizantes.

EG **(1 punto)**

[20]:
```
# Respuesta
```

Ejercicio Opcional

Dada la serie

$$Z_n = z_{n-1}^2 + c$$

y dado un elemento inicial de la serie $z_0 = 0 + i0$

1. Escribe una función $f(c, A, N_{max})$, que devuelva el número iteraciones N de la serie anterior necesarias para que el módulo $|z| > A$, siendo $A \in \mathbb{R}$.
2. Calcula y representa una imagen 2D representante N sobre el dominio de c comprendido por $Re(c)$ $subset[-2, 1]$, $Im(c)$ $subset[-1, 1]$ con los parámetros ($A = 2$, $n_{max} = 100$, $z_0 = i0$, Resolución de la imagen = (800.1200) pixels).

Nota: Te puede ser útil la función `meshgrid` de`numpy`.

EI

[21]:
```
# Respuesta
```