**Pandas- 08.04.2023**

**apply() & applymap() & map()&transform**

apply(): DataFrame veya Seri üzerinde bir satır veya sütun boyunca bir işlem yapmak için kullanılır.

applymap(): DataFrame'in tamamı üzerinde öğe bazında işlem için kullanılır.

map(): Yalnızca Seri lerde kullanılabilir ve Seri nin her bir elemanına uygulanacak bir işlem için kullanılır.

**transform**

#transform ozellikle groupby yapinca anlamli oluyordu

#yanina bir aggregate func kullanmam gerekiyordu.

**#df.groupby("Company").transform(np.mean)**

#transforma agg func yazdigim zaman elementwise calisiyor

**df.groupby("Company")["Age"].transform("mean")**

#burdaki degerleri donustur diyorum, meanini alarak mesela

**Applymap**

#applymap her eleman ozelinde islem yapiyordu.

#seriler uzerinde islem yapamiyordu. hata veriyor.

#transformu bu amacla kullanirken, applymap'i kullanamiyorduk.

##transform ile applymap arasindaki bir fark da, applymap'de agg func kullanamiyorum.

**df.applymap(lambda x: len(str(x)))**

#herbir hucredeki elemanin karakterini saydiriyor mesela bu.

#apply’dan farkli olarak dogrudan her bir elemanin karakter sayisini sayiyor

**Apply mi applymap mi**

#sutun ya da satir bazinda islem yapmak istersen apply,

#herbir elemana islem yapmak istersen applymap kullanirsin

**Apply**

**df.apply(lambda x: len(str(x)))**

#apply ise satir ya da sutun bazinda islem yapar.

#applymap gibi herbir elemana islem yaptirmaz, gruplandirilmis olarak getirir.

#burda sutun bazinda aldi, mesela company'de toplam 124 karakter oldugunu getirdi

#karakter olarak ascii degerlerin uzunlugunu getiriyor.

#applymap'de dogrudan length almisti

# apply'da asciidf.

**groupby("Company").apply(np.mean)**

|  |  |  |
| --- | --- | --- |
|  | Age | Sale |
| Company |  |  |
| Amazon | 28.5 | 200.0 |
| OpenAI | 35.25 | 172.5 |

**df\_num.apply(np.sum)**

#ben bunu apply yaptigim zaman sum olarak kullanabildim, gruplandirarak yapiyor

**df\_num.apply(np.sum, axis=1)**

#bu da axis=1'e gore toplayarak getiriyor

**map**

#map()denersek atama yapiyordu. serideki elemanlari degistirebiliyorduk

**df.Person.map({"Ahmet" : "A", "Fatma" :"F"})**

#icine key ve value iceren sozluk yapisi kullaniriz.

#nan deger istemiyorsam hepsini girmem gerekiyor

#yoksa girdiklerim haricinde nan dodnurur

**df.Department(map{"IT" : "O", "HR" : 1})**

#IT gordugune 0'i, HR gordugune 1'i koydu.

#icindeki tek tek elemanlari degistirmek icin kullaniyoruz

#sutunlari sayisala donustururken kullaniyoruz daha cok

#bazi metodlarda atama yapmamiz gerekiyor degisiklik icin

#ya inplace=True diyeceksin, ya da atama yapacaksin kalici degisiklik icin bunda

**pivot() vs pivot\_table()**

#yeni index degeri koyup, yeni gruplandirmalar yapmak icin kullaniyoruz

#mesela **unique degerlere gore** siniflandirma yapiyor

**d.pivot(index='items', columns='CType', values='USD')**

seklinde kullanimi var. Icine index, columns, values belirliyorum.

#burda mesela, column'lara Ctype getir demek istiyorum, 3 unique deger goruyor (gold, Bronze, Silver), karsiligina degerini getiriyor.

#item1'in bronz ve gold degeri yok o yuzden nan getiriyor

#item0'a ait 2 tane gold gozlemi var. 1 tane olsa onu getirecekti

#ama 2 tane oldugu icin hata veriyor

#boyle birden fazla gozlem varsa pivot degil pivot\_table kullaniyoruz

**#aggfunc=np.mean()** methodu calisiyor pivot\_table’da

ve ortalamasini alip koyuyor bu sefer

#pivot'ta arkaplanda aggfunc methodu olmadigi icin sorun cikiyor

#pivot\_table pivotun daha genellestirilmis hali

#pivot\_table'da duplicate data uzerinden islem yapabiliyoruz

#aggfunc'in defaultu mean

**df.pivot\_table(index="gender", columns="sport", values=["age"], aggfunc="mean")**

#values, index, columns, aggfunc var icinde

#degerleri benim istedigim gibi, gruplandirarak dondurur

#index ve columns'un unique degerlerine gore gruplandirdi.

#gender'da unique degerler male,female'di.

#sport'un unique degeri basket,futbol, tenis ve voleybol, ona gore siraladi

#value olarak da age'i getirdi.

#female'in futbol kisminda kimse olmadigi icin nan degerle doldurdu.

#value degerine, basket ve female'da hangi age degeri varsa onu getirir.

#birden fazla deger olursa ortalamasini alir getirir.

\*

#female’da 2 tane tennis olsun mesela diyelim

**df.pivot\_table(index="gender", columns="sport", values=["age", "weight", "height", ], aggfunc="mean")**

female'e denk gelen 2 tane tennis, dolayisiyla age’de 2 deger vardi

#bunun yasini nasil hesapladi:

#arkasinda bir aggfunc.mean calisti ve ortalamasini aldi.

#birinin degeri 24 digeri 26 idi, ortalamasini alip age'e 25 yazdi

#ama pivot kullanmis olsaydim, arkasinda boyle bir aggfunc calismadigi icin hata verecekti.

**Type of NaN Values**

**NaN = Not-a-Number**

**np.nan,** belirsiz veya tanımlanamaz sayısal değerleri temsil etmek için kullanılır.

np.nan değeri, "sentinel" bir değer olarak kabul edilir ve herhangi bir değerle, hatta kendisiyle bile karşılaştırılamaz.

Sentinel; matematiksel hesaplamalarda belirsiz veya tanımlanamaz değerleri temsil etmek için kullanılır.

None; sentinel bir değerdir ve bir fonksiyonun veya metodun "boş" veya "tanımsız" bir değeri ifade etmesi için kullanılır.

**Sütunun data tipi object ise "nan", sayısal ise "NaN" olur.**

#missing value ile ugrasirken karsimiza cikan degerler nan degerler

#ankette bazi sorular cvpsiz kaliyor mesela.

#ben ankette gelen cvplardan veriseti olustururken deger giremedigim yerlere nan deger giriyorum

#daha sonra cesitli bilgilerle doldurabilirsek dolduruyoruz, dolduramazsak drop yapiyoruz

#mesela gelirini soylemek istemeyen issiz birinin gelirine 0 yazabilirsin sonradan

**df.info()**

**Infosuna bakinca df’in:**

#not-null count da veriyor bize. 11 gozlem var id column’da

#not-null yokmus id ve gender'da mesela.

#status'te bir null deger varmis. 10 non-null. - olanlari null kabul etmiyor.

#department'ta 9 non-null deger varmis. Yani 2 null (NaN ve None)

#var1 sutunda 2 nan var 1 tane - var. Ama 11 non-null var diyor bize

##ipini de object olarak gordugu icin, nan degerleri Null gormuyor.

#var2 datatipi float, o yuzden bana 3 tane null var diyor, NaN null aliyor.

#mesela sayisal bir sutuna str ifade koyarsak, **dtype object** oluyor

#null degerlerin tipi float

#None da null deger kabul edilir

**type(np.nan)**

#np.nan ile nan deger olusturuyoruz

#tipi float

**type(None)**

#None in tipi ise NoneType diye gecer

|  |
| --- |
| print("1", 0 \* np.nan)  print("2", np.nan - np.nan)  print("3", np.nan + np.nan)  print("4", np.nan - 10)  print("5", np.nan + 10)  print("6", np.nan == np.nan)  print("7", 10 > np.nan)  print("8", np.inf > np.nan)  print("9", float('-inf') < np.nan < float('inf'))  print("10", np.nan is np.nan) |

#kiyaslama yaptigin zaman genelde False cikiyor

**Detecting Missing Values**

#datasetindeki null degerleri sorgulama fonksiyonlari:

**df.isnull()**

**df.isna()**

**df.notnull()**

**df.isnull()**

**df.isnull().any()**

#herhangi bir satirda null varsa true, yoksa false

#tek bir null varsa bile True doner

**df.isnull().any(axis=1)**

#satirlarda sutun sutun bakacak, bir tane bile null varsa True doner

**df[df.isnull().any(axis=1)]**

#sadece icinde herhangi bir null deger iceren sutunlari goruyoruz

**df.isnull().sum()**

#kac tane null varsa onu getirir satir satir

**df.isnull().sum().sum()**

#df icindei tiplam null degerleri getirdi

**Converting Improper Values to NaN Values**

**df["var1"]=df["var1"].replace(to\_replace='-', value=np.nan).astype("float")**

# ‘-’ yerine baska bir deger ata diyorum, o da nan olsun

df["var1"]

#artik 3 tane null degerimiz var

# ‘-’ isaretini aldi nan degerine donsuturdu

#cunku - str olarak kabul edildiginden null kabul edilmiyordu, object datatipinde gorunuyordu

#astype float olarak degistirince float olarak degistirmis oldum

**df["status"]=df["status"].replace(to\_replace='-', value=np.nan)**

df["status"]

#status'ta astype("float") yazmiyoruz, cunku icinde str degerler var status columnda

#var1 column’da float degerler vardi.

**df0.isnull().sum().sum()**

#ilk default halinde(df0) 9 tane null vardi.

**df.isnull().sum().sum()**

**Missing Value Handling Methods**

**dropna() & drop()**

#dolduramayacagimiz null degerleri droplamak icin kullaniyoruz

#kategorik veri tipinde, mod degerleri ile doldurabiliyoruz

#en cok tekrar eden mod degeri ile

#sayisal sutunlarda ise mesela, yas degerleri var diyelim

#mod degil de mean'e bakarsak daha anlamli bilgi verebilir

#datanin ruhuna uygun sekilde dolduruyorum

**df.dropna(axis=0, how="any", inplace=False)**

#how parametresi ile any all degerlerini girebiliyoruz

#default degeri any-yani bir tane bile null deger varsa drop etsin demek

#all ise hepsi null ise drop et demek

#bunun da inplace=True parametresi var

**df.dropna(axis=1, how="any", inplace=False)**

#sutunlar bazinda kontrol edip null deger varsa dusuruyor

#condition yaziyorum yani any ve all gibi

**df.drop([1,3,5])** #icine index, column, axis gibi seyler belirtiyorum

#1,3,5.indexleri tamamen dusurdu satirlardan

**df.drop(["var1", "var2"], axis=1)**

#axis=1 yazmasaydim, defaultu axis=0 oldugu ucun, satirlar boyunca arayacak bulamayacakti.

#o yuzden axis=1 olarak yazmam gerek

**Thresh condition**:

#bir sutunun dusmemesi icin icermesi gereken nan degerini yazabiliriz

**df.dropna(axis=1, thresh=9),**

#bu sekilde yazinca, null olmayan degerlerim 9'dan az olmasi halinde dusur demis oluyoruz

**Filling Missing Values (Imputation)**

**df.fillna()**

#null olanlari doldur anlaminda kullaniyoruz bunu

**df["var1"].fillna(0)**

**df[["var1"]].fillna(df["var1"].mean()**

#var1 sutununun ortalamasini almis, bunla doldurtmus

#median'a gore de doldurtabilirsiniz

**df.fillna(df.mean())**

#butun sutunlara gider, her sutunun kendi ortalamasi ile null degerleri doldurur.

**df.fillna({"dept": "Other", "var1": df.var1.mean(), "var2": df.var2.median()})**

#ortalama, median alamayacagim sutunlar icin mesela dept sutunu icin,

#null degerlere "other" desin dedik.

#var1 icin de mean yapsin dedik.

#hepsini tek satirda tanimlamak icin dict formatinin icinde atadik.

**df.dept.fillna(df.dept.mode()[0])**

#mesela kategorik sutunlarda mode ile doldur diyebiliriz

#en cok tekrar eden deger ile doldurmus oldu

#fillna ile mode func bu ise yaradi.

**df.dept.fillna(method="bfill")**

#bfill methodu ile, index sirasina bak, sonra gelen deger ile doldur demis oluyoruz

#backfill

**df.dept.fillna(method="ffill")**

#index sirasina bak, bir onceki deger ile doldur demek

#randomizelik katmak istersek boyle oncekiyle, sonrakiyle dolduruyoruz

#forwardfill

#inplace=True bunun icin de gecerli

#peki en son deger null ise, sonraki deger ile doldur dedik mesela, sonda deger yok, bos kalir o zaman

#napacagiz bu durumda, bfill yaptiysak ffill, ffill yaptiysak da bfill yapmak lazim