

**MULTIVARIATE ANALYSIS AND MACHINE LEARNING TECHNIQUES- FEATURE ANALYSIS IN DATA SCIENCE USING PYTHON**

**DATA SETS AND SOFTWARE PACKAGES**



Version 2.1

Table of Contents

[1 DESCRIPTION OF DATA SETS 2](#_Toc126006136)

[1.1 diamonds 2](#_Toc126006137)

[1.2 Planets 2](#_Toc126006138)

[1.3 Penguins 3](#_Toc126006139)

[1.4 ChickWeight 4](#_Toc126006140)

[1.5 Iris 5](#_Toc126006141)

[1.6 Tips 6](#_Toc126006142)

[1.7 IT Projects 7](#_Toc126006143)

[1.8 Breast Cancer - Gene Prognostic Signature 8](#_Toc126006144)

[1.9 German Breast Cancer Study Group 2 9](#_Toc126006145)

[1.10 West Virginia Student Fraternity 10](#_Toc126006146)

[1.11 The CIFAR-10 dataset and CIFAR-100 dataset 11](#_Toc126006147)

[1.12 MNIST handwritten character recognition 11](#_Toc126006148)

[1.13 PowerConsumption 11](#_Toc126006149)

[1.14 Bank Marketing 12](#_Toc126006150)

[2 SOFTWARE PACKAGE INSTALLATION 14](#_Toc126006151)

[2.1 Package Managers – conda, and pip 14](#_Toc126006152)

[2.2 Install Anaconda 15](#_Toc126006153)

[2.3 Modify the ‘path’ variable 16](#_Toc126006154)

[2.4 Install Python 16](#_Toc126006155)

[2.5 Launch Spyder IDE for program development 16](#_Toc126006156)

[2.6 Install Other Packages 17](#_Toc126006157)

[2.7 Install TensorFlow and Keras 17](#_Toc126006158)

[2.8 Notes on Google Colab, Spyder, and other IDEs 18](#_Toc126006159)

[2.9 Check / Resolve conflicts in package installation 18](#_Toc126006160)

[2.10 Installation from the Command Prompt - for experienced programmers 19](#_Toc126006161)

# DESCRIPTION OF DATA SETS

Note that some of the following datasets are available in this repository.

## diamonds

**Refer:** R documentation

**How to Access the Dataset?**

sb.get\_dataset\_names()

pdf = sb.load\_dataset('diamonds')

pdf.info()

**Description**

Diamonds dataset contains the prices and other attributes of almost 54,000 diamonds. The variables are as follows: -

* price: price in US dollars (326 - 18823)
* carat: weight of the diamond (0.2 - 5.01)
* cut: quality of the cut (Fair, Good, Very Good, Premium, Ideal)
* color: diamond color, from D (best) to J (worst)
* clarity: indicates how clear the diamond is (I1), SI2,..., IF)
* x: length in mm (0 -10.74)
* y: width in mm (0 - 58.9)
* z: depth in mm (0 -31.8)
* depth: total depth % = Z / mean(X,Y) (43 - 79)
* table: width of to the p of the diamond relative to widest point (43-95)

**Sample Data**

carat cut color clarity depth table price x y z

0 0.23 Ideal E SI2 61.5 55.0 326 3.95 3.98 2.43

1 0.21 Premium E SI1 59.8 61.0 326 3.89 3.84 2.31

2 0.23 Good E VS1 56.9 65.0 327 4.05 4.07 2.31

... ... ... ... ... ... ... ... ... ...

53937 0.70 Very Good D SI1 62.8 60.0 2757 5.66 5.68 3.56

53938 0.86 Premium H SI2 61.0 58.0 2757 6.15 6.12 3.74

53939 0.75 Ideal D SI2 62.2 55.0 2757 5.83 5.87 3.64

[53940 rows x 10 columns]

## Planets

**Refer:** seaborn

**How to Access the Dataset?**

import seaborn as sb

pdf = sb.load\_dataset('planets')

pdf.info()

**Description**

RangeIndex: 1035 entries, 0 to 1034

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 method 1035 non-null object Category with 10 labels

1 number 1035 non-null int64

2 orbital\_period 992 non-null float64 \* missing values

3 mass 513 non-null float64 \* missing values

4 distance 808 non-null float64 \* missing values

5 year 1035 non-null int64

**Sample Data**

method number orbital\_period mass distance year

0 Radial Velocity 1 269.300000 7.10 77.40 2006

1 Radial Velocity 1 874.774000 2.21 56.95 2008

2 Radial Velocity 1 763.000000 2.60 19.84 2011

... ... ... ... ... ...

1032 Transit 1 3.191524 NaN 174.00 2007

1033 Transit 1 4.125083 NaN 293.00 2008

1034 Transit 1 4.187757 NaN 260.00 2008

[1035 rows x 6 columns]

## Penguins

**Refer:** https://github.com/allisonhorst/penguins

**How to Access the Dataset?**

import seaborn as sb

pdf = sb.load\_dataset('penguins')

pdf.info()

**Description**

The dataset contain data for 344 penguins. There are 3 different species of penguins in this dataset, collected from 3 islands in the Palmer Archipelago, Antarctica.

Column Non-Null Count Dtype

0 species 344 non-null ['Adelie', 'Chinstrap', 'Gentoo']

1 island 344 non-null ['Biscoe','Dream','Torgersen']

2 bill\_length\_mm 342 non-null float64 \* 2 missing values

3 bill\_depth\_mm 342 non-null float64 \* 2 missing values

4 flipper\_length\_mm 342 non-null float64 \* 2 missing values

5 body\_mass\_g 342 non-null float64 \* 2 missing values

6 sex 333 non-null ['Female','Male'] \* 9 miss.values

**Sample Data**

species island bill\_len… ... flipper\_len… body\_mass\_g sex

0 Adelie Torgersen 39.1 ... 181.0 3750.0 Male

1 Adelie Torgersen 39.5 ... 186.0 3800.0 Female

2 Adelie Torgersen 40.3 ... 195.0 3250.0 Female

.. ... ... ... ... ... ... ...

341 Gentoo Biscoe 50.4 ... 222.0 5750.0 Male

342 Gentoo Biscoe 45.2 ... 212.0 5200.0 Female

343 Gentoo Biscoe 49.9 ... 213.0 5400.0 Male

[344 rows x 7 columns]

## ChickWeight

**Refer:** R Documentation

**How to Access the Dataset?**

import statsmodels.api as sm

pdf =sm.datasets.get\_rdataset("ChickWeight").data

pdf.columns

pdf.info()

**Description**

The body weights of the chicks were measured at birth and every second day thereafter until day 20. They were also measured on day 21. There were four groups of chicks on different protein diets.

* Weight: a numeric vector giving the bodyweight of the chick (gm).
* Time: a numeric vector giving the number of days since birth when the measurement was made.
* Chick: an ordered factor with levels 18 < … < 48 giving a unique identifier for the chick. The ordering of the levels groups chicks on the same diet together and orders them according to their final weight (lightest to heaviest) within the diet.
* Diet: a factor with levels 1, …, 4 indicating which experimental diet the chick received.

RangeIndex: 578 entries, 0 to 577

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 weight 578 non-null int64

1 Time 578 non-null int64 # 0 ... 21

2 Chick 578 non-null int64

3 Diet 578 non-null int64 # values 0,1,2,3

**Sample Data**

weight Time Chick Diet

0 42 0 1 1

1 51 2 1 1

2 59 4 1 1

.. ... ... ... ...

575 234 18 50 4

576 264 20 50 4

577 264 21 50 4

[578 rows x 4 columns]

## Iris

**Refer:** R Documentation

**How to Access the Dataset?**

**(a) Common Method**

import seaborn as sb

pdf = sb.load\_dataset('iris')

pdf.columns

pdf.info()

**(b) For Machine Learning**

from sklearn import datasets

iris = datasets.load\_iris()

X = iris.data

y = iris.target

**Description**

The dataset consists of measurements on 150 iris flowers belonging to three categories – setosa, versicolor, and virginica. Four features are measured. Length and width of sepals and petals.

Column Non-Null Count Dtype

--- ------ -------------- -----

0 sepal\_length 150 non-null float64

1 sepal\_width 150 non-null float64

2 petal\_length 150 non-null float64

3 petal\_width 150 non-null float64

4 species 150 non-null object - setosa, versicolor, virginica

**Sample Data**

sepal\_length sepal\_width petal\_length petal\_width species

0 5.1 3.5 1.4 0.2 setosa

1 4.9 3.0 1.4 0.2 setosa

2 4.7 3.2 1.3 0.2 setosa

.. ... ... ... ... ...

147 6.5 3.0 5.2 2.0 virginica

148 6.2 3.4 5.4 2.3 virginica

149 5.9 3.0 5.1 1.8 virginica

[150 rows x 5 columns]

## Tips

**Refer:** seaborn

**How to Access the Dataset?**

import seaborn as sb

pdf=sb.load\_dataset('tips')

pdf.info( )

**Description**

Food servers’ tips in restaurants may be influenced by many factors. The following dataset shows customers served during two and a half months.

RangeIndex: 244 entries, 0 to 243

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 total\_bill 244 non-null float64

1 tip 244 non-null float64

2 sex 244 non-null category ['Male', 'Female']

3 smoker 244 non-null category ['Yes', 'No']

4 day 244 non-null category ['Thur', 'Fri', 'Sat', 'Sun']

5 time 244 non-null category ['Lunch', 'Dinner']

6 size 244 non-null int64

**Sample Data**

total\_bill tip sex smoker day time size

0 16.99 1.01 Female No Sun Dinner 2

1 10.34 1.66 Male No Sun Dinner 3

2 21.01 3.50 Male No Sun Dinner 3

.. ... ... ... ... ... ... ...

241 22.67 2.00 Male Yes Sat Dinner 2

242 17.82 1.75 Male No Sat Dinner 2

243 18.78 3.00 Female No Thur Dinner 2

[244 rows x 7 columns]

## IT Projects

**Refer**: https://github.com/srikrishnan-punalur/MVA-ML

**How to Access the Dataset?**

import pandas as pd

# Download the file from GitHub

pdf=pd.read\_csv('itprojects.csv')

pdf.info()

**Description**

A study was conducted to explore the variables that influence software project performance, a float. Eleven variables were hypothesized to influence project performance – they are measured on a scale of 0..6. 100 responses were collected from IT consultants working on various projects. Project Type is a category Variable with three labels - 0,1,2

'''

RangeIndex: 100 entries, 0 to 99

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Case\_No 100 non-null int64 Serial Number

1 change\_mgmt 100 non-null int64 feature

2 project\_plan 100 non-null int64 feature

3 tech\_mentoring 100 non-null int64 feature

4 pm\_tools 100 non-null int64 feature

5 dev\_process 100 non-null int64 feature

6 system\_arch 100 non-null int64 feature

7 design\_think 100 non-null int64 feature

8 team\_skills 100 non-null int64 feature

9 core\_team 100 non-null int64 feature

10 prior\_exp 100 non-null int64 feature

11 rewards\_recog 100 non-null int64 feature

12 project\_type 100 non-null int64 [0, 1, 2]

13 project\_perf 100 non-null float64 target variable

dtypes: float64(1), int64(13)

'''

#

**Sample Data**

Case\_No change\_mgmt ... project\_type project\_perf

0 1 5 ... 1 0.374933

1 2 6 ... 1 2.131779

2 3 5 ... 1 1.679185

.. ... ... ... ... ...

97 98 5 ... 2 -1.192247

98 99 6 ... 1 2.401460

99 100 5 ... 1 0.383024

[100 rows x 14 columns]

## Breast Cancer - Gene Prognostic Signature

**Refer:** <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE7390>

scikit-survival

**How to Access the Dataset?**

from sksurv.datasets import load\_breast\_cancer

X, Y = load\_breast\_cancer()

**Description**

A Survival analysis study was conducted on 198 breast cancer patients to predict distant metastases (dm). Distant metastasis (dm) refers to cancer that has spread from the original (primary) tumor to distant organs/lymph nodes. The study included 76-gene prognostic signatures in lymph node-negative (N-) breast cancer patients.

**Features (X0 … X79): 80 features**

* 76 gene characteristics
* age
* er (estrogen-receptor) categories {positive, negative}

Note:- About 80% of all breast cancers are er-positive

* grade (the abnormality of cancer cells) categories:-

{intermediate, poorly differentiated, well differentiated}

* size

**Survival Data (Y)**: structured array with 2 fields

* e.tdm: boolean

True indicates that metastasis has occurred

False indicates that the event time is right-censored

* t.tdm: time to distant metastasis in days

**Features (X)**

Data columns (total 80 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 X200726\_at 198 non-null float64

1 X200965\_s\_at 198 non-null float64

2 X201068\_s\_at 198 non-null float64

3 X201091\_s\_at 198 non-null float64

4 X201288\_at 198 non-null float64

... ... ... ... ...

74 X221916\_at 198 non-null float64

75 X221928\_at 198 non-null float64

76 **age** 198 non-null float64

77 **er** 198 non-null **category** ['negative', 'positive']

78 **grade** 198 non-null **category**

['poorly differentiated',

'intermediate', 'well differentiated', 'unkown']

79 **size** 198 non-null float64

**Sample Data**

**X**

X200726\_at X200965\_s\_at ... grade size

0 10.926361 8.962608 ... poorly differentiated 3.0

1 12.242090 9.531718 ... poorly differentiated 3.0

2 11.661716 10.238680 ... poorly differentiated 2.5

3 12.174021 9.819279 ... poorly differentiated 1.8

4 11.484011 11.489233 ... intermediate 3.0

.. ... ... ... ... ...

193 12.018292 8.323876 ... poorly differentiated 2.2

194 11.711415 10.428482 ... poorly differentiated 3.2

195 11.939616 9.615587 ... well differentiated 2.5

196 11.848449 10.528911 ... intermediate 1.2

197 11.425778 9.901486 ... poorly differentiated 2.5

[198 rows x 80 columns]

**Y**

array([( True, 723.), (False, 6591.), ( True, 524.), (False, 6255.),

... ... ... ...

(False, 2722.), (False, 1781.)]

## German Breast Cancer Study Group 2

**Refer:** <https://ascopubs.org/doi/abs/10.1200/jco.1994.12.10.2086>

scikit-survival

**How to Access the Dataset?**

from sksurv.datasets import load\_gbsg2

x, y = load\_gbsg2()

x.info()

**Description**

A dataset with having 686 samples and 8 features. The endpoint is recurrence free survival, which occurred for 299 patients (43.6%). This dataset is the part of a study by the German Breast Cancer Study Group (GBSG) in 1984. This was a randomized clinical trial to compare the effectiveness of different drugs dosage, on recurrence-free and overall survival

X (features)

0 age 686 non-null float64

1 estrec 686 non-null float64

2 horTh 686 non-null category -> ['no', 'yes']

3 menostat 686 non-null category -> ['Pre', 'Post']

4 pnodes 686 non-null float64

5 progrec 686 non-null float64

6 tgrade 686 non-null category -> ['I', 'II', 'III']

7 tsize 686 non-null float64

Y (outcome)

'cens', Boolean

'time', float (survival in days)

**Sample Data**

**x (small letter x)**

age estrec horTh menostat pnodes progrec tgrade tsize

0 70.0 66.0 no Post 3.0 48.0 II 21.0

1 56.0 77.0 yes Post 7.0 61.0 II 12.0

2 58.0 271.0 yes Post 9.0 52.0 II 35.0

.. ... ... ... ... ... ... ... ...

683 51.0 0.0 no Pre 5.0 43.0 III 25.0

684 52.0 34.0 no Post 3.0 15.0 II 23.0

685 55.0 15.0 no Post 9.0 116.0 II 23.0

[686 rows x 8 columns]

**x (small letter x)**

array([( True, 1814.), ( True, 2018.), ( True, 712.), ( True, 1807.),

... ... ... ...

( True, 727.), ( True, 1701.)]

## West Virginia Student Fraternity

**Refer**: https://github.com/srikrishnan-punalur/MVA-ML

**How to Access the Dataset?**

import pandas as pd

# Download the file from GitHub

d=pd.read\_csv('WV-fraternity.csv')

d.shape # (116, 58)

**Description**

This dataset consists of information about the interaction of students in a student fraternity in West Virginia - the number of times they interacted over a week. This dataset is used for social network analysis in Chapter 10.

## The CIFAR-10 dataset and CIFAR-100 dataset

**Refer:** <https://www.cs.toronto.edu/~kriz/cifar.html>

**How to Access the Dataset?**

from tensorflow.keras import datasets

(X\_train, y\_train), (X\_test, y\_test) = datasets.cifar10.load\_data()

**Description**

The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images. The image classes are: - airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

CIFAR-100 dataset has 100 classes containing 600 images each. The 100 classes are grouped into 20 superclasses.

## MNIST handwritten character recognition

**Refer**: [Refer:- http://yann.lecun.com/exdb/mnist/](https://www.cs.toronto.edu/~kriz/cifar.html)

**How to Access the Dataset?**

This is described in Chapter 12, Artificial Neural Network

**Description**

The MNIST database of handwritten digits houses 60,000 training samples and 10,000 test samples of handwritten digit images in black and white {0,1}. The digits have been size-normalized and centered in a fixed-size image of 28x28. Develop a convolutional neural network to classify the images. Describe the network structure and parameters.

## PowerConsumption

**Refer –** https://github.com/srikrishnan-punalur/MVA-ML

**How to Access the Dataset?**

import pandas as pd

# Download the file from GitHub

d=pd.read\_csv('PowerConsumption\_2013\_19.csv')

d.shape # (2161, 9)

**Description**

The dataset consists of data gathered from the electricity board, for a state in India, reported from 2013 to 2019. The features include daily power demand, shortage, consumption, load, OD/UD, and date. These are typical time-series / sequences and can be used for RNN/ LSTM exercises.

## Bank Marketing

**Refer** – https://archive.ics.uci.edu/ml/datasets/bank+marketing

**How to Access the Dataset?**

import pandas as pd

# Download the file from the url referred above

d = pd.read\_csv ('BankMarketing.csv',delimiter=';')

d.shape #(45211, 17)

**Description**

The dataset consists of data gathered from a bank marketing survey. It has 21 variables and 45211 rows. Feature variables considered in the exercises are 'age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous'. The target (independent variable) is ‘Has the client in the training dataset, subscribed to a term deposit?’ with binary answers ('yes’, ‘no').

X (features)

0 age 45211 non-null int64

1 job 45211 non-null object

2 marital 45211 non-null object

3 education 45211 non-null object

4 default 45211 non-null object

5 balance 45211 non-null int64

6 housing 45211 non-null object

7 loan 45211 non-null object

8 contact 45211 non-null object

9 day 45211 non-null int64

10 month 45211 non-null object

11 duration 45211 non-null int64

12 campaign 45211 non-null int64

13 pdays 45211 non-null int64

14 previous 45211 non-null int64

15 poutcome 45211 non-null object

Y (outcome)

16 y 45211 non-null object

# SOFTWARE PACKAGE INSTALLATION

Software libraries get revised frequently. Therefore, the commands, and software versions mentioned below may undergo changes. Refer to the home page of the respective software projects for current usage. The topics covered in this section are: -

1. Package Managers – conda, and pip

2. Software Package Installation

2.1 Install Anaconda

2.2 Modify the ‘path’ variable

2.3 Install Python

2.4 Launch Spyder IDE for program development

2.5 Install Other Packages

2.6 Install TensorFlow and Keras

3. Notes on Google Colab, Spyder, and other IDEs

4. Check / Resolve conflicts in package installation

5. Installation from the Command Prompt - for experienced programmers

## Package Managers – conda, and pip

Conda and pip are both used for installing Python packages. We can install anaconda and use the conda command from the anaconda prompt to install other packages of the Python ecosystem. Alternatively, the pip command from the command prompt or the !pip from a Python IDE can be used to install packages. The earlier method (conda) is recommended for beginners, though the latter method may be used by professionals for installing specific packages and setting up work environments. The objective of this section is to develop a primary understanding of software installation for building a working Python development environment for machine learning.

As mentioned above, we can use the !pip for package installation in Google Colab, and other IDEs. The command the !pip is also used to check and resolve path dependencies of the installed packages. This ensures that the packages are not broken and are accessible for use.

The Python Package Index (PyPI) is a repository for Python Software packages. PyPI. Python software package developers use PyPI to distribute their software. PyPI has a massive 348,665 projects. **Pip** is the Python Packaging Authority’s recommended package manager for installing python packages from the Python Package Index. Pip installs Python software packaged as wheels or source distributions. The latter may require that the system have compatible compilers, and possibly libraries, installed before invoking pip to succeed.

Conda is a cross-platform package and environment manager that installs and manages software packages from the Anaconda repository and the Anaconda Cloud. The repository has about 1500 software packages. They have Python packages and other software such as C or C++ libraries, and R packages. Conda packages are binaries. Therefore, there is no need for compilation on the host system.

Python must be installed, before using pip manager. We may download Python from python.org and install it. However, Conda can install Python, and Python packages directly using conda commands. Pip depends on other tools like virtualenv or venv to create isolated environments. However, Conda can create isolated environments that consist of different versions of Python/packages.

While installing packages, pip installs dependencies in a recursive, serial loop. This may lead to environments that are broken if the compatibility of packages installed earlier conflicts with the packages installed later. However, conda checks the package metadata and resolves dependencies, thereby creating a robust work environment. The steps involved in software installation are shown below.

## Install Anaconda

It is recommended that we download the latest version of Anaconda and install it before installing Python or Python packages. While installation progresses, be attentive; and answer the questions (like proceed Yes/No). Note the following points:-

* If we accept the default option to install Anaconda, it is installed in your user home directory:

C:\Users\<your-username>\Anaconda3\

* It is recommended that we do not install anaconda in paths that contain spaces such as C:\Program Files. Instead, create your directory in an appropriate folder with an appropriate name. For example,

C:\anaconda

* For doing the above (a) you must select the ‘multiuser’ option and (b) you must have admin authority.
* While installing Anaconda, it is recommended that you may not add the anaconda path (C:\anaconda) to the Windows PATH.

## Modify the ‘path’ variable

The OS uses certain variables, one of which is ‘path’. This variable contains a list of directories where the system will search for executable code. We need to add the path where anaconda scripts are present.

* Modify the windows path by editing 'environment variables' (from the start menu)
* Click on ‘path’ variable
* Click ‘edit’
* Add a 'new' path - the place where anaconda scripts are installed – e.g., it may look like the one below

C:\Users\<Your Usernae>\anaconda3\Scripts\

* Click ok
* Now the 'path' variable is updated. The system knows where to look for anaconda scripts!

## Install Python

After installing anaconda, you may install Python, using Conda prompt. For this, on Windows, open the Start menu and open Anaconda Command Prompt. Execute the following command

conda create -n py39 python=3.9

conda activate py39

(assuming that python 3.9 is the latest stable version of python)

If an older version of Python is already installed, you may upgrade it from the ‘anaconda prompt’ using the above command.

## Launch Spyder IDE for program development

In windows start menu, click on 'anaconda navigator' app. When it opens up, launch 'Spyder' inside 'anaconda navigator'. Now Spyder IDE is available for programming

**Spyder – font size, theme**

Open Spyder.

Select 'Tools' menu.

Then select submenu 'preferences'.

Select 'appearance'

Here you may change font size (fonts -size)

You may also change background and text color layout (icon theme --> 'spyder')

Now click 'OK'

**Spyder – scale**

You may again select 'Tools' menu. Then select submenu 'preferences'.

You may select 'application' to change the scale of display

## Install Other Packages

We can use the anaconda package manager to install any package. See Table 1-1.

Table 1‑1 Using the ‘conda’ command for Package Installation

|  |  |
| --- | --- |
| Run ‘conda’ Package Manager Command | Package Functionality  (See Table1-2 for references |
| conda install -c conda-forge python-graphviz  conda install -c conda-forge pingouin  conda install -c desilinguist factor\_analyzer  conda install -c sebp scikit-survival  conda install -c conda-forge eli5  {Use *pip install apyori* command; See Table 1-2} | To Draw Decision Tree  Partial Correlation coef.  Factor Analysis  Survival Analysis  Survival -Feature Ranking  Market Basket Analysis |

* if a package is already installed, it may be upgraded using the same command used above. For example,

conda install pingouin

## Install TensorFlow and Keras

On Windows, open the Start menu, then open an Anaconda Command Prompt.

* Choose a name for your TensorFlow environment, such as ‘TensorFlow’. Beginners are recommended to install the current release of CPU-only TensorFlow.

conda create -n tf tensorflow

conda activate tf

* After installing anaconda, python, and tensor flow, install Keras

conda install -c conda-forge keras

(or Run conda install -c conda-forge/label/broken keras)

## Notes on Google Colab, Spyder, and other IDEs

We saw how to use conda or pip for package installation. There is yet another method. We can use the !pip command from within an IDE (e.g., google colab or sypder) for package installation. See Table 1-2. So, how do you install python packages, in Google Colab? You must use the above method (!pip command) from within the Colab.

Table 1‑2 !pip - Google Colab and other IDEs, Package Installation / Path Resolution

|  |  |
| --- | --- |
| Installation of Python Packages in Google Colab or other IDEs (the same can be used to resolve code break issues, due to incorrect installation) | |
| !pip3 install --upgrade pip | <https://pypi.org/project/pip/> |
| !pip install graphviz | <https://graphviz.org/> |
| !pip install pingouin | https://pingouin-stats.org/ |
| !pip install factor-analyzer | <https://factor-analyzer.readthedocs.io/en/latest/factor_analyzer.html> |
| !pip install apyori | <https://pypi.org/project/apyori/> |
| !pip scikit-survival | <https://scikit-survival.readthedocs.io/en/stable/install.html> |
| !pip eli5 | (Same as above) |
| !pip3 install --upgrade tf | <https://www.tensorflow.org/install> |

## Check / Resolve conflicts in package installation

When we try to access an installed package from within an IDE, we may get the error - ‘ModuleNotFoundError’. One of the reasons may that the system is not able to resolve the access path. !pip command can be used to check and resolve path dependencies of the installed packages (See Table 1-2). This ensures that the packages are not broken and are accessible for usage. You may do the following inspections to ensure smooth library access.

* To check whether a software package properly installed
* Let’s check ‘FactorAnalyzer’ package
* In the Syder IDE, enter the following command and ‘run’ it

from factor\_analyzer import FactorAnalyzer

>> ModuleNotFoundError: No module named 'factor\_analyzer'

* Now let’s remedy the error. Enter the following commands and ‘run’ it

!pip install factor\_analyzer

>> Successfully installed ...

from factor\_analyzer import FactorAnalyzer

* Check TensorFlow installation

!python -m pip show tf

>> WARNING: Package(s) not found: tf

* TensorFlow – Resolve the error reported above

!pip3 install --upgrade tf

!python -m pip show tf

>> Name: tf

>> Version: 1.0.0

>> Location: c:\users\sri-s\anaconda3\lib\site-packages

* Check Keras installation

!python -c "import keras; print(keras.\_\_version\_\_)"

2.9.0

* Check Access to any Package, from within an IDE

Import pandas as pd

from sksurv.datasets import load\_gbsg2

## Installation from the Command Prompt - for experienced programmers

It is recommended that we use the conda package manager for installing the Python work environment. However, we may need to use the pip package manager from the command prompt for some packages. See Table 1-3. The only difference with Table 1-2, is the absence of the exclamation mark (pip, instead of !pip)

Table 1‑3 System Command Prompt pip Command and package references

|  |  |
| --- | --- |
| Installing Python Packages from system command prompt | Download the package to an appropriate directory. From there issue pip command |
| pip3 install --upgrade pip | Package Manager pip |
| pip install graphviz | <https://graphviz.org/> |
| pip install pingouin | https://pingouin-stats.org/ |
| pip install factor-analyzer | [https://factor-analyzer.readthedocs.io/en/latest/factor\_analyzer.html](https://factor-analyzer.readthedocs.io/en/latest/factor_analyzer.html%20) |
| pip install apyori | https://pypi.org/project/apyori/ |
| pip scikit-survival  pip eli5 | https://scikit-survival.readthedocs.io/en/stable/install.html |

We may download a package from the web, install it, and add the path to the ‘PATH’ variable. For example, download the Graphviz installer to an appropriate folder, e.g., C:\Analytics\graphviz. After downloading the installer, we can use ‘pip’ from the Microsoft Windows command prompt 'cmd'.

* Download-> graphviz-2.50.0 (64-bit) EXE installer [sha256]
* CD C:\Analytics\graphviz
* pip install graphviz
* Check the python version in the IDE

import os

os.environ["PATH"] #check the current access paths

* If graphvz is not present in the path, add it to the path, as follows:-

os.environ["PATH"] += os.pathsep + 'C:\\Analytics\\graphviz\\bin'

* In the same way, as described above, other packages can be installed
* if a package is already installed, it may be upgraded. For example,

pip install --upgrade pingouin

pip install pingouin