Ranking in Pandas

Dataframe.rank()

- The rank() calulates the rank value for each row element in a dataset.
- The rank is calculates on the basis of position of elements after sorting.

> a) The ranking order is by default ascending, so the lowest value is assigned the first rank.

> b) In the case of equality, the rank is determined by taking the average.

Syntax:

> **DataFrame.rank(axis, method, numeric_only, na_option, ascending=True)**

Parameters:

1. **axis:** 0 or 'index' for rows and 1 or 'columns' for Column.
2. **method:** Takes a string input **('average', 'min', 'max', 'first', 'dense')** which tells pandas what to do with same values. Default is average which means assign average of ranks to the similar values.
3. **numeric_only:** Takes a boolean value and the rank function works on non-numeric value only if it's False.
4. **na_option:** Takes 3 string input **('keep', 'top', 'bottom')** to set position of Null values if any in the passed Series.
5. **ascending:** Boolean value which ranks in ascending order, if True.

Different ranking methods

The rank function has 5 different methods that can be used for ranking or in case of equality. These 5 options are:

1. average : average of minimim and maximum ranks of the group (default ranking method)

2. min : lowest rank in the group

3. max : highest rank in the group

4. first : ranks assigned in order they appear in the array (serial-wise)

5. dense : rank always increases by 1 between groups (user-type)

How to rank the group of records that have the same value (i.e. ties)?

The option is selected with the method parameter and the default value is "average",

Let us Create a DataFrame to understand Ranking of elements

```
import pandas as pd
```

```
df2 = pd.DataFrame([1,2,2,3,3,3,4,4,4,4,5,6,7,7,7,8,8,9], columns=['Sample'])
df2
```

	Sample
0	1
1	2
2	2
3	3
4	3
5	3
6	4
7	4
8	4
9	4
10	5
11	6
12	7
13	7
14	7
15	8
16	8
17	9

Here, five new columns are added in the df2 DataFrame. Name of new columns are average_rank, min_rank, max_rank, first_rank and dense_rank.

Each column has the rank of the Sample column, calculated using the given method.

- Dense ranking is the ranking given by the user.
- First ranling is serial-wise.
- Min Ranking is the minimum index value (numbering of elements) among group members.
- Max ranking is the maximum index value (numbering of elements) among group members.
- Average ranking is the average of minimum and maximum rank values.

```
df2['average_rank'] = df2['Sample'].rank(method='average')

df2['min_rank'] = df2['Sample'].rank(method='min')

df2['max_rank'] = df2['Sample'].rank(method='max')

df2['first_rank'] = df2['Sample'].rank(method='first')

df2['dense_rank'] = df2['Sample'].rank(method='dense')

df2
```

	Sample	average_rank	min_rank	max_rank	first_rank	dense_rank
0	1	1.0	1.0	1.0	1.0	1.0
1	2	2.5	2.0	3.0	2.0	2.0
2	2	2.5	2.0	3.0	3.0	2.0
3	3	5.0	4.0	6.0	4.0	3.0
4	3	5.0	4.0	6.0	5.0	3.0
5	3	5.0	4.0	6.0	6.0	3.0
6	4	8.5	7.0	10.0	7.0	4.0
7	4	8.5	7.0	10.0	8.0	4.0
8	4	8.5	7.0	10.0	9.0	4.0
9	4	8.5	7.0	10.0	10.0	4.0
10	5	11.0	11.0	11.0	11.0	5.0
11	6	12.0	12.0	12.0	12.0	6.0
12	7	14.0	13.0	15.0	13.0	7.0
13	7	14.0	13.0	15.0	14.0	7.0
14	7	14.0	13.0	15.0	15.0	7.0

Average Rank Formula:

> **Average of Minimum rank and Maximum Rank of the group**

Example:

average rank is 2.5 = (2.0 + 2.0)/2

average rank is 5.0 = (4.0 + 6.0)/2

and so on.

Let’s creating another sample DataFrame and learn to apply ranking

```
import pandas as pd
df = pd.DataFrame({
    "name": ["John","Jane","Emily","Lisa","Matt","Jenny","Adam"],
    "current": [92,94,87,82,90,78,84],
    "overall": [184,173,184,201,208,182,185],
    "group":["A","B","C","A","A","C","B"]
})

print(df)
```

```
    name  current  overall group
0   John       92      184     A
1   Jane       94      173     B
2  Emily       87      184     C
3   Lisa       82      201     A
4   Matt       90      208     A
5  Jenny       78      182     C
6   Adam       84      185     B
```

We have created a DataFrame with 7 rows and 4 columns.

Let’s start with the default settings and assign a rank to the rows based on the overall column.

```
df["rank_default"] = df["group"].rank()

print(df)
```

```
    name  current  overall group  rank_default
0   John       92      184     A           2.0
1   Jane       94      173     B           4.5
2  Emily       87      184     C           6.5
3   Lisa       82      201     A           2.0
4   Matt       90      208     A           2.0
5  Jenny       78      182     C           6.5
6   Adam       84      185     B           4.5
```

You may have noticed that we did not write the method parameter when creating the "rank_default" column.

Since it is the default value, we do not need to specify it but it also works if you write it as follows:

> df["rank_default"] = df["group"].rank(method="average", ascending=False)

Let's change the order and sort them in descending order so that the person with the highest score is ranked 1st.

```
df["rank_default_desc"] = df["group"].rank(ascending=False)

df = df.sort_values(by="rank_default_desc", ignore_index=True)

print(df)
```

```
    name  current  overall group  rank_default  rank_default_desc
0  Emily       87      184     C           6.5                1.5
1  Jenny       78      182     C           6.5                1.5
2   Jane       94      173     B           4.5                3.5
3   Adam       84      185     B           4.5                3.5
4   John       92      184     A           2.0                6.0
5   Lisa       82      201     A           2.0                6.0
6   Matt       90      208     A           2.0                6.0
```

Applying Different Ranking Methods

```
# create DataFrame
df = pd.DataFrame({
    "name": ["John","Jane","Emily","Lisa","Matt","Jenny","Adam"],
    "current": [92,94,87,82,90,78,84],
    "overall": [184,173,184,201,208,182,185],
    "group":["A","B","C","A","A","C","B"]
})

# create rank columns
df["rank_default"] = df["group"].rank(ascending=False)
df["rank_min"] = df["group"].rank(method="min", ascending=False)
df["rank_max"] = df["group"].rank(method="max", ascending=False)

# sort rows
df = df.sort_values(by="rank_default", ignore_index=True)
df
```

	name	current	overall	group	rank_default	rank_min	rank_max
0	Emily	87	184	C	1.5	1.0	2.0
1	Jenny	78	182	C	1.5	1.0	2.0

The other two options for the method parameter are “first” and “dense”.

```
df["rank_first"] = df["overall"].rank(method="first", ascending=False)
df["rank_dense"] = df["overall"].rank(method="dense", ascending=False)

df
```

	name	current	overall	group	rank_default	rank_min	rank_max	rank_first	rank_dense
0	Emily	87	184	C	1.5	1.0	2.0	4.0	4.0
1	Jenny	78	182	C	1.5	1.0	2.0	6.0	5.0
2	Jane	94	173	B	3.5	3.0	4.0	7.0	6.0
3	Adam	84	185	B	3.5	3.0	4.0	3.0	3.0
4	John	92	184	A	6.0	5.0	7.0	5.0	4.0
5	Lisa	82	201	A	6.0	5.0	7.0	2.0	2.0
6	Matt	90	208	A	6.0	5.0	7.0	1.0	1.0

Additional Examples:

Applying Ranking on CSV files

Example #1: Ranking Column with Unique values

In the following example, a new rank column is created which ranks the Name of every Player. All the values in Name column are unique and hence there is no need to describe a method.

```
# importing pandas package
import pandas as pd

# making data frame from csv file
data = pd.read_csv("/content/drive/MyDrive/nba.csv")

# creating a rank column and passing the returned rank series
data["Rank"] = data["Name"].rank()

# display
data

# sorting w.r.t name column
data.sort_values("Name", inplace = True)

# display after sorting w.r.t Name column
data
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary	Rank
152	Aaron Brooks	Chicago Bulls	0.0	PG	31.0	6-0	161.0	Oregon	2250000.0	1.0
356	Aaron Gordon	Orlando Magic	0.0	PF	20.0	6-9	220.0	Arizona	4171680.0	2.0
328	Aaron Harrison	Charlotte Hornets	9.0	SG	21.0	6-6	210.0	Kentucky	525093.0	3.0
404	Adreian Payne	Minnesota Timberwolves	33.0	PF	25.0	6-10	237.0	Michigan State	1938840.0	4.0
312	Al Horford	Atlanta Hawks	15.0	C	30.0	6-10	245.0	Florida	12000000.0	5.0

Example #2: Sorting Column with some similar values

In the following example, data frame is first sorted with respect to team name and first the method is default (i.e. average) and hence the rank of same Team players is average. After that min method is also used to see the output.

```
# importing pandas package
import pandas as pd

# making data frame from csv file
data = pd.read_csv("/content/drive/MyDrive/nba.csv")

# sorting w.r.t team name
data.sort_values("Team", inplace = True)

# creating a rank column and passing the returned rank series
# change method to 'min' to rank by minimum
data["Rank"] = data["Team"].rank(method ='average')

# display
data
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary	Rank
317	Lamar Patterson	Atlanta Hawks	13.0	SG	24.0	6-5	225.0	Pittsburgh	525093.0	8.0
309	Kent Bazemore	Atlanta Hawks	24.0	SF	26.0	6-5	201.0	Old Dominion	2000000.0	8.0
310	Tim Hardaway Jr.	Atlanta Hawks	10.0	SG	24.0	6-6	205.0	Michigan	1304520.0	8.0
311	Kirk Hinrich	Atlanta Hawks	12.0	SG	35.0	6-4	190.0	Kansas	2854940.0	8.0
312	Al Horford	Atlanta Hawks	15.0	C	30.0	6-10	245.0	Florida	12000000.0	8.0
...	...	...	...	...	...	...	...	...	...	...
369	Bradley Beal	Washington Wizards	3.0	SG	22.0	6-5	207.0	Florida	5694674.0	450.0
368	Alan Anderson	Washington Wizards	6.0	SG	33.0	6-6	220.0	Michigan State	4000000.0	450.0
382	John Wall	Washington Wizards	2.0	PG	25.0	6-4	195.0	Kentucky	15851950.0	450.0
370	Jared Dudley	Washington Wizards	1.0	SF	30.0	6-7	225.0	Boston College	4375000.0	450.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 10 columns

```
# importing pandas package
import pandas as pd

# making data frame from csv file
df2 = pd.read_csv("/content/drive/MyDrive/nba.csv")
```

```
df2['average_rank'] = df2['Age'].rank(method='average')
df2['min_rank'] = df2['Age'].rank(method='min')
df2['max_rank'] = df2['Age'].rank(method='max')
df2['first_rank'] = df2['Age'].rank(method='first')
data["Rank"] = data["Age"].rank(method='dense')

print(df2)
```

```
              Name            Team  Number Position   Age Height  Weight  \
0    Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2   180.0
1      Jae Crowder  Boston Celtics    99.0       SF  25.0    6-6   235.0
2     John Holland  Boston Celtics    30.0       SG  27.0    6-5   205.0
3      R.J. Hunter  Boston Celtics    28.0       SG  22.0    6-5   185.0
4    Jonas Jerebko  Boston Celtics     8.0       PF  29.0   6-10   231.0
..             ...             ...     ...      ...   ...    ...     ...
453   Shelvin Mack       Utah Jazz     8.0       PG  26.0    6-3   203.0
454      Raul Neto       Utah Jazz    25.0       PG  24.0    6-1   179.0
455   Tibor Pleiss       Utah Jazz    21.0        C  26.0    7-3   256.0
456    Jeff Withey       Utah Jazz    24.0        C  26.0    7-0   231.0
457            NaN             NaN     NaN      NaN   NaN    NaN     NaN

               College     Salary  average_rank  min_rank  max_rank  \
0                Texas  7730337.0         177.0     155.0     199.0
1            Marquette  6796117.0         177.0     155.0     199.0
2    Boston University        NaN         256.0     236.0     276.0
3        Georgia State  1148640.0          53.5      41.0      66.0
4                  NaN  5000000.0         321.5     308.0     335.0
..                 ...        ...           ...       ...       ...
453             Butler  2433333.0         217.5     200.0     235.0
454                NaN   900000.0         131.0     108.0     154.0
455                NaN  2900000.0         217.5     200.0     235.0
456             Kansas   947276.0         217.5     200.0     235.0
457                NaN        NaN           NaN       NaN       NaN

     first_rank
0         155.0
1         156.0
2         236.0
3          41.0
4         308.0
..          ...
453       233.0
454       154.0
455       234.0
456       235.0
457         NaN

[458 rows x 13 columns]
```