PANDAS

23-04-2021

Introduction:

1. **Data science**:Data science or data analytics is a process of analysing a large set of data points to get answers on the question related to that data set.
2. **Dataframe** is a main object in python. It is used to represent data in rows and columns.
3. The process of cleaning messy data is called **data munging** and **data wrangling.**

Dataframes

Dataframes is a data structure in python used to tabulate the data.
Example :

Code:

```
import pandas as pd
student_info = {'Name':['Neha','Shravan','Rakshith','Prarthana','Rohit'],
          'USN':['1JT19CS055','1JT19CS086','1JT19CS069','1JT19CS063','1JT19CS074'],
          'Attendance':[98,94,86,86,80],
           'remarks':['excellent','very good','very good','very good','good']}
df = pd.DataFrame(student_info)
print(df)
```

output:

	Name	USN	Attendance	remarks
0	Neha	1JT19CS055	98	excellent
1	Shravan	1JT19CS086	94	very good
2	Rakshith	1JT19CS069	86	very good
3	Prarthana	1JT19CS063	86	very good
4	Rohit	1JT19CS074	80	good

Note :Use ctrl + enter to execute a line in the jupyter notebook.

24-04-2021

Types of operations in DataFrames:

1. *df.shape* - displays (rows,columns)
2. *df.head(num)*- displays num rows from top
3. *df.tail(num)*- displays num rows from bottom
4. *df.columns*- displays the index of columns
5. *df.['col name'] / df.col_name*- displays that particular column
6. *df.['col_name'].max()*- displays maximum value in that column
7. *df.['col_name'].min()*- displays minimum value in that column
8. *df.['col_name'].mean()*- displays mean value of that column

9. *df.['col_name'].std()*- displays standard deviation value of that column
10. *df.describe()*- prints the statistics of the numerical values(integer data) in the given table(count,mean,std,min,25%,75%,50%,max)
11. *df[df.col_name>=num]*- display the row which satisfies the given condition
12. *df.set_index('col_name')*- the given column will be the index
13. *df.reset_index* - resets previous index

25-04-2021

DataFrames can be created using:

1. Calling a .csv file
2. Calling a .xlsx file
3. Using dictionaries
4. Using list of tuples(note: while creating mention the columns names)
5. Using a list of dictionary

26-04-2021

.csv file operations

csv stands for "comma-separated values".

Read

1. *read_csv("filename.csv")* - reads the values of a given csv file.
2. *read_csv("filename.csv", skiprows=1*)- skips row number 1 and prints values from the 2nd row of the csv file. (there are several operations using skip)
3. *read_csv("filename.csv", header=None,names["col_names"...])*- if the header row is not present, that is the tabular columns directly begins with data, we use this operations to allot names for the particular columns. If we do not know the names, by stating the header is "None" the columns will get the index values by default.
4. *read_csv("filename.csv", nrows=num)*= the file reads only the "num"number of rows.
5. *read_csv("filename.csv", na_values=["not available",0])/read_csv("filename.csv", na_values=[{ dictionary }]* = the value in the table which read as "not available or 0 are written as NaN. We can also make this possible by entering the values in the form of a dictionary specifying each column.

Write

1) *Read fn…..to.csv("newfilename")*- new csv file is created with the new values as mentioned by the read function.
    a) *index=false* - index value won't be displayed
    b) Header =false

Excel file operations

1. **Converters :** This function converts a given value of the excel to another value according to the function defined to convert it.

2. *startrow=n* - the rows in the excel sheet starts from nth row.
3. We can export only a portion of the data frame to the excel
4. We can create 2 DataFrames and write that to excel in 2 separate sheets.

27-04-2021

Handle missing data

1. Fillna to fill the missing value using different ways
2. Interpolation to make a guess on the missing values using interpolation
3. Dropna to drop rows with missing values

Note:
parse_dates - converts dates from string format to Timestamp
Operation

1. new_df=df.fillna(x)- a new file is created with all the NaN value as “x”
2. We can use a dictionary to specify the columns in which a particular change is desired.
3. new_df=df.fillna(method=”ffill”)- ffill means forward fill, the data of the previous row will be filled in the empty space. [bfill(backward fill) works similarly]
4. axis=”columns” - fills data according to the columns
5. Limit - this restricts the number of spaces that can be filled.
6. interpolate()- this guesses the probable value in case of integers.
7. dropna()- drops or removes all the rows that have NaN.
    a. how - parameter used to decide how the rows are supposed to be dropped
    b. thresh=num - drops a rows if it doesn't have even ‘num’ number of data

28-04-2021

Replace function

1. replace(num,np.NaN)- replaces that num value by NaN
2. replace({ dict },np.NaN- dictionary is used to specify the column in which the specific value should be replaced.
3. Regex or regular expression- these are used to detect patterns.(watch tutorial)
4. replace([list1],[list2]) - the values of list1 will be replaced by list2

29-04-2021

Group by

1. To create group according to column - df.groupby(‘col’)
2. To access the group= for col, col_df in group
3. To access any specific group - g.get_group(‘grp’)
4. %matplotlib inline - to plot graph
   g.plot()

30-04-2021

Concatenate DataFrame

1. **pd.concat()-** joins two dataframes to one dataframe
2. ignore_index=True - all the index number will be continuous in the dataframe
3. key[“”,””]= separate index for the 2 different concatenated df
4. To append columns: use axis argument -> axis=1
5. To create a series and append it with a given dataframe:
   pd.Series([“..”,”..”,”..”...], name=”.....”)

1-05-2021

Merge DataFrames

pd.merge(df1 , df2 , on=”col_name”) - merges to dataframes, in this function it is not necessary to mention the index even if the data are mixed up, it will be rightly allocated in the output.

But this only displays the common elements among the 2 dataframes(intersection or inner join). So, if there is uncommon data, it won't be displayed hence we use **how=”outer”** (union).

Similarly we have how=”left” or how=”right”.

indicator=True - used to indicate from which dataframe did the new dataframe originate from.

2-05-2021

Pivot and pivot table:

1. Pivot = this allows us to transform or reshape the data.

df.pivot(index=” “,columns=” “, values” “)= the given dataframe will be sorted according to the information given.

2. Pivot table: it is used to summarise and aggregate the data inside a dataframe.

df.pivot_table(index=” “,columns=” “)

aggfunc=“...”-it can be sum,count,dif. The default argument is mean.

margins-True- shows aggregate of the given column values.

pd.Grouper(freq=“M”,keys=”...”)= shows the average value of the 2 specified keys

03-05-2021

Reshaping DataFrame using Melt

Pandas melt function provides a way to transform and reshape dataframe

pd.melt(df,id_vars=[“”])

We can change variable name and value name by **var_name** and **value_name** arguments respectively.

id_vars= identifier variables

04-05-2021

Stack / unstack

It is a reshaping method

df.stack()

df.unstack()

05-05-2021

Crosstab

Also called a contingency table. It displays the frequency distribution of the variables.

pd.crosstab(df.___,df.___)

pd.crosstab(df.___,[df.___,df.___])

margin=True -> displays total

normalise='index' -> displays average the given values.

pd.crosstab(df.___,df.___,values=df.___, aggfunc=np.average)

Eg: pd.crosstab(df.Sex,df.Handedness,values=df.Age, aggfunc=np.average)

This displays the average age of people of a particular sex who are left/right handed.

Note= press **shift+tab** to open the documentation of a function.

06-05-2021

Read write data from database

Pandas' **read_sql, read_sql_table, read_sql_query** methods provide a way to read records in a database directly into a dataframe.

to_sql on a dataframe can be used to write dataframe records into a sql table Using sqlalchemy engine.

07-05-2021

Time series analysis

Time series is a set of data points indexed in time order.

Time series analysis is crucial in financial data analysis space. Pandas has built-in support of time series functionality that makes analyzing time serieses extremely efficient.

Datetime index and resample

parse_dates=["name"]-converts string format to datetime/timestamp format

df["yyyy-mm-dd"] - displays the values of that specific date.

df["yyyy-mm"]- displays the values of that specific month.

df["yyyy-mm"].close=shows average

df["yyyy-mm-dd:yyyy-mm-dd"]= displays value for the given range of dates

df.___.resample('M').mean()= displays all the mean values of specified column of a particular month(M).similarly we can find quarterly(Q),business days(B),daily(D),hourly(H)etc.
%matplotlib inline
df.Close.resample('Q').mean().plot(kind="pie")=plots a pie chart for quarterly values.

08-05-2021

Function: date_range
pd.date_range(start="mm/dd/yyyy",end="mm/dd/yyyy",freq='B')=prints the business dates between the given range.
df.asfreq('D',method='pad')=displays daily data, but as value for weekends may be null.. It places the previous days' value in empty spaces.
periods= specifies the number of days
Note: date_range does not cover holidays.

09-05-2021

Holidays
CustomBusinessDay - this is a class documentation which lets us specify or customise the dates.
According to documentation of CustomBusinessDay(weekmask) , we can change the definition of the working days, that is we can change the weekend to the desired day.
holidays=['yyyy-mm-dd']= this day will be set as a holiday
Observance=nearest_workday - if a particular holiday is on a weekend, that holiday will be set on the nearest working day.

10-05-2021

Function: **to_datetime**

1. **to_datetime** function can convert date time string into datetime object or DatetimeIndex.
2. Used to handle invalid datetime strings, different date formats etc.
3. **pd.to_datetime()**
4. **format='%Y$%m$%d'-**for a date 2017$4$1, will be then converted to datetime.
5. **errors="ignore"** = this ignores the errors but doesn't execute for the rest of the code as well
6. **error="coerce"**= this only ignores the part which has error,rest of the code is executed normally

Note: epoch(unix time) is the number of seconds that have passed since Jan 1 1970.

11-05-2021

Period and period index
pd.Period('yyyy')

y.start_time
y.is_leap_year
pd.Period('yyyy-mm',freq='M')
q1.asfreq('M',how='start')
df.T= transposes the table
to_timestamp()

12-05-2021

Timezone handling
Two types of date time in python:

1. Naive (no timezone)
2. Time zone aware datetime

df.tz_localize()
tz_convert()
pytz=In pytz you can find a list of common (and less common) time zones using from pytz import common_timezones, all_timezones

dateutil= dateutil uses the OS timezones so there isn't a fixed list available. For common zones, the names are the same as pytz

13-05-2021

Shift function
df.shift(1):this shifts the column by one

14-05-2021

Memory optimisation
Often datasets that you load in pandas are very big and you may run out of memory.
Link:
pythonspeed.com/articles/pandas-load-less-data/