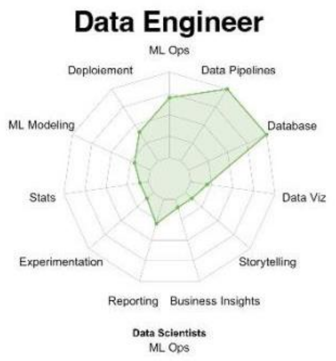
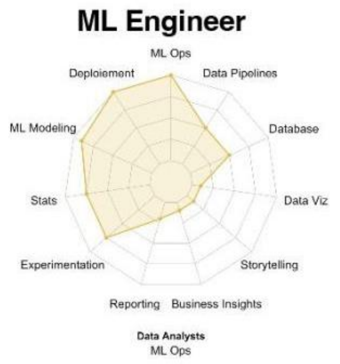
Data Science

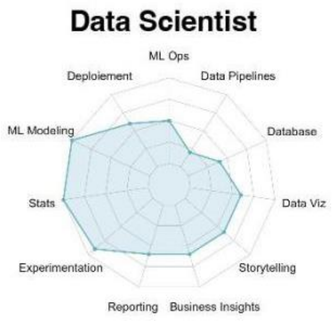
0

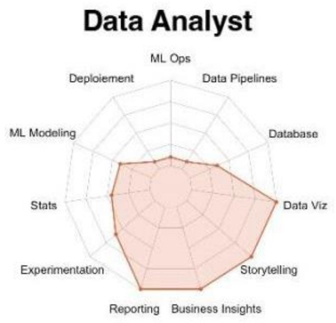
Introduction

Data Professionals







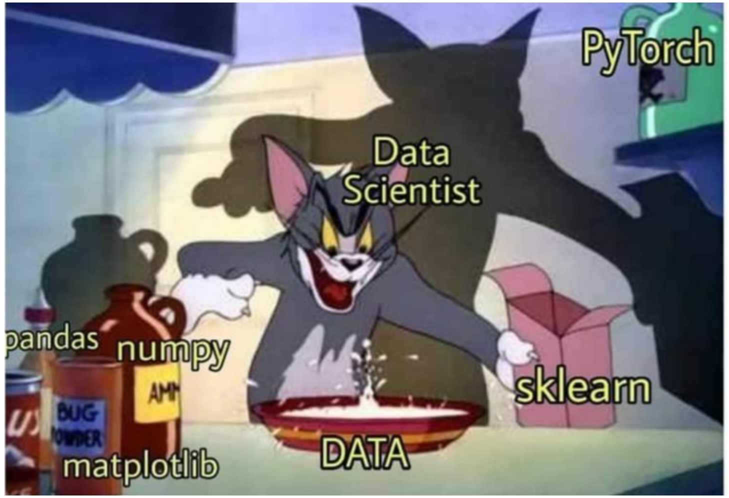


Data Science

A Data Scientist helps companies with data-driven decisions, to make their business better.

Data Science is a combination of multiple disciplines that uses statistics, data analysis, and machine learning to analyze data and to extract knowledge and insights from it.

Data Science is about data gathering, finding patterns in data, data analysis, make future predictions and decision-making.



1

Numpy

Introduction

NumPy stands for Numerical Python.

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy aims to provide an array object that is up to 50x faster than lists. Arrays are very frequently used in data science. The array object in NumPy is called ndarray.

Numpy documentation: https://numpy.org/doc/

Import

```
import numpy as np
```

Arrays

Create

```
arr0 = np.array(1)
arr1 = np.array([1, 2, 3])
arr2 = np.array([[1, 2, 3], [2, 3, 4]])
arr3 = np.array([[[1, 2, 3], [2, 3, 4]], [[3, 4, 5], [4, 5, 6]]])
print(arr2.ndim)
```

Slice

```
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(arr[1, 5], arr[1, 1:4], arr[0:2, 1:4])
```

Data Types

Check

```
arr = np.array([1, 2, 3, 4])
print(arr.dtype)
```

Define

```
arr = np.array([1, 2, 3, 4], dtype = 'S')
print(arr.dtype)
```

Convert

```
arr = np.array([1, 2, 3, 4])
newArr = arr.astype('S')
print(arr.dtype, newArr.dtype)
```

S	String
U	Unicode string
i	Integer
u	Unsigned integer
f	Float
c	Complex float
m	Timedelta
M	Datetime
O	Object
b	Boolean
V	Void

Copy and View

Copy is a new array, and view is just a view of the original array.

```
arr = np.array([1, 2, 3, 4, 5])
viewArr = arr.view()
copyArr = arr.copy()
arr[0] = 10
print(viewArr, copyArr)
print(viewArr.base, copyArr.base)
```

Shape

Shape

```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(arr.shape)
```

Reshape

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
print(arr.reshape(2, 4))
print(arr.reshape(2, 2, -1))
```

Flatten

```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(arr.reshape(-1))
```

Loops

Iterating arrays

```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
for x in arr:
    print(x)
for x in np.nditer(arr):
    print(x)
for x in np.nditer(arr[:, ::2]):
    print(x)
for x in np.ndenumerate(arr):
    print(x)
```

Joins and Splits

Joining arrays

```
arr1, arr2 = np.array([[1, 2], [3, 4]]), np.array([[5, 6], [7, 8]])
print(np.concatenate((arr1, arr2)))
print(np.concatenate((arr1, arr2), axis = 1))
print(np.stack((arr1, arr2), axis = 1))
```

Splitting arrays

```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]])
print(np.array_split(arr, 2))
print(np.array_split(arr, 3))
print(np.array_split(arr, 6))
print(np.array_split(arr, 2, axis = 1))
```

Sort

Sorting arrays

```
arr = np.array([[1, 3], [2, 4], [6, 4], [4, 0]])
print(np.sort(arr))
```

Search Sorted

```
arr = np.array([1, 3, 2, 4, 6, 4, 4, 0])
print(np.searchsorted(arr, 3))
print(np.searchsorted(arr, 5))
```

Search and Filter

Searching arrays

```
arr = np.array([[1, 3], [2, 4], [6, 4], [4, 0]])
print(np.where(arr == 4))
print(np.where(arr % 2 == 0))
```

Filtering arrays

```
arr = np.array([1, 3, 2, 4, 6, 4, 4, 0])
print(arr[arr % 2 == 0])
```

Random Numbers

Generate Random Number

```
print(np.random.rand())
print(np.random.rand(2, 3))
print(np.random.randint(50, 100))
print(np.random.randint(50, 100, size = (2, 3)))
```

Generate Random Number From Array

```
print(np.random.choice([3, 5, 7, 9]))
print(np.random.choice([3, 5, 7, 9], size = (2, 3)))
print(np.random.choice([3, 5, 7, 9], p = [0.1, 0.3, 0.6, 0.0]))
```

Universal Functions

Converting iterative statements into a vector based operation is called vectorization.

It is faster as modern CPUs are optimized for such operations.

ufuncs are used to implement vectorization in NumPy.

```
x, y = [1, 2, 3, 4], [5, 6, 7, 8]
print(np.add(x, y))
```

Some useful ufuncs

```
add()  subtract()  multiply()  divide()  power()  mod()  abs()
sum()  cumsum()  prod()  cumprod()  diff()
```

2

Pandas

Introduction

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

It allows us to analyze big data and make conclusions based on statistical theories.

It can clean messy data sets, and make them readable and relevant.

Pandas documentation: http://pandas.pydata.org/pandas-docs/stable/

Import

```
import pandas as pd
```

Series

A column in a table

```
calories = [420, 380, 390]
s = pd.Series(calories)
print(s)
print(s[0])
calories = {'day1': 420, 'day2': 380, 'day3': 390}
s = pd.Series(calories)
print(s['day2'])
print(s.loc['day2'])
```

DataFrames

A table with rows and columns

```
data = {'calories': [420, 380, 390], 'duration': [50, 40, 45]}
df = pd.DataFrame(data)
print(df['calories'])
print(df[['calories']])
print(df.loc[0])
print(df.loc[[0, 2]])
df = pd.DataFrame(data, index = ['day1', 'day2', 'day3'])
print(df.loc['day2'])
print(df.iloc[1])
```

Read and Write

Read files

```
df = pd.read_csv('data.csv')
df = pd.read_excel('data.xlsx')
df = pd.read_json('data.json')
```

Write files

```
df.to_csv('data.csv')
df.to_excel('data.xlsx')
df.to_json('data.json')
```

Analyze

```
df = pd.DataFrame({'calories': [420, 380, 390], 'duration': [50, 50, 45]})
```

View

```
print(df.head())
print(df.tail())
print(df.index)
print(df.columns)
print(df['duration'].value_counts())
```

Information

```
print(df.info())
print(df.describe())
```

Sort and Filter

```
data = {'calories': [420, 380, 390], 'duration': [50, 40, -45]}
df = pd.DataFrame(data, index = ['day2', 'day3', 'day1'])
```

Sort

```
print(df.sort_index())
print(df.sort_values(by = 'calories'))
```

Filter

```
df[df['duration'] > 0]
df[df < 0] = -df
```

Operations

```
df = pd.DataFrame({'calories': [420, 380, 390], 'duration': [50, 40, 45]})
```

Stats

```
print(df.sum())
print(df.sum(axis = 1))
print(df.mean())
print(df.max())
print(df.min())
```

Apply

```
print(df.apply(np.cumsum))
print(df.apply(lambda x: x.max() - x.min(), axis = 1))
print(df['calories'].apply(lambda x: x * 2))
```

Data Cleaning

The data set contains some empty cells: row 18, 22, and 28.

`isna()` `dropna()` `fillna()`

The data set contains wrong format: row 26.

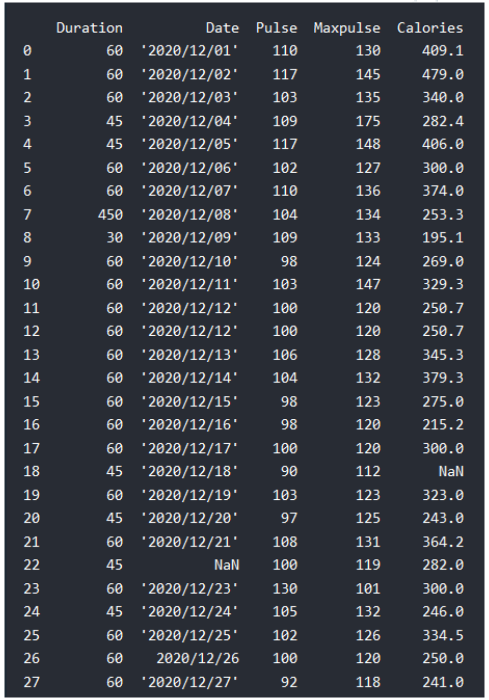
`dtype` `astype()` `to_numeric()` `to_datetime()`

The data set contains wrong data: row 7.

`loc[]` `iloc[]`

The data set contains duplicates: row 11 and 12.

`duplicated()` `drop_duplicates()`



```
    Duration          Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
7        450  '2020/12/08'    104       134     253.3
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
10        60  '2020/12/11'    103       147     329.3
11        60  '2020/12/12'    100       120     250.7
12        60  '2020/12/12'    100       120     250.7
13        60  '2020/12/13'    106       128     345.3
14        60  '2020/12/14'    104       132     379.3
15        60  '2020/12/15'     98       123     275.0
16        60  '2020/12/16'     98       120     215.2
17        60  '2020/12/17'    100       120     300.0
18        45  '2020/12/18'     90       112       NaN
19        60  '2020/12/19'    103       123     323.0
20        45  '2020/12/20'     97       125     243.0
21        60  '2020/12/21'    108       131     364.2
22        45           NaN    100       119     282.0
23        60  '2020/12/23'    130       101     300.0
24        45  '2020/12/24'    105       132     246.0
25        60  '2020/12/25'    102       126     334.5
26        60    2020/12/26    100       120     250.0
27        60  '2020/12/27'     92       118     241.0
```

3

Matplotlib

Introduction

Matplotlib is a graph plotting library in python that serves as a visualization utility.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias.

Matplotlib documentation: https://matplotlib.org/stable/

Import

```
import matplotlib as mpl

import matplotlib.pyplot as plt
```

Plotting

```
xpoints = np.array([0, 10, 8])
ypoints = np.array([0, 15, 4])
```

Plotting x and y points

```
plt.plot(xpoints, ypoints)
plt.show()
```

Plotting without line

```
plt.plot(xpoints, ypoints, 'o')
plt.show()
```

Markers and Line

fmt parameter

```
xpoints = np.array([0, 10, 8])
ypoints = np.array([0, 15, 4])
marker, line, color = '*', '--', 'g'
plt.plot(xpoints, ypoints, f'{marker}{line}{color}')
plt.show
```

fmt reference

```
Marker:  o  *  .  ,  x  X  +  P  s  D  d  p  h  H  v  ^  <  >  1  2  3  4  |  _
Line:  -  :  --  -.
Color:  r  g  b  c  m  y  k  w
```

Markers and Line

Multiple Lines

```
y1 = np.array([3, 8, 1, 10])
y2 = np.array([6, 2, 7, 11])
plt.plot(y1, marker = '', linestyle = ':', color = '#4CAF50')
plt.plot(y2, 'o-.r', markersize = 9, linewidth = 3)
plt.show()
```

Text

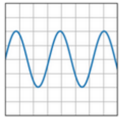
Title and labels

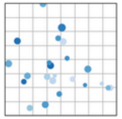
```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 255, 259, 265, 270, 290, 305, 306, 325, 330])
titleFont = {'family': 'Trebuchet MS', 'color': '#0091EA', 'size': 14}
labelFont = {'family': 'Consolas', 'color': 'black', 'size': 11}
plt.plot(x, y)
plt.title('Sports Watch Data', titleFont)
plt.xlabel('Average Pulse', labelFont)
plt.ylabel('Calorie Burnage', labelFont)
plt.show()
```

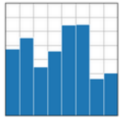
Plot Types

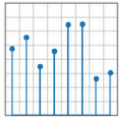
Documentation

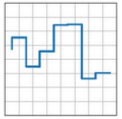
https://matplotlib.org/stable/plot_types/index.html

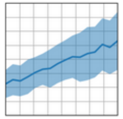

plot(x, y)

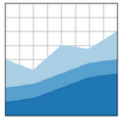

scatter(x, y)

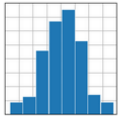

bar(x, height)

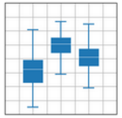

stem(x, y)

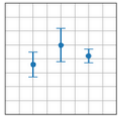

step(x, y)

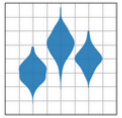

fill_between(x, y1, y2)

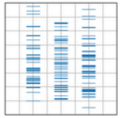

stackplot(x, y)

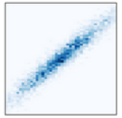

hist(x)

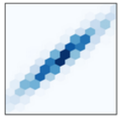

boxplot(X)

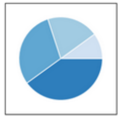

errorbar(x, y, yerr, xerr)


violinplot(D)


eventplot(D)


hist2d(x, y)


hexbin(x, y, C)


pie(x)

Thanks!

You can find me at:

github.com/AleeRezaa

t.me/Alee_Rezaa

alee_rezaa@outlook.com