Python Handy Tips

Mahdi Dor Emami

August 27, 2020

Contents

1 Introduction

This literature is not a "0 to 100" python learning cookbook or textbook or anything! This is made of only the most useful commands being able to work with python, Written by Mahdi Dor Emami. I hope it contains useful information for you ladies and gentlemen.

Also, there are some python files that contain some good examples on specified topic (by the name of the file) with deep explanations inside the files.

2 Python General Commands

2.1 print command tips

print() ⇒ Simply prints... but let's review some tips through examples.

Example:

>>>print("tell %s(or %d or %f) people" %5)

'tell 5 people'

>>>print("tell {} people" .format(5))

'tell 5 people'

>>>print("tell %s(or %d or %f) + {} people" .format(6)%5

'tell 5 + 6 people'

>>>print("tell {} + %s(or %d or %f) people" .format(6)%5

'tell 6 + 5 people'

Better than all of the above, and most robus technique :

>>>print(f"tell {variable1} + {variable2} (or {variable3}or {variable4}) people"

In this way, you can easily add variable name inside the {} without needing to write further things like % or .format{}.

2.2 Measuring Time

You have to use library **timeit** to measure time, not **time** library. Because it's way more accurate. Do as below:

Example:

>>>import timeit
Imports the library

>>>timeit.timeit('2*3')
Returns the duration of the statement 2*3 to be done.

timeit.timeit(stmt, setup,timer, number) :
stmt:
This will take the code for which you want to measure the execution time. The default value is "pass".
setup:
: This will have setup details that need to be executed before stmt. The default value is "pass."
timer:
: This will have the timer value, timeit() already has a default value set, and we can ignore it.
number:
: The stmt will execute as per the number is given here. The default value is 1000000.

>>>timeit.timeit('char in text', setup='text = "sample string"; char = "g"')
Calculates the time that is necessary for the statement "char in text" that says if char is in text or

not. But what is char and text? They are pre-settings that need to be passed in setup.

```
>>>timeit.repeat(stmt,setup,timer,number,repeat)
```
This function, repeats measuring the time for "repeat" number. Consider below example:

```
>>>timeit.repeat('char in text', setup='text = "sample string"; char = "g"',number=1000000,repeat=10)
```
This function, measures the time for the operation "char in text" to be done for one million times (if 'char in text' is measured one time, it's a very small number representing its duration). repeat, does it for 10 times. It will give you a list with 10 time measurements that measured the time for 'char in text' command running one million times.

2.3 .join functionality

Just pay attention to the examples! Run them in your Python IDE and see the results.

Example:

```
>>>"".join(['Fuck You %d times '% i for i in range(1000)])
```
Prints Fuck You 1 until Fuck You 999

```
>>>"".join(['Fuck You' + ' ' + str(i)+' ' for i in range(1000) ])
```
Same as before. Pay attention that there's a space in between the quotes after + sign.

```
>>>'-'.join(str(m) for m in range(100))
```
The result is as below:

'0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-
26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-
51-52-53-54-55-56-57-58-59-60-61-62-63-64-65-66-67-68-69-70-71-72-73-74-75-
76-77-78-79-80-81-82-83-84-85-86-87-88-89-90-91-92-93-94-95-96-97-98-99'

2.4 Lambda Function

It's a super cool feature, it acts like a function. Just do the examples below to find out. **Example:**

```
>>>x = lambda a : 3*a

>>>print( x(10) )

    30
>>>def myfunc(n):

>>>     return lambda a : a * n

>>>mydoubler = myfunc(2)

>>>print(mydoubler(11))

    22
```

2.5 Creating a new variable in each iteration of a for loop

Sooo cooooooooooooooL : **Example:**

```
>>>for x in range(0, 9):
>>>    globals()['var%s' % x] = 'Hello'
```

Aother example :

```
>>>for x in range(0, 9):
>>>    globals()['var%s' % x] = 'Hello%s'%i
```

Imagine what you can do with this capability and the pickle library :

```
>>>for i in range(5):
>>>    dat = i
>>>    tmp = 'data_%s'%i
>>>    with open(tmp,'wb') as f:
>>>        pickle.dump(dat,f)
>>>    f.close()
```

2.6 Iterator

x=iter(mylist) ⇒ make an iteration on mylist. You can use the following commands to be able to see the content of mylist:

Example:

```
>>>x = iter(mlist)
>>>next(x)
```

2.7 Python Generator

You must define it inside a function... The main item of Generator is 'yield' command that must be inside a function. **Example:**

```
>>>def indexer(text):
>>>    for i,char in enumerate(text):
>>>        yield i,char
>>>res = indexer('Mahdi')
>>>next(res)
```

Just run this function and run next(res) as many times as the length of res. see what will happen...

3 Sublime Text Editor Hotkeys

3.1 First, install Package Control

Install Package Control since it's a very useful module to install other modules. You can install it while you are inside Sublime Text environment by pressing **Ctrl+Shift+P**.

3.2 Conda

After installing Package Control, press **Ctrl+Shift+P**, then type Package Control. Now, all the commands related to Package Control, including installing a package, or removing a package will appear. Select install Package and type Conda in the upcoming window. By installing Conda, you can have your Anaconda running inside.
Now, in Tools¿Build System, select Conda.
You can configure settings to paths if necessary in Tools¿Package Settings> Conda> Settings - User. (Only if you changed paths when you were installing Anaconda).

After Conda is installed, by pressing **Ctrl+Shift+P**, and typing Conda, related commands will pop up and you can select them. Creating an environment, Activating it, etc. Select Activate Environment, and among all environments you have, select the one you want to work with.
Finally, save your file in a location with .py extension.
Now, everything is up and running. You can **compile the code by pressing Ctrl+B**.
Enjoy coding in Sublime.

3.3 Some Handy tips and Hotkeys

By pressing the **'Middle-Mouse-Button'** and selecting a few things, you will understand that you are able to select different things from different lines in a specific row or column, and edit them all together..... Suuuuperb!!

Also, by **holding Ctrl, and click multiple places**, you can edit all those places at once.

You can have **multiple .py files opened in different tabs**. By pressing **Ctrl+P**, you can easily **navigate through them from the pane** that will pop up.

By pressing **Ctrl+D**, you can select a word your typing indicator stays on and by pressing **Ctrl+D** multiple times, all the same words will be selected from later lines and you are able to change them at once... Awesome!

By pressing **Ctrl+Shift+D**, you can **copy a line**... without even selecting the whole line; It's just enough that your typing indicator to be in that line...

By pressing **Ctrl+/, you can comment the lines**.

Moving a code line up or down... Now, this is a real deal here... Remember when you wanted to bring a line above... All that cut and paste shit... Now, by pressing **Ctrl+Shift+up/down**, you can really save your time.

Speak of **cutting** sth, without highlighting, only if your indicator is at the line, just press **Ctrl+X** and enjoy this powerful editor.

Select current lines and keep selecting next lines by **Ctrl+L**.
Also, you can edit these selected lines by **Ctrl+Shift+L**.

If you are in the middle of the line and you want to remove the rest till the end of it, you usually use Shift+end buttons to select that and then delete it. But, by pressing **Ctrl+Shift+delete, it will remove the rest of the line**.
Ctrl+Shift+Backspace will delete texts from your indicator to the head of the line

Ctrl+Backspace, will delete a word.

Multicopying... Shit... Just select whatever you want by **holding Ctrl, and select sth by mouse**, and then copy them. Now, select all other things you want to change with what you just coppied in the same way, and now paste... Brilliant!! Becareful that order matters.

There will be a beter PDF file in here inside the Misc folder. Thanks for reading.

4 Have your files with Glob Library

4.1 import glob

Imports the library.

4.2 glob.glob('*')

Shows whatever thing is in the Current Working Directory (CWD).

4.3 glob.glob('*.png')

Shows all .png files in the CWD.
You can do it with any other extension.

4.4 glob.glob('R*')

Shows all the files is in the CWD starting with R

5 matplotlib Library

5.1 import matplotlib.pypelot as plt

import matplotlib.pypelot as plt ⇒ Imports the library matplotlyb.pypelot and names it as plt

5.2 plt.plot(...)

plt.plot(...) ⇒ plots something... you can use "scatter","stem" etc instead of "plot" in "plt.plot()" command

5.3 plt.show()

plt.show() ⇒ plt.plot(variable) will make it ready to be shown by plt.show(), so if you want to see your figures, use it!

5.4 plt.xlabel(name)

plt.xlabel(name) ⇒ Puts a label on x-axis. You can do the same with plt.ylabel(name) for y-axis.

5.5 Subplots

plt.subplot(mnp) ⇒ m*n grid plot, which the pĥ elements (column-wise) would be plotted now. **Remember to put the settings in yout IDE (Mine was Spyder) to use "Qt" to show the plots. It is available in Spyder IDE in "Tools (tab)>Preferences>Ipython console>Graphics (tab)> Graphics Backend> Put it on Qt4 or Qt5".It affects new kernels.**

Example:

```
>>>plt.subplot(131)

>>>plt.plot(Something)

>>>plt.subplot(132)

>>>plt.plot(Something)

>>>plt.subplot(133)

>>>plt.plot(Something)

>>>plt.show()
```

6 os Library

6.1 import os

import os ⇒ Imports the library

6.2 os.getcwd()

os.getcwd() ⇒ Would show the current working directory

6.3 os.mkdir('name')

Creates a directory with the name 'name'.

6.4 os.removedirs('name')

Removes the directory with the name 'name'.

6.5 os.chdir(Newdirectory)

os.chdir(Newdirectory) ⇒ Would change the current directory to the new one. Remember to put two \s instead of one **Example:**

```
>>>os.chdir('C:\\Users\\Mahdi\\Desktop\\Datasets to work on')
```

6.6 os.path.join(Further Address)

os.path.join(Further Address) ⇒ Adds a piece of address to your previous address.

Example:

```
>>>Address = os.chdir('C:\\Users\\Mahdi\\Desktop\\Datasets to work on')

>>>NewAddress = os.path.join(Address,"Cars")

>>>print(NewAddress)

    'C:\Users\Mahdi\Desktop\Datasets to work on\Cars')
```

6.7 os.walk

You have to use it in the way explained below to be able to get all directories, subdirectories, and all files in a directory.

Example:

```
>>>for a,b,c in os.walk(os.getcwd()):
>>>        print(f"a:a")
>>>        print(f"b:b")
>>>        print(f"c:c")
```

Just test it to see how it works.

6.8 To Loop over files in a directory

Do as below:

Example:

```
>>>directory = os.getcwd()
# Comment
>>>for filename in os.listdir(directory):
# os.listdir(directory) lists everything inside the directory
>>>        if filename.endswith(".asm") or filename.endswith(".py"):
# filename.endswith('extension') searches for the files that have the specified extension in the listed items
in directory
>>>            print(os.path.join(directory, filename))
```

YOU CAN DO THIS WITH pathlib LIBRARY TOO. JUST GO TO THE pathlib SECTION.

There is also a good other function that can give the filename and extensions:

```
>>>for file in os.listdir():
>>>        filename,file_extension = os.path.splitext(file)
>>>        print(f'Filename = [filename] && File_extension =[file_extension]')
```

7 numpy Library

7.1 import numpy as np

import numpy as np ⇒ Imports numpy library and makes np as the contraction name for numpy library

7.2 np.array(list)

x=np.array(list) ⇒ Makes the 'list' a numpy array

7.3 Dimension and Shape

np.dim() && np.shape() ⇒ We can use both of these commands in order to get the dimension or shape of any types of data like tuple, list, dict, etc...

7.4 np.array(list).astype('float')

x=np.array(list).astype('float') ⇒ Makes the 'list' a numpy array with float type elements. There are some good functions for numpy arrays like:

```
>>>np.max()# Gives the maximum value in the array (or matrix)

>>>np.min()# Gives the minimum value.

>>>np.mean()# Calculates the mean.

>>>np.argmax()# Gives the index refering to the maximum value in the array (or matrix).
```

7.5 Reshape

x.reshape(size,order=) ⇒ Reshapes the data in specific size with sepecific order. order values are : order='C', order = 'F',order='A'

C means to read / write the elements using C-like index order, with the last axis index changing fastest, back to the first axis index changing slowest. F means to read / write the elements using Fortran-like index order, with the first index changing fastest, and the last index changing slowest. Note that the C and F options take no account of the memory layout of the underlying array, and only refer to the order of indexing. A means to read / write the elements in Fortran-like index order if a is Fortran contiguous in memory, C-like order otherwise.

7.6 np.loadtxt(rawdata,delimiter=",")

x=np.loadtxt(rawdata,delimiter=",") ⇒ Turns a CSV file which is loaded by open(filename,'rt') in the directory, into a numpy array with float values.

7.7 np.dstack(TUP)

The code concatenates two vectors,matrices, whatever, in 3rd axis (depth). Consider a,b be both vectors of dimension (M,) . np.dstack((a,b)) returns a (1,M,2) tensor since first, a dimension will be added to both so they become (M,1). Then another dimension is inserted, this time to the head and makes it (1,M,1) for each a,b. And finally, both will be concatenated along the third axis. The result is a (1,M,2) tensor.

There is a difference between np.dstack(TUP) and np.concatenate(TUP,axis=2). First, in **np.concatenate(TUP,axis** the axis values are in range (0,1,2,...) where 0 is rows, 1 columns, 2 depth ,3 higher dimension,etc. So now, we want to concatenate a,b vectors with dimension (M,) on the third dimension. a[:,None] will add another dimension to a , so a becomes (M,1). **np.concatenate((a[:,None,None],b[:,None,None]),axis=2)** will get us a tensor of shape (M,1,2).

np.hstack((a,b)) concatenates a and b on rows so it will give a (2*M,) dimension matrix.

np.vstack((a,b)) concatenates a and b on columns so it will give a (2,M) dimension matrix. It goes vertically... So, we have two flat vecs a=[1,2,3] and b=[4,5,6], It'll go vertically like [[1,4],[2,5],[3,6]].

7.8 np.ndarray VS np.array

No differences... But, np.array is much more high-level, so use np.array all the time.

8 pandas Library

8.1 import pandas

import pandas ⇒ Imports pandas library

8.2 from pandas import read_csv

from pandas import read_csv ⇒ Imports a sublibrary that is expert to open csv files and turn them into "Pandas Data Frames"

8.3 pandas.Series(data=Data, index=labels)

panda.Series ⇒ Makes a labeld vector from one dimensional Data.

8.4 pandas.DataFrame(data=Data, index = rowlabels , columns = column labels)

pandas.DataFrame ⇒ Makes a row&column labled dataset. One can refer to the labeled data in "method" way:

Example:

```
>>>a=[[2,4,5],[4,3,2]]

>>>rows = ['first','second']

>>>column = ['one','two','three']

>>>Data=pandas.DataFrame(a,index=rows,columns=column)

>>>Data

        one two three
first     2   4     5
second  4   3     2

>>>Data.one

first    2
second 4
Name: one, dtype: int64

>>>Data.three

first    5
second 2
Name: one, dtype: int64

#Notice that this method only works for column labels, not for rows
```

8.5 read_csv(filename,names=Names)

Dataset = read_csv(...) ⇒ Makes a DataFrame out of a CSV file (Comma Seperated Value file). It loads the dataset and tags the Names on each column of it. If the dataset exists in the current directory, filename is only the name of the file plus its extension; Otherwise, you have to refer the whole address of the dataset plus name and extension of the file. One can use a URL address instead of filename as well.

8.6 DATA.head(#)

DATA.head(#) ⇒ If DATA is a pandas DataFrame, this .head(#) method would show the first # rows of the dataset, in order to have a quick perspective.

8.7 Data.dtypes

Data.dtypes ⇒ Represents every column's datas' types.

8.8 Data.describe()

x=Data.describe() ⇒ Gives the necessary statistical information on the data like "mean" , "standard deviation" , etc. You can also set some options related to the percision like : set_option('precision',3) which sets the decimal point precision.

8.9 Data.groupby('GROUP NAME').size()

Data.groupby('GROUP NAME').size() ⇒ Returns the number of occurance for each value in the specified group (column) of the data. "GROUP NAME" is the name of the column in the dataset.

8.10 Data.pop('GROUP NAME')

Picks the specified column from the dataset and stores it in any variable you want.

8.11 Data.loc[index]

Shows the specified row from the dataset woith its full features.

8.12 Data.isna()

Shows "unknown" or "None" values in the dataset.

8.13 Data.isna().sum()

Counts "unknown" or "None" values in the dataset.

8.14 Data.dropna()

We'll get rid of "unknown" or "None" values in the dataset.

8.15 Data.corr(method='pearson')

Data.corr(method='pearson') ⇒ gives the correlation of the data with each other by using "pearson" method. There are other methods for calculating the correlation like "spearman" or "kendall".

8.16 Plotting Correlation Graph

Imagine you loaded the dataset and named it as DATA.Watch the example below to learn how to do it:

Example:

```
>>>correlations = DATA.corr()

>>>fig = plt.figure()
# Have to store in order to make changes in it.

>>>ax = fig.add_subplot(111)
# A 1 by 1 plot.

>>>cax = ax.matshow(correlations , vmin = -1 , vmax = 1)
# Plots the correlation matrix

>>>fig.colorbar(cax)
# Places a colorbar next to the main plot relevant to the correlation matrix plotted there.

>>>names = ['preg','plas','pres','skin','test','mass','pedi','age','class']
# Makes names

>>>ticks = np.arange(0,9,1)
# Arange the numbers based on the classes of your database. Mine are nine classes.

>>>ax.set_xticks(ticks)
# You have to pass this step to correctly be able to label your data. Otherwise, the labels are goanna
be tagged in a wrong manner. This would tell the correlation chart to count the classes from 0 to 9

>>>ax.set_yticks(ticks)
# Same manners like previous line of code.

>>>ax.set_xticklabels(names)
# Names all the columns

>>>ax.set_yticklabels(names)
# Names all the rows
```

>>>ax.set_xlabel('DATA CLASSES')
Sets the main label for x-axis

>>>ax.set_ylabel('DATA CLASSES')
Sets the main label for y-axis

8.17 Pipeline in pandas

Pipelines generally are used to do a number of functions sequentially. Imagine you have two different functions : func1 and func2 , whose parameters are (DataFrame , Column). If your Pandas.DataFrame structure name is "data" for example, the following command will apply all functions on the column "Age" of "data" DataFrame.

Example:

>>>automation = data.pipe(func1 , Col='Age') , data.pipe(func2 , Col='Age')

In this way, you will have an automatic way of doing several functions on several groups of data.

8.18 Data.skew()

Data.skew() ⇒ Gives the skew of a Normal Gaussian distributed data. Poitive skew means that the data histogram is right taled; Better to say it's concentration is on the left part. And Negative skew means that the data histogram is left taled; It means that the concentration is on the right side. The skew formula is $\frac{(mean-median)}{std}$

8.19 Normalizing

We can normalize the data:

Example:

Everything is clear from the code.

>>>des = dataset.describe().transpose()

>>>new_dataset = (dataset - des['mean']) / des['std']

8.20 Test,Train Split

You can do splitting on the dataset:

Example:

Everything is clear from the code.

>>>train = dataset.sample(frac=0.6 , random_state = seed)

>>>valid = dataset.drop(train.index)

>>>validation = valid.sample(frac=0.4 , random_state = seed)

>>>test = valid.drop(validation.index)

8.21 One Hot Encoding

You can do a one hot encoding by two ways for your dataset:

Method I:

Imagine that the 'sex' feature of your dataset is a categorical feature containing only 'male' and 'felmale' as values.

>>>dataset['sex'] = dataset['sex'].map({'female':0 , 'male':1})

Easy peasy

Method II:

You can do it automatically by this command of pandas:

>>>new_data = pd.get_dummies(dataset)

It's actually doing a one hot encoding for it... And added them to your features.

Method III:

This method is for categorical features that are represented as numbers. For example, the samples are in 4 classes determined by 0,1,2,3 digits. You can do a one hot encoidng on this feature as below:
>>>dataset['class'].unique()# Checks the unique things in class. In our example, it will return 0,1,2,3

array([0,2,1,3] , dtype=int64)

>>>dataset['class'] = dataset['class'].map({0:'class1' , 1:'class2' , 2:'class3' , 3:'class4'})

First, we replace the 0,1,2,3 with class1,class2,class3,class4.

>>>new_data = pd.get_dummies(dataset)

Finally, we change it.

There are some arguments in pd.get_dummies() like prefix , prefix_sep. If you pass nothing, prefix will be the previous feature name and prefix_sep will be _. You can change these like for example: prefix='@' , prefix_sep ='^'

8.22 data.hist()

data.hist() ⇒ Prepares the histogram plots for all the columns (groups or classes) of the data in one figure. This will be shown by the command plt.show().

8.23 data.plot(Options)

data.plot(kind = 'density' , subplots = True , layout = (subplot dimension... like (3,3)) , sharex = False or True , sharey = False or True) ⇒ Would plot different figures for all the groups with a few settings. Like density figure which is declared in "kind" option. There are other "kind"s as well, like " line " , " bar , " barh " , " box " , " kde" , "density" , "area" , "pie" , "scatter" , "hexbin". subplot option must be turned on if you want to plot for all of the groups, and the dimension should match the number of your sub classes. The last options are sharey and sharex that if they are set to "True", the axis x or y related to sharex or sharey, wouldn't be scaled properly. Otherwise, if you set it

to "False", the scale is shown. density plot, is like histogram with continues lines. And box, is a special type of plot that draws a box around the middle 50% of the data; Meaning it covers the 25th percentile upto 75th percentile. Also there is a line inside the box remarking the median.

8.24 DATA.values

x=DATA.values ⇒ Turns a DataFrame into a numpy array. So, it would be indexed normally with ""[]"". Otherwise, if it was a DataFrame, it would have been refrenced by its class names. (DATA.park or DATA['park']

9 seaborn Library

9.1 import seaborn as sns

import seaborn as sns ⇒ Imports the seaborn library useful for statistical data analysis.

9.2 Correlation Matrix

sns.heatmap(CorrMat , annot=True , vmin=-1 , vmax = 1) ⇒ Returns a figure containing the correlation of all of the groups of a DataFrame within (-1,1). CorrMat can be data.corr(). annot, means the number in each cell. Run the code with and without it to understand.

9.3 Scatter Matrix

sns.pairplot(dataset) ⇒ Returns the scatter-matrix related to the dataset.

9.4 Scatter Matrix II

sns.pairplot(dataset,hue='class') ⇒ All attributes in relation with "class" for classification possibility checking

10 sklearn Library

10.1 from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import MinMaxScaler ⇒ Would import the "MinMaxScaler" algorithm from preprocessing part of sklearn library suitable for scaling the data between 0 and 1.

Example:

```
>>>scaler = MinMaxScaler(feature_range(0,1))# Sets the range of normalization

>>>rescaled = scaler.fit_transform(DATA)# Applies the normalization to the DATA.
```

10.2 from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import StandardScaler ⇒ Imports the StandardScaler library that will scale the dataset with the mean being 0 and the std being 1.**Linear Regression or Logistic Regression are the algorithms the StandardScaler is useful and the times one wants to make a standard Gaussian Distributed input.**

Example:

```
>>>scaler = StandardScaler().fit(DATA)# Sets the required setting.

>>>rescaled = scaler.transform(DATA)# Applies the changes.
```

10.3 from sklearn.preprocessing import Normalizer

from sklearn.preprocessing import Normalizer ⇒ Imports the library to normalize the data. **Normalizing would be used for sparsed Data and K-Nearest Algorithms.**

Example:

```
>>>scaler = Normalizer().fit(DATA)# Sets the required setting.

>>>normalized = scaler.transform(DATA)# Applies the changes.
```

10.4 from sklearn.preprocessing import Binarizer

from sklearn.preprocessing import Binarizer ⇒ Imports the library to make a binary representation of the data.

Example:

```
>>>scaler = Binarizer(threshold=0.0).fit(DATA)# Sets the required setting.

>>>binarized_data = scaler.transform(DATA)# Applies the changes.
```

10.5 SelectKBest(score_func=chi2,k=4)

The above code ⇒ Would be used for feature selection. It would use the scoring statistical function "chi2" in order to select the k best scored features and keep them as the necessary features for all of the classes in dataset. **It will select the features which has the strongest relationships with the output.**You will have to import two libraries and then apply this algorithm on the training and target

sets as below:

Example:

```
>>>from sklearn.feature_selection import SelectKBest# Imports the library SelectKBest

>>>from sklearn.feature_selection import chi2# Imports the scoring function chi2

>>>array = DATA.values# Turns the pandas DataFrame into an array

>>>X = array[:,0:8]
# Selects 8 classes of the dataset to be as training sets

>>>Y = array[:,8]
# Puts the last to be as the target set.

>>>test = SelectKBest(score_func_chi2 , k=4)
#Enables the algorithm that selects the 4 best classes for all of the dataset with chi2 statistical algorithm

>>>fit = test.fit(X,Y)
# The algorithm is applied to X,Y... So best scores to get from X to Y are selected

>>>print(fit.scores_)
# Prints out the scores related to each class of the dataset X. The top score classes are goanna be selected.

>>>features = fit.transform(X)
# Now applies the transform to X. So we have the features we want for training.

>>>print(array.shape)
# Main dataset dimension

    (768,9)

>>>print(X.shape)
# Training dataset dimension

    (768,8)

>>>print(features.shape)
# The selected features for training shape.

    (768,4)
```

10.6 PCA(n_components=N)

The code ⇒ Is for PCA algorithm to reduce data size. It chooses N

Example:

```
>>>Python code
# Comment

    result
```

10.7 OneHotEncoding

Makes a categorical thing into numbers. Like male,female seperation which names them 01,10 for example.

```
>>>from sklearn.preprocessing import OneHotEncoder
# Imports library

>>>OneHotEncoder().fit(X).transform(X).toarray()
# Returns the encoded array

>>>OneHotEncoder().fit_transform(X).toarray()
# Returns the encoded array (Just like above)
```

11 OpenCV Library

This section, is mainly from "Mastering OpenCV4 with Python" from Alberto Fernandez Villian.

11.1 import cv2

import cv2 ⇒ Imports the OpenCV library.

11.2 Primary Works on Image

Due to a lot of codes, I'll be explaining this part a little bit different. Mostly, one line code one line comment, and further, with examples.

```
>>>image = cv2.imread('1.jpg')
# Imports the image in the directory. Becareful that imported image (frames of the videos as well), are in BGR format; Not RBG!! Because it was more popular with cameras.

>>>gray_image = cv2.imread('1.jpg' , cv2.IMREAD_GRAYSCALE)
# Imports the image in gray-scale

>>>height,width,channels=image.shape
# Gives hiehgt, width, and number of channels of the image (RGB num of channels:3 grayscale:1)

>>>total_pixels = image.size
# Number of pixels

>>>image_type = image.dtype
# Image type

>>>image_grayed = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
# Turns an image into gray. It's a kind of process and you can change any image in everywhere of your code. Just notice at BGR2GRAY. If you had an RGB image, write RGB2GRAY.

>>>modified_image = image
>>>modified_image[400:500,400:500,:] = image[600:700,600:700,:]
# You can modify images.

>>>B,G,R = cv2.split(image)
# You can split the channels of an image

>>>RGB_image = cv2.merge([R,G,B])
# You can merge the channels and create a colorful image with any format you like! RGB or BGR. In this code, we created an RGB image.

>>>b = image[:,:,0]
>>>g = image[:,:,1]
>>>r = image[:,:,2]
>>>rgb_image = image[:,:,::-1]
>>>rgb_image_gray = cv2.cvtColor(rgb_image,cv2.COLOR_RGB2GRAY)
# Working with Numpy library to split channels, and build an RGB image. Last line of code is making the RGB image in gray-scale.

>>>con_image_1 = np.concatenate((gray_image,image_grayed),axis=1)
>>>con_image_2 = np.concatenate((RGB_image,rgb_image),axis=1)
>>>con_image_3 = np.concatenate((RGB_image_gray,rgb_image_gray),axis=1)
```

A little numpy to stick a few pictures together horizontally(axis = 1) or vertically(axis = 0)

>>>cv2.imshow("Title",image)
Shows the image. The title must be set. It's the title of the windows displaying the image.

>>>cv2.waitKey()
Without this command, you can not see the displayed image. You can set time in mili-seconds inside parantheses to specify the duration of the display.

>>>cv2.destroyAllWindows()
It's necessary if you want to get rid of the display window and shut it with any keys on your keyboard.

>>>plt.figure,plt.subplot(nmp),plt.imshow(image)
You can display images with pyplot.imshow and subplots too.

>>>cv2.imwrite('Gray.jpg',image_grayed)
Finally, you can export the image with cv2.imwrite command, in your working directory. It needs a name and the image to export as arguments.

11.3 Primary Works on Video

Explainations will be like the last sub-section.

>>>video = cv2.VideoCapture('1.mp4')
Imports the video. If you put a URL address related to your public camera, it would get that video for you. Or, if you put 0 (zero) as the argument, it would get your webcam as the input video.

>>>video.isOpened()
Checks if video is loaded correctly and available to process

>>>ret, frame = video.read()
reads the first frame and saves it as frame. and ret is True if this action is done

Codes below would give you properties about the video. The **Bold** ones are more important than the others.

>>>**video.get(cv2.CAP_PROP_FRAME_WIDTH)**
>>>**video.get(cv2.CAP_PROP_FRAME_HEIGHT)**
>>>**video.get(cv2.CAP_PROP_FPS)**
>>>**video.get(cv2.CAP_PROP_POS_MSEC)**
>>>**video.get(cv2.CAP_PROP_POS_FRAMES)**
>>>**decode_fourcc(capture.get(cv2.CAP_PROP_FOURCC))**
>>>**video.get(cv2.CAP_PROP_FRAME_COUNT)**
>>>video.get(cv2.CAP_PROP_MODE)
>>>video.get(cv2.CAP_PROP_BRIGHTNESS)
>>>video.get(cv2.CAP_PROP_CONTRAST)
>>>video.get(cv2.CAP_PROP_SATURATION)
>>>video.get(cv2.CAP_PROP_HUE)
>>>video.get(cv2.CAP_PROP_GAIN)
>>>video.get(cv2.CAP_PROP_EXPOSURE)
>>>video.get(cv2.CAP_PROP_CONVERT_RGB)
>>>video.get(cv2.CAP_PROP_RECTIFICATION)
>>>video.get(cv2.CAP_PROP_ISO_SPEED)
>>>video.get(cv2.CAP_PROP_BUFFERSIZE)

Most of them are clear. The important ones (Bolded ones) are width, height, and FPS of the image; Along side with FOURCC command wich will give the codec of the video. The function "decode_fourcc"

is as below:

```
>>>def decode_fourcc(fourcc):
        """Decodes the fourcc value to get the four chars identifying it"""
        # Convert to int:
>>>       fourcc_int = int(fourcc)
        # We can do this one line command below, or go the usual way as for loop:
        # One line:
>>>       return "".join([chr((fourcc_int ¿¿ 8 * i) & 0xFF) for i in range(4)])
        # Typical:
>>>       fourcc_decode = ""
>>>       for i in range(4):
>>>       int_value = fourcc_int ¿¿ 8 * i & 0xFF
>>>       fourcc_decode += chr(int_value)
>>>       return fourcc_decode
# It's the way to decode the fourcc code into the ASCII mode we all know. We have XDIV, AVC1, AVI etc fourcc codes.
```

11.3.1 How to display a video

```
>>>while video.isOpened():
>>>       ret,frame = video.read()
>>>       cv2.imshow('Main Video',frame)
>>>       gray_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
>>>       cv2.imshow('Gray Video',gray_frame)
>>>       if cv2.waitKey(30) & 0xFF == ord('q'):
# By 0xFF == ord('q') command, you can end and exit the video-displaying window by pressing the 'q' botton. It's just a complicated thing to explain why, so just learn it. You can not put cv2.waitKey() outside of if, otherwise by pressing 'q' botton, you'll only go to the next frame. You can not replace "ord('q')" with 'q' either because 'ord' is necessary.
>>>           break
# It'll break out of the while loop.
>>>video.release()
>>>cv2.destroyAllWindows()
# This is the way to show a video; Frame by frame. Because cv2.waitKey() is a method (function) so break doesn't work in it,You have to bring cv2.waitKey() inside "if" procedure.
```

11.3.2 How to save desired frames

You can save videos too. Just enter the codes inside while loop, before "if cv2.waitKey(30) & 0xFF == ord('q')" code. The piece of code is like this example:

```
# Press c on keyboard to save current frame
>>>if cv2.waitKey(20) & 0xFF == ord('c'):
>>>frame_name = "camera_frame_{}.png".format(frame_index)
>>>gray_frame_name = "grayscale_camera_frame_{}.png".format(frame_index)
>>>cv2.imwrite(frame_name, frame)
>>>cv2.imwrite(gray_frame_name, gray_frame)
>>>frame_index += 1
# This frame_index is only a growing number for naming the frame you want to save. Remember to put its value to 0 (zero) before the while loop.
```

11.3.3 How to backward-play a video

Now, a code to review the video in reverse:

```
    # We get the index of the last frame of the video file:
    >>>frame_index = capture.get(cv2.CAP_PROP_FRAME_COUNT) - 1
    >>>print("starting in frame: '{}'".format(frame_index))

    # Read until video is completed:
    >>>while capture.isOpened() and frame_index ¿= 0:

    # We set the current frame position:
    >>>         capture.set(cv2.CAP_PROP_POS_FRAMES, frame_index)

    # Capture frame-by-frame from the video file:
    >>>         ret, frame = capture.read()

        >>>         if ret is True:

    # Print current frame number per iteration
#print("CAP_PROP_POS_FRAMES : '{}'".format(capture.get(cv2.CAP_PROP_POS_FRAMES)))

    # Get the timestamp of the current frame in milliseconds
# print("CAP_PROP_POS_MSEC : '{}'".format(capture.get(cv2.CAP_PROP_POS_MSEC)))

    # Display the resulting frame:
    >>>             cv2.imshow('Original frame', frame)

    # Convert the frame to grayscale:
    >>>             gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
# Display the grayscale frame:
    >>>             cv2.imshow('Grayscale frame', gray_frame)
# Decrement the index to read next frame:
    >>>             frame_index = frame_index - 1
    >>>             print("next index to read: '{}'".format(frame_index))
# Press q on keyboard to exit the program:
    >>>             if cv2.waitKey(25) & 0xFF == ord('q'):
    >>>                 break
# Break the loop
    >>>             else:
    >>>                 break
    >>>video.release()
    >>>cv2.destroyAllWindows()
# Hope everything is clear :)
```

11.3.4 How to export a video file

You first have to specify the fourcc code you want the video to be coded into. The whole procedure is displayed to you below:

```
        >>>fourcc = cv2.VideoWriter_fourcc(*'XVID')
# *'XVID' returns 'X' 'V' 'I' 'D'. You have to enter the fourcc code like that.
    >>>out_gray = cv2.VideoWriter("PATH", fourcc, int(fps), (int(frame_width), int(frame_height)),
False)
# Last one is for grayscale (True = grayscale... False = Colorful; This doesn't work without cv2.cvtColor(frame,cv2.COLO

        >>>while capture.isOpened():
# Read the frame from the camera
    >>>         ret, frame = capture.read()
    >>>         if ret is True:
```

```
    # Convert the frame to grayscale
    >>>             gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
# Write the grayscale frame to the video
    >>>             out_gray.write(gray_frame)
# We show the frame (this is not necessary to write the video)
# But we show it until 'q' is pressed
    >>>             cv2.imshow('gray', gray_frame)
    >>>             if cv2.waitKey(1) & 0xFF == ord('q'):
    >>>                 break
    >>>             else:
    >>>                 break

    # Release everything:
    >>>capture.release()
    >>>out_gray.release()
    >>>cv2.destroyAllWindows()
```

12 Pillow Library

12.1 from PIL import Image

The code imports the library we want.

12.2 Image.open(image)

myimage = Image.open('file.jpg') ⇒Opens the image and stores it in myimage.

12.3 myimage.show()

Shows the image.

12.4 myimage.save('filename.anyformat'

You can store the loaded image in any formats you like; .png, .jpg, .jpeg, etc.

12.5 Resize(reduce the size only)

myimage.thumbnail(size_in_tuple) ⇒ Resizes myimage with the size in tuple, like (300,300), passed in the thumbnail function.

12.6 Rotate

myimage.rotate(90) ⇒ Rotates the image for you :)

12.7 Convert

myimage.convert(model='argument') ⇒ You have to pass the type of conversion you want as 'argument' to get it. A list of arguments is as below:

- **1**$(1 - bitpixels, blackandwhite, storedwithonepixelperbyte)$
- **L**$(8 - bitpixels, blackandwhite)$
- **P**$(8 - bitpixels, mappedtoanyothermodeusingacolorpalette)$
- **RGB**$(3x8 - bitpixels, truecolor)$
- **RGBA**$(4x8 - bitpixels, truecolorwithtransparencymask)$
- **CMYK**$(4x8 - bitpixels, colorseparation)$
- **YCbCr**$(3x8 - bitpixels, colorvideoformat)$

etc...

12.8 Filtering

from PIL import ImageFilter ⇒ With this library you can put so many filters on your picture.

Example:

```
>>>from PIL import Image ImageFilter
# Imports both "Image" and "ImageFilter" libraries.

>>>myimage = Image.open('image.jpg')

>>>new_image = myimage.filter(ImageFilter.GaussianBlur(#))
# Makes it blur and more blur as # increases.
```

13 Pickle Library

13.1 Pickle

Good for loading and a kind of algorithm to serialize and save a data. Let's say we have something like a dictionary to save. Let's go as below:

Example:

```
>>>import pickle# Imports the library

# To Save a Pickle File

>>>outpickle=pickle.open('Mydesiredpickle','wb')# Opens a pickle protocol in writing mode to save our pickle with desired name in CWD.

>>>pickle.dump(dictionary,outpickle)# Creates the pickle for us.

>>>outpickle.close()# Closes the outpickle. You have to use this command at the end to close the protocol.

# To Load a Pickle File

>>>picklein = open(filename,'rb')# Opens the pickled file in reading mode (rb stands for readig binary and good for numeric data)

>>>mydict = pickle.load(picklein)# Loads and decodes the pickle file into whatever it was.

>>>picklein.close()# To close the opened file.
```

Note that, if you want to compress the file alongside saving that, all you need to do is to import a library called "bz2" standing for Bzip2 and do as follows:

Example:

```
# To Save the Pickle for Compressed File

>>>import bz2# Imports the library

>>>zipped = bz2.BZ2File(filename , 'w')# Opens a zip protocoled file.

>>>pickle.dump(mydict,zipped)# Pickles mydict dictionary into the compressed file.

>>>zipped.close()# Closes the file.

# To Load the Pickle for Compressed File

>>>zipped = bz2.BZ2File(filename , 'rb')# Opens the zipped file.

>>>mydict=pickle.load(zipped)# Loads and decodes the pickle file into whatever it was.

>>>zipped.close()# Closes the file.
```

14 Pathlib Library

14.1 Pathlib

from pathlib import Path ⇒Very handy in data controling

14.2 Home & CWD

Path.cwd() & Path.home() ⇒ These two will print current working directory and home directory respectively.

Example:

```
>>>print(f'cwd is {Path.cwd()}

    cwd is C:\\Users\\Mahdi
>>>print(f'cwd is {Path.home()}# Prints home path.
```

14.3 Testing that if this is a real address

path.is_dir() ⇒ Does the job. Pay attention to the example:

Example:

```
>>>px = Path()

>>>px = px / 'GUBOLI'
# There is no folder GUBOLI. But px will have the address.

>>>px
# To obserce px

    WindowsPath('GUBOLI')

>>>px.is_dir()
# Shows that it's not a real address.

    False
```

14.4 Joining paths, getting all subfolders and return back to parent folder

Code below does it:

Example:

```
>>>path = Path()
# path becomes current working directory

>>>path = Path('E:/subfolder/subsubfolder')
# path becomes the required directory. But, in order to change CWD, you must use os library and
nothing else.

>>>directory = [x for x in path.iterdir() if x.is_dir()]
# All the subfolders inside the current working direcotry, if they are real addresses will be joined to
```

directory list.

```
>>>file = [x for x in path.iterdir() if x.is_file()]
#All the files inside the current working direcotry, if they are files not folders.

>>>path = path / 'newfolder'
# Now, our path is in the newfolder. Like we opened newfolder in windows. If you run that path.iterdir()
code, you will get the things inside the newfolder.

>>>path = path.parent
# Returns to the parent folder

>>>path = path.parent.parent.parent
# You can use whatever number of .parents you like to get back and further back
```

14.5 Making a new directory (New folder)

path.mkdir() ⇒ Makes it. Look at the example:

Example:

```
>>>mypath = Path()

>>>mypath = mypath / 'Mynewfolder'# This folder doesn't exist yet!

>>>mypath.mkdir()# The folder will be created in the path.
```

14.6 Copying files

Done by importing shutil library. Straight to examples:

Example:

```
>>>from shutil import copyfile

>>>path = Path()

>>>path = path / 'Mahdi' / 'Desktop'

>>>Source = path / '1.mp4'# Selects an mp4 file as source file

>>>Destination = path / '2.mp4'# Selects the destination

>>>copyfile(Source,Destination)# Copy is done :)
```

14.7 Creating an empty file

We can create empty files. MP3, MP4, txt and whatever format you pass to your path. Imagine path is in the Desktop folder. **Example:**

Just go to your desired path and make sure your CWD is the path you like by Path.cwd() command. I must recall that you can go to the desired path and select it as your CWD by os library like os.chdir(MyDesiredAddress). And if you type Path.cwd(), you will be in there.

You are in the place.

>>>Path('myfile.txt').touch()# Creates the file for you in your place.

You are not in the place, act like below

>>>address = os.getcwd()# saves CWD in address

>>>newaddress = os.path.joint(address,'Desktop')# If address was $C:\\Users\\UserName$, you can add Desktop to it as well.

>>>os.chdir(newaddress)# Now CWD is on Desktop

>>>Path('myfile.txt').touch()# Creates the file for you in your Desktop.

14.8 Rename a file

Imagine you are in the directory the file exists.

Example:

>>>path = Path('file1.txt')# Selects file1

>>>path.rename('file100.txt')# Renames it to file100

14.9 Remove a file or folder

Imagine you are in the directory the file exists.

Example:

>>>path = Path('file1.txt')# Selects file1

>>>path.unlink()# Removes the file

>>>path = Path('Foler')# Selects Folder

>>>path.rmdir()# Removes the Folder

14.10 Prints paths cleaner

Use path.as_posix()

14.11 path parts

Can see the root, drive and parts of the path

Example:

>>>path = Path.cwd()

>>>path.parts# Shows the path parts

('C : \\','Users','Mahdi','Desktop')

>>>path.drive# The drive in which the path is

'C:'

>>>path.root# It's most used in Linux systems but, for windows it only shows a simple thing:

'\\'

14.12 Globbing and finding specific types of files

Glob patterns specify sets of filenames with wildcard characters. For example, the *.txt represents all files with names ending in .txt. The * is a wildcard standing for any string of characters. Read comments because they provide you with important materials.

Example:

```
>>>for e in path.rglob('*.py'):
>>>        print(e)

# Prints all the python files in the CWD AND in the subfolders of the CWD.

>>>path.rglob('*.py') == path.glob('**/*.py')# The two commands are equivalent.

>>>for e in path.glob('*.py'):
>>>        print(e)

# Prints all the python files only in the CWD and not in subfolders.

>>>for e in path.glob('image_*'):
>>>        print(e)

# Prints all the files that their name starts with image_. Good for dataset loading in machine learning!!
```

14.13 Counting files by extension

Very Interesting:

Example:

```
>>>import collections# Imports the library collections. We can use it to count for us.

>>>files = [x.suffix for x in docs.iterdir() if path.is_file() and x.suffix]# If (and x.suffix) is not placed in code, the files that have the extension will be selected. But now, the extension alone are selected.

>>>data = collections.Counter(files)

>>>print(data)# See the result yourself :) .

# If you want to only have the counts and not to have Counter({ at the begining, do as follows:

>>>for key, val in data.items():
>>>        print(f'{key}: {val}')
```

14.14 Read & Write a text file

As below:

Example:

```
# To read a text file :

>>>path = Path('1.txt')# Selects 1.txt file in CWD.

>>>content = path.read_text()# Reads the whole text

>>>print(content)# The text will be shown :)

# To write a text file :

>>>path = Path('2.txt')# Selects 2.txt file which is not in CWD and not created yet.

>>>path.touch()# Creates the file.

>>>path.write_text('Hello Friends')# It's done!
```

14.15 To Loop over files in a directory

Do as below:

Example:

```
>>>pathlist = Path(directory_in_str).glob('**/*.asm')
# Lists whatever .asm file is in the directory (directory_in_str is means you have to insert directory as a string).

>>>for path in pathlist:

>>>        path_in_str = str(path)
# Since path is not string and is the file itself.

>>>        print(path_in_str)
```

14.16 Acknowledgements about pathlib tutorial

For further reading about pathlib, go to : http://zetcode.com/python/pathlib/
It really gave useful information. Thanks zetcode.

15 Tensorflow

There's a quick telling, and that's that the Tensorflow.org/overview tutorials are absolutely great and perfect. **The secret to read documentations fast is** to actually **Read once to the end and try to get them line by line, and if you didn't understand a thing, just look at it's title, to see if it's related to data loading or model building or what... And then pass it quickly, just remember the headline... Once you were done with the documentation, google that part you didn't understand.**

15.1 Importing Libraries

Below are a few libraries to import

Example:

```
    >>>import tensorflow as tf
# Main one

    >>>from tensorflow import keras
# Keras library

    >>>from tensorflow.keras import layers
# Importing Layers for easier handlings

    >>>import tensorflow_datasets as tfds
# Importing tfds useful library

    >>>import tensorflow_docs as tfdocs
# Documentation Library

    >>>import tensorflow_docs.plots as plotting
# tf plotter

    >>>from tensorflow.keras import regularizers
# Imports regularizers for avoiding overfitting
```

15.2 General Commands

A list of general commands:

Example:

```
    >>>a = tf.Variable([[2,3,4],[1,6,7]])
# Builds a tensorflow variable.

    >>>a.shape
# Gives its shape

    >>>tf.rank(a)
# Gives the rank (somehow depth of it) of the tensor.

    >>>b = tf.reshape(a , [-1,2])
# We specify the important dimension for us and we leave the other one as -1 for tensorflow to calculate what should be placed in there... E.g : A tensor is 6*8 dimensioned so it has got 48 elements. We want
```

a tensor with 2 columns only, we write tf.reshape(tensor , [-1,2]) , so automatically -1 would be 48/2 = 24

>>>tf.ones([shape])
Creates a tensor full of ones with the shape.

>>>tf.zeros([shape])
Creates a tensor full of zeros with the shape.

>>>randomer = tf.random.normal((size1,size2) , mean=0 , stddev = 1)
For random operations. There are different random distributions like normal and uniform

15.3 Tensorflow Datasets

There are various techniques for dataset handelings. We'll go through each one.

Example:

With these commands, you will be able to create a data object from your data.

>>>tmp = tf.data.Dataset.from_tensors(tensor)
The tensor would be saved as a single element of tf.data object. We use these objects to train model later.

>>>tmp = tf.data.Dataset.from_tensor_slices(tensor)
The tensor's elements would be saved as data objects. The previous command would take the whole tensor as one object but this command will take the elements of the tensor as objects

By this below command, you will be able to see the data in data object (in 'tmp'):

>>>for i in tmp:

>>> print(i.numpy())
# Prints the data.

You can see the data with this below command as well:

>>>it = iter(tmp)
Makes a Python iterable

>>>print(next(it).numpy())
Tada...

I have to mention that if you made a data object from a dictionary, you have to use below command:

>>>for i in tmp:

>>> print(list(i.values())[0].numpy())
# Prints the data's first key's elements.

You can inspect the data object type and shape by:

>>>tmp.element_spec

You can combine a few data objects with eachother:

>>>com = tf.data.Dataset.zip((dataset1,dataset2,dataset3,....))

Now, the most useful methods for this tf.data.Dataset class:

- The .repeat(#n) method that will repeat the dataset for n times.

```
>>> counter = 0

>>>for i in tmp.repeat(2):

>>>    print(i.numpy())

>>>counter += 1

>>> print(counter)
```

- The .shuffle(#n) method that will shuffle the data for n times.

```
>>>for i in tmp.shuffle(2):

>>>    print(i.numpy())
```

- The .batch(#n) method that will batch the data in n-element batches.

If we had a dataset in which its elements were with different lengths, we should use .padded_batch(batch_size , padded_shapes([size1,size2])) where batch_size is batch size, and padded_shapes, is a way to make all the data elements to have unique length specified by size1,size2 where size1 or size2 could be set to None if each is not important.

```
>>>for i in tmp.batch(5):

>>>    print(i.numpy())
```

- The .take(#n) method that will select the n first elements of the tmp.

```
>>>for i in tmp.take(3):

>>>    print(i.numpy())
# It will show the first 3 elements of the data
```

You can do a mixture of operations in one line:

```
>>>counter = 0

>>>for i in tmp.repeat(3).shuffle(4).batch(5).take(10):

>>>    counter += 1

>>>    print(counter , '\n' , i.numpy() , '\n')
```

This will first, repeat the dataset for 3 times, then shuffle all of it, then turn it to batches of 5 elements, then takes the first 10 and represents it.

You can build data object from Python Generators as well:

```
# This is the generator:

>>>def count(stop):

>>>    i = 0

>>>    while i¡stop:
```

>>> yield i

>>> i += 1

>>>ds_counter = tf.data.Dataset.from_generator(count, args=[10], output_types=tf.int32, output_shapes = (),)

This is the data object created from the generator. It's arguments are first, the generator itself (count is my generator now...); Second, is the number of 'next's to go (in my case, 10 means to run the generator for 10 times, which would be from 0 to 9). Then the output type and shape.

You can list the files in the folder by this command:

>>>list_ds = tf.data.Dataset.list_files(str(data_dir/'*'))
Makes a tensorflow list out of files.

>>>for f in list_ds.take(5):

>>> print(f.numpy())# Shows the first 5 of them

A very handy tool for opening CSV files (after pandas of course) is as below:

>>>titan = tf.data.experimental.make_csv_dataset('titanic.csv' , batch_size = 5 , label_name = 'Survived' , select_columns = ['Pclass' , 'Sex' , 'Age' , 'Frare'])
I think everything is clear...

This part is for images... Verrry handy

Read the documentations on the web for more info...

>>>x = tf.keras.preprocessing.image.ImageDataGenerator(IMAGE PROCESSING OPERATIONS)# Stores the settings related to the image processing things for you.

>>>gen = x.flow_from_directory(batch_size = batch_size , directory = mydatadir , shuffle = True , target_size = (img_height , img_width) , class_mode = 'binary')

class_mode is for labeling. Can be set to 'sparse' (int) , 'binary' (1D) , 'categorical' (2D). Categorical is for multi-classification, binary for two-classification and sparse,... I don't know much about it.. Google it :)

This .flow_from can be from directory or DataFrame. in each case, pass the required information.

After creating your own model, you can use:

>>>history = model.fit_generator(gen_data , steps_per_epoch = ,epochs = , validation_data = valid_gen_data, validation_steps =)

Parameters are clear mostly. gen_data is the train set turned into generator by ImageDataGenerator of tensorflow. validation_data is the same story for validation set. Other pararmeters are obvious... epochs and steps in each epoch.

You can use plotting options to visualize your data as well:

>>>acc = history.history['accuracy']

>>>val_acc = history.history['val_accuracy']

```
>>>loss=history.history['loss']

>>>val_loss=history.history['val_loss']

>>>epochs_range = range(epochs)

>>>plt.figure(figsize=(8, 8))
>>>plt.subplot(1, 2, 1)
>>>plt.plot(epochs_range, acc, label='Training Accuracy')
>>>plt.plot(epochs_range, val_acc, label='Validation Accuracy')
>>>plt.legend(loc='lower right')
>>>plt.title('Training and Validation Accuracy')

>>>plt.subplot(1, 2, 2)
>>>plt.plot(epochs_range, loss, label='Training Loss')
>>>plt.plot(epochs_range, val_loss, label='Validation Loss')
>>>plt.legend(loc='upper right')
>>>plt.title('Training and Validation Loss')
>>>plt.show()
```

15.4 Keras Library

15.4.1 Keras Datasets

Keras has a couple of datasets up in the cloud and it's possible for you to download and work on them. For getting them, use below commands:

Example:

```
>>>(xtrain , ytrain) , (xtest , ytest) = keras.datasets.mnist.load_data()
# Loads the train and test sets.
```

15.4.2 Keras URLS

It's possible to download a dataset from a URL:

Example:

```
>>>DATA = keras.utils.get_file('name' , 'URL address')
# Downloads the data file.
```

15.4.3 Keras Model

The Tensorflow Tutorial Page has a strong tutorial on tf.keras... Very Complete; We, would only highlight what is where on this webpage.

• On the left-side, there is a pane. From Tensorflow 2 part, the only handy one is Effective Tensorflow that introduces some how some techniques for transfer learning with 'head' and 'trunk' things... Take a look if it's good for you.

The Keras part compeletely is the most useful thing on earth.... :)
It contains :

• Keras Review: tf.keras.Sequential structure, keras optimizers, loss functions, suitable modes for classification and regression, saving, almost everything...

Enough is enough... :)

16 The End

16.1 Acknowledgements

Dear readers. Thank you for reading this article. I will not continue this documentation and the rest of the things, related to each library we discussed, will be upon yourselves.

I wanted to gather a meaningful and handy pack to show you some basics and little bit of everything from most useful libraries in python. Although most of them are not listed like Tensorflow, Pytorch, etc. But, these would cover a good range of python materials.

16.2 Final Words

To sum it up, I should encourage you all to get used to read documentations. There are a lot of persons out there gathering too much informations in details to provide you with. This is one of the main reasons I decided to finish this article. It's too much time consuming to work like this. Just get used to work with documentations and grasp the ones that fits you.

With Best Regards
Mahdi Dor Emami