



Python and Basic Statistics

Contents

- What is Python & History
- Installing Python & Python Environment
- Basic commands in Python
- Data importing
- Basic details of the data frames
- Basic statistics
- Measures of dispersion
- Data exploration
- Data cleaning



Introduction to Python & History

What is python

- It's a language
- Human-readable syntax and well Documented
- Open Source (Free)
- Powerful scripting language with simple Syntax
- www.python.org
- Used by many data scientists and developers



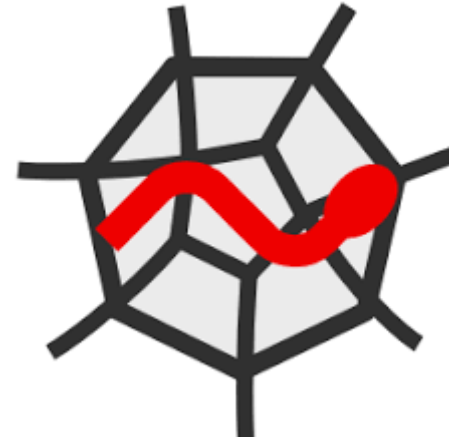
Installing Python & Python IDEs

Writing and executing python programs

- Python has many options to write and execute a program
- You can use Text Editors or Command line interfaces or Notebook or an IDE
- We will use Spyder IDE in our course
- Anaconda distribution has all the required software's inbuilt. We just need to download and install it.





Installing Python, Anaconda

- Download and install Anaconda3
- It automatically installs
 - Ipython
 - Jupyter notebook
 - Spyder IDE
 - Google Colab





Python on Cloud – Google Colab




← → ↻ colab.research.google.com/notebooks/intro.ipynb# 🔍 ☆ 🌙 📄 🟢 ⚙️ 👤 ⋮

 **Welcome To Colaboratory**  Share  

File Edit View Insert Runtime Tools Help

 **Table of contents** 

- 🔍 Getting started
- Data science
- <> Machine learning
- 📁 More Resources
- Machine Learning Examples
- + Section

+ Code + Text |  Copy to Drive Connect ▾ |  Editing 

What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

▼ **Getting started**



Basic Commands in Python

Before you code

- Python is case sensitive
- Be careful while using the Variable names and Function names
 - `Sales_data` is not same as `sales_data`
 - `Print()` is not same as `print()`

Basic Commands

```
571+95
```

```
19*17
```

```
print(57+39)
```

```
print(19*17)
```

```
print("Statinfer")
```

```
# use hash(#) for comments
```

```
#Division example
```

```
34/56
```



Important Packages

Packages

- To be a good data scientist on python, one needs to be very comfortable with the following packages
 - numpy
 - pandas
 - scikit-Learn
 - matplotlib
 - nltk

Important Packages- NumPy

- NumPy is for fast operations on vectors and matrices, including mathematical, logical, shape manipulation, sorting, selecting.
- It is the foundation on which all higher level tools for scientific Python packages are built

```
import numpy as np
```

```
income = np.array([9000, 8500, 9800, 12000, 7900, 6700, 10000])
```

```
expenses=income*0.6525
```

```
print(expenses)
```

```
savings=income-expenses
```

```
print(savings)
```

Important Packages- Pandas

- Data frames and data handling
- Pandas has Data structures and operations for manipulating numerical tables and time series.

```
import pandas as pd
#bank= pd.read_csv('D:\\Datasets\\Bank Tele Marketing\\bank_market.csv')

bank= pd.read_csv('https://raw.githubusercontent.com/venkatareddykonasani/Datasets/master/Bank%20Tele%20Marketing/bank_market.csv')
print(bank)
```

Important Packages- Matplotlib

Plotting library similar to MATLAB plots

```
X = np.random.rand(50)
```

```
Y = np.random.rand(50)
```

```
print("X Array \n" ,X )
```

```
print("Y Array \n", Y)
```

```
import matplotlib as mp
```

```
mp.pyplot.scatter(X,Y)
```


Important Packages- Scikit-Learn

- Machine learning algorithms made easy

```
import sklearn as sk
import pandas as pd
```

```
air = pd.read_csv("D:\\Google
Drive\\Training\\Datasets\\AirPassengers\\AirPassengers.csv")
air
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(air[["Promotion_Budget"]], air[["Passengers"]])
```

```
#Coefficients
print(lr.coef_)
print(lr.intercept_)
```

Basic Commands on Datasets

- Is the data imported correctly? Are the variables imported in right format? Did we import all the rows?
- Once the dataset is inside Python, we would like to do some basic checks to get an idea on the dataset.
- Just printing the data is not a good option, always.
- Is a good practice to check the number of rows, columns, quick look at the variable structures, a summary and data snapshot

Check list after Import

Dataset - Sales_by_country_v1.csv

Code	Description
<code>Sales.shape</code>	To check the number of rows and columns
<code>Sales.info()</code>	All important information
<code>Sales.head(10)</code>	First few observations of data
<code>Sales.tail(10)</code>	Last few observations of the data

Descriptive statistics

- The basic descriptive statistics to give us an idea on the variables and their distributions
- Permit the analyst to describe many pieces of data with a few indices
- Central tendencies
 - Mean
 - Median
- Dispersion
 - Range
 - Variance
 - Standard deviation

Central tendencies

- Mean
 - The arithmetic mean
 - $\text{Sum of values} / \text{Count of values}$
 - Gives a quick idea on average of a variable

Mean in Python

Import “Census Income Data/Income_data.csv”

```
gain_mean=Income["capital-gain"].mean()  
print(gain_mean)
```

Guess the mean

1.5, 1.7, 1.9, 0.8, 0.8, 1.2, 1.9, 1.4, 9, 0.7, 1.1

Median

- Mean is not a good measure in presence of outliers
- For example Consider below data vector
 - 1.5, 1.7, 1.9, 0.8, 0.8, 1.2, 1.9, 1.4, 9, 0.7, 1.1
- 90% of the above values are less than 2, but the mean of above vector is 2
- There is an unusual value in the above data vector i.e 9
- It is also known as outlier.
- Mean is not the true middle value in presence of outliers. Mean is very much effected by the outliers.
- We use median, the true middle value in such cases
- Sort the data either in ascending or descending order

Median

1.5		0.7
1.7		0.8
1.9		0.8
0.8		1.1
0.8		1.2
1.2	➡	1.4
1.9		1.5
1.4		1.7
9		1.9
0.7		1.9
1.1		9

- Mean of the data is 2
- Median of the data is 1.4
- Even if we have the outlier as 90, we will have the same median
- Median is a positional measure, it doesn't really depend on outliers
- When there are no outliers then mean and median will be nearly equal
- When mean is not equal to median it gives us an idea on presence of outliers in the data

Mean and Median

Import “Census Income Data/Income_data.csv”

```
gain_median=Income["capital-gain"].median()  
print(gain_median)
```



Dispersion Measures : Variance and Standard Deviation

Dispersion

- Just knowing the central tendency is not enough.
- Two variables might have same mean, but they might be very different.
- Look at these two variables. Profit details of two companies A & B for last 14 Quarters in MMs

															Mean
Company A	43	44	0	25	20	35	-8	13	-10	-8	32	11	-8	21	15
Company B	17	15	12	17	15	18	12	15	12	13	18	18	14	14	15

- Though the average profit is 15 in both the cases
- Company B has performed consistently than company A.
- There was even losses for company A
- Measures of dispersion become very vital in such cases

Variance and Standard deviation

- Dispersion is the quantification of deviation of each point from the mean value.
- Variance is average of squared distances of each point from the mean
- Variance is a fairly good measure of dispersion.
- Variance in profit for company A is 352 and Company B is 4.9

Value	Value-Mean	(Value-Mean)^2
43	28	784
44	29	841
0	-15	225
25	10	100
20	5	25
35	20	400
-8	-23	529
13	-2	4
-10	-25	625
-8	-23	529
32	17	289
11	-4	16
-8	-23	529
21	6	36
15.0		352

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}$$

Value	Value-Mean	(Value-Mean)^2
17	2	4
15	0	0
12	-3	9
17	2	4
15	0	0
18	3	9
12	-3	9
15	0	0
12	-3	9
13	-2	4
18	3	9
18	3	9
14	-1	1
14	-1	1
15.0		4.9

Standard Deviation

- Standard deviation is just the square root of variance
- Variance gives a good idea on dispersion, but it is of the order of squares.
- Its very clear from the formula, variance unites are squared than that of original data.
- Standard deviation is the variance measure that is in the same units as the original data

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}}$$

LAB: Variance and Standard deviation

- Dataset: ". / Online Retail Sales Data / Online Retail.csv"
 - Use option `pd.read_csv("C:/Online Retail Sales Data/Online Retail.csv", encoding = "ISO-8859-1")`
- What is the variance and s.d of "UnitPrice"
- What is the variance and s.d of "Quantity"

LAB: Variance and Standard deviation

```
#var and sd UnitPrice
```

```
var_UnitPrice=Online_Retail['UnitPrice'].var()  
print("Variance of UnitPrice", var_UnitPrice)
```

```
std_UnitPrice=Online_Retail['UnitPrice'].std()  
print("S.D of UnitPrice", std_UnitPrice)
```

```
#var and sd Quantity
```

```
var_Quantity=Online_Retail['Quantity'].var()  
print("Variance of Quantity", var_Quantity)
```

```
std_Quantity=Online_Retail['Quantity'].std()  
print("S.D of Quantity", std_Quantity)
```




Percentiles

Percentiles

- A student attended an exam along with 1000 others.
 - He got 68% marks? How good or bad he performed in the exam?
 - What will be his rank overall?
 - What will be his rank if there were 100 students overall?
- For example, with 68 marks, he stood at 90th position. There are 910 students who got less than 68, only 89 students got more marks than him
- He is standing at 91 percentile.
- Instead of stating 68 marks, 91% gives a good idea on his performance
- Percentiles make the data easy to read

Percentiles

- p^{th} percentile: p percent of observations below it, $(100 - p)\%$ above it.
- Marks are 40 but percentile is 80%, what does this mean?
- 80% of CAT exam percentile means
 - 20% are above & 80% are below
- Percentiles help us in getting an idea on outliers.
- For example the highest income value is 400,000 but 95th percentile is 20,000 only. That means 95% of the values are less than 20,000. So the values near 400,000 are clearly outliers

Percentiles

```
Income['capital-gain'].quantile([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```


Lab: Outlier detection

- Import “Give me some Credit\cs-training.csv”
- Look at the percentiles of the variable monthly_utilization
- Are there any outliers?

Code: Outlier detection

```
loans['monthly_utilization'].quantile([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```



Data Cleaning and Imputation

X1
11.0
11.1
11.9
10.9
10.8
.
11.5
11.6
11.6
11.4
11
12
11.8
11.4
11.9

Missing Value Imputation

- **Standalone imputation**

- Mean, median, other point estimates
- Convenient, easy to implement
- **Assume:** Distribution of the missing values is the same as the non-missing values.
- Does not take into account inter-relationships
- **Eg:** The average of available values is 11.4. Can we replace the missing value in this table by **11.4** ?

X1
11.0
11.1
11.9
10.9
10.8
.
11.5
11.6
11.6
11.4
11
12
11.8
11.4
11.9

LAB: Outlier Treatment

- Perform imputation on monthly_utilization

Code: Outlier Treatment

```
loans['util_new']=loans['monthly_utilization']  
loans['util_new'][loans['util_new']>1]=median_util
```

Model Building Life Cycle

Background and Objective

Business Objective

Set Goals

Project Plan

Budget & Resources

Data Exploration

Collect data

Explore data

Basic Summary

Identify outliers and missing values

Preparing data for analysis

Validate data

Outlier treatment

Missing value treatment

Clean the data

Prepare data for Analysis

Building the model

Select the right model

Variable selection

Model building and finetuning

Model iterations

Validating the model

In time validation

Out of time validation

Model finetuning

Deployment

Deploy model

Maintenance of model

Model monitoring

Conclusion

- In this session we discussed basics of python
- Basic statistics
- Basics of data exploration and cleaning