ES-PAC4-enun

December 14, 2020

1 Programación para la ciencia de datos - PEC4

En este Notebook encontraréis un ejercicio que supone la cuarta actividad de evaluación continuada (PEC) de la asignatura. Esta PEC consta de un único ejercicio a resolver, que engloba muchos de los conceptos cubiertos durante la asignatura.

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

Se nos pide que implementemos un paquete de Python que sea capaz de realizar un análisis de datos con información genética sobre el Bacilo de Koch. En particular, nos centraremos en las *pautas abiertas de lectura* de los genes del Bacilo de Koch. Las *pautas abiertas de lectura* son más conocidas por su acrónimo *ORF* que viene de su nombre en inglés *open reading frame*. El ORF es una secuencia de nucleótidos que potencialmente puede codificar una proteína. En los organismos eucariontes, como somos los seres humanos, cada gen tiene un único ORF. Pero este no es el caso de las bacterias, que son organismos procariontes.

2 Los datos

Los datos a analizar nos son proporcionados en dos colecciones de datos separadas: `tb_functions.pl` y `orfs/tb_data_0x.pl`. Estos datos provienen del repositorio UCI Machine Learning y tienen formado de *datalog*.

`tb_functions.pl` contiene información general sobre los genes y sus clases funcionales. Mientras que `tb_data_0X.pl` contiene información detallada sobre todos los genes indicados.

Echando un vistazo a los archivos proporcionados, podréis ver que los diferentes archivos contienen bastante información. Para resolver el ejercicio propuesto, sólo será necesario usar una cantidad muy reducida de esta.

2.1 tb_functions.pl

El archivo `tb_functions.pl` contiene información sobre 123 clases funcionales de ORFs y tiene la siguiente estructura:

```
class([1,0,0,0],"Small-molecule metabolism ").
class([1,1,0,0],"Degradation ").
class([1,1,1,0],"Carbon compounds ").
function(tb186,[1,1,1,0],'bglS',"beta-glucosidase").
function(tb2202,[1,1,1,0],'cbhK',"carbohydrate kinase").
function(tb727,[1,1,1,0],'fucA',"L-fuculose phosphate aldolase").
class([1,1,2,0],"Amino acids and amines ").
function(tb1905,[1,1,2,0],'aao',"D-amino acid oxidase").
function(tb2531,[1,1,2,0],'adi',"ornithine/arginine decarboxylase").
function(tb2780,[1,1,2,0],'ald',"L-alanine dehydrogenase").
function(tb1538,[1,1,2,0],'ansA',"L-asparaginase").
...
```

Donde hay dos tipos de entrada:

`class`: tiene 2 elementos separados por comas siempre presentados en el siguiente orden: * *identificador de la clase*: lista de 4 números que describe la clase en 4 dimensiones diferentes (separados por comas y entre corchetes), y * *descripción de la clase*: string que contiene la descripción de la clase, ninguna clase comparte descripción con otra clase (string entre comillas dobles).

`function`: tiene 4 elementos separados por comas siempre presentados en el siguiente orden: * *ORF*: pauta abierta de lectura (en inglés *open reading frame*) (string sin comillas), * *identificador de la clase*: lista de 4 números que describe la clase en 4 dimensiones diferentes (separados por comas y entre corchetes), * *nombre del gen*: nombre del gen o valor *null* si el gen no tiene nombre (string entre comillas simples), y * *descripción ORF*: descripción de la pauta de lectura (string entre comillas dobles).

2.2 tb_data_0X.pl

Los archivos `tb_data_0X.pl` tienen la siguiente estructura:

```
begin(model(tb4)).
tb_protein(tb4).
function(5,0,0,0,'null','null').
coding_region(4434,4994).
tb_mol_wt(19934).
...
sequence_length(187).
amino_acid_pair_ratio(a,a,24.8).
amino_acid_pair_ratio(a,c,0.0).
amino_acid_pair_ratio(a,d,0.0).
amino_acid_pair_ratio(a,e,18.6).
amino_acid_pair_ratio(a,f,0.0).
amino_acid_pair_ratio(a,g,12.4).
...
tb_to_tb_evalue(tb3671,1.100000e-01).
tb_to_tb_evalue(tb405,4.300000e-01).
tb_to_tb_evalue(tb3225,5.600000e-01).
...
```

```
species(p35925,'streptomyces_coelicolor').
classification(p35925,bacteria).
classification(p35925,firmicutes).
classification(p35925,actinobacteria).
classification(p35925,actinobacteridae).
classification(p35925,actinomycetales).
classification(p35925,streptomycineae).
classification(p35925,streptomycetaceae).
classification(p35925,streptomyces).
mol_wt(p35925,19772).
keyword(p35925,'hypothetical_protein').
db_ref(p35925,embl,127063,g436026,null).
signalip(c,35,no).
signalip(y,35,no).
signalip(s,54,no).
signalip(ss,1,34,no).
signalip(cleavage,null,null).
hydro_cons(-0.498,-0.474,0.624,3.248,0.278).
end(model(tb4)).
begin(model(tb5)).
...
end(model(tb3915)).
```

Donde los datos para un único ORF están capturados entre los delimitadores:

```
begin(model(ORF))
end(model(ORF))
```

Y el atributo `tb_tono_tb_evalue(ORF, E-value)` muestra la relación con otros ORFs.

3 Ejercicio

Primero, será necesario que leáis los archivos facilitados de la forma más óptima teniendo en cuenta las tareas pedidas. Tendréis que justificar vuestra decisión.

Después, tendréis que generar funciones que os permitan hacer los siguientes cálculos:

1. Dada la colección de clases funcionales:

   1.1 Calcular cuántos ORFs pertenecen a cada clase.

   1.2 Dado que el Bacilo de Koch afecta sobre todo a los pulmones, queremos que mostréis por pantalla cuántos ORFs pertenecen a la clase que tiene *Respiration* como descripción. Mostrad el resultado por pantalla debidamente formateado (utilizando el método `format()` u otro similar), incluyendo un mensaje explicativo de los valores que enseñáis.

2. Para cada patrón listado*, calcular:

   2.1 El número de clases que contienen como mínimo un ORF con el patrón indicado en su descripción.

2.2 El número promedio de ORFs con los cuales se relacionan los ORFs con el patrón indicado en su descripción.

* Los patrones para los cuales tendréis que resolver los cálculos 2.1 y 2.2 son:

  - La descripción contiene el término *protein*. Por ejemplo, el ORF con descripción *electron transfer flavoprotein alpha subunit* encajaría con esta definición.
  - La descripción contiene una palabra de 13 caracteres y esta contiene el término *hydro*. Por ejemplo, el ORF con descripción *3-hydroxyacyl-CoA dehydrogenase* encajaría con esta definición.

3. Para cada entero M entre 2 y 9 (ambos incluidos), calcula el número de clases que tienen como mínimo una *dimensión* mayor estricta (>) que 0 y a la vez múltiple de M. Con el término dimensión nos referimos a cada uno de los 4 números que forman el identificador de la clase (explicado en la sección anterior). Este cálculo tendrá que resultar en algo como:

```
M=2: ? clases
M=3: ? clases
...
M=9: ? clases
```

   donde ? representa un entero.

Además, tendréis que generar código que permita representar todos los resultados gráficamente (excepto para el cálculo 1.1). Para cada función hará falta que penséis y justifiquéis qué tipo de gráfica es la más adecuada para representar el resultado.

El código tendrá que estar correctamente comentado, incluyendo documentación de funciones, y correctamente testado usando la librería `unittest`. Los tests proporcionados tendrán que dar cobertura como mínimo al 50% de la funcionalidad propuesta.

3.1 Cobertura de los tests

La medición de la cobertura de los tests se utiliza para evaluar la eficacia de los tests propuestos. En particular, sirve para determinar la calidad de los tests desarrollados y para determinar las partes críticas del código que no han sido testadas. Para medir este valor, proponemos el uso de la herramienta `Coverage.py`. En la documentación podréis encontrar cómo instalarla y cómo usarla.

Para evaluar la calidad de los tests desarrollados en la PAC4, pedimos un mínimo del 50% de cobertura.

4 Uso de Git

Para poner en práctica lo que habéis aprendido en la Unidad 6 sobre `Git`, proponemos el uso de `GitHub Classroom` para desarrollar vuestro paquete de Python. GitHub Classroom es una herramienta gratuita de código abierto que ayuda a simplificar el uso educativo de GitHub. Hemos usado GitHub Classroom para crear una aula como esta y dónde hemos creado una tarea para la PAC4. Para hacer uso de este espacio que hemos creado, os aconsejamos seguir los pasos indicados en esta guía donde se explica cómo crear un repositorio para trabajar en la tarea que hemos

preparado. El enlace a la tarea lo encontraréis en el mensaje de la PAC4 publicado en el tablón del aula.

El uso de esta herramienta no es obligatorio para la evaluación de la PAC4, pero creemos que es una muy buena oportunidad para poner en práctica vuestros conocimientos en un entorno vital para todo el mundo que trabaje o quiera trabajar en el ámbito de la ciencia de datos.

5 Criterios de corrección

Esta PEC se valorará siguiendo los criterios siguientes:

- **Funcionalidad** (5 puntos): Se valorará que el código implemente correctamente lo que pide el enunciado.
  - Lectura ficheros (1 punto)
  - Ejercicio 1 (0.5 puntos)
  - Ejercicio 2 (1.5 puntos)
  - Ejercicio 3 (1 punto)
  - Visualizaciones (1 punto)
- **Documentación** (0.5 puntos): Todas las funciones de los ejercicios de esta PEC tendrán que estar correctamente documentadas utilizando docstrings (en el formato que prefiráis).
- **Modularidad** (1 punto): Se valorará la modularidad del código (tanto la organización del código en ficheros 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** (2 puntos): El código tiene que contener una o varias *suites* de tests que permitan comprobar el buen funcionamiento de las funciones implementadas, obteniendo un mínimo del 50% de cobertura.
- **Requerimientos** (0.5 puntos): Es necesario crear un fichero de requerimientos que liste (sólo) las librerías necesarias para ejecutar el código.
- **README** y **licencia** (0.5 puntos): Se valorará la creación de un fichero README, que presente el proyecto y explique cómo ejecutarlo, así como la inclusión de la licencia bajo la cual se distribuye el código (podéis elegir la que queráis).

5.1 Importante

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

Por ejemplo, si el código únicamente implementa la mitad de la funcionalidad pedida, y la documentación de esta parte es perfecta, entonces la puntuación correspondiente a la parte de documentación será de 0.25.

Nota 2: Es imprescindible que el paquete que entreguéis se ejecute correctamente en la máquina virtual, y que el fichero de README que incluyáis explique claramente cómo se tiene que ejecutar vuestro código para generar las gráficas resultantes del análisis.

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