22.403 - Programació per a Data Science PAC 4

Aquest és l'enunciat de la quarta activitat d'avaluació continuada de l'assignatura (PAC). Aquesta PAC esta composta de 6 exercicis relacionats amb la majoria de continguts vistos durant el curs.

Us recomanem que resoleu els exercicis per ordre. Tot i que no cal que sigueu totalment estrictes en el seguiment d'aquesta recomanació si que es important que feu l'exercici 1 el primer de tots donat que les funcions que s'hi demanen es fan servir en els exercicis posteriors. A més , l'exercici 6 es un exercici "obert" que us serà més fàcil de resoldre si ja teniu resolts els anteriors.

Aquesta PAC inclou una serie de fitxers de test que us serviran per comprovar que les funcions que programeu tenen un resultat correcte. Al final del document també trobareu detalls sobre *pylint*, l'eina que farem servir en aquest cas per comprovar que es segueix la guia d'estil PEP8.

A l'aula hi podeu trobar un video que repassa aspectes pràctics de programació fora de l'entorn dels jupyter notebooks. Aquest video també conte informació sobre com executar testos i com fer servir *pylint*. Us recomanem que hi feu un cop d'ull abans de començar la PAC.

Context

Treballeu per a una empresa que crea solucions "Data Science" per a projectes de tot tipus. L'equip directiu vol centrar-se en els propers anys en el món dels e-sports, de ràpid creixement, i ha decidit començar amb un que no els resultat una mica més conegut, el FIFA.

Ens han demanat que dissenyem eines per treballar amb aquest tipus de dades, tant en termes d'anàlisi exploratòria de dades com a llarg termini. Per això, ens han proporcionat un conjunt de dades que inclou tota la informació sobre els jugadors i les jugadores del videojoc entre els anys 2016 i 2022.

Exercici 1: lectura i pre-procés

El primer que cal fer es crear funcions que llegeixin i processin les nostre dades. Concretament, us demanem que implementeu:

a) Una funció que llegeixi un fitxer de jugadors que afegeix dues columnes a la informació obtinguda. Aquesta informació afegida s'expressara en dues noves columnes "gender" i "year" (que fan referencia al genere dels esportistes en el fitxer i a l'any en que es van publicar les dades). La funció ha de tornar el dataframe de pandas resultant d'afegir les dues columnes mencionades a la informacion inicial .

read_add_year_gender(filepath: str, gender: str, year: int) -> pd.DataFrame

- filepath: string amb la ruta de l'arxiu que volem llegir
- gender: 'M' o 'F' (segons les sigles de "Male" or "Female")
- year: Any al que corresponen les dades en format XXXX (per exemple, 2020)

b) Una funció que creï un sol dataframe amb les dades de tots els jugadors/es d'un mateix any. Aquest dataframe també ha de contenir informació sobre el gènere i any a què es corresponen les dades de cada futbolista.

join_male_female(path: str, year: int) -> pd.DataFrame

- path: ruta a la carpeta que conté les dades
- year: any del que es volen llegir les dades, format XXXX (per exemple, 2020)

c) Una funció que llegeixi la informació corresponent a futbolistes d'ambdós gèneres durant diversos anys i que torni un únic dataframe. Aquest dataframe ha de contenir, a més, informació sobre el gènere i any a què es corresponen les dades de cada futbolista.

join_datasets_year(path: str, years: list) -> pd.DataFrame

- path: ruta a la carpeta que conté les dades
- years: llista d'anys que es volen incloure en el dataframe, en format [XXXX,...]

Exercici 2: estadistica basica

Volem crear una sèrie de funcions que ens permetin obtenir certes estadístiques bàsiques del conjunt de futbolistes. En concret:

a) Una funció que, rebi un dataframe i, donat el nom d'una columna numèrica, ens retorni la/es fila/es en què el seu valor és màxim. A més, la funció rebrà com a argument una llista de noms de columnes. El dataframe que retorni la funcion només ha de contenir aquestes columnes.

find_max_col(df: pd.DataFrame, filter_col: str, cols_to_return: list) -> pd.DataFrame

- df: *dataframe* que conté les dades
- filter_col: nom de la columna de la que volem saber el màxim
- cols_to_return: llista de columnes que cal retornar

b) Una funció per a filtrar les dades amb filtres avançats. La funció rebrà una query (o consulta) en forma de tupla. El primer element de la tupla serà una llista de columnes sobre les que volem filtrar. El segon element serà la llista de valors que volem fer servir al filtre. Si la columna és categòrica, el valor serà un string. Si és numèrica, serà una tupla amb el valor mínim i màxim (tots dos han de ser inclosos al filtre).

Per exemple, la query

([“league_name”, “weight_kg”], [“English Premier League”, (60, 70)])

Tornaria els jugadors i les jugadores de la lliga “English Premier League” amb un pes entre 60 i 70 quilos (inclosos).

Igual que a l'exercici 2a, s'inclou un argument que ens indica quines columnes volem que ens torni la funció.

find_rows_query(df: pd.DataFrame, query: tuple, cols_to_return: list) -> pd.DataFrame

- df: dataframe que conté les dades
- query: tupla que conté la query
- cols_to_return: llista de columnes que cal retornar

c) Considerant tot el conjunt de dades proporcionat (des de l'any 2016 fins al 2022, ambdós inclosos), mostreu per pantalla el “short_name”, “year”, “age”, “overall” i “potential” de:

- Els jugadors de nacionalitat belga menors de 25 anys màxim “potential” al futbol masculí.
- Les porteres majors de 28 anys amb “overall” superior a 85 al futbol femení.

Exercici 3: BMI

Els/Les esportistes professionals segueixen un rigorós control dietètic i realitzen més activitat física que la població general. Volem estudiar si això resulta en un índex de massa corporal (BMI per les sigles en anglès) diferent del de la població general.

En primer lloc, cal crear una funció que, donats: un dataframe amb les dades, un gènere i un any,retorni un dataframe que inclogui una columna amb l'índex de massa corporal (BMI) de cada futbolista d'aquest gènere i any. La funció també rebrà com a argument una llista de columnes. El dataframe que retorni la funció han de contenir aquestes columnes, a més de la nova columna BMI.

El BMI d'una persona es calcula de la següent manera:

$$BMI = \frac{pes}{alçada * alçada}$$

Nota: Cal expressar el pes en quilograms i l'alçada en metros.

calculate_BMI(df: pd.DataFrame, gender: str, year: int, cols_to_return: list) -> pd.DataFrame

- df: dataframe que conté les dades
- gender: gènere que volem estudiar
- year: any que volem estudiar en format XXXX (per exemple 2020)
- cols_to_return: llista de columnes que cal retornar (sense columna BMI)

a) Mostreu una gràfica amb el BMI màxim per país. Filtreu per gènere masculí i any 2022. La informació sobre el país on juga cada futbolista (no confondre amb la nacionalitat del jugador) es pot extreure de la columna club_flag_url. Per exemple, la url https://cdn.sofifa.net/flags/fr.png correspon a França (fr). No cal obtenir el nom complet del país, amb l'abreviatura n'hi ha prou.

Considerant la següent classificació:

Category	BMI
underweight	< 18.5
normal weight	[18.5, 25)
overweight	[25, 30)
obese	≥ 30

Trobeu sorprenents els valors obtinguts? Per què?

b) Compareu en una gràfica el BMI del conjunt de futbolistes que desitgeu amb el de la població espanyola. Podeu descarregar les dades de la pàgina de l'INE:

https://www.ine.es/jaxiPx/Tabla.htm?path=/t15/p420/a2019/p03/l0/&file=01001.px&L=1

Noteu que les dades de l'INE estan separades entre homes i dones. Noteu també que el BMI varia amb l'edat. Qualsevol comparació que feu deu ser entre dades el més properes possible. No obstant això, no cal que considereu informació sobre la nacionalitat o el país on juguen. Indiqueu clarament a la llegenda de la gràfica quines dades heu triat per fer la comparació.

Exercici 4: diccionaris

Volem poder seguir l'evolució dels/de les jugadors/es que apareixen en diferents anys al nostre conjunt de dades. En un primer pas crearem una funció per seleccionar amb quins jugadors (identificats amb la columna “sofifa_id”) volem treballar i quines columnes volem mostrar:

a) Creeu una funció que, donat un dataframe amb les dades, una llista d'identificadors i una llista de columnes:

players_dict(df: pd.DataFrame, ids: list, cols:list) -> dict

- df: *dataframe* que conté les dades
- ids: llista d’iidentificador “sofifa_id”
- cols: llista de columnes de les que volem informació

retorni un diccionari:

- que tingui com a claus els identificadors “sofifa_id” continguts a la llista “ids”
- que tingui com a valors diccionaris amb la informació corresponent a cada jugador/a. Concretament, les claus de cadascun d'aquests diccionaris seran els noms de les columnes incloses a “col” i els seus valors seran la informació de tots els anys disponibles al dataframe per a cada futbolista.

Nota 1: Cal fer servir la columna “sofifa_id exclusivament com a clau. No s'inclourà a la llista “cols” en cap cas.

El diccionari anterior conté informació redundant. Per exemple, la columna “short_name” contindrà tantes repeticions com anys aparegui cada futbolista a les dades. En altres casos, com ara la columna “player_positions”, el diccionari contindrà informació parcialment repetida però amb la possibilitat d'incloure nous valors. Finalment, en columnes com a “potential”, cadascun dels valors obtinguts representa informació rellevant.

b) Per tal d’eliminar només la informació redundant, creeu la funció:

clean_up_players_dict(player_dict: dict, col_query: list) -> dict

- player_dict: diccionari amb el formato de l’apartat (a) *players_dict*
- col_query: llista de tuples amb detalls sobre la información que cal simplificar

La funció ha de recórrer un a un els elements del diccionari i fer els canvis indicats per la llista de tuples col_query.

Cada tupla de la llista estarà composta per dos valors: 1) el nom duna columna i 2) una cadena de caràcters. Aquesta cadena pot contenir dos valors possibles, els quals ens indiquen l'operació a realitzar sobre la columna/clau del diccionari:

- "one": ens hem de quedar només amb el primer valor que aparegui
- "del_rep": primer hem de descompondre la informació i després eliminar les repeticions
- La informació referent a les columnes que no apareixen a col_query es retornarà sense canvis.

Per exemple si tinguéssim la query

[("player_positions","del_rep"), ("short_name","one")]

y l'entrada de diccionari

41: {'short_name': ['Iniesta', 'Iniesta', 'Iniesta'], 'overall': [88, 88, 87], 'potential': [88, 88, 87], 'player_positions': ['CM', 'CM', 'CM, LM'], 'year': [2016, 2017, 2018]}

La nostra funció hauria de retornar:

41: {'short_name': 'Iniesta', 'overall': [88, 88, 87], 'potential': [88, 88, 87], 'player_positions': {'CM', 'LM'}, 'year': [2016, 2017, 2018]}

c) Considerant el dataframe amb ambdós gèneres i els anys 2016, 2017 i 2018, mostreu per pantalla:

- El diccionari construït amb la funció de l'apartat 4a amb la informació de les columnes ["short_name", "overall", "potential", "player_positions", "year"] i els ids = [226328, 192476, 230566].
- La query que passaríeu a la funció de l'apartat 4b per netejar aquest diccionari.
- El diccionari "net".

Nota: es recomana utilitzar el mòdul pprint per mostrar els diccionaris.

Exercici 5: evolució

Volem estudiar l'evolució d'algunes característiques dels/les esportistes:

a) Implementeu la funció:

top_average_column(data: dict, identifier: str, col: str, threshold: int) -> list

- data: diccionari "net" que conté la informació de diversos sofifa_id
- identifier: columna/clau que es fara servir com identificador
- col: nom d'una columna/clau numérica
- threshold: mínim número de dades necessàries

Aquesta funció cal que:

- Per a cada sofifa_id, calculi el valor mitjà de la característica col si teniu informació de threshold o més anys. Si no, s'ignora aquest sofifa_id. Si algun element de la llista té el valor NaN, també s'ignora aquest sofifa_id.
- Retorni una llista de tuples formades per tres elements: valor de la columna identifier; mitjana de la característica; i un diccionari compost per la clau year que contingui la llista d'anys corresponents als valors i la clau value amb aquests valors.
- Ordeni la llista de tuples a retornar en ordre descendent segons la mitjana calculada.

Per exemple, si tinguéssim identifier = "short_name" i col = "shooting", un possible element de la llista seria

('L. Schelin', 85.0, {'value': [87.0, 84.0, 84.0], 'year': [2016, 2017, 2018]})

b) Feu servir la funció anterior per obtenir l'evolució dels 4 futbolistes amb millor mitjana de movement_sprint_speed entre el 2016 i el 2022 (inclosos). Utilitzeu "short_name" com a identificador i mostreu el resultat per pantalla. Representeu gràficament l'evolució obtinguda.

Exercici 6: La millor defensa.

En aquest exercici final de l'assignatura us demanem que utilitzeu les vostres habilitats com a científics de dades per:

a) Formalitzar un problema a partir d'objectius generals.
b) Implementar una solució seguint els principis vistos a l'assignatura.
c) Trobar una solució al problema inicial i comunicar-ne els resultats de manera efectiva.

A continuació teniu un enunciat amb una mica de literatura que descriu el problema "general". Al final de l'exercici trobareu un resum dels punts més importants:

Un dels vostres clients representa un misteriós grup d'inversió que ha comprat un conegut club de futbol "en hores baixes". El client us indica que ha decidit començar la reconstrucció de l'equip per la defensa i que, per tant, necessita saber quins futbolistes formen la millor línia defensiva del món, els diners no són un obstacle.

El client us deixa molt clars els punts següents:

- ***Origen de Dades**: La millor manera de jutjar la qualitat dels futbolistes és consultar la base de dades d'aquesta PAC. Concretament, el client està interessat únicament en les dades del 2022 (female_players_22.csv i players_22.csv).*
- ***Definició de "defensa"**. Una línia defensiva es compon de 4 jugadors/es: 2 defenses centrals (CB segons la notació de la columna "player_positions"); 1 defensa lateral dret (RB); i 1 defensa lateral esquerre (LB).*
- ***No s'han de repetir jugadors/es**: Si un jugador/a pot jugar en més d'una d'aquestes posicions es pot considerar en més d'un rol, però sempre com a màxim en una posició a cada línia defensiva. Per exemple, a l'arxiu female_players_22.csv s'indica que "Magdalena Lilly Eriksson" pot ocupar les posicions "LB" i "CB". Aquesta jugadora podrà, per tant, optar a ser part de la nostra "defensa ideal" tant com a lateral dret com a central, però una alineació que la inclogui alhora en les dues posicions serà considerada errònia.*
- ***Masculí/Femení/Veterans/es**: El club disposa d'un equip masculí en hores particularment baixes, i un equip femení entre els millors del continent. Per tant, el client vol conèixer les millors línies defensives masculina i femenina. A més, el client sap del cert que un projecte anomenat "European VetLeague", on competiran equips mixtos de jugadores/es a partir de 30 anys, està a punt de veure la llum. Per això, vol conèixer també la millor línia defensiva de jugadors/es de qualsevol sexe que tinguin 30 anys o més.*

Pregunteu al client quins criteris defineixen la qualitat d'una línia defensiva, però aquest gesticula vagament en direcció de les columnes F a B7 del nostre dataset obert en excel i us indica que cal triar les (entre 3 i 5) característiques més importants per a cada posició (LB, RB,CB). El client vol tenir una estimació de quina aportació produeix cada possible línia defensiva en els tres criteris següents: 1) defensa; 2) control de pilota (possessió); i 3) atac. Si fos possible, la defensa i l'atac haurien d'estar dividits en tres zones: esquerra, central i dreta.

Els vostres intents d'aconseguir més concreció del vostre cap resulten infructuosos doncs està molt ocupat comptant feixos de bitllets que treu de la bossa d'esport que el client ha deixat. Només us indica que aquest és el projecte més important de l'empresa i que la vostra feina depèn del seu resultat, per no parlar de la felicitat dels centenars de milers de seguidors del club.

Punts importants:

a) ***Formalització del problema:*** *El primer que heu de fer és determinar quins jugadors poden jugar a cadascuna de les posicions d'interès. De la mateixa manera, heu de decidir, per a cada posició, quines variables (de les representades per les columnes del dataset) són més importants. Us recomanem triar un reduït nombre de candidats (no més de 50) a ocupar cadascuna de les 4 posicions seguint algun criteri de la vostra elecció. Obtinguda la llista de candidats per a cada posició, el següent pas seria determinar totes les possibles línies defensives que s'hi puguin constituir. Després, caldrà fer una estimació de la contribució a l'atac, la possessió i la defensa de cada línia en base al vostre criteri propi. Així, podreu ordenar les línies de millor a pitjor i quedar-vos amb la millor.*

b) ***Implementació:*** *El vostre lliurament ha d'incloure el codi necessari per trobar la solució al problema implementant els criteris subjectius que heu definit. Recordeu que es valoraran els mateixos aspectes de modularitat, documentació i eficiència que a la resta de la PAC.*

c) ***Presentació dels resultats:*** *El vostre lliurament ha d'incloure també un petit informe (de no més de 2 pàgines, en format pdf) que expliqui quins criteris heu utilitzat per modelar cadascun dels aspectes del problema i que presenti els resultats obtinguts per a les millors línies defensives masculina, femenina i de jugadors veterans. Es valorarà positivament la inclusió d'alguns gràfics que acompanyin l'explicació.*

d) ***Cal tenir carnet d'entrenador?*** *No cal tenir coneixements avançats de futbol per fer aquest exercici. La feina del Data Scientist consisteix a treure el màxim d'informació de les dades recopilades per els experts en el tema. Tampoc no hi ha una solució única ni es comparará el resultat amb el d'altres estudiants. El que es valorarà és que la idea plantejada tingui sentit, que l'informe comuniqui el que s'ha fet de manera efectiva i que la implementació sigui correcta.*

Tests

La PAC inclou un fitxer amb un conjunt de tests públics a tests/test_public.py. Per a la seva execució, cal tenir instal·lats els mòduls unittest i HTMLTestRunner. Aquest últim es pot instal·lar executant

pip install HTMLTestRunner-rv

Per córrer els tests cal executar la seguent comanda des d'un terminal situat al directori que conte el main.py:

python3 -m tests.test_public

De manera automàtica es crearà una carpeta anomenada reports amb informació sobre l'execució dels tests en html.

Per corregir un apartat és imprescindible que passi satisfactòriament tots els tests públics corresponents. Us animem a obrir el fitxer ia explorar els tests per entendre què s'espera que donin les funcions. Això sí, no es permet cap modificació del fitxer test_public.py. De fet, per corregir la PAC substituirem l'arxiu amb la nostra còpia, així que no serveix de res intentar modificar a mà aquests tests.

Tingueu en compte que els tests proporcionats no inclouen proves per a totes les funcionalitats que es demanen en aquesta PAC. A l'hora de corregir, passarem un conjunt de tests privats que comprovaran altres aspectes de les funcions. Això, a més, evita que es programin funcions pensades per passar únicament els tests públics.

Finalment, us proposem que creeu els vostres propis tests per comprovar alguna d'aquestes funcionalitats no contingudes als públics. Per això, s'inclou el fitxer test_custom.py que podeu modificar com vulgueu. Encara que podeu crear tots els que vulgueu, per aconseguir la nota màxima en aquest apartat heu de crear, com a mínim, els tests següents:

- Comproveu que la funció 2a dóna el resultat correcte quan cols_to_return conté, com a mínim, dues columnes (per exemple, short_name i potential).
- Comproveu que la funció 2a dóna el resultat correcte si el resultat inclou més d'una fila.
- Comproveu que la funció calculate_BMI dóna el resultat correcte quan gender = 'F'
- Comproveu que la funció clean_up_players_dict dóna el resultat correcte si la query conté l'operació "one".

Important: cal que el fitxer test_public.py importi totes les funcions que feu. Per evitar que hagueu de modificar el fitxer (recordeu que el substituirem per la nostra còpia), s'inclou el fitxer testing_imports.py. Haureu de posar-hi tots els imports que us calguin (al vídeo hi teniu un exemple).

Linter

Atès que ja no estem a l'entorn dels Jupyter notebooks, per avaluar si el codi segueix la guia d'estil PEP8 farem servir el linter pylint.. Per instal·lar-lo a la màquina virtual simplement heu d'executar en un terminal:

```
sudo apt install pylint
```

Després, per executar-lo, només cal anar al directori on es trobin els fitxers *.py i cridar en un terminal:

```
pylint *.py
```

El resultat serà una llista d'errors (si n'hi hagués) juntament amb una nota numèrica sobre 10. La puntuació corresponent a l'apartat d'estil en aquesta PAC es basarà en aquesta nota.

Nota: no s'avaluarà l'estil dels fitxers del directori test.

Criteris de correcció

Aquesta PAC es valorarà seguint els criteris següents:

- **Funcionalitat** (7 punts): Es valorarà que el codi implementi correctament el que es demana a l'enunciat. Per això, s'usarà un conjunt de tests públics, privats i es valoraran els resultats gràfics proposats. **És imprescindible per corregir un exercici que passi tots els tests públics corresponents**.
    - Tasca 1 (0.75 punts)
    - Tasca 2 (1.25 punts)
    - Tasca 3 (1.25 punts)
    - Tasca 4 (1 punt)
    - Tasca 5 (0.75 punts)
    - Tasca 6 (2 punts)
- **Estilo i documentació** (0.75 punts): S'usarà un analitzador automàtic de codi, pylint, el qual proporciona una nota sobre 10 que és prendra com a nota d'estil i documentació Tingueu en compte que l'analitzador també avalua la presència de docstrings.
- **Modularitat** (0.25 punts): Es valorarà l'organització del codi en mòduls temàtics seguint algun criteri raonable.
- **Tests** (1 punt): Es proposa implementar 4 tests que permetin avaluar la funcionalitat de parts del codi que els tests públics no comproven. Tots tenen el mateix pes en aquest apartat.
- **Llicència + Requeriments** (0.5 punts): Cal que el projecte inclogui la llicència sota la qual es distribueix el codi (podeu triar la que vulgueu). Cal incloure un fitxer de requeriments que llisti únicament les llibreries necessàries per executar el codi.
- **README** (0.5 punts): Cal que el projecte inclogui un fitxer README en format markdown que expliqui com instal·lar-lo i executar-lo.