FUNDAMENTAL DATA ANALYSIS WITH PYTHON

By Muhamad Irfan Fadhullah

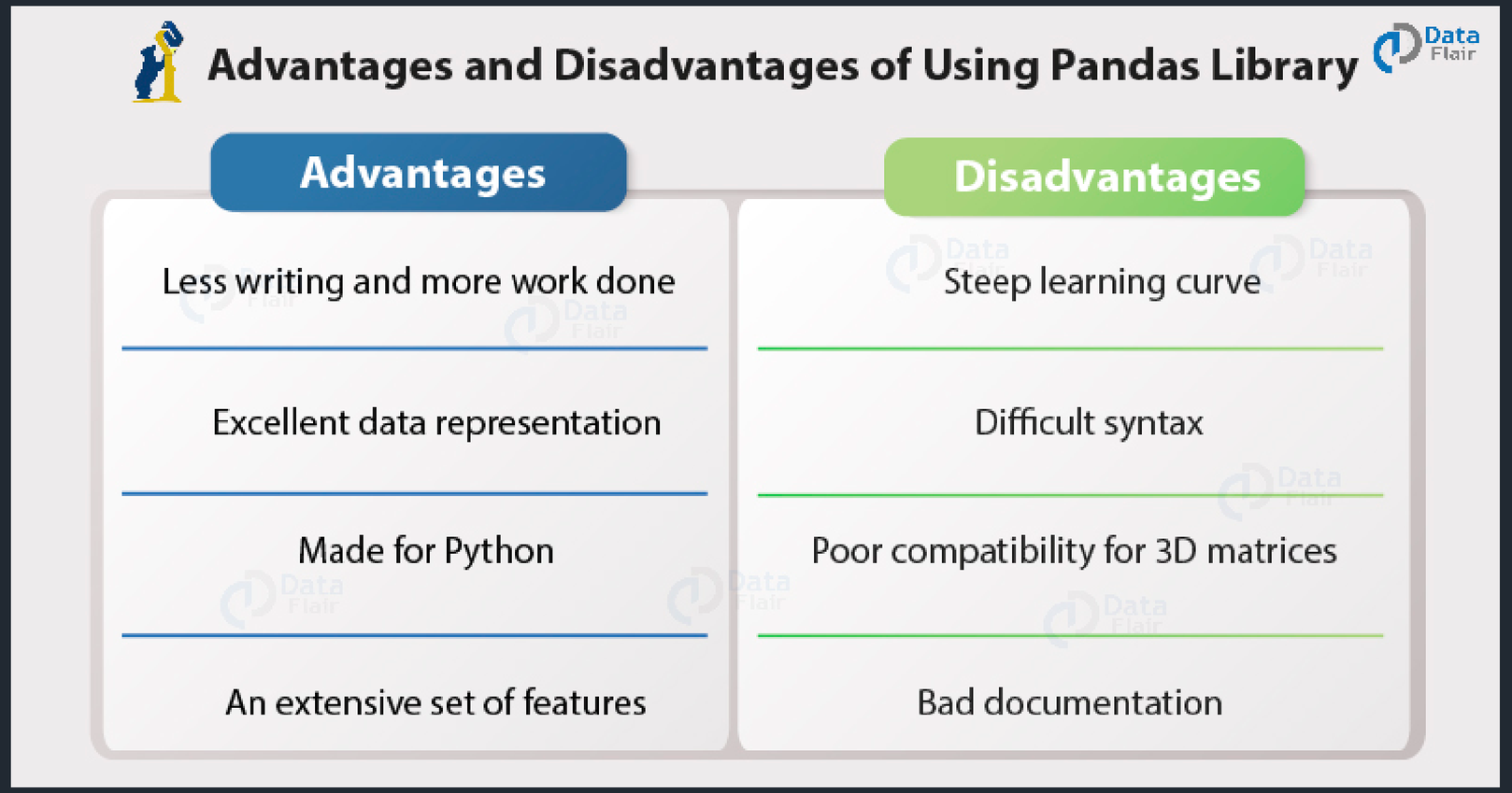
Outline

- Pandas Foundation
- Manipulating Data Using Pandas
- Fandas Functionality
- Data Cleansing
- Data Wrangling
- Data Visualization
- Exploratory Data Analysis

Pandas Foundation

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool built on top of the Python programming language.

Why Pandas?



Manipulating Data Using Pandas

Examples:

SELECTION

```
year = datac[['adr','reservation_status_date']]
```

ADDITION

```
df['gender'] = ['male','male','female']
df
```

RENAME

```
cek=high.groupby([high['reservation_status_date'].dt.year.rename('Year'),
                  high['reservation_status_date'].dt.month.rename('Month')]).agg('count')
```

Manipulating Data Using Pandas

DELETION

```
datac=data.drop(columns=['company'],axis=1)
```

SORTING

```
df.sort_index(ascending =False)
```

PANDAS FUNCTIONALITY

BASIC FUNCTION

DATA ENGINEERING

SIMPLE PLOTTING

BASIC FUNCTION

Basic Information

`>>> df.shape`	(rows,columns)
`>>> df.index`	Describe index
`>>> df.columns`	Describe DataFrame columns
`>>> df.info()`	Info on DataFrame
`>>> df.count()`	Number of non-NA values

Summary

`>>> df.sum()`	Sum of values
`>>> df.cumsum()`	Cummulative sum of values
`>>> df.min()/df.max()`	Minimum/maximum values
`>>> df.idxmin()/df.idxmax()`	Minimum/Maximum index value
`>>> df.describe()`	Summary statistics
`>>> df.mean()`	Mean of values
`>>> df.median()`	Median of values

DATA ENGINEERING

CHECK MISSING VALUES

df.notnull()

DROP MISSING VALUES

df.dropna()

FILLING MISSING VALUES — DIRECT REPLACE

df.fillna(ScalarValue)

COMPUTING UNIQUE VALUES

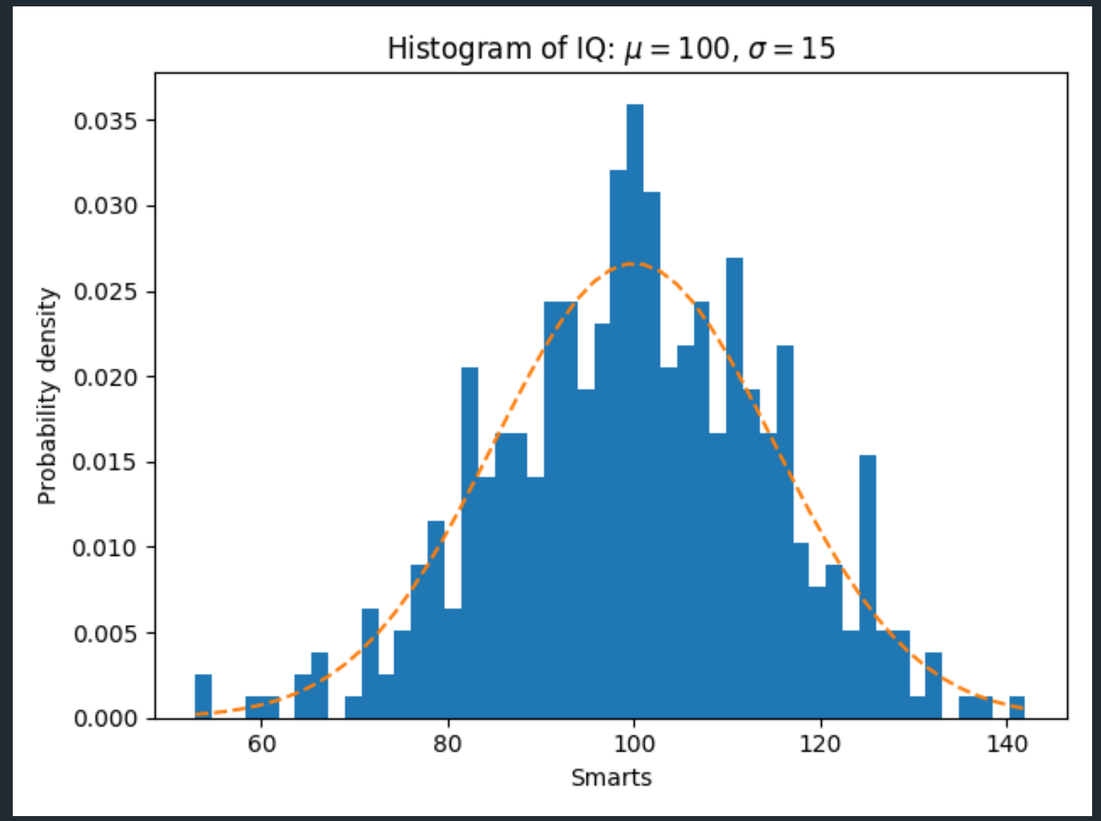
df.unique()

df.value_counts()

SIMPLE PLOTTING

Pandas Dataframe offers a range of graphical plotting options.We can plot, box plot, area, scatter plots, stacked charts, barcharts, histograms, etc.

- df.plot.scatter() #plots a scatter chart
- df.plot.line() # plots a line chart
- df.boxplot() # plots a box plot



iykra

CHEAT SHEET PANDAS FROM DATACAMP

Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com

Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.

Use the following import convention:

```
>>> import pandas as pd
```

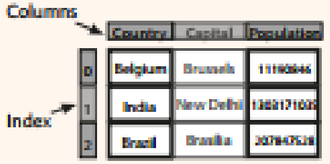
Pandas Data Structures

Series

A one-dimensional labeled array capable of holding any data type


```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame



A two-dimensional labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b']
 -5

>>> df[1:]
   Country     Capital  Population
1   India   New Delhi  1303171035
2  Brazil    Brasilia   207847528
```

Get one element

Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[[0],[0]]
'Belgium'
>>> df.iat[[0],[0]]
'Belgium'
```

Select single value by row & column

By Label

```
>>> df.loc[[0], ['Country']]
'Belgium'
>>> df.at[[0], ['Country']]
'Belgium'
```

Select single value by row & column labels

By Label/Position

```
>>> df.ix[2]
Country       Brazil
Capital     Brasilia
Population  207847528
```

Select single row of subset of rows

```
>>> df.ix[:,'Capital']
0      Brussels
1     New Delhi
2      Brasilia
```

Select a single column of subset of columns

```
>>> df.ix[1,'Capital']
'New Delhi'
```

Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
>>> s[(s < -1) | (s > 2)]
>>> df[df['Population']>1200000000]
```

Series `s` where value is not >1
`s` where value is <-1 or >2
Use filter to adjust DataFrame

Setting

```
>>> s['a'] = 6
```

Set index `a` of Series `s` to 6

Dropping

```
>>> s.drop(['a', 'c'])
>>> df.drop('Country', axis=1)
```

Drop values from rows (axis=0)
Drop values from columns(axis=1)

Sort & Rank

```
>>> df.sort_index()
>>> df.sort_values(by='Country')
>>> df.rank()
```

Sort by labels along an axis
Sort by the values along an axis
Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
>>> df.index
>>> df.columns
>>> df.info()
>>> df.count()
```

(rows,columns)
Describe index
Describe DataFrame columns
Info on DataFrame
Number of non-NA values

Summary

```
>>> df.sum()
>>> df.cumsum()
>>> df.min()/df.max()
>>> df.idxmin()/df.idxmax()
>>> df.describe()
>>> df.mean()
>>> df.median()
```

Sum of values
Cummulative sum of values
Minimum/maximum values
Minimum/Maximum index value
Summary statistics
Mean of values
Median of values

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
>>> df.applymap(f)
```

Apply function
Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
a    10.0
b     NaN
c     5.0
d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
a    10.0
b    -5.0
c     5.0
d     7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> df.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

`read_sql()` is a convenience wrapper around `read_sql_table()` and `read_sql_query()`

```
>>> df.to_sql('myDf', engine)
```

DataCamp
Learn Python for Data Science Interactively

DATA CLEANSING

Data cleaning or cleansing
is the process of detecting and correcting (or removing) corrupt or inaccurate
records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

DATA WRANGLING

Data wrangling
is one of the most important components in the data science workflow. It
involves the processing of data in various formats like concatenating, grouping, merging, etc. for the purpose of getting them used with another set
of data or for analysing.

DATA CLEANSING

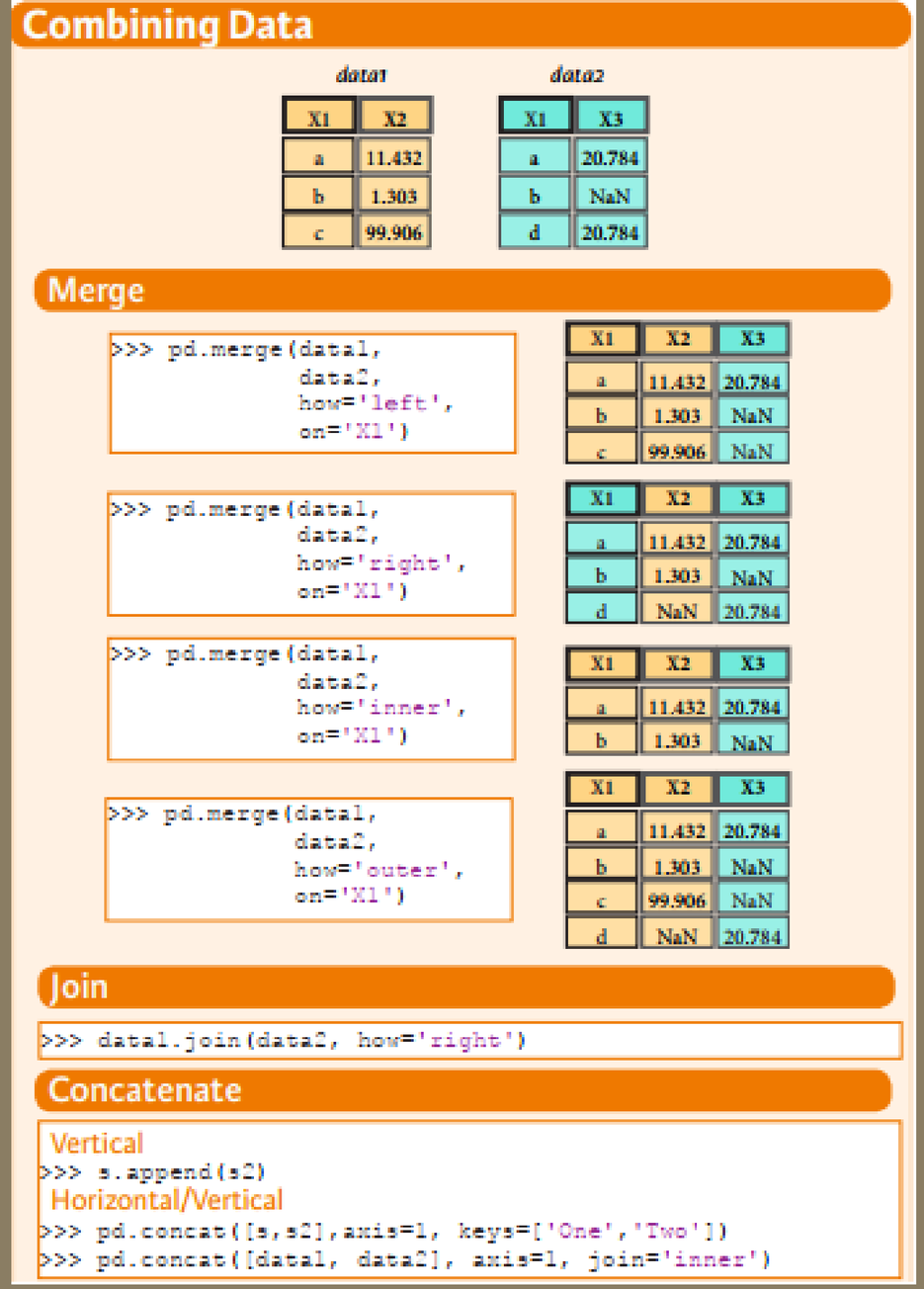
Common Problems in the Data Cleansing:

1. Duplicate Dataset
2. Missing Data
3. Outliers
4. Data Type

METHODS IN COMBINING DATA:

1. JOIN
2. MERGE
3. CONCAT

SOURCE : DATACAMP



DATA WRANGLING

Data Visualization With Python

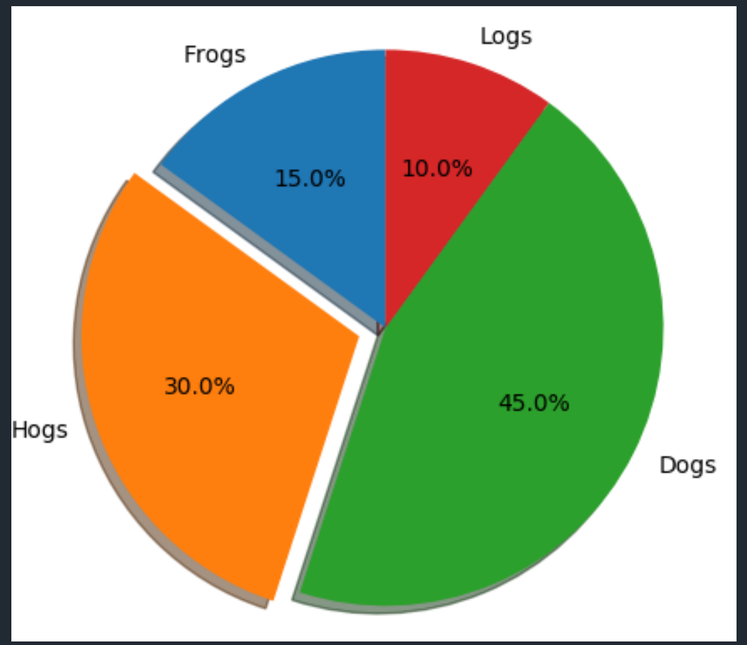
Introduction

How to create right Data Visualization

- Understand the context!
- Choose the appropriate plot type. If there are various options, we can try to compare them,and choose the one that fits our model the best.
- When we choose your type of plot, one of the most important things is to label your axis.
- Add a title to make our plot more informative.
- Add labels for different categories when needed.
- Optionally we can add a text or an arrow at interesting data points.
- In some cases we can use some sizes and colors of the data to make the plot more informative.

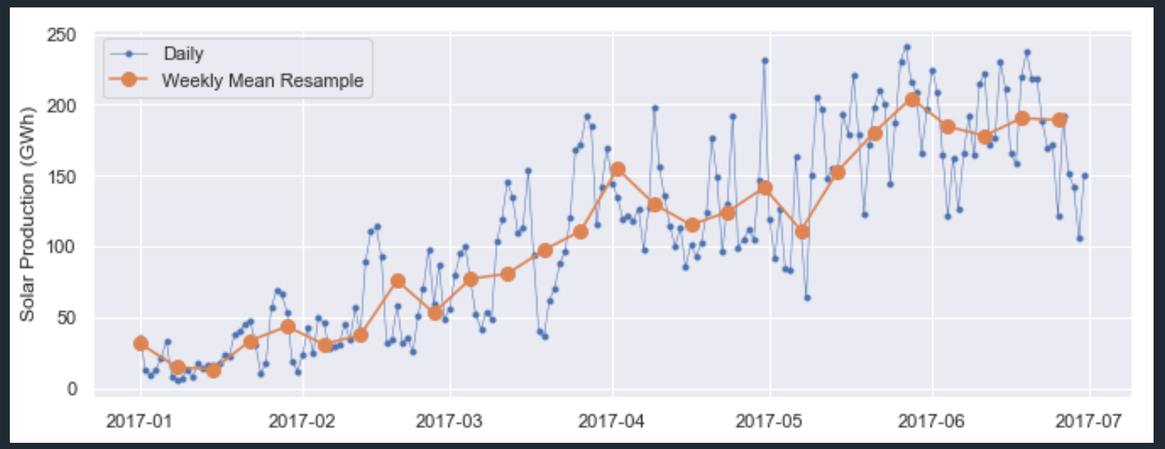
Pie Chart

Pie chartshould be used seldom usedas It is difficult to comparesections of the chart. Barplot is used instead ascomparing sections is easy.eg: Market share in Films



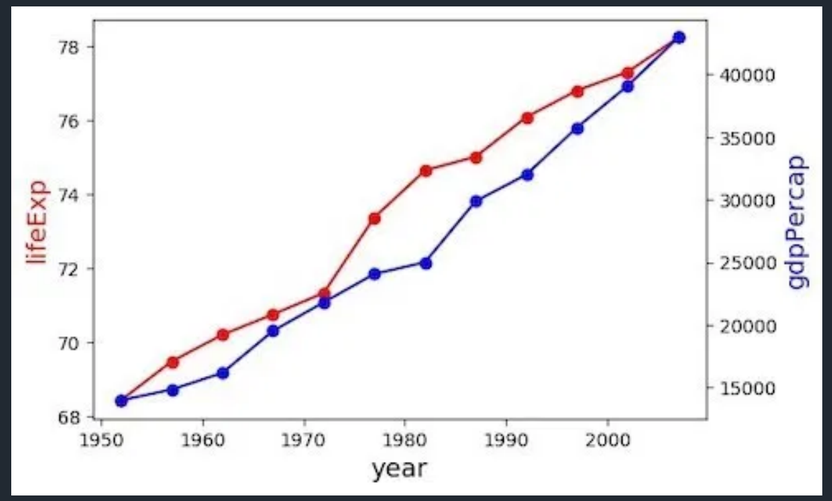
LINE CHART

TIME SERIES



It should when werequire 2 plots or grouped data inthe same direction.
e.g : Population and GDP data in the same x-axis

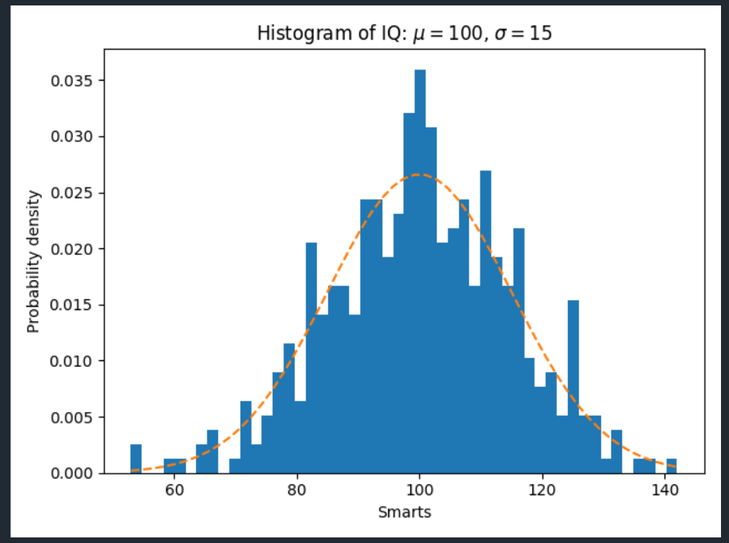
TWIN AXIS



Time Seriesshould be used when singleor multiple variables are tobe plotted over time.Eg: Stock Market Analysis ofCompanies, WeatherForecasting.

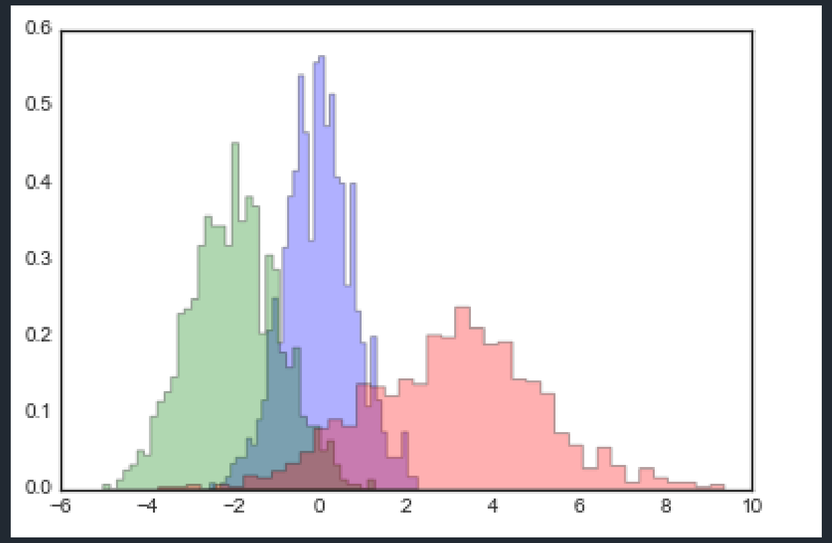
HISTOGRAM

SIMPLE



We should use histogram when weneed the count of thevariable in a plot.eg: Number of particulargames sold in a store.

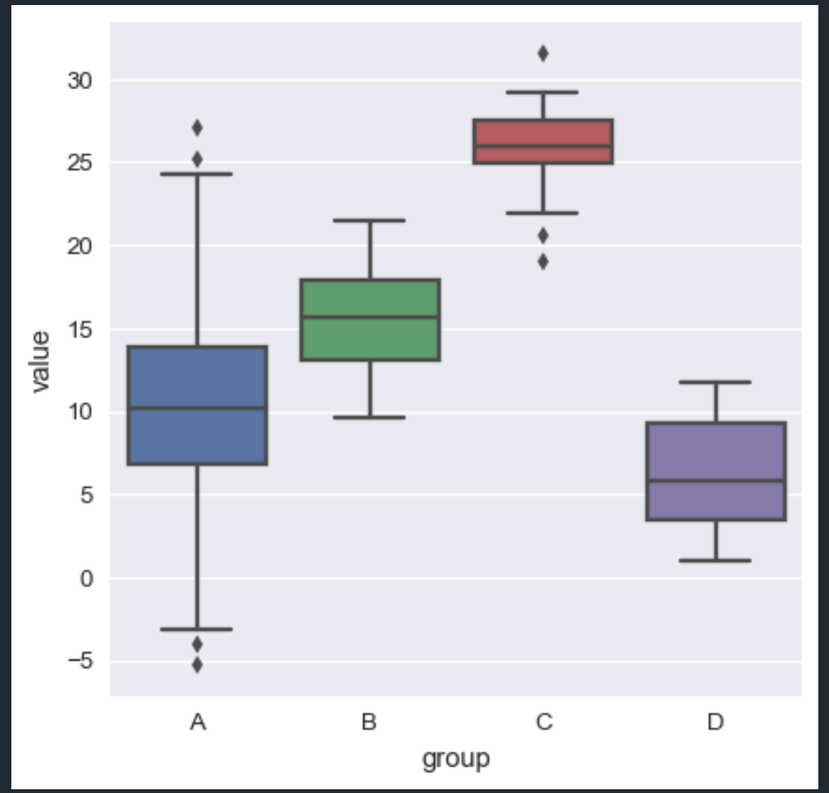
MULTIPLE HISTOGRAM



When weneed to understand thedistributions between 2entity variables. We can see that Grand Canyonhas comparably more visitorsthan Bryce Canyon

BOX PLOT

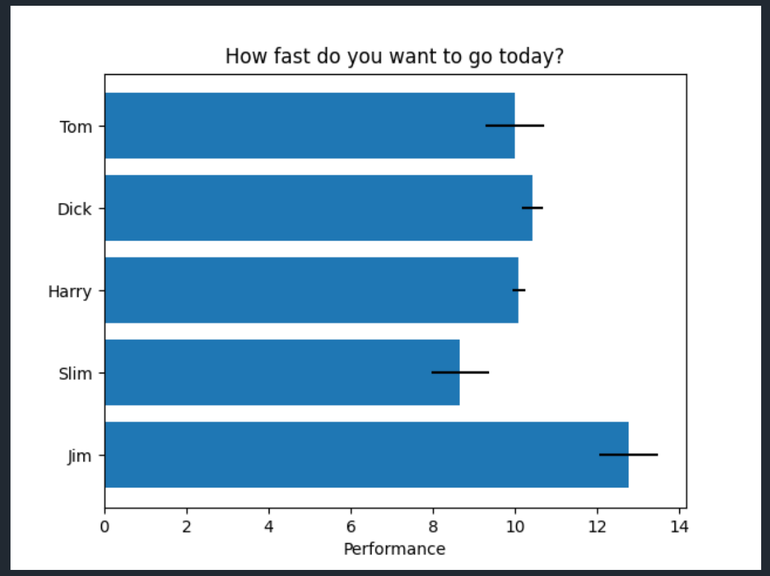
It should be usedwhen we require to use theoverall statistical informationon the distribution of the data.It can be used to detectoutliers in the data.Eg: Credit Score of Customer.We can get the max, min andmuch more information aboutthe mark



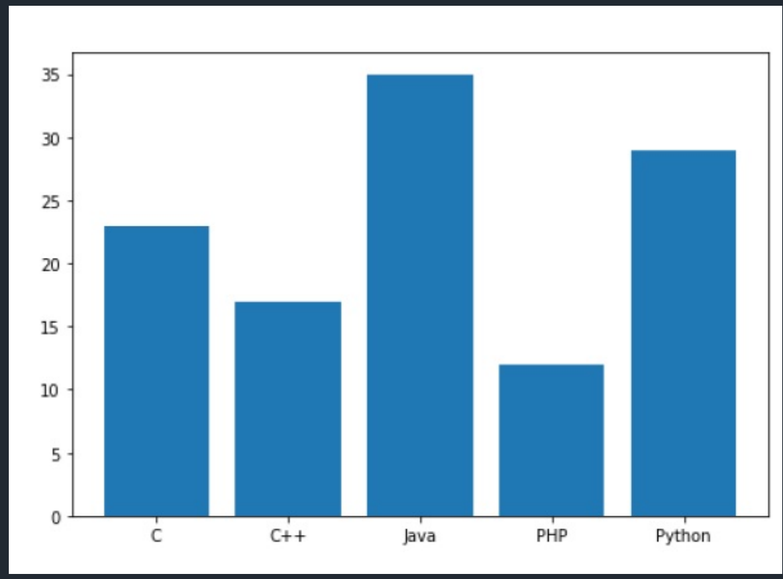
BAR PLOT

It is used when to compare between several groups.
Eg: Student marks in an exam

HORIZONTAL

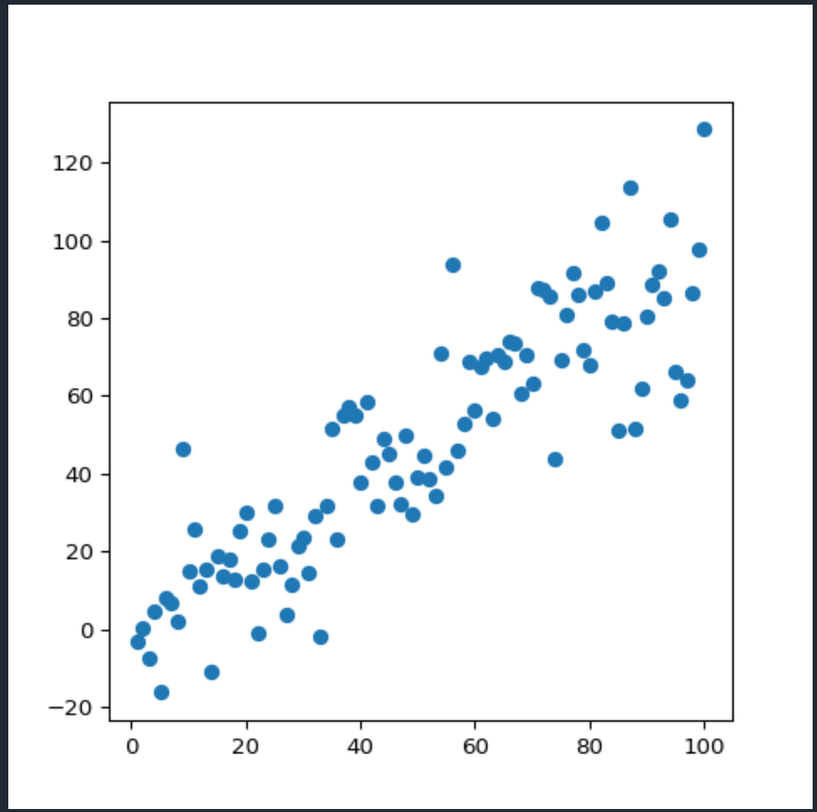


VERTIKAL

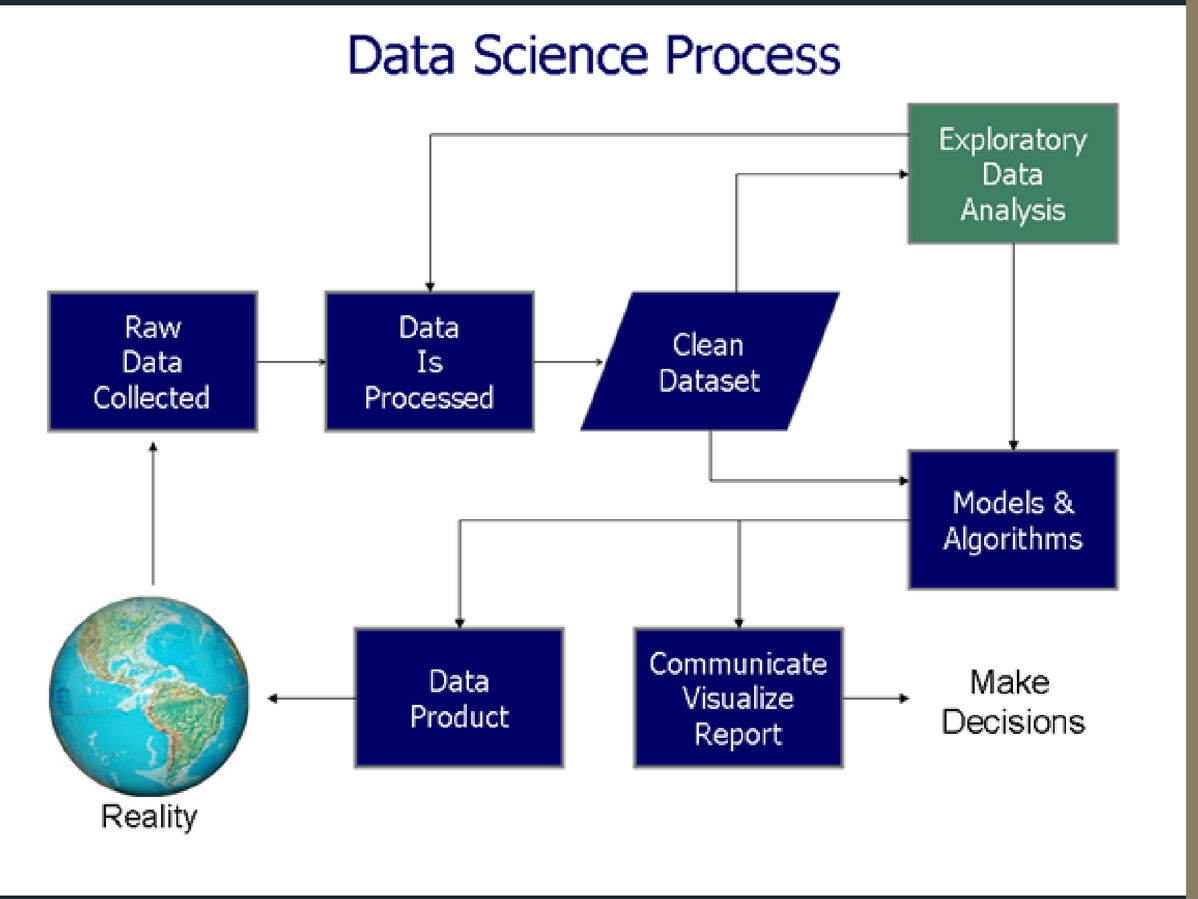


SCATTER PLOT

It is used inMachine learning concepts likeregression, where x and y arecontinuous variables



Exploratory Data Analysis



Exploratory Data Analysis is an approach to oanalyzing data sets to summarize their main characteristics, often with visual methods

OBJECTIVES

QUICKLY DESCRIBE A DATASET	CLEAN CORRUPTED DATA	VISUALIZE DATA DISTRIBUTIONS	CALCULATE AND VISUALIZE CORRELATION
number of rows/columns, missing data, datatypes, preview of the data	handle missing data, invalid data types, incorrectvalues	bar chart, histogram, box plots	Calculate and visualize correlation (relationships) between variables(heatmap)

REFERENCES

- https://iykra.com
- datacamp.com
- towardsdatascience.com
- www.analyticsvidhya.com
- pandas.pydata.org
- stackoverflow.com
- https://data-flair.training/blogs

Thank You

Contact Information

EMAIL ADDRESS

irfanfadhullah@gmail.com

For any questions or concerns