ES-PEC4-enun

June 1, 2024

```
<div style="float: left; width: 50%;">
   <img src="https://www.uoc.edu/portal/system/modules/edu.uoc.presentations/resources/img/bra
</div>

<p style="margin: 0; padding-top: 22px; text-align:right;">22.403 · Programació para la Ciencia
<p style="margin: 0; text-align:right;">Grado en Ciencia de Datos Aplicada</p>
<p style="margin: 0; text-align:right; padding-button: 100px;">Estudios de Informática, Multime
```

1 Programación para la ciencia de datos - PEC4

En este Notebook encontraréis el ejercicio que supone la cuarta y última actividad de evaluación continuada (PEC) de la asignatura. Esta PEC intenta presentaros un pequeño proyecto en el cual debéis resolver diferentes ejercicios, que engloba muchos de los conceptos cubiertos durante la asignatura.

El objetivo de este ejercicio será desarrollar un **paquete de Python** fuera del entorno de Notebooks, que nos permita resolver el problema dado. Trabajaréis en archivos Python planos `.py`. Éste tendrá que incluir el correspondiente código organizado lógicamente (separado por módulos, organizados por funcionalidad,…), la documentación del código (*docstrings*) y tests. Además, tendréis que incluir los correspondientes archivos de documentación de alto nivel (`README`) así como los archivos de licencia y dependencias (`requirements.txt`) comentados en la teoría.

Hacer un `setup.py` es opcional, pero si se hace se valorará positivamente de cara a la nota de la práctica y del curso.

2 Enunciado

Queremos estudiar el comportamiento de la población de Estados Unidos respecto al uso de armas de fuego. Utilizaremos el siguiente dataset que ya ha sido incorporado en la carpeta del proyecto, y que proviene del siguiente enlace: * https://www.kaggle.com/datasets/pedropereira94/nics-firearm-background-checks

El dataset representa la acumulación de información (por fecha y estado) de la verificación de antecedentes de gente que quiere sacarse el permiso de armas. Se quiere ver si se puede sacar algún tipo de conclusión sobre diferencias de estado, evolución temporal, etc.

- la columna *permit* significa los permisos (verificación de antecedentes)
- la columna *handgun* serían las peticiones de armas cortas (pistolas)
- la columna *long_gun* serían las peticiones de armas largas

Estas serán las columnas que nos interesan a parte del mes (*month*) y del estado (*state*).

Además, en el ejercicio 5 utilizaremos también datos poblacionales para calcular los datos relativos. Para ello utilizaremos el siguiente dataset que también hemos incluido en el proyecto: * https://gist.githubusercontent.com/bradoyler/0fd473541083cfa9ea6b5da57b08461c/raw/fa5f59ff1ce7ad9ff792e223b state-populations.csv

El dataset presenta la población (2014) de los diferentes estados de los Estados Unidos y contiene las siguientes columnas:

- la columna *code*, un string de dos letras que identifica a cada uno de los estados (Por ejemplo California se identifica como CA y Florida como FL).
- la columna *state* con el nombre del estado sin abreviar.
- la columna *pop_2014*, representando el número de habitantes en el año 2014.

3 Proyecto Python, funcionalidad

Para hacer la entrega más fácil y homogénea os pedimos que organicéis el código de tal manera que **desde el fichero principal retorne todas las respuestas que se os pida en la PEC** haciendo uso de funciones que tendréis que definir en módulos. Para eso, en cada ejercicio os indicaremos el formato que tiene que tener cada respuesta, de tal manera que ejecutando `main.py` se vaya respondiendo a toda la PEC. Por defecto, `main.py` debe ejecutar todas las funciones de la PEC mostrando cómo funcionan pero también debe permitir ejecutarlas una a una si se desea. Debéis documentarlo todo muy bien en el *README* para que el profesor pueda ejecutar el código sin problemas y sin dudas. Os recordamos que en el *README* también tenéis que indicar cómo ejecutar los test y comprobar la cobertura de éstos.

3.1 Ejercicio 1: Lectura y limpieza de datos. (0.75 puntos)

3.1.1 Ejercicio 1.1. (0.25 puntos)

Implementad una función llamada *read_csv*: - **Inputs:** La función recibirá como datos de entrada un único parámetro que será la url del fichero que queremos leer. - **Funcionalidad:** La función deberá ser capaz de leer el fichero csv *nics-firearm-background-checks.csv*. Para comprobar que se han cargado correctamente esos datos, la función deberá mostrar por pantalla las cinco primeras filas de la base de datos así como su estructura. - **Outputs:** La función devolverá el dataframe que se ha leído.

3.1.2 Ejercicio 1.2. (0.25 puntos)

Implementad una función llamada *clean_csv*: - **Inputs:** Como datos de entrada la función recibirá la estructura de datos (dataframe). - **Funcionalidad:** La función deberá ser capaz de limpiar el dataset inicial, eliminando todas sus columnas excepto las cinco que utilizaremos a lo largo del ejercicio: *month, state, permit, handgun, long_gun*. Para aseguraros de que la función es correcta, se deberá mostrar por pantalla dentro de la misma el nombre de todas las columnas del dataframe. - **Outputs:** La función devolverá el dataframe conteniendo unicamente las columnas *month, state, permit, handgun, long_gun*.

3.1.3 Ejercicio 1.3. (0.25 puntos)

Implementad una función llamada *rename_col*: - **Inputs:** El dataframe con todas sus columnas. - **Funcionalidad:** La función deberá ser capaz de cambiar el nombre de la columna *longgun* por *long_gun* Para aseguraros de que la función es correcta, nos debemos de asegurar de que esa columna efectivamente existe en el dataframe. Asimismo, deberemos mostrar por pantalla el nombre de todas las columnas del dataframe dentro de la misma función. - **Outputs:** La función devolverá el dataframe con el nombre de la columna cambiado.

3.2 Ejercicio 2: Procesamiento de datos (1 punto)

La información de la columna meses se encuentra en un formato que no es demasiado manejable. Por ejemplo Febrero del año 2020 aparece como *2020-2*. Vamos a solucionar este problema:

3.2.1 Ejercicio 2.1 (0.5 puntos)

Implementad una función llamada *breakdown_date*: - **Inputs:** La función recibirá el dataframe conteniendo la columna *month* con el formato de datos igual que en el ejemplo. - **Funcionalidad:** La función dividirá la información que hay en la columna *month* creando dos nuevas columnas en el dataframe. Una de ellas llamada *year* y que contendrá el número del año y la otra columna llamada *month*. y que será el número del mes. Siguiendo nuestro ejemplo, para el valor *2020-2* la columna *year* deberá tener el valor **2020** y la columna *month* deberá tener el valor **2**. Para asegurarnos de que la función es correcta, será necesario mostrar las cinco primeras filas del dataframe resultante. - **Outputs:** El dataframe con la información de la fecha dividida en las dos columnas.

3.2.2 Ejercicio 2.2 (0.5 puntos)

Implementad una función llamada *erase_month*: - **Inputs:** La función recibirá el dataframe conteniendo la columna *month*. - **Funcionalidad:** Eliminar la columna month. Para comprobar que se ha realizado correctamente, la función deberá mostrar por pantalla también las cinco primeras filas de datos y el nombre de todas sus columnas. - **Outputs:** El dataframe sin la columna *month*.

3.3 Ejercicio 3: Agrupamiento de datos (1 punto)

3.3.1 Ejercicio 3.1 (0.5 puntos)

Implementad una función llamada *groupby_state_and_year* - **Inputs:** La función recibirá el dataframe obtenido en el ejercicio 2.2. - **Funcionalidad:** La función deberá ser capaz de calcular los valores acumulados totales agrupando los datos por año y por estado: (columnas *year* y *state*). - **Outputs:** El dataframe resultante con los datos agrupados.

3.3.2 Ejercicio 3.2 (0.25 puntos)

Implementad una función llamada *print_biggest_handguns* - **Inputs:** La función recibirá el dataframe con los datos agrupados por estado y por año como resultado del ejercicio 3.1. - **Funcionalidad:** La función deberá imprimir por pantalla un mensaje informativo indicando el nombre del estado y el año en donde se ha registrado un mayor numero de *hand_guns*. - **Outputs:** Esta función no devolverá ningún valor.

3.3.3 Ejercicio 3.3 (0.25 puntos)

Implementad una función llamada *print_biggest_longguns* - **Inputs:** La función recibirá el dataframe con los datos agrupados por estado y por año como resultado del ejercicio 3.1. - **Funcionalidad:** La función deberá imprimir por pantalla un mensaje informativo indicando el nombre del estado y el año en donde se ha registrado un mayor numero de *long_guns*. - **Outputs:** Esta función no devolverá ningún valor.

3.4 Ejercicio 4: Análisis temporal (1 punto)

Para este ejercicio se pedirá hacer un análisis temporal para ver la evolución de las licencias, pistolas y rifles de asalto a lo largo de los años. Para ello será necesario:

3.4.1 Ejercicio 4.1 (0.75 puntos)

Implementad una función llamada *time_evolution()* que cree un gráfico con las siguientes características: - El eje X será el número del año (que en el caso de este dataframe debería variar desde 1998 hasta 2020), mientras que en el eje y se mostrarán tres series temporales con el número total de *permit*, *hand_gun* y *long_gun* registrado por cada uno de los años.

3.4.2 Ejercicio 4.2 (0.25 puntos)

Comenta el gráfico generado en el ejercicio 4.2. ¿Vemos una correlación en las licencias, pistolas y rifles de asalto a lo largo de los años? ¿Es la tendencia ascendente o descendente? ¿Ha habido cambios durante la pandemia? ¿Que podríamos esperar en los próximos años?

Nota: En https://cnnespanol.cnn.com/2024/02/15/cultura-armas-estados-unidos-mundo-trax/ hay una gráfica sobre el número de víctimas de tiroteos masivos. En el 2017 hay un máximo, que parece coincidir con los resultados que habréis obtenido.

3.5 Ejercicio 5: Análisis de los estados (1.25 puntos)

A lo largo de este ejercicio aplicaremos un poco de ciencia de datos y sacaremos una serie de conclusiones agrupando los datos por cada uno de los estados:

3.5.1 Ejercicio 5.1 (0.25 puntos)

Implementad una función llamada *groupby_state* - **Inputs:** La función recibirá el dataframe con los datos agrupados por estado y por año como resultado del ejercicio 3.1. - **Funcionalidad:** La función mostrará los valores totales agrupando los valores unicamente por estado y no por año. Para comprobar que la función es correcta se pedirá también que muestre por pantalla las 5 primeras filas del dataframe resultante. - **Outputs:** Esta función deberá devolver el dataframe con los valores agrupados unicamente por estados.

Nota Los resultados obtenidos en la función del ejercicio 5.1 nos muestran únicamente los valores absolutos. Sin embargo, también hay que tener en cuenta que no todos los estados son igual de poblados. Para establecer una comparación justa, deberíamos de tener en cuenta también la población total de cada estado, para calcular así los valores relativos. Para ello, utilizaremos un nuevo conjunto de datos que hemos obtenido de la siguiente dirección: * https://gist.githubusercontent.com/bradoyler/0fd473541083cfa9ea6b5da57b08461c/raw/fa5f59ff1ce7ad9ff792e223b
state-populations.csv

3.5.2 Ejercicio 5.2 (0.25 puntos)

Los siguientes estados no aparecen en el archivo *us-state-populations.csv*: Guam, Mariana Islands, Puerto Rico y Virgin Islands. Por tanto, necesitaremos eliminarlos del dataframe para poder continuar con nuestro análisis de datos.

Implementad una función llamada *clean_states*: - **Inputs:** La función recibirá el dataframe con los datos agrupados por estado como resultado del ejercicio 5.1. - **Funcionalidad:** La función primero comprobará si existen esos cuatro estados (Guam, Mariana Islands, Puerto Rico y Virgin Islands) y, en el caso de que existan los eliminará. Para comprobar que la funcionalidad se ha implementado con éxito, la función también mostrará por pantalla el número de estados diferentes. - **Outputs:** Esta función devolverá el mismo dataset pero sin los cuatro estados mencionados.

3.5.3 Ejercicio 5.3 (0.25 puntos)

Ahora nuestro objetivo será fusionar los dos datasets:

Implementad una función llamada *merge_datasets*: - **Inputs:** La función recibirá como parámetros de entrada el conjunto de datos resultante del ejercicio ejercicio 5.2 y el conjunto de datos poblacionales provenientes del fichero: *us-state-populations.csv*. (Para leer los datos de la población puedes utilizar la función creada en el ejercicio 1.1). - **Funcionalidad:** La función fusionará los datos de los dos datasets recibidos como parámetros de entrada, incluyendo por cada estado toda la información procedente de las dos fuentes de datos. Para comprobar que se ha hecho correctamente, la función imprimirá por pantalla las cinco primeras filas del dataset resultante. - **Outputs:** Esta función devolverá el dataset resultante al fusionar los datos.

3.5.4 Ejercicio 5.4 (0.25 puntos)

A continuación necesitaremos calcular los valores relativos:

Implementad una función llamada *calculate_relative_values*: - **Inputs:** La función recibirá como parámetros de entrada, el conjunto de datos resultante del ejercicio ejercicio 5.3. - **Funcionalidad:** La función creará 3 nuevas columnas llamadas *permit_perc*, *longgun_perc* y *handgun_perc* (por si hay algún despistado que se confunda con la regla de tres como ya pasó con la PEC2 os voy a dar una pista, por ejemplo, en el caso de *permit_perc* los valores relativos se calcularían con la fórmula: (permit * 100) / poblacionTotal). - **Outputs:** Esta función devolverá el dataset resultante con las tres columnas nuevas: *permit_perc*, *loggun_perc* y *shotgun_perc* y los valores relativos ya calculados.

3.5.5 Ejercicio 5.5 (0.25 puntos)

1 - En primer lugar, calcularemos la media de permisos *permit_perc* con dos decimales y mostraremos el resultado en pantalla. 2 - En segundo lugar, mostraremos por pantalla toda la información relativa al estado de *Kentucky*.

Nota ¡Tenemos un problema técnico! El estado de *Kentucky* es lo que se llama un *outlier* o valor atípico. Los *outliers* son valores atípicamente altos que distorsionan cualquier tipo de métricas estadísticas. En este caso, la media está inflada debido a los valores que tiene este estado. Los *outliers* no solamente distorsionan las métricas estadísticas, también hacen que algoritmos de aprendizaje máquina lleguen a conclusiones erróneas y eso es un problema.

3- Reemplazar el valor *permit_perc* de *Kentucky* con el valor de la media de esta columna. 4- Volveremos a calcular la media con dos decimales. 5- ¿Ha cambiado mucho el valor? ¿Entiendes el proceso de quitar valores atípicos? Escribe tus conclusiones.

3.6 Ejercicio 6: Mapas coropléticos (1.5 puntos)

Geographic Data Science (GDS) es la rama de Ciencia de Datos que utiliza datos con información geográfica. En este enlace tienes disponible un curso de GDS, con un docker para que utilicéis los materiales (para cuando tengáis tiempo): *https://darribas.org/gds_course/content/home.html

Los mapas coropléticos son los mapas donde pintamos un área (estado, región, provincia, país) de un color, dentro de un mapa de colores, para obtener información visual.

Vamos a hacer 3 mapas coropléticos para *permit_perc*, *handgun_perc* y *longgun_perc*. Hay diferentes soluciones. Proponemos hacerlo con el código disponible en: * https://python-graph-gallery.com/292-choropleth-map-with-folium/

Para ello, vas a necesitar instalar las librerías *folium* y *selenium*.

El fichero *us-states.json* contiene toda la información de las fronteras de los estados. Lo primero que puedes hacer es hacer funcionar el ejemplo que se propone (*US_Unemployment_Oct2012.csv*) y luego adaptarlo a la información que disponemos.

Puedes generar 3 mapas (m1, m2 y m3), uno para cada variable. Luego estos mapas los puedes guardar a imagen con el siguiente código:

Tienes que obtener 3 imágenes, una para cada variable que tienes (*permit_perc*, *handgun_perc* y *longgun_perc*).

4 Criterios de corrección

Esta PEC se evaluará siguiendo los siguientes criterios:

- **Funcionalidad** (6.5 puntos): Se valorará que el código implemente todo lo que se pide.
  - Ejercicio 1 (0.75 puntos)
  - Ejercicio 2 (1 punto)
  - Ejercicio 3 (1 punto)
  - Ejercicio 4 (1 punto)
  - Ejercicio 5 (1.25 puntos)
  - Ejercicio 6 (1.5 puntos)
- **Documentación** (0.5 puntos): Todas las funciones de los ejercicios de esta PEC tendrán que estar debidamente documentadas utilizando docstrings (en el formato que prefiráis).
- **Modularidad** (0.5 puntos): Se valorará la modularidad del código (tanto la organización del código en módulos como la creación de funciones).
- **Estilo** (0.5 puntos): El código tiene que seguir la guía de estilo de Python (PEP8), exceptuando los casos donde hacerlo complique la legibilidad del código.
- **Tests** (1.5 puntos): El código tiene que contener una o diversas *suites* de tests que permitan comprobar que el código funciona correctamente, con un mínimo del 50% de cobertura.
- **Requerimientos** (0.5 puntos): Tenéis que incluir un fichero *requirements.txt* que contenga la lista de librerías necesarias para ejecutar el código.

- **README** y **LICENSE** (0.25 puntos): Tenéis que añadir también un fichero README, que presente el proyecto y explique cómo ejecutarlo, así como la inclusión de la licencia bajo la que se distribuye el código (podéis escoger la que queráis).

4.1 Importante

Nota 1: De la misma manera que en las PECs anteriores, los criterios transversales se valorarán de manera proporcional a la parte de funcionalidad implementada.

Por ejemplo, si el código sólo implementa la mitad de la PEC y la documentación está perfecta la puntuación correspondiente a documentación será de 0.25.

Nota 2: Es imprescindible que el paquete que libréis se ejecute correctamente en la máquina virtual y que el fichero de README explique claramente cómo ejecutar el código para generar los resultados pedidos. Además en el README tiene que explicarse también cómo se ejecutarán los test y cómo se comprueba su cobertura. **Lo primero que hará el profesor cuando corrija es leer el fichero README y seguir las instrucciones que allá se especifica.**

Nota 3: Entregad el paquete como un único archivo .zip que contenga sólo el código en el Registro de Evaluación Continua. **El código de Python tendrá que estar escrito en ficheros planos de Python.**