pandas_cheat_sheet

February 20, 2021

0.1 Pandas Cheat Sheet

Installation Pandas is built on top of Numpy, for better use we need to have NumPy installed.

[ ]:
```
!pip install numpy
!pip install pandas #Installs the library
```

Import

[ ]:
```
import numpy as np
import pandas as pd
```

Data Structure

- Pandas supports up to two-dimentions DataFrame
- 1D objects are called Series.
- 2D objects are called DataFrame.
- The structure is Rows and Columns.

Reading data from a file It supports:

- cvs, sql, json, html, etc.

[ ]:
```
df = pd.read_csv('file.csv') #regular import
df = pd.read_csv('file.csv', index_col=0) #takes the column 0 as the index
df = pd.read_html(url) #needs beautifulsoup for manipulation
#options but not unique
df = pd.read_json()
df = pd.read_sql()
df = pd.read_excel()
```

Saving dataframe as a file

[ ]:
```
df.to_csv('file_name.csv',index=False) #This will save the file as dataframe␣
 ↪without an index
#options
df.to_json()
df.to_sql()
```

```
pd.to_excel()
```

Data type change

[ ]:
```
pd.to_datetime
pd.to_timedelta
```

Atributes

[ ]:
```
df.columns #Returns the columns
df.dtypes #Returns the datatypes
df.index #Retuns the idex
df.shape #Returnsthe shape
df.T #Returns the dataframe inverted
df.values #Returns the values
```

Join, Merge and Concat Common adjustments:

- `ignore_index = True`- When the index is not relevant for the join
- `axis= 0`: adds up rows | `axis= 1` Adds up the columns
- `keys = ['a', 'b', 'c']`- Adds up the DataFrame on certain keys
- `left` join - Use keys from left frame only
- `right` join - Use keys from right frame only
- `outer` - Use union of keys from both frames
- `inner` - Use intersection of keys from both frames

This will return the index and column/row content. In some methods, if we want to modify the dataframe `inplace=True` needs to be specified

[ ]:
```
df_1.join(df_2) #Joins the dataframes based in Index - They must have the same␣
 ↪index

pd.concat([df_1, df_2], axis=1, join='outer', ignore_index=False, keys=None,
          levels=None, names=None, verify_integrity=False, copy=True) #Values␣
 ↪can be changed as needed

df_1.pd.append(df_2, sort = True, inplace = True) #This adds dataframes with the␣
 ↪same column structure and names

dataframes = {'a': df_1, 'b': df_2, 'c': df_3}
new_df  = pd.concat(dataframes) #Concats the dataframes indicated in the␣
 ↪dataframes dictionary and create a new column indicating the origin of the␣
 ↪data

pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None,
         left_index=False, right_index=False, sort=True,
         suffixes=('_x', '_y'), copy=True, indicator=False,
         validate=None)
```

Columns addition and creation

```
[ ]: df['sum_col1_col2'] = df['col_1'] + df['col_2']
```

Inspecting and Visualizing data without change in dataframe Use this if you want to see the values of certain columns or rows.

```
[ ]: df.head() #Returns the first 5 values

df.tail() #Returns the last 5 values

df['column_name'] #Returns all the data in the column
df.column_name #Returns all the data in the column
df['column_name', 'second_column'] #Returns columns as a new DataFrame
df[7:9] # Displays the values of rows 7 to 9
df.value_counts(dropna=False)  #Retuns unique values and counts
df.sort_index(axis=0, ascending=False) #Returns dataframe sorted by index
df.apply(pd.Series.value_counts) #Returns values and counts for all columns
df.sort_values(by='column_name') #Returns dataframe sorted by the column selected
df.groupby('column_name').mean() #Returns dataframe grouped by column name and␣
 ↪the mean
df.pivot_table
df.iloc[0] #Selection by position
df.loc[`index_one`] # Selection by index
df.iloc[0,:]  #Returns First row
df.iloc[0,0] #Returns element of first column

#Simple examples that can be adapted as needed

df[df['is_muy_value'] == 1][['what_im_looking_for']]

df[df['column_1'] < 10].groupby('column_2').mean()[['what_im_looking_for']]
df[df['column_1'] == 0].sort_values(by='column_2', ascending=False).head()
```

Replacing and renaming

```
[ ]: df.columns = ['column_1', 'column_2'] #Renames columns
df.rename(columns={'old_name': 'new_ name'}) # Selective renaming

df.replace(1,'one') #Replace all values equal to 1 with 'one'
df.replace([1,3],['one','three']) #Replace all 1 with 'one' and 3 with 'three'


df.set_index('column_1') #Changes the index
df.astype(int) #Converts the datatype of the series to integer - It can be␣
 ↪changed to any datatype

df['column_1'].astype(int) #changes the datatype
```

Null Values

- isna = isnull
- notna = notnull

```
[ ]: df.fillna(value = 'my_selected_value', inplace = True) #Fills all NANs with the␣
      ↪value we selected and make the change permanent
     df.fillna(x) # Replace all null values with x
     df.notna().sum() #Sums of nas per column
     df.interpolate()
     df.isna().sum() #Returns True/false to NAs
     df.isnull()
     df.dropna(inplace = True) # Drops null values permanently
     df.isnull().sum() #Prints null values agregated by column
     df.isnull().sum()[df.isnull().sum() !=0].sort_values().plot(kind='barh'); #Plots␣
      ↪the null values

     #Advance replacing:
     df.fillna(df.mean()) #Replace all null values with the mean

     #Other way to overwrite the dataframe without the NAs in specific column
     df.column.fillna(value='no_info', inplace=True)
     df= df.loc[df['column'] != 'no_info']
```

Dropping

```
[ ]: df.drop(['column_1', 'column_2'], axis=1, inplace = True) #drops specific columns
     df.drop_duplicates(inplace=True) #drops duplicates permanently
     df.drop('row_1', axis=0, inplace = True) #drops the row permanently
```

Agregation Methods, stadistical methods and summaries Can be used in way `df.sum` and `df.sum()` way

```
[ ]: df.count() # Returns the number of non-null values in each DataFrame column
     df.describe() # Summary statistics for numerical columns
     df.max() #Returns the highest value in each column
     df.mean() #Returns the mean of all columns
     df.median() # Returns the median of each column
     df.min() # Returns the lowest value in each column
     df.mode() #Returns mode
     df.std() # Returns the standard deviation of each column
     df.var() #retuns varianza
     df.abs() #Returns absolute values
     df.corr() # Returns the correlation between columns in a DataFrame
     df.round() #rounds the number
```

0.1.1 Other to explore

```
df.clip()
df.nunique()
df.idxmax()
df.idxmin()
df.cov()
df.cummax()
df.cummin()
df.cumprod()
df.cumsum()
df.diff()
df.nlargest()
df.nsmallest()
df.pct_change()
df.prod()
df.quantile()
df.rank()
```

0.1.2 Simple Functions

Dummy Variables https://socialresearchmethods.net/kb/dummyvar.php

```
df = pd.get_dummies(df, columns=['my_column'], drop_first=True) #I dummy variables we drop the first column to not to make it redundant
```

```
#Does this column have the value I'm looking for?
def is_the_value_im_looking(i):
    val = i.split()
    if 'value' in str(val):
        return 1
    else:
        return 0

#Create a column called as the value I'm looking for and adds 0 or 1
df['value'] = df['col_1'].apply(is_the_value_im_looking)
```

```
#Extracts the title from everyone's name and create dummy columns, made with list comprehension.
#This can be adapted as needed

df['Title'] = [each.split(',')[1].split('.')[0].strip() for each in df['Name']]
```

```
#Rate per column.
#this can be adapted as needed.

for i in ['column_1', 'column_2', 'column_3']:
```

```
    print(i, ':')
    print(df[df[i] == 1][['the_value_im_lookingfor']].mean())
    print()
```

0.1.3 Plotting

Simple plotting examples

[ ]:
```
df.groupby('column_1').mean()[['value']].plot(kind='barh')
plt.title("plot title");
```

[ ]:
```
df.groupby(['column_1', 'colum_2']).mean()[['value']].plot(kind='barh');
```