Sarcini Pandas

Serii

* noțiune de serie
* indexul numeric si indexul etichetă
* crearea seriei dintr-un dicționar
* vizualizarea indecșilor etichetă ***keys()***
* operații aritmetice cu un scalar
* sumarea seriilor cu indecși eticheta diferiți
* metodele operațiilor aritmetice si parametru ***fill\_value***
* tipul elementelor după operații

DataFrame

* noțiune de DataFrame
* crearea si personalizarea DataFrame dintr-un 2d np array
* crearea DataFrame din fișiere .csv
* afișarea coloanelor ***df.columns***
* schimbarea denumirii unei coloane ***df.rename(columns={nume\_vechi:nume\_nou})***
* afișarea indexurilor ***df.index***
* afisarea tipului de date pe coloane ***df.dtypes***
* afisarea nr de linii si coloane ***df.shape***
* metodele ***head(), tail(), info(), describe()***
* schimba indexurile cu coloanele ***transpose()***
* citirea datelor unei coloane
* citirea datelor din mai multe coloane
* crearea unei noi coloane
* operații aritmetice cu coloane numerice
* ștergerea unei coloane
* parametrul inplace pentru realizarea modificărilor
* setarea unei coloane ca index ***df.set\_index(col***)
* restabilirea indexurilor implicite ***df.reset\_index()***
* citirea datelor de pe linia cu indexul eticheta ***df.loc[index\_eticheta]***
* citirea datelor de pe linia cu indexul numeric ***df.iloc[index\_numar]***
* citirea datelor dintr-o lista de linii cu index eticheta ***df.loc[lista\_etichete]***
* citirea mai multor linii după indexul numeric ***df.iloc[slicing]***
* ștergerea unei linii după indexul eticheta
* excluderea unei linii după indexul numeric
* adăugarea unei linii sub forma de serie a cărei indecși coincid cu coloanele ***df.append(linie)***
* actualizarea datelor unei celule conform indexului si coloanei ***df.at[index, col]=valoarea***

Filtrarea condiționată

* filtrarea liniilor după o condiție pe o coloana ***df[df[coloana] condiție]***
* filtrare linii după îndeplinirea mai multor condiții de pe mai multe coloane ***df[(df[col1]cond1)&(df[col2]cond2)]***
* filtrare linii după îndeplinirea uneia din mai multe condiții de pe mai multe coloane ***df[(df[col1]cond1)|(df[col2]cond2)]***
* filtrare linii după îndeplinirea unei condiții când o coloana ia valori dintr-o lista ***df[df[col1].isin(lista)]***

Metode utile

* aplicarea unei funții a utilizatorului asupra unei coloane ***df[col].apply(func)***
* aplicarea unei funcții a utilizatorului asupra mai multor coloane ***df[list\_col].apply(lambda df:func(df[col1], df[col2],...))***
* transformarea unei funcții de utilizator in funcție np - ***np.vectorize(func)***
* aplicarea unei funtii de utilizator vectorizate asupra mai multor coloane ***np.vectorize(func)(df[col1], df[col2],...)***
* sortarea liniilor in funcție de valorile unei coloane ***df.sort\_values(col, ascending)***
* sortarea liniilor in funcție de valorile mai multor coloane in ordinea scrierii lor ***df.sort\_values(list\_col, ascending)***
* determinarea valorii maxime/minime pe o coloana ***df[col].max()/min()***
* determinarea indexului valorii maxime/minime pe o coloana ***df[col].idxmax()/idxmin()***
* determinarea corelației dintre coloanele numerice ***df.corr()***
* determinarea numărului de valori de fiecare categorie in coloane categoriale ***df[col].value\_counts()***
* afișarea categoriilor dintr-o coloana categoriala ***df[col].unique()***
* determinarea numărului de categorii dintr-o coloana categoriala ***df[col].nunique()***
* substituirea unei valori a unei coloane cu o alta valoare ***df[col].replace(val\_veche, val\_noua)***
* substituirea mai multor valori a unei coloane cu alte valori utilizând replace() ***df[col].replace(list\_val\_vechi, list\_val\_noi)***
* substituirea mai multor valori a unei coloane cu alte valori utilizând map() ***df[col].map({val\_veche1:val\_noua1, val\_veche2:val\_noua2, ...})***
* transformarea tuturor coloanelor categoriale in numerice confrom one-hot-encoder ***pd.get\_dummies(df)***
* transformarea tuturor coloanelor categoriale in numerice confrom one-hot-encoder cu stergerea unei coloane numerice pe categorie ***pd.get\_dummies(df,drop\_first=True)***
* verificarea prezentei liniilor identice ***df.duplicated()***
* ștergerea liniilor identice ***df.drop\_duplicates()***
* verificarea daca valorile unei coloane numerice se includ într-un interval ***df[col].between(inf, sup, inclusive)***
* selectarea liniilor unde valorile unei coloane numerice se includ într-un interval ***df[df[col].between(inf, sup, inclusive)]***
* afișarea primelor n linii unde valorile unei coloane sunt maxime/minime ***df.nlargest(n, col)/nsmallest(n, col)***
* selectarea aliatoare a unui număr n de linii ***df.sample(n)***
* selectarea aliatoare a unui procentaj de linii din numărul total ***df.sample(frac=0.1)***

Date lipsă

* păstrarea lipsurilor vs ștergerea lipsurilor vs completarea lipsurilor
* compararea a 2 valori lipsa (== vs is)
* verificare prezenței lipsurilor în df ***df.isnull()/df.notnull()/df.isna()/df.notna()***
* prezentarea numărului datelor lipsa pe coloane ***df.isna().sum()***
* verificare prezentei lipsurilor pe o coloana ***df[col].isna()/notna()***
* afișarea liniilor care nu au lipsuri pe o anumita coloană ***df[df[col].notna()]***
* ștergerea tuturor liniilor cu lipsuri ***df.dropna()***
* ștergerea liniilor care au lipsuri si numărul de valori non-lipsa este mai mic decât n ***df.dropna(thresh=n)***
* ștergerea tuturor coloanelor care au lipsuri ***df.dropna(axis=1)***
* ștergerea liniilor care au lipsuri pe o anumita coloana ***df.dropna(subset=[col])***
* completarea tuturor lipsurilor cu o anumita valoarea val ***df.fillna(val)***
* completarea lipsurilor unei coloane cu o anumita valoarea val ***df[col].fillna(val)***
* completarea lipsurilor unei coloane cu media valorilor vecine din coloana ***df[col].interpolate()***

Operația de grupare

* Esența grupării datelor coloanelor numerice în funcție de datele unei coloane categoriale
* Determinarea valorii medii a grupurilor de valori a tuturor coloanelor numerice ce corespund valorilor unei coloane categoriale ***df.groupby(col\_categ).mean()***
* Determinarea valorii medii a grupurilor de valori a unei coloane numerice ce corespund valorilor unei coloane categoriale ***df.groupby(col\_categ).mean()[col\_num]***
* Funcții ce pot fi aplicate valorilor unui grup ***mean(), sum(), size(), count(), std(), var(), sem(), describe(), first(), last(), nth(), min(), max()***
* Determinarea valorii medii a grupurilor de valori a tuturor coloanelor numerice ce corespund valorilor combinațiilor mai multor coloane categoriale ***df.groupby(lista\_col\_categ).mean()***
* Noțiunea de multiindex, atributele ***df.index.name***, ***df.index.levels***
* Citirea liniilor cu multiindex ***df.loc[index\_prim]***, ***df.loc[[list\_index\_prim]]***, ***df.loc[(index\_prim, index\_sec)]***
* Citirea liniilor cu multiindex dupa o valoarea a unui singur index ***df.xs(key=val\_index, level=nume\_index)***
* Sortarea linilor cu multiindex in funcție de un index ***df.sort\_index(level=index, ascending)***
* Aplicarea unor functii statistice (mean(), sum(), size(), count(), std(), var(), sem(), describe(), first(), last(), nth(), min(), max()) asupra tuturor coloanelor ***df.agg(list\_string\_nume\_funct)***
* Aplicarea unor funcții statistice asupra unei coloane ***df.agg(list\_string\_nume\_funct)[col]***
* Aplicarea anumitor funcții statistice asupra anumitor coloane ***df.agg({col1: list\_string\_nume\_funct, col2: list\_string\_nume\_funct,...})***

Combinarea DataFrame

* Concatenarea 2 df după coloane (cu aceiași indecși) ***pd.concat([df1,df2], axis=1)***
* Concatenarea 2 df după linii (cu aceleași coloane) ***pd.concat([df1,df2], axis=0)***
* Combinarea a 2 df după datele comune dintr-o coloana comună ***pd.merge(df1, df2, how=’inner’, on=col\_comuna)***
* Combinarea a 2 df după datele din df1 dintr-o coloana comună ***pd.merge(left=df1, right=df2, how=’left’, on=col\_comuna)***
* Combinarea a 2 df după datele din df2 dintr-o coloana comună ***pd.merge(left=df1, right=df2, how=’right’, on=col\_comuna)***
* Combinarea a 2 df după toate datele dintr-o coloana comună ***pd.merge(df1, df2, how=’outer’, on=col\_comuna)***
* Combinarea a 2 df după datele comune din coloane diferite a acestora ***pd.merge(df1, df2, how=’inner’, left\_on=col\_df1, right\_on=col\_df2)***
* Adăugarea în rezultatul combinării după o coloana comuna a unui sufix celorlalte coloane comune în ambele df ***pd.merge(df1, df2, how=’inner’, on=col1\_comuna, suffixes=(‘sufix\_df1’, ‘sufix\_df2’))***

Metode pentru string sau datetime

* Tipul de date string în Pandas – Object sau StringDtype
* Utilizarea .str pentru aplicarea metodelor string din Python asupra datelor in pandas
* Divizarea unei coloane cu string în mai multe coloane cu substringuri formate prin divizare după un caracter ***df[col].str.split(character, expand=True)***
* Utilizarea modulului datetime pentru crearea obiectelor datetime
* Conversia unui string in obiect datetime in Pandas ***pd.to\_datetime(df[col])***
* Format european (zz-ll-aaaa) vs format american (ll-zz-aaaa) ***pd.to\_datetime(df[col], dayfirst=True)***
* Conversia unui string in obiect datetime in Pandas cu precizare formatului stringului ***pd.to\_datetime(df[col], format=’%d--%b--%Y’)***
* Gruparea după an când coloana cu datatime este index ***df.resample(rule=’A’).mean()***
* Citirea unei componente (an, luna, ...) dintr-un tip de datedatetime ***df[col].dt.month***

Importul și exportul datelor în Pandas

* Determinarea caii fișierului jupyter notebook ***pwd***
* Vizualizarea conținutului folderului cufișierul jupyter notebook ***ls***
* Importul fisierelor .csv ***df=pd.read\_csv(cale\_csv)***
* Importul csv fara setarea numelui coloanelor ***df=pd.read\_csv(cale\_csv, header=None)***
* Importul csv cu setarea ca index a unei coloane ***df=pd.read\_csv(cale\_csv, index\_col=index\_col)***
* Exportul datelor în fisiere .csv ***df.to\_csv(‘cale\_nume.csv’)***
* Exportul datelor în fisiere .csv cu salvarea indexului etichetă ***df.to\_csv(‘cale\_nume.csv’, index=True)***
* Importul fișierelor .html necesită instalarea ***lxml***
* Importul tuturor tabelelor de pe o pagina html ***tabele=pd.read\_html(adresa\_html)***
* Selectarea tabelului dorit din lista de tabele ***df=tabele[index\_tabel]***
* Exportul datelor în tabele .html ***df.to\_html(‘cale\_nume.html’)***
* Importul fișierelor Excel necesită instalarea ***openpyxl*** și ***xlrd***
* Importul unui tabel din fișierele Excel .xlsx ***df=pd.read\_excel(cale\_excel, sheet\_name=nume\_tabel)***
* Vizualizarea tabelelor (sheets) a unui fișierExcel ***pd.ExcelFile(cale\_excel).sheet\_names***
* Importul sub formă de dicționar a tuturor tabelelor unui fișier Excel ***df=pd.read\_excel(cale\_excel, sheet\_name=None)***
* Exportul datelor în tabele Excel ***df.to\_excel(‘cale\_nume.xlsx’, sheet\_name=nume\_tabel)***
* Importul datelor din baze SQL necesită instalarea bibliotecilor drive corespunzătoare ***pymysql – MySQL, psycopg2 – PostgreSQL, pyodbc – MS SQL Server*** etc.
* Importul datelor din baze SQL necesită instalarea ***sqlalchemy*** pentru stabilirea conexiunii prin intermediul unui Engine
* Importul ***create\_engine*** din ***sqlalchemy***
* Crearea conexiunii cu baza de date MySQL ***conexiune=create\_engine(‘mysql+pymysql://<username>:<password>@<host>/<dbname>[?<options>]’, pool\_recycle=3600)***
* Exportul datelor în tabele SQL ***df.to\_sql(name=nume\_tabel, con=nume\_conexiune, if\_exists)***
* Importul unui tabel din MySQL ***df=pd.read\_sql(sql=nume\_tabel, con=nume\_conexiune)***
* Importul unor coloane a unui tabel din MySQL cu interogari SQL ***df=pd.read\_sql\_query(sql=’SELECT col1, col2 FROM nume\_tabel’, con=nume\_conexiune)***

Reorganizarea tabelelor

* Esența reorganizării tabelului cu metoda pivot()
* Transformarea unui df astfel încât valorile col1 sa fie index, valorile col2 sa fie numele coloanelor iar valorile col3 sa fie valorile df ***pd.pivot(data=df, index=col1, columns=col2, values=col3)***
* Transformarea unui df astfel încât valorile col1 sa fie index iar pentru celelalte col numerice sa se aplice o funcție de agregare ***pd.pivot\_table(data=df, index=col1, aggfunc=’nume\_funct’, values=lista\_col\_num)***