prog_datasci_8_vis_entrega_es

June 14, 2023

1 Unidad 8. Visualización

2 Fundamentos de Programación

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.

2.0.1 Ejercicio 1

Tenemos un conjunto de datos llamado `worldcities.xlsx` que contiene información sobre la mayoría de las ciudades de todo el mundo. A partir de estos datos, nos han solicitado que dibujemos un mapa de las ciudades españolas utilizando las librerías `pandas`, `geopandas` y `matplotlib`. **(2,5 puntos)**

El mapa resultado debe tener marcadas todas las ciudades del país con puntos de color azul, pero las que sean más importantes (`primary` y `admin`), tendrán los puntos de color rojo, de mayor tamaño que los demás y con los nombres al lado. Elija un tamaño de puntos que sea legible y no se superpongan mucho unos con otros.

El estilo del mapa debe ser de color blanco (sin relieve) con las líneas delimitadoras de color negro.

Para tener una guía del proceso, nos han proporcionado una serie de puntos a seguir:

a) Carga del conjunto de datos y visualización de las 10 primeras filas. NM **(0,25 puntos)**

b) Filtrar los datos para obtener las ciudades españolas. NM **(0,25 puntos)**

c) Dibujar mapa de España de color blanco con los contornos negros. EI **(0,5 puntos)**

d) Marcar todas las ciudades españolas con los puntos azules. EI **(0,5 puntos)**

e) Añadir los puntos rojos algo mayores de las ciudades más importantes. EI **(0,5 puntos)**

f) Poner los nombres en las ciudades más relevantes junto a cada punto rojo. EI **(0,5 puntos)**

Nota: Quizás tengas que instalar geopandas, como por ejemplo con `!pip install geopandas==0.13.0` desde el notebook

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

3 Ejercicio 2

Utilizando el DataFrame de worldcities, responde lo siguiente **(2,5 puntos)**:

a) Calcula aquellas capitales que tienen más de 1 millón de habitantes, da el resultado estratificado por tipos de capital. Utiliza las funciones de pandas para encontrar esta información.

NM **(0,5 puntos)**

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

b) Haz una gráfica usando seaborno que represente la distribución de población en función del tipo de capital usando violinplot y mostrando los datos de la distribución mediante por swarmplot. EG **(1 punto)**

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

c) Utiliza seaborn FacetGrid para crear un gráfico que muestre la distribución de población en función del tipo de capitalidad por cada país. El gráfico debe cumplir las siguientes condiciones:

- Aplica un filtro para mostrar únicamente los países con más de 1000 entradas en el conjunto de datos original.
- Utiliza un histograma para representar la distribución de población y define 10 bins.
- Añade etiquetas adecuadas a los ejes y títulos para las filas y columnas del FacetGrid.
- El eje X debe contener los nombres de los países y el eje Y debe representar el recuento de ciudades de cada país.
- Cada figura debe ser representada en escala logarítmica en ambos ejes.

EG **(1 punto)**

[23]:
```
# Respuesta
```

4 Ejercicio 3

El siguiente conjunto de datos (`attack_sample.csv`) contiene información sobre ataques que se han realizado en una red entre diferentes nodos durante un experimento. A continuación, responde a los siguientes puntos: **(2,5 puntos)**

a) Carga los datos a un dataframe y muestra las primeras 5 filas. NM **(0,25 puntos)**

[27]:
```
# Respuesta
```

b) Queremos visualizar los ataques que se han realizado entre los nodos en forma de grafo. Construye un grafo dirigido con la librería `networkx` donde los nodos (etiquetados) representen a los diferentes emisores/receptores de los ataques y las aristas indiquen entre qué nodos se ha producido. EI **(1 punto)**

[29]:
```
# Respuesta
```

c) Responde a las siguientes preguntas haciendo uso de las funciones y propiedades de la librería `networkx`: EI **(0,75 puntos)**

- Visualizando el grafo, ¿qué nodos no han recibido ningún ataque? También demuéstralo a través de código.
- ¿Cuántos ataques se han producido?
- ¿Qué nodo ha recibido más ataques? ¿Cuál ha producido más?

[31]:
```
# Respuesta
```

d) Vuelve a dibujar el grafo pero esta vez muestra el tamaño de los nodos en función de su grado. Es decir, un nodo con grado superior se pintará mayor que un nodo con grado inferior. EI **(0,5 puntos)**

[ ]:
```
# Respuesta
```

5 Ejercicio 4

Considera el archivo `evolucion.csv`. A partir del archivo:

a) importa el archivo a pandas y descríbelo. NM **(0,5 puntos)**

b) Usa ipywidgets para realizar un pequeño panel de datos en el que:

- Puedas selecciona una o más provincias disponibles basadas en los datos leídos con un widget
- Subas seleccionar una o más de las curvas disponibles con un segundo widget
- Crea un panel interactivo en el que al seleccionar las provincias y curvas a representar, se cree una figura mostrando la evolución de las curvas seleccionadas a lo largo del tiempo.
- Dentro de esta función, se filtra los datos en función de las provincias seleccionadas y se trazan las curvas correspondientes para cada provincia. Por último, se configura el estilo de la gráfica, se muestran las etiquetas y el título, y se muestra la gráfica.

Recuerda etiquetar los plotes, ejes e incluir leyendas. NM **(2 puntos)**

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

EG **(2 puntos)**

[ ]:
```
# Respuesta
```

Ejercicio Opcional

El triángulo de Sierpinski es un objeto fractal, que fue introducido por primera vez en 1915 por el matemático polaco Waclaw Sierpiński. El Tetraedro de Sierpiński es una generalización del fractal que se genera a partir de la subdivisión recursiva de un tetraedro. Empieza con un tetraedro inicial y después se divide repetidamente en tetraedros más pequeños, siguiendo un patrón específico. A cada paso de la subdivisión, se toman los puntos medio de las aristas para construir los nuevos tetraedros.

El resultado es un fractal en forma de tetraedros que se repiten a distintos niveles de detalle. Cada tetraedro más pequeño contiene una versión en miniatura del fractal completo. El Tetraedro de Sierpiński exhibe una estructura fractal autosimilar, con la misma forma que se repite en diferentes dimensiones y en una escala decreciente.

A medida que se incrementa el número de subdivisiones o niveles de detalle, el fractal resulta más complejo y muestra una intrincada estructura geométrica. Esta propiedad fractal, junto a su belleza visual y simetría, hace del Tetraedro de Sierpiński una figura fascinante y cautivadora para los matemáticos y entusiastas de los fractales.

Crea una representación del Tetraedro de Sierpiński usando matplotlib

Nota: Puedes generar una representación 3d con el argumento 'projection='3d' de subplot

EI

[ ]:
```
# Respuesta
```