Machine Learning -1

Course Syllabus and Materials

Follow link (ctrl + click)

Students Labs

https://docs.google.com/spreadsheets/d/10LpJrdobC28Ttrnghw3fpL9UeN39qD64uqvFlixQROY/edit?usp=sharing

Google Colaboratory

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education.

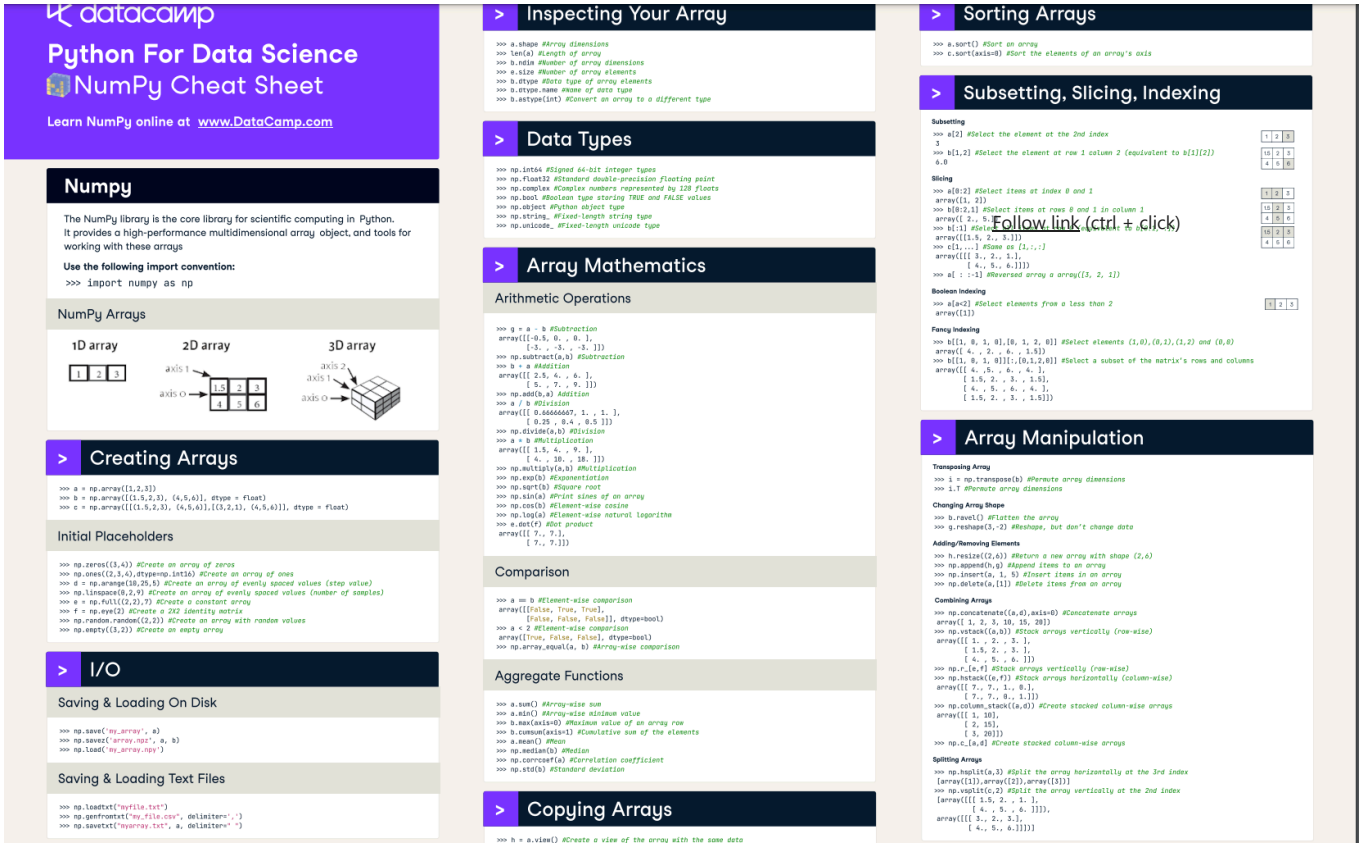
1. https://jupyter.org/
2. https://colab.research.google.com/drive/16pBJQePbqkz3QFV54L4NIkOn1kwpuRrj#scrollTo=u1wdmFKqylSl
3. https://colab.research.google.com/notebooks/basic_features_overview.ipynb#scrollTo=JyG45Qk3qQLS

Double-click (or enter) to edit

Scentific Computing- NumPy

NumPy is the fundamental package for scientific computing in Python. It is a **Python library** that provides a **multidimensional array object**, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

1. https://numpy.org/doc/stable/user/index.html#user
2. https://images.datacamp.com/image/upload/v1676302459/Marketing/Blog/Numpy_Cheat_Sheet.pdf

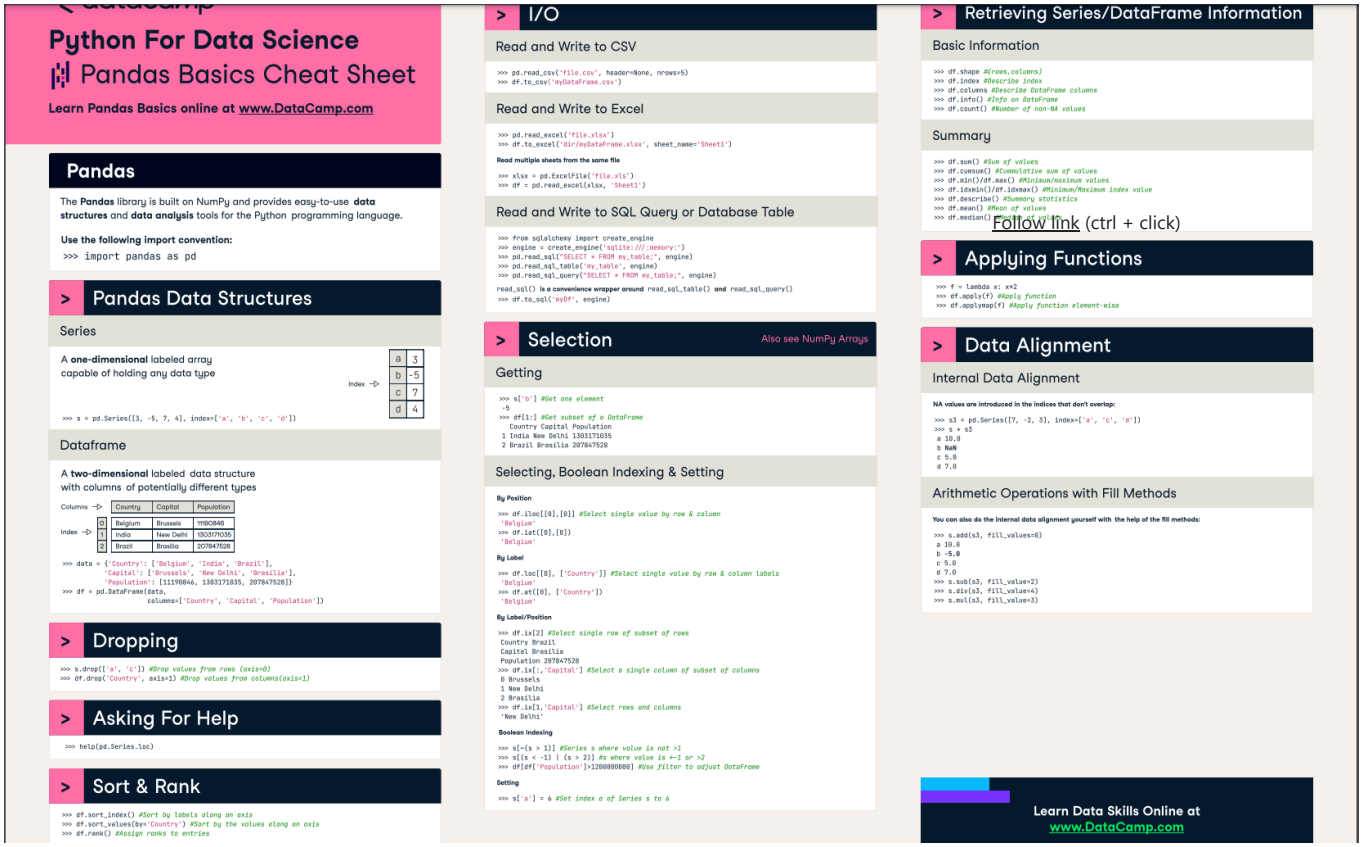


Follow link (ctrl + click)

Data Analysis-Pandas

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.

1. https://pandas.pydata.org/docs/getting_started/intro_tutorials/index.html
2. https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf
3. https://images.datacamp.com/image/upload/v1676302204/Marketing/Blog/Pandas_Cheat_Sheet.pdf



Python For Data Science
Pandas Basics Cheat Sheet

Learn Pandas Basics online 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

Index → | a | 3 | b | -5 | c | 7 | d | 4 |

```
>>> 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

Index	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasília	207847528

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasília'],
            'Population': [11190846, 1303171035, 207847528]}
>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Dropping

```
>>> s.drop(['a', 'c']) #Drop values from rows (axis=0)
>>> df.drop('Country', axis=1) #Drop values from columns(axis=1)
```

Asking For Help

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

Sort & Rank

```
>>> df.sort_index() #Sort by labels along an axis
>>> df.sort_values(by='Country') #Sort by the values along an axis
>>> df.rank() #Assign ranks to entries
```

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)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b'] #Get one element
-5
>>> df[1:] #Get subset of a DataFrame
  Country Capital Population
1 India New Delhi 1303171035
2 Brazil Brasília 207847528
```

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[[0],[0]] #Select single value by row & column
'Belgium'
>>> df.iat([0],[0])
'Belgium'
```

By Label

```
>>> df.loc[[0], ['Country']] #Select single value by row & column labels
'Belgium'
>>> df.at([0], ['Country'])
'Belgium'
```

By Label/Position

```
>>> df.ix[2] #Select single row of subset of rows
Country Brazil
Capital Brasília
Population 207847528
>>> df.ix[:,'Capital'] #Select a single column of subset of columns
0 Brussels
1 New Delhi
2 Brasília
>>> df.ix[1,'Capital'] #Select rows and columns
'New Delhi'
```

Boolean Indexing

```
>>> s[~(s > 1)] #Series s where value is not >1
>>> s[(s < -1) | (s > 2)] #s where value is <-1 or >2
>>> df[df['Population']>1200000000] #Use filter to adjust DataFrame
```

Setting

```
>>> s['a'] = 6 #Set index a of Series s to 6
```

Retrieving Series/DataFrame Information

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
```

Follow link (ctrl + click)

Applying Functions

```
>>> f = lambda x: x+2
>>> df.apply(f) #Apply function
>>> df.applymap(f) #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)
```

Learn Data Skills Online at www.DataCamp.com

Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4, 5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
        index = [1, 2, 3])
```
Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```
Specify values for each row.

		a	b	c
N	v			
D	1	4	7	10
	2	5	8	11
e	2	6	9	12

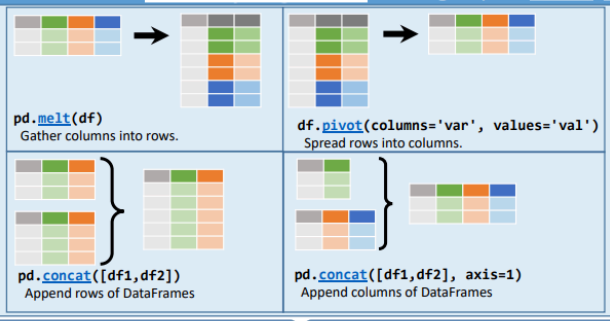
```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d', 1), ('d', 2),
         ('e', 2)], names=['n', 'v']))
```
Create DataFrame with a MultiIndex

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable':'var',
                'value':'val'})
        .query('val >= 200')
    )
```

Reshaping Data – Change layout, sorting, reindexing, renaming



pd.melt(df)
Gather columns into rows.

df.pivot(columns='var', values='val')
Spread rows into columns.

pd.concat([df1,df2])
Append rows of DataFrames

pd.concat([df1,df2], axis=1)
Append columns of DataFrames

df.sort_values('mpg')
Order rows by values of a column (low to high).

df.sort_values('mpg', ascending=False)
Order rows by values of a column (high to low).

df.rename(columns = {'y':'year'})
Rename the columns of a DataFrame

df.sort_index()
Sort the index of a DataFrame

df.reset_index()
Reset index of DataFrame to row numbers, moving index to columns.

df.drop(columns=['Length', 'Height'])
Drop columns from DataFrame

Subset Observations - rows

df[df.Length > 7]
Extract rows that meet logical criteria.

df.drop_duplicates()
Remove duplicate rows (only considers columns).

df.sample(frac=0.5)
Randomly select fraction of rows.

df.sample(n=10) Randomly select n rows.

df.nlargest(n, 'value')
Select and order top n entries.

df.nsmallest(n, 'value')
Select and order bottom n entries.

df.head(n)
Select first n rows.

df.tail(n)
Select last n rows.

Subset Variables - columns

df[['width', 'length', 'species']]
Select multiple columns with specific names.

df['width'] *or* **df.width**
Select single column with specific name.

df.filter(regex='*regex*')
Select columns whose name matches regular expression *regex*.

Using query

query() allows Boolean expressions for filtering rows.

```
df.query('Length > 7')
df.query('Length > 7 and Width < 8')
df.query('Name.str.startswith("abc")',
         engine="python")
```

Subsets - rows and columns

Use **df.loc[]** and **df.iloc[]** to select only rows, only columns or both.
Use **df.at[]** and **df.iat[]** to access a single value by row and column.
First index selects rows, second index columns.

df.iloc[10:20]
Select rows 10-20.

df.iloc[:, [1, 2, 5]]
Select columns in positions 1, 2 and 5 (first column is 0).

df.loc[:, 'x2':'x4']
Select all columns between x2 and x4 (inclusive).

df.loc[df['a'] > 10, ['a', 'c']]
Select rows meeting logical condition, and only the specific columns .

df.iat[1, 2] Access single value by index

df.at[4, 'A'] Access single value by label

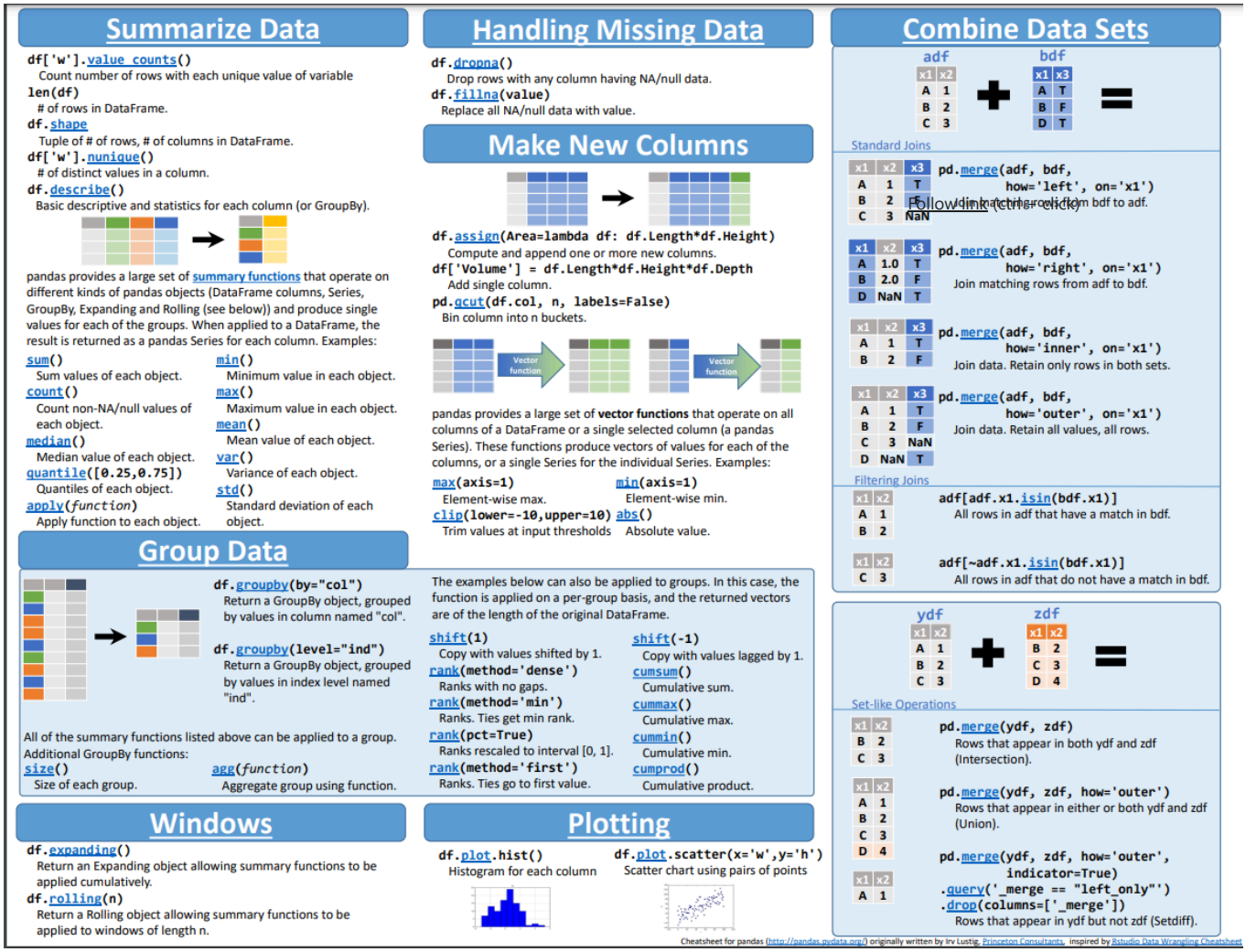
Logic in Python (and pandas)

<	Less than	!=	Not equal to
>	Greater than	df.column.isin(*values*)	Group membership
==	Equals	pd.isnull(*obj*)	Is NaN
<=	Less than or equals	pd.notnull(*obj*)	Is not NaN
>=	Greater than or equals	&,|,~,^,df.any(),df.all()	Logical and, or, not, xor, any, all

regex (Regular Expressions) Examples

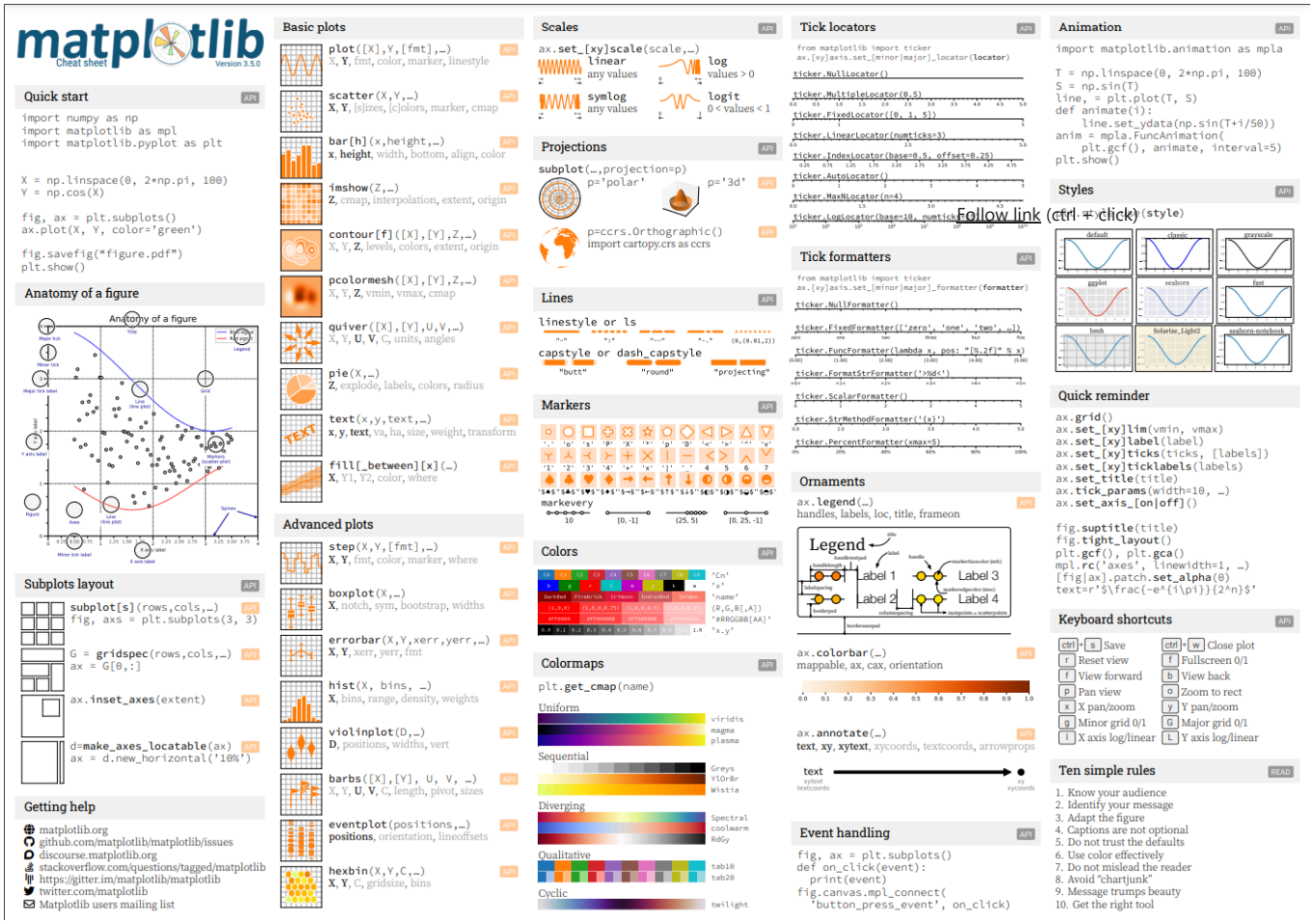
'\.'	Matches strings containing a period '.'
'Length$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
'^(?!Species$).*'	Matches strings except the string 'Species'

Cheatsheet for pandas (http://pandas.pydata.org/ originally written by Irv Lustig, Princeton Consultants, inspired by Rstudio Data Wrangling Cheatsheet

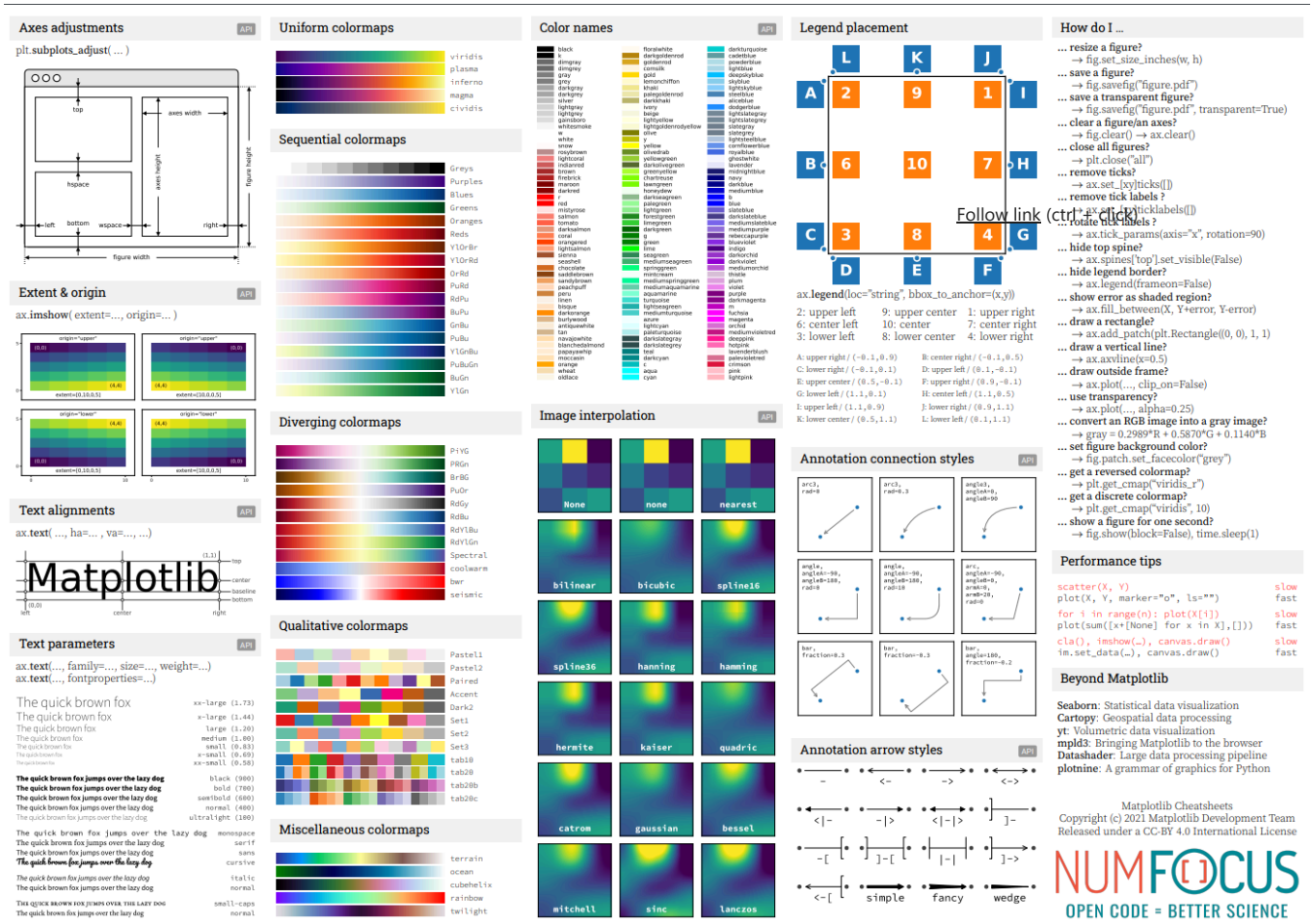


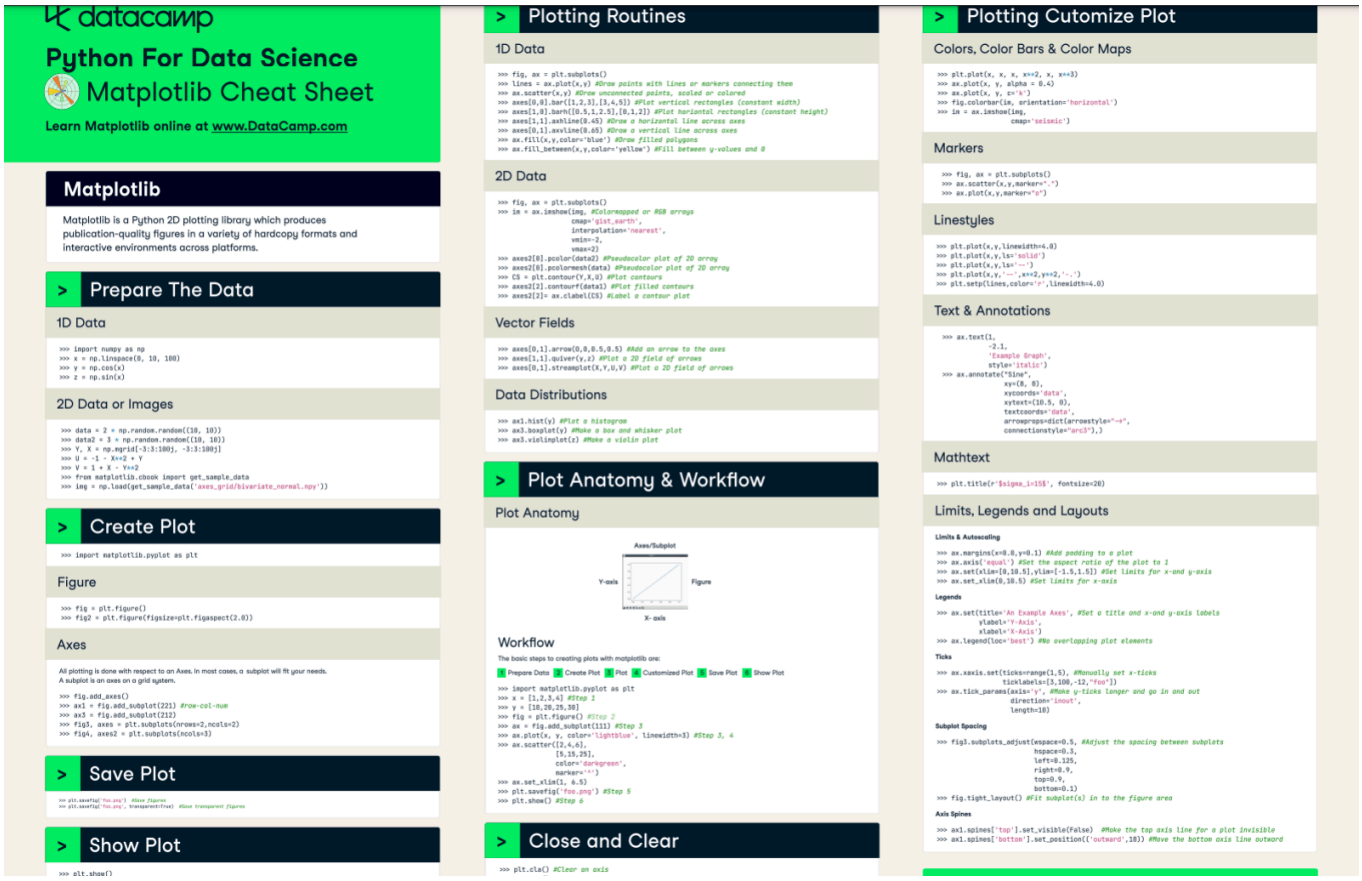
Data Visualization- Matplotlib

1. https://matplotlib.org/stable/users/index
2. https://matplotlib.org/cheatsheets/cheatsheets.pdf
3. https://images.datacamp.com/image/upload/v1676360378/Marketing/Blog/Matplotlib_Cheat_Sheet.pdf



Follow link (ctrl + click)


Axes adjustments
Uniform colormaps
Color names
Legend placement
How do I ...
Extent & origin
Sequential colormaps
Image interpolation
Text alignments
Diverging colormaps
Annotation connection styles
Text parameters
Qualitative colormaps
Annotation arrow styles
Miscellaneous colormaps
Performance tips
Beyond Matplotlib
Follow link (ctrl + click)
Matplotlib Cheatsheets
Copyright (c) 2021 Matplotlib Development Team
Released under a CC-BY 4.0 International License
NUMFOCUS
OPEN CODE = BETTER SCIENCE


datacamp
Python For Data Science
Matplotlib Cheat Sheet
Learn Matplotlib online at www.DataCamp.com
Matplotlib
Prepare The Data
Create Plot
Save Plot
Show Plot
Plotting Routines
Plot Anatomy & Workflow
Close and Clear
Plotting Cutomize Plot

Lab 1. Working with public datasets

1. Homework - https://www.kaggle.com/code/kashnitsky/a1-demo-pandas-and-uci-adult-dataset/notebook

Datasets

Follow link (ctrl + click)

Dataset Search by Google – allows searching not only by the keywords, but also filtering the results based on the types of the dataset that you want (e.g., tables, images, text), or based on whether the dataset is available for free from the provider.

Visual Data Discovery – specializes in Computer Vision datasets, all datasets are explicitly categorized and are easily filtered.

OpenML – as stated in the documentation section it's 'an open, collaborative, frictionless, automated machine learning environment'. This is a whole resource that allows not only sharing data, but also working on it collaboratively and solving problems in cooperation with other data scientists.

UCI: Machine Learning Repository – a collection of datasets and data generators, that is listed in the top 100 most quoted resources in Computer Science.

Awesome Public Datasets on Github- it would be weird if Github didn't have its own list of datasets, divided into categories.

Kaggle – one of the best, if not the best, resource for trying ML for yourself. Here you can also find data sets divided into categories with usability scores (an indicator that the dataset is well-documented).

Amazon Datasets – lots of datasets stored in S3, available for quick deployment if you're using AWS.

```
#https://drive.google.com/file/d/1DIGLFkdu3d77gXXwWZzSxwqDIrbXCxvn/view?usp=sharing
```

```
#Read dataset and show content
url = "https://drive.google.com/file/d/1DIGLFkdu3d77gXXwWZzSxwqDIrbXCxvn/view?usp=sharing"
#1DIGLFkdu3d77gXXwWZzSxwqDIrbXCxvn
data_url = "https://drive.google.com/uc?export=download&id=1DIGLFkdu3d77gXXwWZzSxwqDIrbXCxvn"
import pandas as pd
df = pd.read_csv(data_url)
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S

```
url_split= url.split("/")
url_split[-2]
```

```
'1DIGLFkdu3d77gXXwWZzSxwqDIrbXCxvn'
```

```
#Read dataset and show content
url = "https://drive.google.com/file/d/1DIGLFkdu3d77gXXwWZzSxwqDIrbXCxvn/view?usp=sharing"
import pandas as pd
df = pd.read_csv(url)
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	Follow link (ctrl + click) 113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

```
#Showing first n rows
df.head(15)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda	female	14.0	0	0	350406	7.8542	NaN	S

Follow link (ctrl + click)

```
df.tail(15)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
876	877	0	3	Gustafsson, Mr. Alfred Ossian	male	20.0	0	0	7534	9.8458	NaN	S
877	878	0	3	Petroff, Mr. Nedelio	male	19.0	0	0	349212	7.8958	NaN	S
878	879	0	3	Laleff, Mr. Kristo	male	NaN	0	0	349217	7.8958	NaN	S
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767 Follow link (ctrl + click)	83.1583	C50	C
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	230433	26.0000	NaN	S
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.5167	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
df['Sex'].value_counts()
```

```
male      577
female    314
Name: Sex, dtype: int64
```

```
df['Sex'].value_counts(normalize=True)
```

```
male      0.647587
female    0.352413
Name: Sex, dtype: float64
```

Follow link (ctrl + click)

```
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

Follow link (ctrl + click)

Follow link (ctrl + click)

Follow link (ctrl + click)

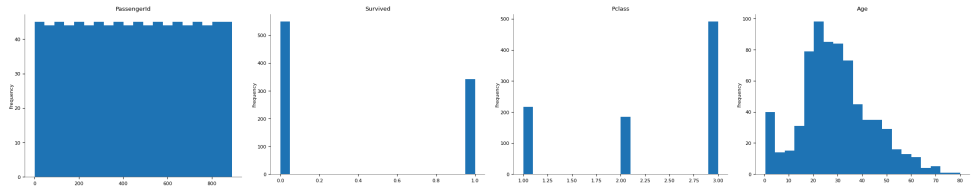
```
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

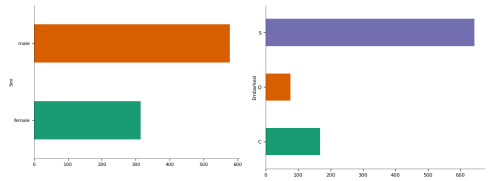
Follow link (ctrl + click)

891 rows × 12 columns

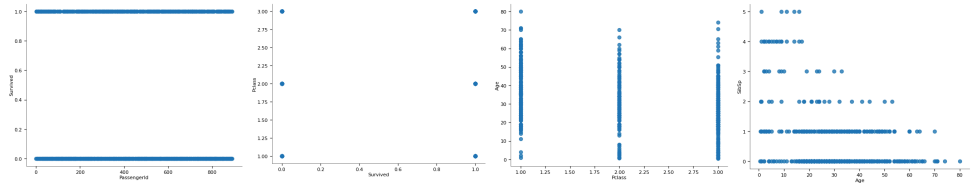
Distributions



Categorical distributions



2-d distributions



```
df['Age']
```

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

```
df[['Name','Age','Pclass']]
```

	Name	Age	Pclass
0	Braund, Mr. Owen Harris	22.0	3
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1
2	Heikkinen, Miss. Laina	26.0	3
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1
4	Allen, Mr. William Henry	35.0	3
...	...	...	...
886	Montvila, Rev. Juozas	27.0	2
887	Graham, Miss. Margaret Edith	19.0	1
888	Johnston, Miss. Catherine Helen "Carrie"	NaN	3
889	Behr, Mr. Karl Howell	26.0	1
890	Dooley, Mr. Patrick	32.0	3

891 rows × 3 columns

Follow link (ctrl + click)

```
df.iloc[20]
```

```
PassengerId                     21
Survived                         0
Pclass                           2
Name           Fynney, Mr. Joseph J
Sex                            male
Age                            35.0
SibSp                             0
Parch                             0
Ticket                       239865
Fare                           26.0
Cabin                           NaN
Embarked                          S
Name: 20, dtype: object
```

```
df.iloc[10:20]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	NaN	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	NaN	S
16	17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.1250	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S
18	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1	0	345763	18.0000	NaN	S

```
df.iloc[10:20,[1,5,6]]
```

	Survived	Age	SibSp
10	1	4.0	1
11	1	58.0	0
12	0	20.0	0
13	0	39.0	1
14	0	14.0	0
15	1	55.0	0
16	0	2.0	4
17	1	NaN	0
18	0	31.0	1
19	1	NaN	0

Follow link (ctrl + click)

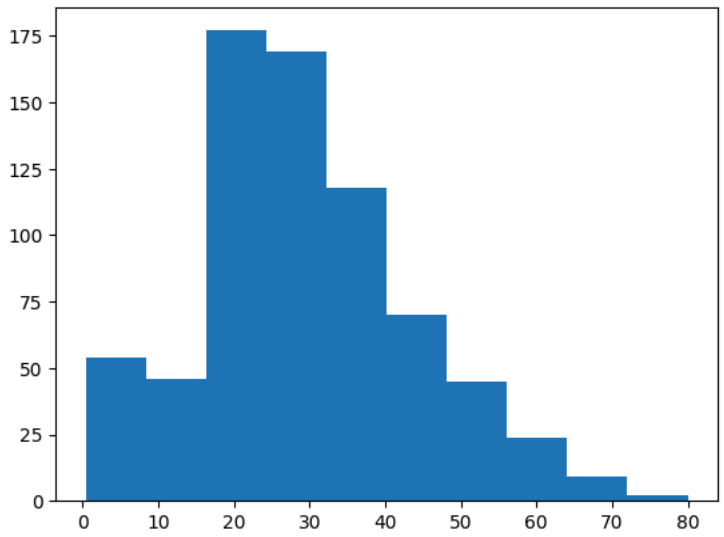
```
df.corr()
```

```
<ipython-input-46-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versio
  df.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

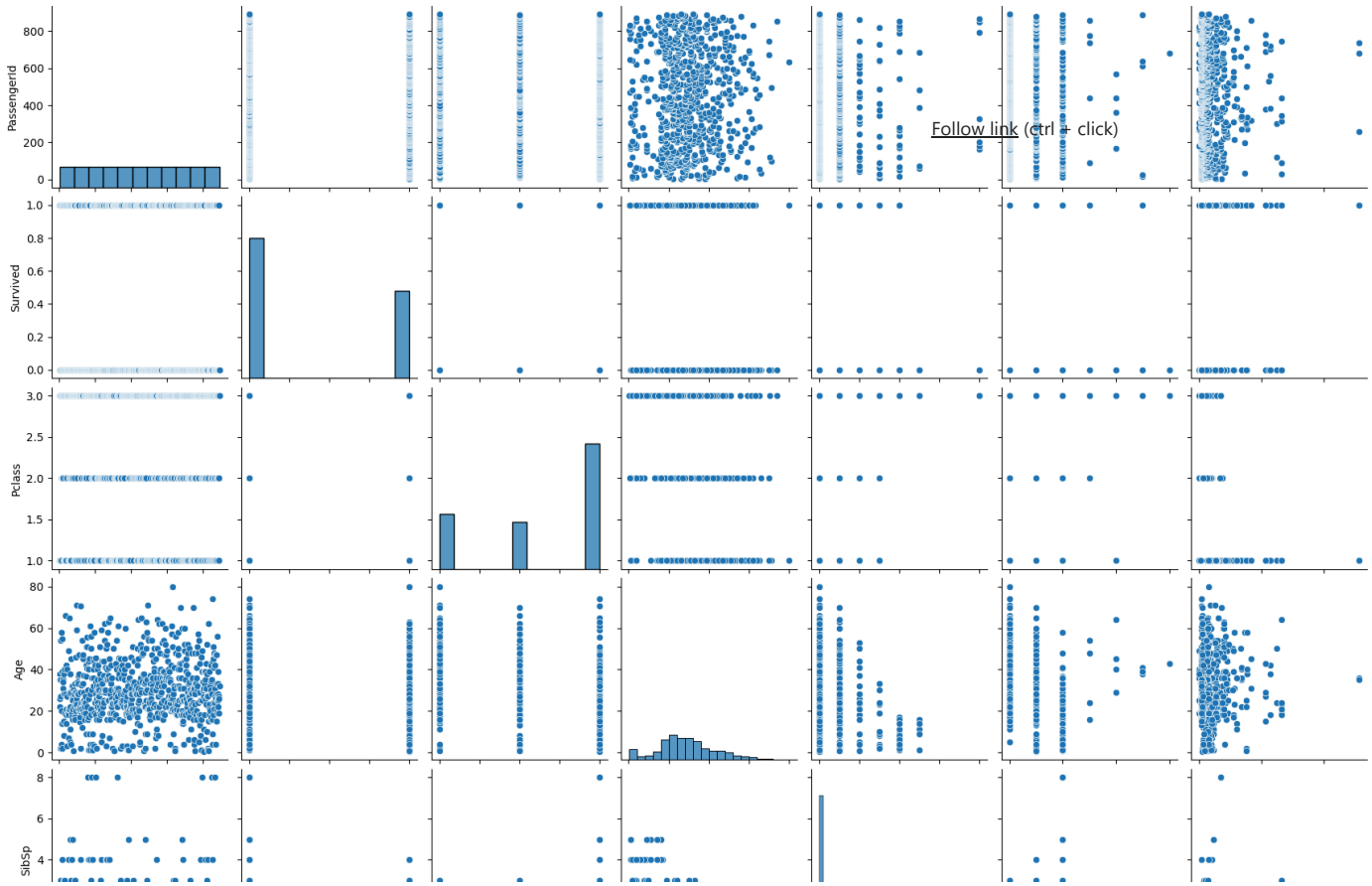
```
import matplotlib.pyplot as plt
plt.hist(df['Age'])
```

```
(array([ 54.,  46., 177., 169., 118.,  70.,  45.,  24.,   9.,   2.]),
 array([ 0.42 ,  8.378, 16.336, 24.294, 32.252, 40.21 , 48.168, 56.126,
        64.084, 72.042, 80.   ]),
 <BarContainer object of 10 artists>)
```



```
import seaborn as sns
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7ae85e2528c0>
```



Lab 1. Group A Working with public datasets

```
import pandas as pd
url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
df = pd.read_csv(url)
```

```
#Showing first 15 rows
df.head(15)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda	female	14.0	0	0	350406	7.8542	NaN	S

Follow link (ctrl + click)

```
#shwoing last 20 rows
df.tail(20)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	D35	S
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	B51 B53 B55	S
873	874	0	3	Vander Cruyssen, Mr. Victor	male	47.0	0	0	345765	9.0000	NaN	S
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	Follow link (ctrl + click) P/PP 3381	24.0000	NaN	C
875	876	1	3	Najib, Miss. Adele Kiamie "Jane"	female	15.0	0	0	2667	7.2250	NaN	C
876	877	0	3	Gustafsson, Mr. Alfred Ossian	male	20.0	0	0	7534	9.8458	NaN	S
877	878	0	3	Petroff, Mr. Nedelio	male	19.0	0	0	349212	7.8958	NaN	S
878	879	0	3	Laleff, Mr. Kristo	male	NaN	0	0	349217	7.8958	NaN	S
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	230433	26.0000	NaN	S
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.5167	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	Follow link (ctrl + click) 113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

Follow link (ctrl + click)

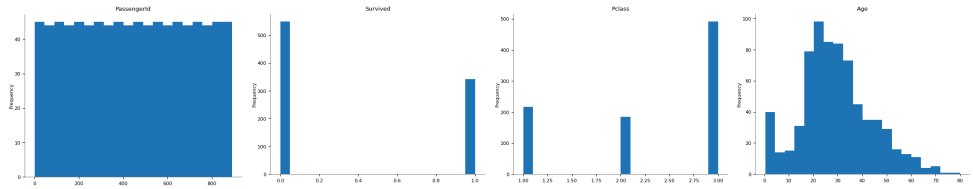
```
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

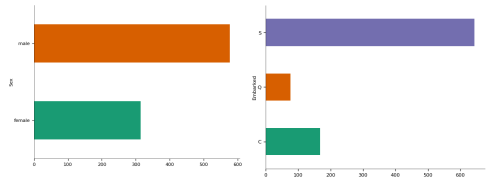
891 rows × 12 columns

Follow link (ctrl + click)

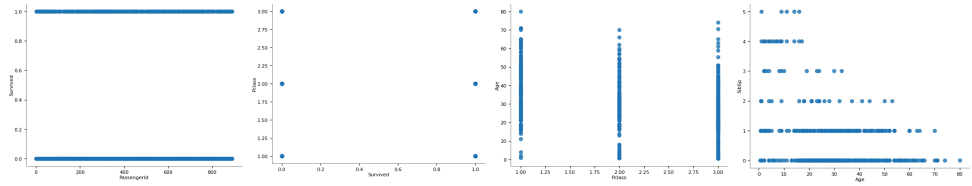
Distributions



Categorical distributions



2-d distributions



```
df['Name']
```

```
0                              Braund, Mr. Owen Harris
1    Cumings, Mrs. John Bradley (Florence Briggs Th...
2                               Heikkinen, Miss. Laina
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                             Allen, Mr. William Henry
                           ...
886                              Montvila, Rev. Juozas
887                       Graham, Miss. Margaret Edith
888            Johnston, Miss. Catherine Helen "Carrie"
889                              Behr, Mr. Karl Howell
890                                Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```



```
df['Sex'].value_counts()
```

```
male      577
female    314
Name: Sex, dtype: int64
```



```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Follow link (ctrl + click)

```
df.sample(20)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
481	482	0	2	Frost, Mr. Anthony Wood "Archie"	male	NaN	0	0	239854	0.0000	NaN	S
588	589	0	3	Gilinski, Mr. Eliezer	male	22.0	0	0	14973	8.0500	NaN	S
257	258	1	1	Cherry, Miss. Gladys	female	30.0	0	0	110152	86.5000	B77	S
252	253	0	1	Stead, Mr. William Thomas	male	62.0	0	0	113514	26.5500	C87	S
802	803	1	1	Carter, Master. William Thornton II	male	11.0	1	2	113760	120.0000	B96 B98	S
615	616	1	2	Herman, Miss. Alice	female	24.0	1	2	220845	65.0000	NaN	S
837	838	0	3	Sirota, Mr. Maurice	male	NaN	0	0	392092	8.0500	NaN	S
437	438	1	2	Richards, Mrs. Sidney (Emily Hocking)	female	24.0	2	3	29106	18.7500	NaN	S
531	532	0	3	Toufik, Mr. Nakli	male	NaN	0	0	2641	7.2292	NaN	C
338	339	1	3	Dahl, Mr. Karl Edwart	male	45.0	0	0	7598	8.0500	NaN	S
220	221	1	3	Sunderland, Mr. Victor Francis	male	16.0	0	0	SOTON/OQ 392089	8.0500	NaN	S
284	285	0	1	Smith, Mr. Richard William	male	NaN	0	0	113056	26.0000	A19	S
676	677	0	3	Sawyer, Mr. Frederick Charles	male	24.5	0	0	342826	8.0500	NaN	S
342	343	0	2	Collander, Mr. Erik Gustaf	male	28.0	0	0	248740	13.0000	NaN	S
521	522	0	3	Vovk, Mr. Janko	male	22.0	0	0	349252	7.8958	NaN	S
573	574	1	3	Kelly, Miss. Mary	female	NaN	0	0	14312	7.7500	NaN	Q
852	853	0	3	Boulos, Miss. Nourelain	female	9.0	1	1	2678	15.2458	NaN	C
187	188	1	1	Romaine, Mr. Charles Hallace ("Mr C Rolmane")	male	45.0	0	0	111428	26.5500	NaN	S
324	325	0	3	Sage, Mr. George John Jr	male	NaN	8	2	CA. 2343	69.5500	NaN	S

```
df[['Age','Name']]
```

	Age	Name
0	22.0	Braund, Mr. Owen Harris
1	38.0	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	26.0	Heikkinen, Miss. Laina
3	35.0	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	35.0	Allen, Mr. William Henry
...	...	...
886	27.0	Montvila, Rev. Juozas
887	19.0	Graham, Miss. Margaret Edith
888	NaN	Johnston, Miss. Catherine Helen "Carrie"
889	26.0	Behr, Mr. Karl Howell
890	32.0	Dooley, Mr. Patrick

891 rows × 2 columns

Follow link (ctrl + click)

```
df.iloc[10:30]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	NaN	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	NaN	S
16	17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.1250	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S
18	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1	0	345763	18.0000	NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
20	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.0000	NaN	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000	D56	S
22	23	1	3	McGowan, Miss. Anna "Annie"	female	15.0	0	0	330923	8.0292	NaN	Q
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S
24	25	0	3	Palsson, Miss. Torborg Danira	female	8.0	3	1	349909	21.0750	NaN	S
25	26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	female	38.0	1	5	347077	31.3875	NaN	S
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.2250	NaN	C
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	S
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.8792	NaN	Q
29	30	0	3	Todoroff, Mr. Lalio	male	NaN	0	0	349216	7.8958	NaN	S

```
df.iloc[:, [2,5,6]]
```

	Pclass	Age	SibSp
0	3	22.0	1
1	1	38.0	1
2	3	26.0	0
3	1	35.0	1
4	3	35.0	0
...	...	...	...
886	2	27.0	0
887	1	19.0	0
888	3	NaN	1
889	1	26.0	0
890	3	32.0	0

891 rows × 3 columns

Follow link (ctrl + click)

```
df.query('Age>30')
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	C103	S
...	...	...	...	...	...	...	...	...	...	...	...	...
873	874	0	3	Vander Cruyssen, Mr. Victor	male	47.0	0	0	345765	9.0000	NaN	S
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	C50	C
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

```
df.corr()
```

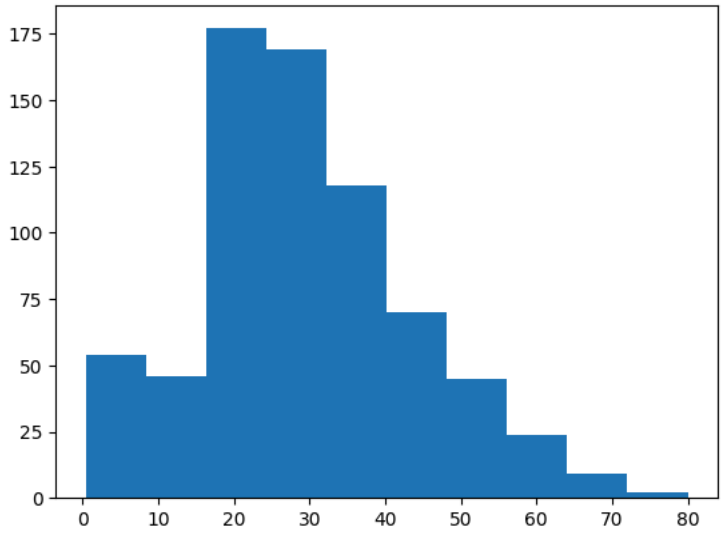
```
<ipython-input-26-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future versio
  df.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

Follow link (ctrl + click)

```
import matplotlib.pyplot as plt
plt.hist(df['Age'])
```

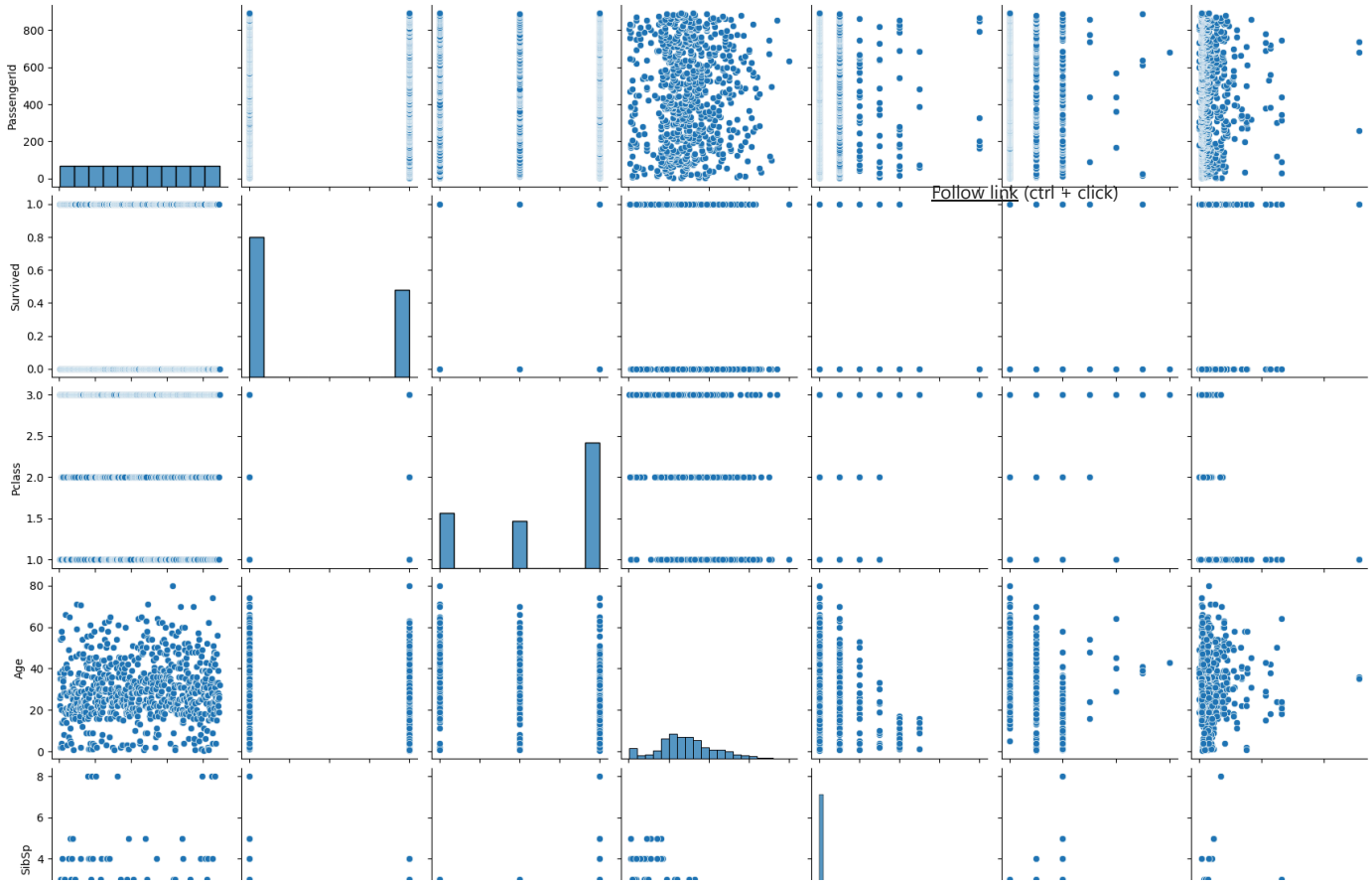
```
(array([ 54.,  46., 177., 169., 118.,  70.,  45.,  24.,   9.,   2.]),
 array([ 0.42 ,  8.378, 16.336, 24.294, 32.252, 40.21 , 48.168, 56.126,
        64.084, 72.042, 80.   ]),
 <BarContainer object of 10 artists>)
```




```
# prompt: seaborn pairplot

import seaborn as sns
sns.pairplot(df)
```

```
import seaborn as sb
sb.pairplot(df)
```

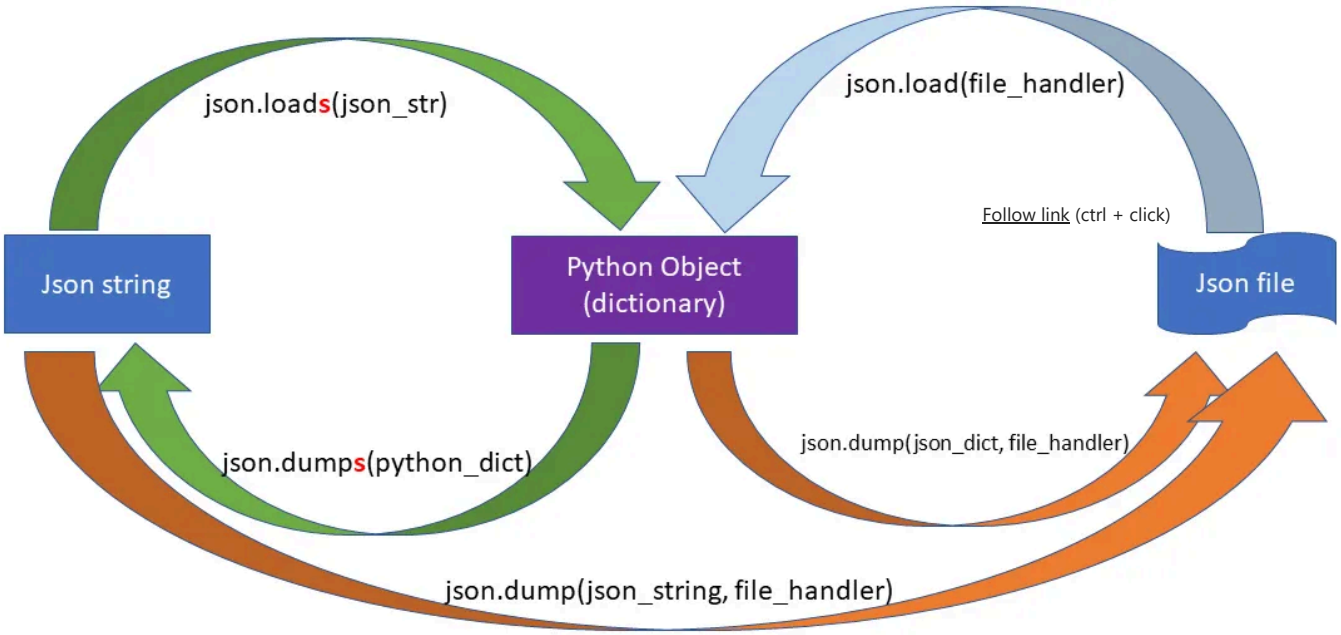
```
<seaborn.axisgrid.PairGrid at 0x7ae866df5060>
```

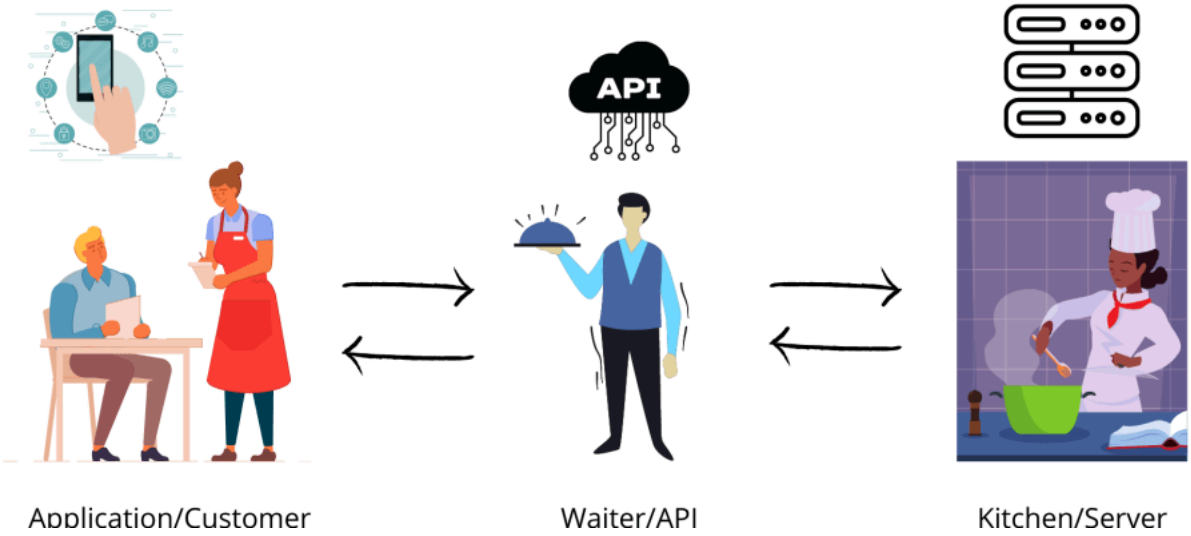


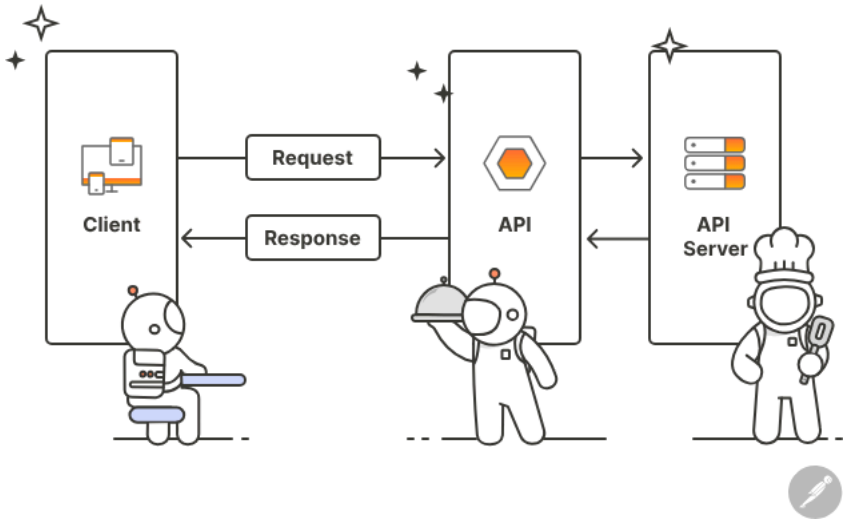
Follow link (ctrl + click)

Lab 2. Data Collection (API, JSON Parsing)

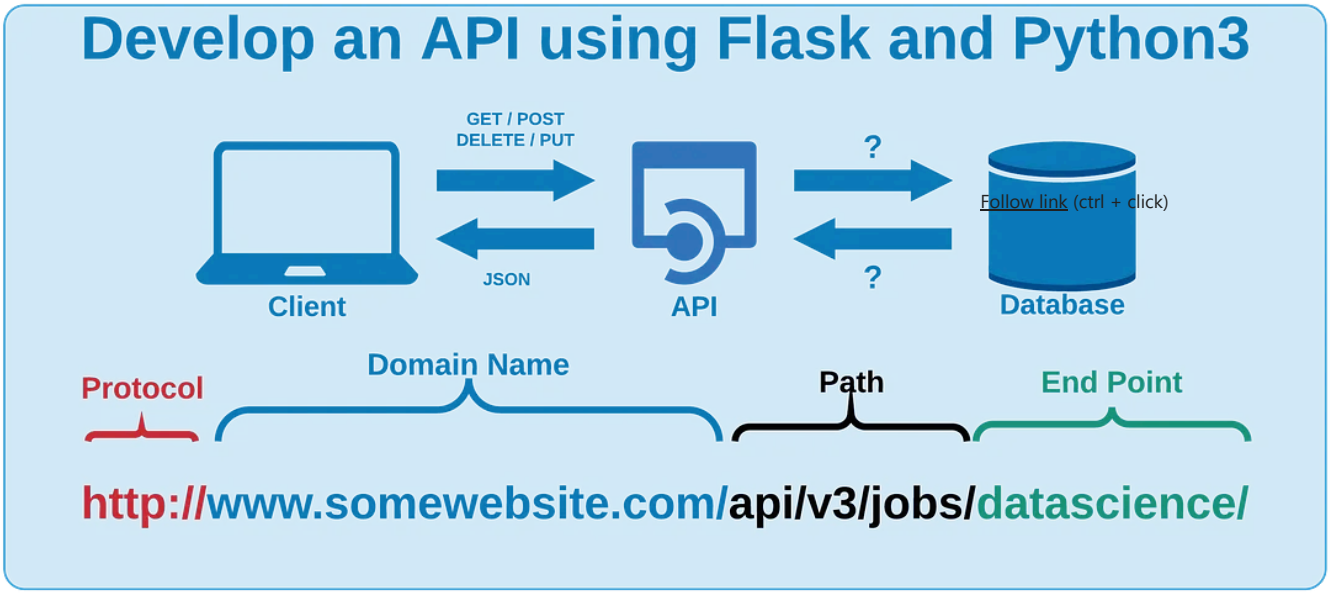
1. https://www.json.org/json-en.html
2. https://www.realpythonproject.com/a-cheat-sheet-for-working-with-json-data-in-python/
3. https://www.postman.com/what-is-an-api/
4. https://jsonplaceholder.typicode.com/todos/
5. https://realpython.com/api-integration-in-python/
6. https://www.dataquest.io/blog/python-api-tutorial/
7. https://jsoneditoronline.org/#left=local.vadace&right=local.bigofo
8. https://any-api.com/
9. https://rapidapi.com/hub







Double-click (or enter) to edit



Read the content of JSON file

```
import pandas
```

Lab 2

```
import pandas as pd
file_url = "/content/sample_data/anscombe.json"
df = pd.read_json(file_url)
df
```

	Series	X	Y
0	I	10	8.04
1	I	8	6.95
2	I	13	7.58
3	I	9	8.81
4	I	11	8.33
5	I	14	9.96
6	I	6	7.24
7	I	4	4.26
8	I	12	10.84
9	I	7	4.81
10	I	5	5.68
11	II	10	9.14
12	II	8	8.14
13	II	13	8.74
14	II	9	8.77
15	II	11	9.26
16	II	14	8.10
17	II	6	6.13
18	II	4	3.10
19	II	12	9.13
20	II	7	7.26
21	II	5	4.74
22	III	10	7.46
23	III	8	6.77
24	III	13	12.74
25	III	9	7.11
26	III	11	7.81
27	III	14	8.84
28	III	6	6.08
29	III	4	5.39
30	III	12	8.15

Follow link (ctrl + click)

```
import requests
api_url = "https://jsonplaceholder.typicode.com/todos/"
response = requests.get(api_url)
response
```

```
<Response [200]>
```

```
print(response.status_code)
```

```
200
```

```
import requests
api_url = "https://jsonplaceholder.typicode.com/todos/1"
response = requests.get(api_url)
response
```

```
<Response [200]>
```

```
print(response.status_code)
```

```
200
```

```
print(response.content)
```

```
b'[\n  {\n    "userId": 1,\n    "id": 1,\n    "title": "delectus aut autem",\n    "completed": false\n  },\n  {\n    "userId": 1,\n
```

Follow link (ctrl + click)

```
print(response.json())
```

```
[{'userId': 1, 'id': 1, 'title': 'delectus aut autem', 'completed': False}, {'userId': 1, 'id': 2, 'title': 'quis ut nam facilis et off
```

```
import requests

url = "https://accuweatherstefan-skliarovv1.p.rapidapi.com/getCurrentConditionsForTopCities"

headers = {
    "X-RapidAPI-Key": "90259ec543mshc0489c0f1fd8a95p1c6e5djsnfa79bbdbc717",
    "X-RapidAPI-Host": "AccuWeatherstefan-skliarovV1.p.rapidapi.com"
}

response = requests.post(url, headers=headers)

print(response.json())
```

```
{'messages': 'The API is unreachable, please contact the API provider', 'info': 'Your Client (working) ---> Gateway (working) ---> API
```

```
print(response.content)
```

```
b'{\n  "userId": 1,\n  "id": 1,\n  "title": "delectus aut autem",\n  "completed": false\n}'
```

```
import requests
api_url = "https://jsonplaceholder.typicode.com/todos/1"
response = requests.get(api_url)
response.json()
```

```
{'userId': 1, 'id': 1, 'title': 'delectus aut autem', 'completed': False}
```

```
response.json()
```

```
{'userId': 1, 'id': 1, 'title': 'delectus aut autem', 'completed': False}
```

```
import requests

url = "https://accuweatherstefan-skliarovv1.p.rapidapi.com/getCurrentConditionsForTopCities"

headers = {
    "X-RapidAPI-Key": "90259ec543mshc0489c0f1fd8a95p1c6e5djsnfa79bbdbc717",
    "X-RapidAPI-Host": "AccuWeatherstefan-skliarovV1.p.rapidapi.com"
}

response = requests.post(url, headers=headers)

print(response.json())
```

```
{'messages': 'The API is unreachable, please contact the API provider', 'info': 'Your Client (working) ---> Gateway (working) ---> API
```

Lab3. Data Engineering-Web Scraping (Beautiful Soup)

- https://pypi.org/project/beautifulsoup4/
- https://realpython.com/beautiful-soup-web-scraper-python/
- https://realpython.github.io/fake-jobs/

- https://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
import requests
url = "https://realpython.github.io/fake-jobs/"
data = requests.get(url)
data
```

```
<Response [200]>
```

Follow link (ctrl + click)

```
print(data.text)
```

```
from bs4 import BeautifulSoup
```

```
url = "https://realpython.github.io/fake-jobs/"
page = requests.get(url)
```

```
soup = BeautifulSoup(page.content,"html.parser")
```

```
type(soup)
```

```
bs4.BeautifulSoup
```

```
results = soup.find(id="ResultsContainer")
results
```

```
type(results)
```

```
bs4.element.Tag
```

```
job_elements = results.findAll("div", class_="card-content")
```

```
job_elements
```

```
type(job_elements)
```

```
bs4.element.ResultSet
```

```
for item in job_elements:
  title_element = item.find("h2",class_="title")
  company_element = item.find("h3", class_="subtitle")
  location_element = item.find("p", class_="location")
  print(title_element)
  print(company_element)
  print(location_element)
  print()
```

```
for item in job_elements:
  title_element = item.find("h2",class_="title")
  company_element = item.find("h3", class_="subtitle")
  location_element = item.find("p", class_="location")
  print(title_element.text)
  print(company_element.text)
  print(location_element.text)
  print()
```

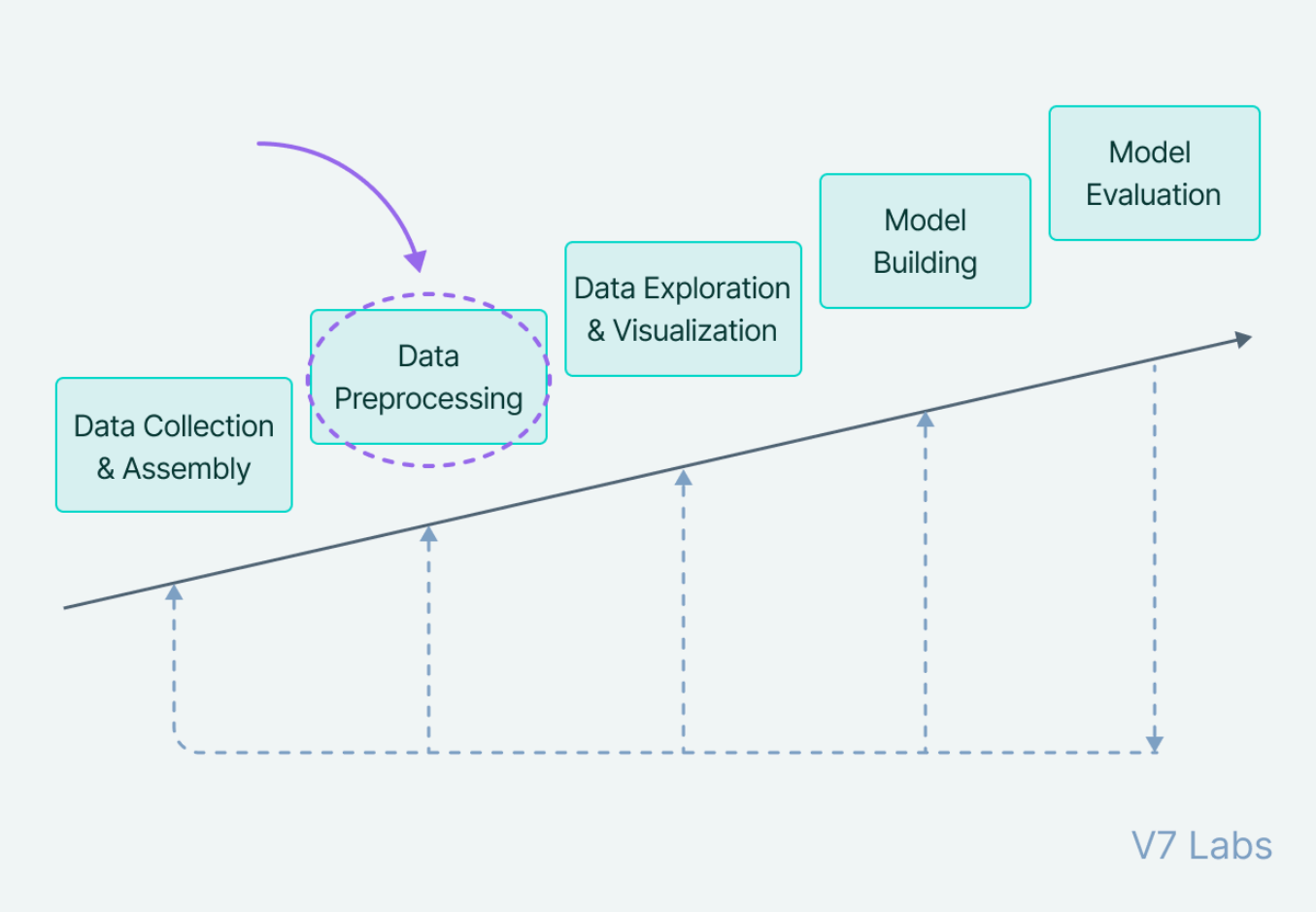
```
for item in job_elements:
  title_element = item.find("h2",class_="title")
  company_element = item.find("h3", class_="subtitle")
  location_element = item.find("p", class_="location")
  print(title_element.text.strip())
  print(company_element.text.strip())
  print(location_element.text.strip())
  print()
```

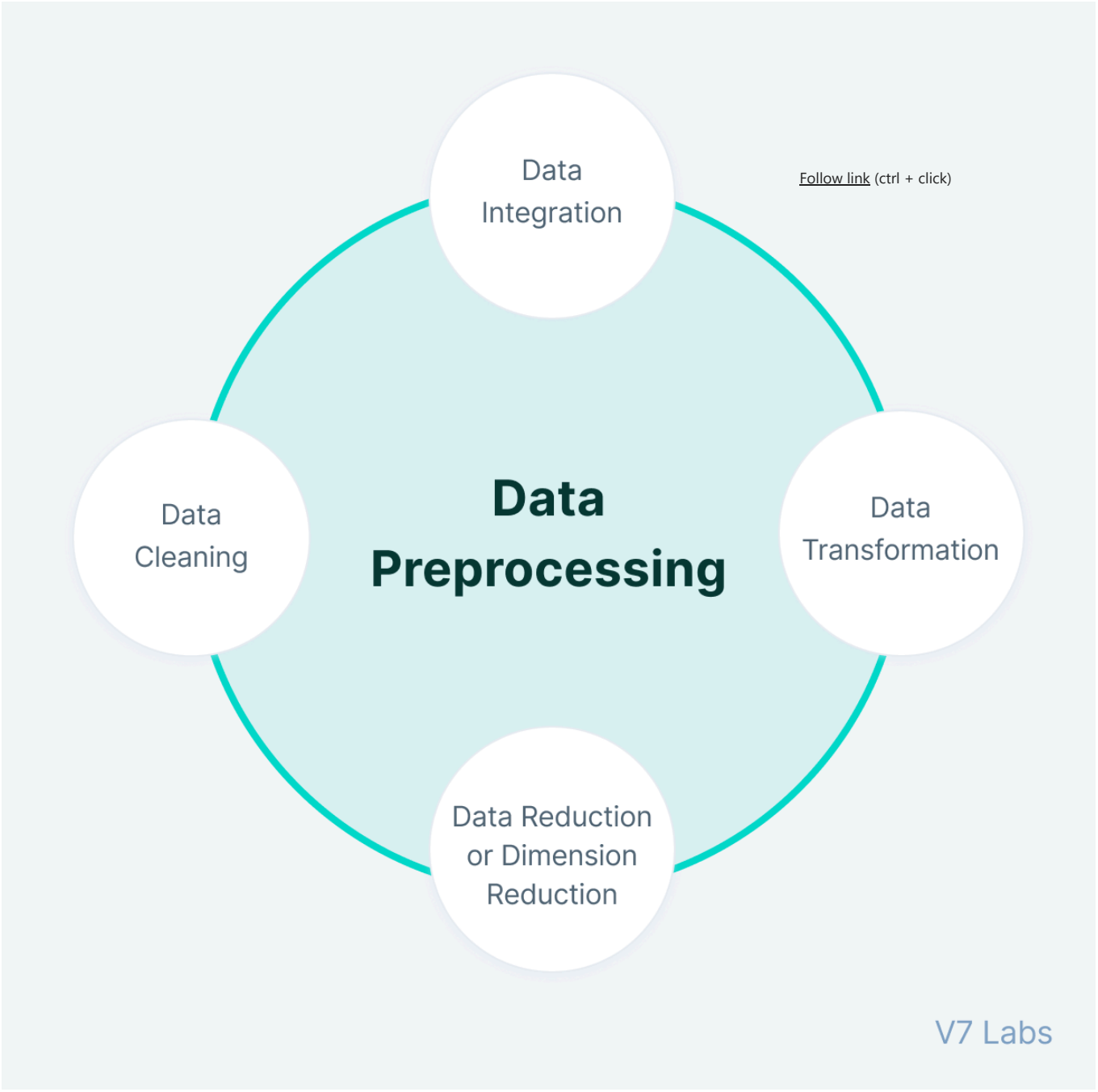
Lab 4. Data Preprocessing

1. https://www.w3schools.com/python/pandas/pandas_cleaning.asp
2. https://neptune.ai/blog/data-preprocessing-guide
3. https://www.v7labs.com/blog/data-preprocessing-guide
4. https://www.javatpoint.com/data-preprocessing-machine-learning
5. https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/
6. https://www.jcchouinard.com/preprocessing-in-scikit-learn/

Follow link (ctrl + click)

Data Preprocessing includes the steps we need to follow to transform or encode data so that it may be easily parsed by the machine.





What is handled in data preprocessing:

Missing data: Remove, fix and impute missing data

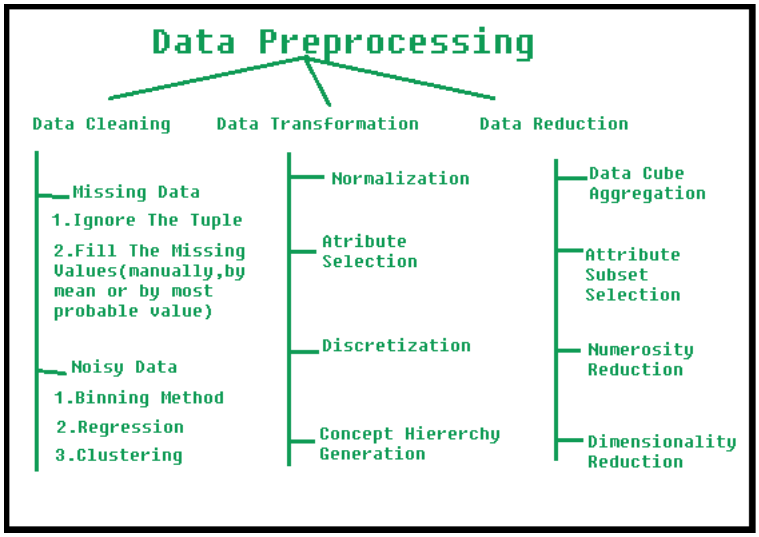
Feature engineering: Infer additional features from raw data

Data formatting: The data might not be in the format that you need. For example, the Scikit-learn API requires the data to be a Numpy array or a pandas DataFrame.

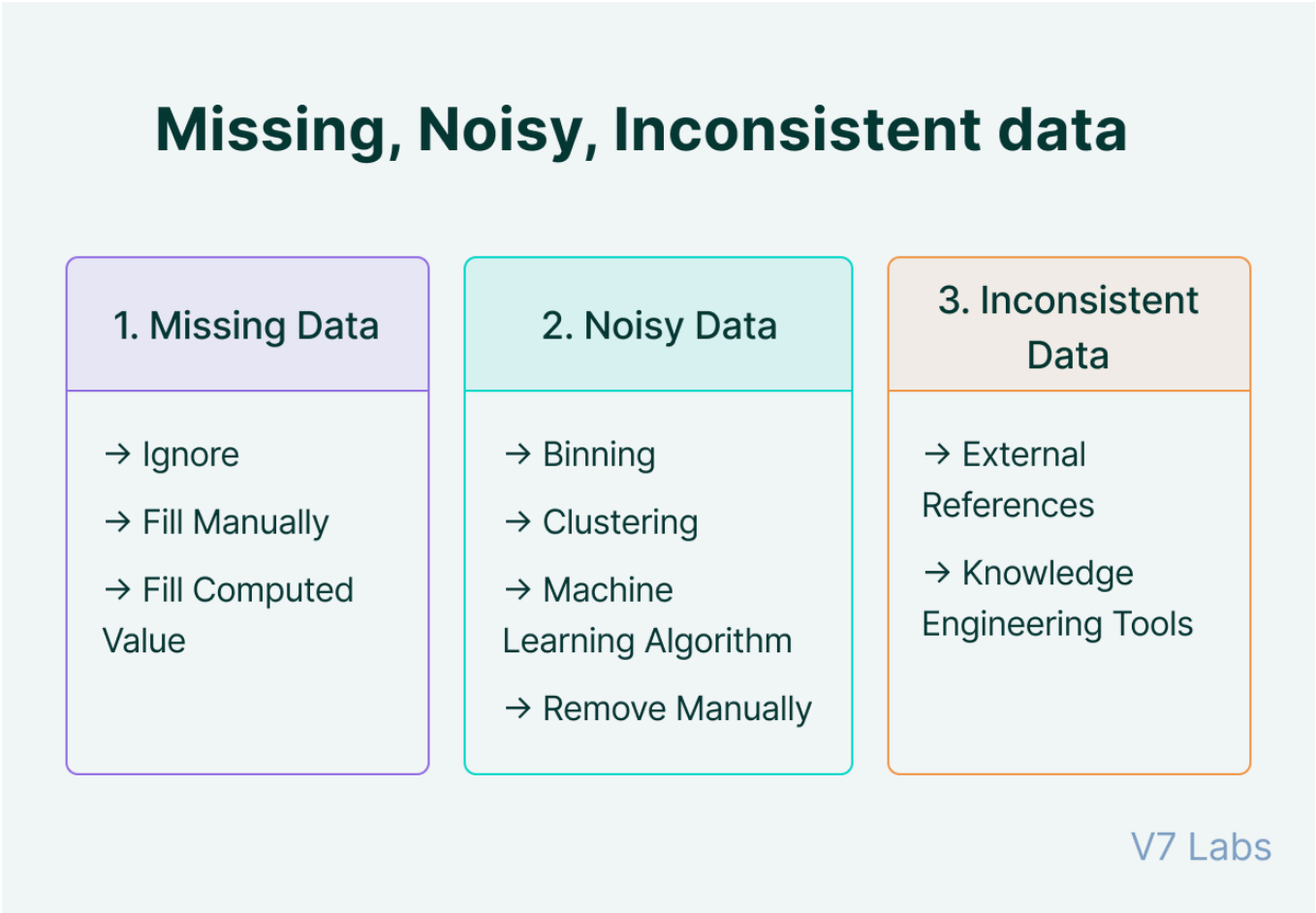
Scaling the data: The data may not all be on the same scale. For instance, kilograms and pounds can be put on same scale from 0 to 1.

Decomposition:i.e. keep only hour of the day from datetime

Aggregation: i.e. aggregate by loggedin vs non logged in



Follow link (ctrl + click)



Standardization in sklearn

StandardScaler: sklearn scaler, scaling values to be centered around mean 0 with standard deviation 1 - .fit(), .transform()

Follow link (ctrl + click)

Transform: $x_{scaled} = \frac{x - x_{mean}}{x_{std}}$

```
from sklearn.preprocessing import StandardScaler
stdsc = StandardScaler()

raw_data = np.array([[-3.4], [4.5], [50], [24], [3.4], [1.6]])
scaled_data = stdsc.fit_transform(raw_data)
print(scaled_data.reshape(1,-1))
```

```
[[-0.90560498 -0.47848383  1.98151777  0.57580257 -0.53795639 -0.63527514]]
```

MinMax Scaling in sklearn

MinMaxScaler: sklearn scaler, scaling values so that minimum value is 0 and maximum value is 1 - .fit(), .transform()

Transform: $x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$

```
from sklearn.preprocessing import MinMaxScaler
minmaxsc = MinMaxScaler()

raw_data = np.array([[-3.4], [4.5], [50], [24], [3.4], [1.6]])
scaled_data = minmaxsc.fit_transform(raw_data)
print(scaled_data.reshape(1,-1))
```

```
[[0.         0.14794007 1.         0.51310861 0.12734082 0.09363296]]
```

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing

```
import pandas as pd
from sklearn.datasets import fetch_openml
df = fetch_openml('titanic', version=1, as_frame=True)['data']
df.head(5)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     1309 non-null   float64
 1   name       1309 non-null   object
 2   sex        1309 non-null   category
```

```
 3   age        1046 non-null   float64
 4   sibsp      1309 non-null   float64
 5   parch      1309 non-null   float64
 6   ticket     1309 non-null   object
 7   fare       1308 non-null   float64
 8   cabin      295 non-null    object
 9   embarked   1307 non-null   category
 10  boat       486 non-null    object
 11  body       121 non-null    object
 12  home.dest  745 non-null    object
dtypes: category(2), float64(5), object(6)
memory usage: 115.4+ KB
```

Follow link (ctrl + click)

```
df.isnull().sum()
```

```
pclass          0
name            0
sex             0
age           263
sibsp           0
parch           0
ticket          0
fare            1
cabin        1014
embarked        2
boat          823
body         1188
home.dest     564
dtype: int64
```

```
df_clean = df.dropna()
```

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 0 entries
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     0 non-null      float64
 1   name       0 non-null      object
 2   sex        0 non-null      category
 3   age        0 non-null      float64
 4   sibsp      0 non-null      float64
 5   parch      0 non-null      float64
 6   ticket     0 non-null      object
 7   fare       0 non-null      float64
 8   cabin      0 non-null      object
 9   embarked   0 non-null      category
 10  boat       0 non-null      object
 11  body       0 non-null      object
 12  home.dest  0 non-null      object
dtypes: category(2), float64(5), object(6)
memory usage: 256.0+ bytes
```

```
df_clean = df.dropna(subset=['age'])
```

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1046 entries, 0 to 1308
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     1046 non-null   float64
 1   name       1046 non-null   object
 2   sex        1046 non-null   category
 3   age        1046 non-null   float64
 4   sibsp      1046 non-null   float64
 5   parch      1046 non-null   float64
 6   ticket     1046 non-null   object
 7   fare       1045 non-null   float64
 8   cabin      272 non-null    object
 9   embarked   1044 non-null   category
 10  boat       417 non-null    object
 11  body       120 non-null    object
 12  home.dest  685 non-null    object
dtypes: category(2), float64(5), object(6)
```

```
memory usage: 100.4+ KB
```

```
df_clean.isnull().sum()
```

```
pclass         0
name           0
sex            0
age            0
sibsp          0
parch          0
ticket         0
fare           1
cabin        774
embarked       2
boat         629
body         926
home.dest    361
dtype: int64
```

```
df_clean['age'].isnull().sum()
```

```
0
```

```
df.isnull().sum()
```

```
pclass          0
name            0
sex             0
age           263
sibsp           0
parch           0
ticket          0
fare            1
cabin        1014
embarked        2
boat          823
body         1188
home.dest     564
dtype: int64
```

```
df.dropna(subset = ['age'], inplace=True)
```

Follow link (ctrl + click)

```
Start coding or generate with AI.
```

```
df.select_dtypes('number').head()
```

	pclass	age	sibsp	parch	fare
0	1.0	29.0000	0.0	0.0	211.3375
1	1.0	0.9167	1.0	2.0	151.5500
2	1.0	2.0000	1.0	2.0	151.5500
3	1.0	30.0000	1.0	2.0	151.5500
4	1.0	25.0000	1.0	2.0	151.5500

```
df.select_dtypes('category').head()
```

	sex	embarked
0	female	S
1	male	S
2	female	S
3	male	S
4	female	S

```
df['parch'] = df['parch'].astype(int)
df['sibsp'] = df['sibsp'].astype(int)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     1309 non-null   float64
 1   name       1309 non-null   object
 2   sex        1309 non-null   category
 3   age        1046 non-null   float64
 4   sibsp      1309 non-null   int64
 5   parch      1309 non-null   int64
 6   ticket     1309 non-null   object
 7   fare       1308 non-null   float64
 8   cabin      295 non-null    object
 9   embarked   1307 non-null   category
 10  boat       486 non-null    object
 11  body       121 non-null    object
 12  home.dest  745 non-null    object
dtypes: category(2), float64(3), int64(2), object(6)
memory usage: 115.4+ KB
```

Follow link (ctrl + click)

```
df['age'].isnull().sum()
```

```
263
```

```
df['age'].mean()
```

```
29.8811345124283
```

```
df['age'].fillna(df['age'].mean,inplace = True)
```

```
df['age'].isnull().sum()
```

```
0
```

Start coding or generate with AI.

Start coding or generate with AI.

```
#W3school
dirty_data_url = "https://raw.githubusercontent.com/sharadkhare/Complete-Python-Pandas-Tutorial-in-Hindi-With-Notes-/main/dirtydata.csv"
```

```
import pandas as pd
from sklearn.datasets import fetch_openml
df = fetch_openml('titanic', version = 1, as_frame = True)['data']
df.head(5)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The default value of `parser` will change from
  warn(
```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	None	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON
3	1.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	135.0	Montreal, PQ / Chesterville, ON
4	1.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	S	None	NaN	Montreal, PQ / Chesterville, ON

```
#Titanic
import pandas as pd
from sklearn.datasets import fetch_openml
df = fetch_openml('titanic', version=1, as_frame=True)['data']
df.head(3)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The default value of `parser` will change from
  warn(
```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	None	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON

Follow link (ctrl + click)

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     1309 non-null   float64
 1   name       1309 non-null   object
 2   sex        1309 non-null   category
 3   age        1046 non-null   float64
 4   sibsp      1309 non-null   float64
 5   parch      1309 non-null   float64
 6   ticket     1309 non-null   object
 7   fare       1308 non-null   float64
 8   cabin      295 non-null    object
 9   embarked   1307 non-null   category
 10  boat       486 non-null    object
 11  body       121 non-null    object
 12  home.dest  745 non-null    object
dtypes: category(2), float64(5), object(6)
memory usage: 115.4+ KB
```

```
df['sibsp']=df['sibsp'].astype(int)
df['parch']=df['parch'].astype(int)
```

```
df.info()
```

```
df.select_dtypes('number').head()
```

	pclass	age	sibsp	parch	fare
0	1.0	29.0000	0	0	211.3375
1	1.0	0.9167	1	2	151.5500
2	1.0	2.0000	1	2	151.5500
3	1.0	30.0000	1	2	151.5500
4	1.0	25.0000	1	2	151.5500

```
df.select_dtypes('category').head()
```

	sex	embarked
0	female	S
1	male	S
2	female	S
3	male	S
4	female	S

Follow link (ctrl + click)

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     1309 non-null   float64
 1   name       1309 non-null   object
 2   sex        1309 non-null   category
 3   age        1046 non-null   float64
 4   sibsp      1309 non-null   int64
 5   parch      1309 non-null   int64
 6   ticket     1309 non-null   object
 7   fare       1308 non-null   float64
 8   cabin      295 non-null    object
 9   embarked   1307 non-null   category
 10  boat       486 non-null    object
 11  body       121 non-null    object
 12  home.dest  745 non-null    object
dtypes: category(2), float64(3), int64(2), object(6)
memory usage: 115.4+ KB
```

```
df.isnull().sum()
```

```
pclass          0
name            0
sex             0
age           263
sibsp           0
parch           0
ticket          0
fare            1
cabin        1014
embarked        2
boat          823
body         1188
home.dest     564
dtype: int64
```

```
df_clean = df.dropna()
```

```
df_clean.isnull().sum()
```

```
pclass       0.0
name         0.0
sex          0.0
age          0.0
sibsp        0.0
parch        0.0
ticket       0.0
fare         0.0
cabin        0.0
embarked     0.0
boat         0.0
body         0.0
home.dest    0.0
dtype: float64
```

```
df.isnull().sum()
```

```
pclass          0
name            0
sex             0
age           263
sibsp           0
parch           0
```

```
ticket          0
fare            1
cabin        1014
embarked        2
boat          823
body         1188
home.dest     564
dtype: int64
```

Follow link (ctrl + click)

```
df.dropna(inplace = True)
```

```
df.isnull().sum()
```

```
pclass       0.0
name         0.0
sex          0.0
age          0.0
sibsp        0.0
parch        0.0
ticket       0.0
fare         0.0
cabin        0.0
embarked     0.0
boat         0.0
body         0.0
home.dest    0.0
dtype: float64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 0 entries
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     0 non-null      float64
 1   name       0 non-null      object
 2   sex        0 non-null      category
 3   age        0 non-null      float64
 4   sibsp      0 non-null      int64
 5   parch      0 non-null      int64
 6   ticket     0 non-null      object
 7   fare       0 non-null      float64
 8   cabin      0 non-null      object
 9   embarked   0 non-null      category
 10  boat       0 non-null      object
 11  body       0 non-null      object
 12  home.dest  0 non-null      object
dtypes: category(2), float64(3), int64(2), object(6)
memory usage: 256.0+ bytes
```

```
#Titanic
import pandas as pd
from sklearn.datasets import fetch_openml
df = fetch_openml('titanic', version=1, as_frame=True)['data']
df.head(3)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The default value of `parser` will change from
  warn(
```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	S	2	None	St Louis, MO
1	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON

```
df=df.dropna(subset=['age'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1046 entries, 0 to 1308
```

```
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     1046 non-null   float64
 1   name       1046 non-null   object
 2   sex        1046 non-null   category
 3   age        1046 non-null   float64
 4   sibsp      1046 non-null   float64
 5   parch      1046 non-null   float64
 6   ticket     1046 non-null   object
 7   fare       1045 non-null   float64
 8   cabin      272 non-null    object
 9   embarked   1044 non-null   category
 10  boat       417 non-null    object
 11  body       120 non-null    object
 12  home.dest  685 non-null    object
dtypes: category(2), float64(5), object(6)
memory usage: 100.4+ KB
```

Follow link (ctrl + click)

```
# prompt: pandas dropna for specific column

df = df.dropna(subset=['age'])
```

Lab 5. Intro to Scikit-learn

1. https://scikit-learn.org/stable/user_guide.html
2. https://images.datacamp.com/image/upload/v1676302389/Marketing/Blog/Scikit-Learn_Cheat_Sheet.pdf
3. https://scikit-learn.org/stable/datasets.html
4. https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html#sphx-glr-auto-examples-datasets-plot-iris-dataset-py

A

Feature vector of the 1st training example

Class label

Index	Sepal length	Sepal width	Petal length	Petal width	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
...	...	...	...	...	...
150	5.9	3	5.1	1.8	Iris-virginica

B


```
from sklearn.datasets import load_iris
data = load_iris()
```

```
type(data)
```

```
sklearn.utils._bunch.Bunch
```

```
#Features matric
#Data
data.data
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
```

```
            [5.4, 3.9, 1.7, 0.4],
            [4.6, 3.4, 1.4, 0.3],
            [5. , 3.4, 1.5, 0.2],
            [4.4, 2.9, 1.4, 0.2],
            [4.9, 3.1, 1.5, 0.1],
            [5.4, 3.7, 1.5, 0.2],
            [4.8, 3.4, 1.6, 0.2],
            [4.8, 3. , 1.4, 0.1],
            [4.3, 3. , 1.1, 0.1],
            [5.8, 4. , 1.2, 0.2],
            [5.7, 4.4, 1.5, 0.4],
            [5.4, 3.9, 1.3, 0.4],
            [5.1, 3.5, 1.4, 0.3],
            [5.7, 3.8, 1.7, 0.3],
            [5.1, 3.8, 1.5, 0.3],
            [5.4, 3.4, 1.7, 0.2],
            [5.1, 3.7, 1.5, 0.4],
            [4.6, 3.6, 1. , 0.2],
            [5.1, 3.3, 1.7, 0.5],
            [4.8, 3.4, 1.9, 0.2],
            [5. , 3. , 1.6, 0.2],
            [5. , 3.4, 1.6, 0.4],
            [5.2, 3.5, 1.5, 0.2],
            [5.2, 3.4, 1.4, 0.2],
            [4.7, 3.2, 1.6, 0.2],
            [4.8, 3.1, 1.6, 0.2],
            [5.4, 3.4, 1.5, 0.4],
            [5.2, 4.1, 1.5, 0.1],
            [5.5, 4.2, 1.4, 0.2],
            [4.9, 3.1, 1.5, 0.2],
            [5. , 3.2, 1.2, 0.2],
            [5.5, 3.5, 1.3, 0.2],
            [4.9, 3.6, 1.4, 0.1],
            [4.4, 3. , 1.3, 0.2],
            [5.1, 3.4, 1.5, 0.2],
            [5. , 3.5, 1.3, 0.3],
            [4.5, 2.3, 1.3, 0.3],
            [4.4, 3.2, 1.3, 0.2],
            [5. , 3.5, 1.6, 0.6],
            [5.1, 3.8, 1.9, 0.4],
            [4.8, 3. , 1.4, 0.3],
            [5.1, 3.8, 1.6, 0.2],
            [4.6, 3.2, 1.4, 0.2],
            [5.3, 3.7, 1.5, 0.2],
            [5. , 3.3, 1.4, 0.2],
            [7. , 3.2, 4.7, 1.4],
            [6.4, 3.2, 4.5, 1.5],
            [6.9, 3.1, 4.9, 1.5],
            [5.5, 2.3, 4. , 1.3],
            [6.5, 2.8, 4.6, 1.5],
            [5.7, 2.8, 4.5, 1.3],
            [6.3, 3.3, 4.7, 1.6],
            [4.9, 2.4, 3.3, 1. ],
```

Follow link (ctrl + click)

```
#Feature names
data.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

```
#Target
data.target
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
type(data.target)
```

```
numpy.ndarray
```

```
#Target Names
data.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
array = np.array(['France', 'Spain', 'Germany', 'Spain', 'Germany', 'France',
       'Spain', 'France', 'Germany', 'France'])
# Transform the array
transformed_array = le.fit_transform(array)
print(transformed_array)
```

Follow link (ctrl + click)

```
[0 2 1 2 1 0 2 0 1 0]
```

```
del(data)
```

```
data.data.shape
```

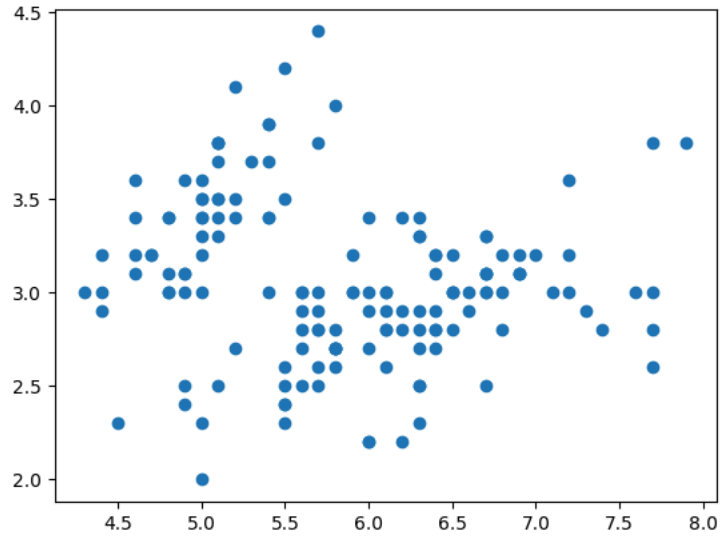
```
(150, 4)
```

```
print("Number of rows:",data.data.shape[0])
```

```
Number of rows: 150
```

```
import matplotlib.pyplot as plt
x_1 = data.data[:,0] #sepal length
x_2 = data.data[:,1] #sepal width
plt.scatter(x_1,x_2)
```

```
<matplotlib.collections.PathCollection at 0x7c96a3575ea0>
```



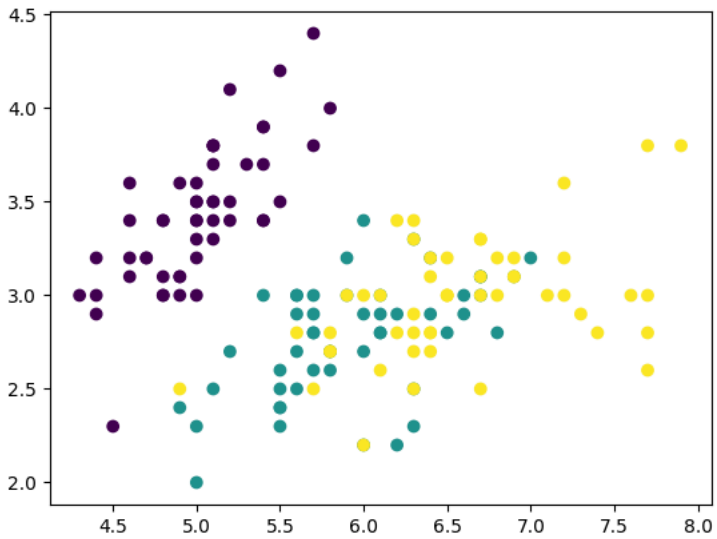

```
target_values = data.target
```

```
target_values
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
plt.scatter(x_1,x_2,c = target_values)
```

```
<matplotlib.collections.PathCollection at 0x7c96a1254d30>
```



Follow link (ctrl + click)

Start coding or generate with AI.

Group B

```
from  sklearn.datasets import load_iris
iris = load_iris()
```

```
type(iris)
```

```
sklearn.utils._bunch.Bunch
```

```
#Feature names
iris.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

```
#Feature matrice
X = iris.data
X
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
```

```
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
```

Follow link (ctrl + click)

```
#Target names
iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
iris.target
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing

```
# prompt: label encoding example
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
# Define a list of strings
data = ["France", "Spain", "Germany", "Spain", "Germany", "France",
       "Spain", "France", "Germany", "France"]
# Fit the encoder to the data
le.fit(data)
# Transform the data
encoded_data = le.transform(data)
# Print the encoded data
print(encoded_data)
```

```
type(X)
```

```
numpy.ndarray
```

```
X.shape
```

```
(150, 4)
```

```
print("Number of rows: ", X.shape[0])
```

```
Number of rows:  150
```

```
#get first 10 rows
X[:10,:]
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1]])
```

Follow link (ctrl + click)

```
iris.target[:10]
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

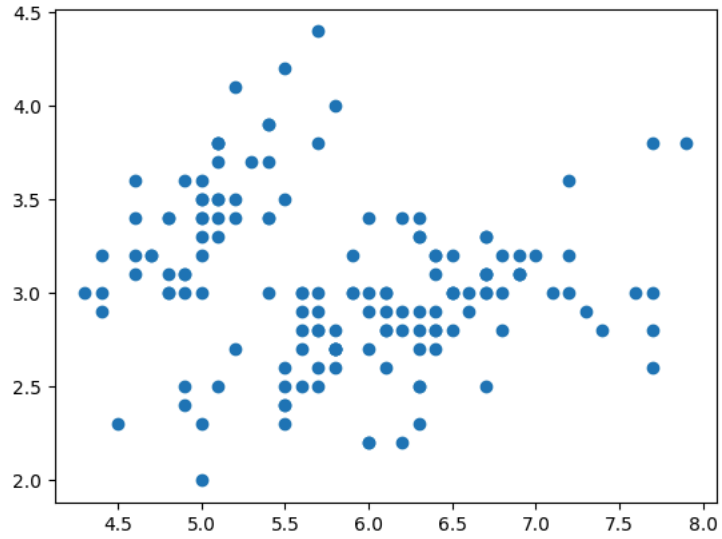
```
iris.data[:,:2]
```

```
array([[5.1, 3.5],
       [4.9, 3. ],
       [4.7, 3.2],
       [4.6, 3.1],
       [5. , 3.6],
       [5.4, 3.9],
       [4.6, 3.4],
       [5. , 3.4],
       [4.4, 2.9],
       [4.9, 3.1],
       [5.4, 3.7],
       [4.8, 3.4],
       [4.8, 3. ],
       [4.3, 3. ],
       [5.8, 4. ],
       [5.7, 4.4],
       [5.4, 3.9],
       [5.1, 3.5],
       [5.7, 3.8],
       [5.1, 3.8],
       [5.4, 3.4],
       [5.1, 3.7],
       [4.6, 3.6],
       [5.1, 3.3],
       [4.8, 3.4],
       [5. , 3. ],
       [5. , 3.4],
       [5.2, 3.5],
       [5.2, 3.4],
       [4.7, 3.2],
       [4.8, 3.1],
       [5.4, 3.4],
       [5.2, 4.1],
       [5.5, 4.2],
       [4.9, 3.1],
       [5. , 3.2],
       [5.5, 3.5],
       [4.9, 3.6],
       [4.4, 3. ],
       [5.1, 3.4],
       [5. , 3.5],
       [4.5, 2.3],
       [4.4, 3.2],
       [5. , 3.5],
       [5.1, 3.8],
       [4.8, 3. ],
       [5.1, 3.8],
       [4.6, 3.2],
       [5.3, 3.7],
       [5. , 3.3],
       [7. , 3.2],
       [6.4, 3.2],
       [6.9, 3.1],
       [5.5, 2.3],
       [6.5, 2.8],
       [5.7, 2.8],
       [6.3, 3.3],
       [4.9, 2.4],
```

```
x_1 = iris.data[:,0]#sepal length
x_2 = iris.data[:,1]#sepal width
y_values = iris.target[:]
```

```
import matplotlib.pyplot as plt
plt.scatter(x_1, x_2)
```

```
<matplotlib.collections.PathCollection at 0x7c96a1254a90>
```



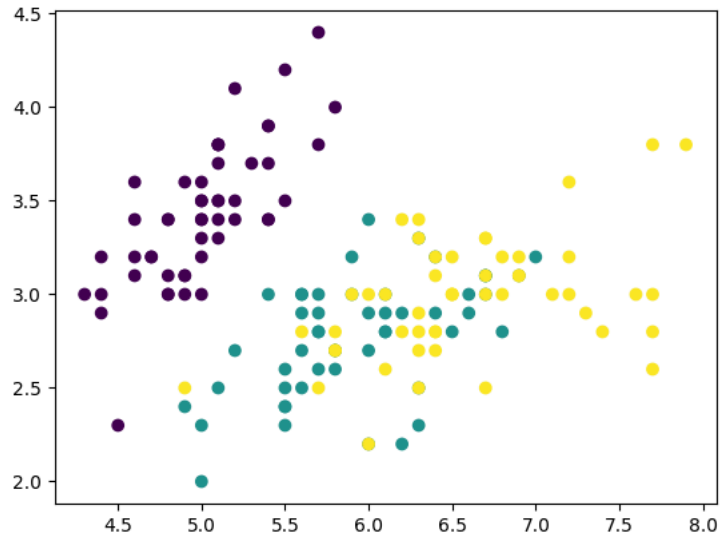
Follow link (ctrl + click)

```
target_values
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
plt.scatter(x_1,x_2, c = y_values)
```

```
<matplotlib.collections.PathCollection at 0x7c96a0d58250>
```




```
Start coding or generate with AI.
```

Lab 6.

The project aims to showcase proficient skills in data analysis, covering the entire spectrum from data collection and cleaning to exploratory data analysis (EDA). While excluding machine learning aspects, the focus is on extracting data from diverse sources, performing thorough data preprocessing, conducting in-depth analysis, and presenting the results coherently. The project will be documented meticulously, culminating in a detailed report and a presentation for defense.

Start coding or generate with AI.

Lab 7.ML model training (Decision Trees)

1. https://images.datacamp.com/image/upload/v1676302389/Marketing/Blog/Scikit-Learn_Cheat_Sheet.pdf
2. https://www.datacamp.com/cheat-sheet/scikit-learn-cheat-sheet-python-machine-learning
3. https://builtin.com/data-science/train-test-split
4. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#
5. https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html
6. https://scikit-learn.org/stable/modules/tree.html
7. https://scikit-learn.org/stable/modules/generated/sklearn.inspection.DecisionBoundaryDisplay.html

Follow link (ctrl + click)

Goal: Learn the basics of fitting, predicting, and visualizing a decision tree model using scikit-learn.

Steps:

Data Preparation:

Load a dataset suitable for a classification or regression task. Split the dataset into features (X) and the target variable (y).

Train-Test Split:

Split the data into training and testing sets using train_test_split from scikit-learn.

Create and Fit a Decision Tree Model:

Import DecisionTreeClassifier or DecisionTreeRegressor depending on the task. Create an instance of the model. Fit the model to the training data using the fit method.

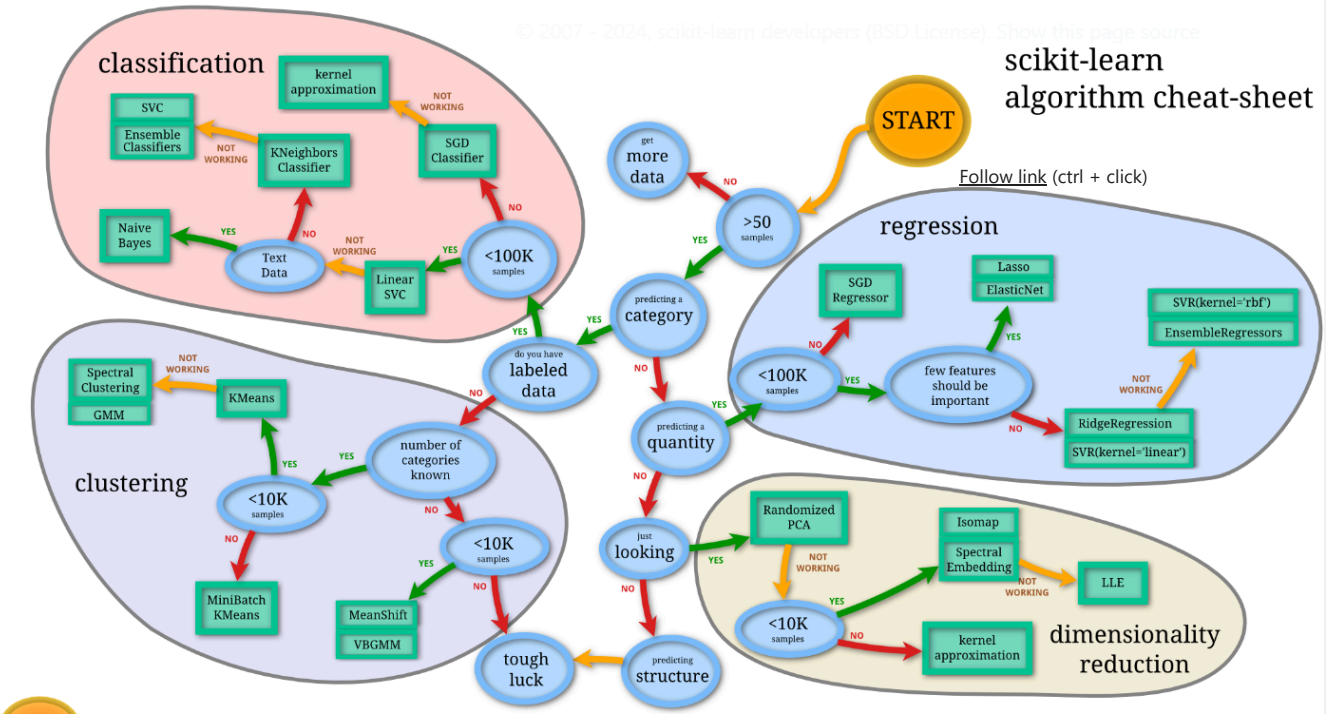
Make Predictions:

Use the trained model to make predictions on the test set using the predict method. **Visualize the Decision Tree:**

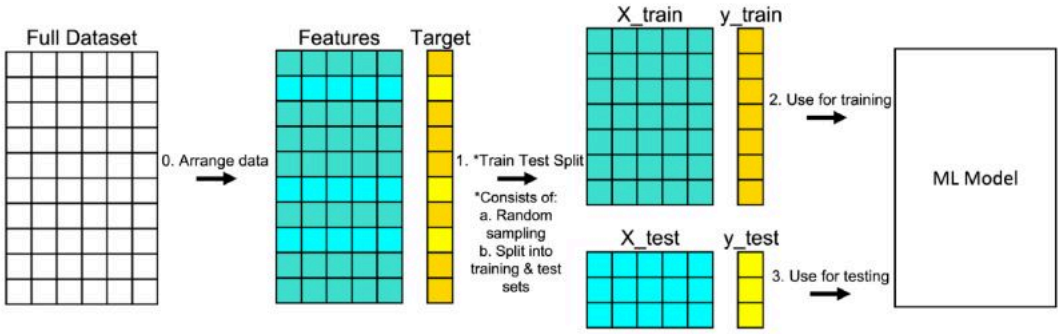
Use plot_tree or other visualization tools to create a visual representation of the decision tree structure. Optionally, export and display the decision tree rules as text using export_text.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a decision tree classifier
clf = DecisionTreeClassifier()

# Fit the model to the training data
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Visualize the decision tree
plt.figure(figsize=(10, 7))
plot_tree(clf, filled=True, feature_names=iris.feature_names, class_names=iris.target_names, rounded=True)
plt.show()
```

Click on any estimator in the chart below to see its documentation.

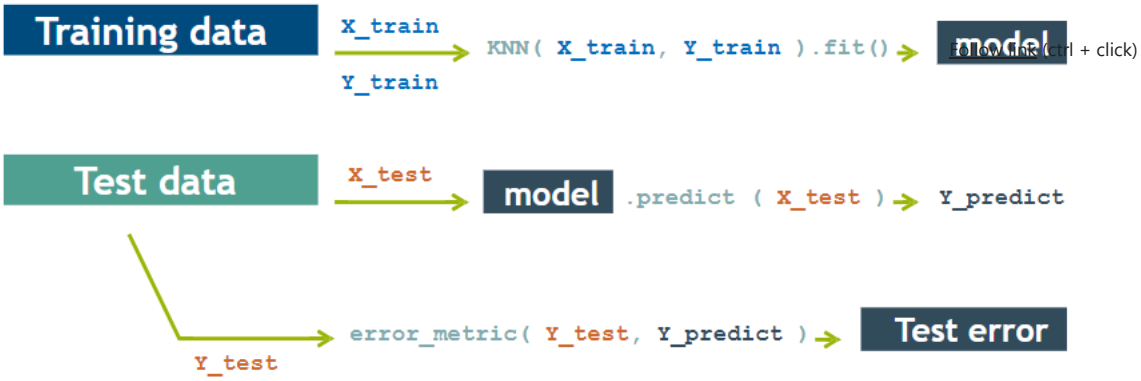


Train test split is a model evaluation procedure that allows you to simulate how a model would perform on new/unseen data.



Train and Test Splitting

Fitting Training and Test Data



Train and Test Splitting: The Syntax

Import the train and test split function

```
from sklearn.model_selection import train_test_split
```

Split the data and put 30% into the test set

```
train, test = train_test_split(data, test_size=0.3)
```

Other method for splitting data:

```
from sklearn.model_selection import ShuffleSplit
```

Decision Trees: The Syntax

Import the class containing the classification method

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

Create an instance of the Decision Tree classifier

```
dt_classifier = DecisionTreeClassifier(max_depth=3)
```

Fit the classifier on your data

```
dt_classifier.fit(X_data, y_data)
```

Predict the labels

```
y_predict = dt_classifier.predict(X_data)
```

Calculate accuracy

```
accuracy = accuracy_score(y_data, y_predict)
```

Print the accuracy

```
print(f'Accuracy = {accuracy:.2f}')
```

Start coding or generate with AI.

```
from sklearn import tree
X = [[0,0],[0,1],[1,1]]
y = [0,0,1]
clf = tree.DecisionTreeClassifier().fit(X,y)
```

Follow link (ctrl + click)

```
type(clf)
```

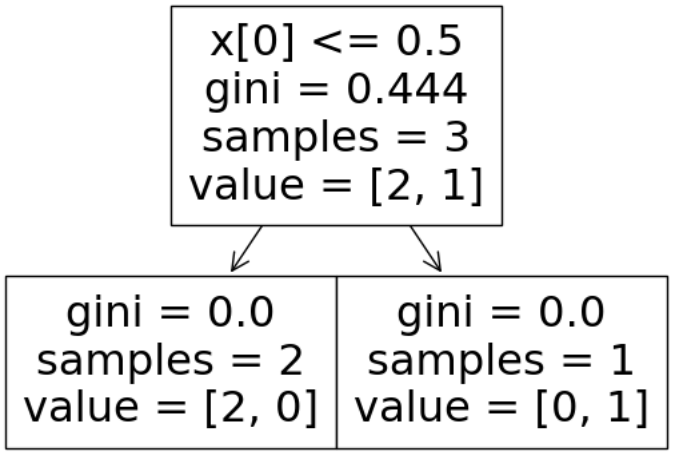
```
sklearn.tree._classes.DecisionTreeClassifier
```

```
X_new = [[1,1],[1,0]]
y_pred = clf.predict(X_new)
y_pred
```

```
array([1, 1])
```

```
tree.plot_tree(clf)
```

```
[Text(0.5, 0.75, 'x[0] <= 0.5\ngini = 0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(0.25, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.75, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]')]
```




```
from sklearn.datasets import load_iris
from sklearn import tree
from sklearn.model_selection import train_test_split
```

```
iris = load_iris()
X = iris.data
y = iris.target
```

```
X.shape
```

```
(150, 4)
```

```
x_train, x_test, y_train,  y_test = train_test_split(X,y, test_size=0.2)
```

```
x_train.shape
```

```
(120, 4)
```

```
y_train.shape
```

```
(120,)
```

```
clf = tree.DecisionTreeClassifier().fit(x_train,y_train)
```

```
x_train
```

```
array([[7.7, 2.6, 6.9, 2.3],
       [7.7, 3.8, 6.7, 2.2],
       [6.2, 2.9, 4.3, 1.3],
       [6.4, 3.1, 5.5, 1.8],
       [7.3, 2.9, 6.3, 1.8],
       [7.2, 3.2, 6. , 1.8],
       [6.4, 2.7, 5.3, 1.9],
       [6.4, 2.8, 5.6, 2.1],
       [6.7, 2.5, 5.8, 1.8],
       [5.5, 2.6, 4.4, 1.2],
       [6.3, 2.5, 4.9, 1.5],
       [4.4, 3.2, 1.3, 0.2],
       [6.2, 3.4, 5.4, 2.3],
       [5.7, 2.5, 5. , 2. ],
       [5.6, 3. , 4.1, 1.3],
       [6. , 2.2, 4. , 1. ],
       [6.5, 2.8, 4.6, 1.5],
       [4.6, 3.4, 1.4, 0.3],
       [4.9, 2.5, 4.5, 1.7],
       [5.4, 3.7, 1.5, 0.2],
       [5.9, 3. , 5.1, 1.8],
       [4.7, 3.2, 1.3, 0.2],
       [6.7, 3.1, 4.7, 1.5],
       [4.3, 3. , 1.1, 0.1],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [6.2, 2.2, 4.5, 1.5],
       [5. , 3.5, 1.3, 0.3],
       [6.5, 3. , 5.2, 2. ],
       [6.3, 2.9, 5.6, 1.8],
       [5.1, 3.4, 1.5, 0.2],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.3, 4.4, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [7.7, 3. , 6.1, 2.3],
       [6.8, 3. , 5.5, 2.1],
       [5. , 3.4, 1.5, 0.2],
       [6. , 2.7, 5.1, 1.6],
       [5.7, 2.9, 4.2, 1.3],
       [5.6, 2.9, 3.6, 1.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.2, 4.1, 1.5, 0.1],
       [6. , 2.2, 5. , 1.5],
       [5.1, 3.3, 1.7, 0.5],
       [6.7, 3.1, 5.6, 2.4],
       [7.1, 3. , 5.9, 2.1],
       [4.7, 3.2, 1.6, 0.2],
       [6.3, 3.4, 5.6, 2.4],
       [5.7, 3.8, 1.7, 0.3],
       [5.3, 3.7, 1.5, 0.2],
       [6.4, 3.2, 5.3, 2.3],
       [5.4, 3.9, 1.7, 0.4],
       [5.5, 2.5, 4. , 1.3],
       [4.6, 3.2, 1.4, 0.2],
       [6.3, 3.3, 4.7, 1.6],
       [6.5, 3. , 5.5, 1.8],
       [6.5, 3.2, 5.1, 2. ],
       [5.1, 3.8, 1.9, 0.4],
```

Follow link (ctrl + click)

```
clf.predict([[5.7, 4.4, 1.5, 0.4]])
```

```
array([0])
```

```
y_pred = clf.predict(x_test)
```

```
y_pred
```

```
array([1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 2, 1, 1, 0, 1, 1, 0, 2, 0,
       0, 0, 2, 0, 1, 0, 1, 2])
```

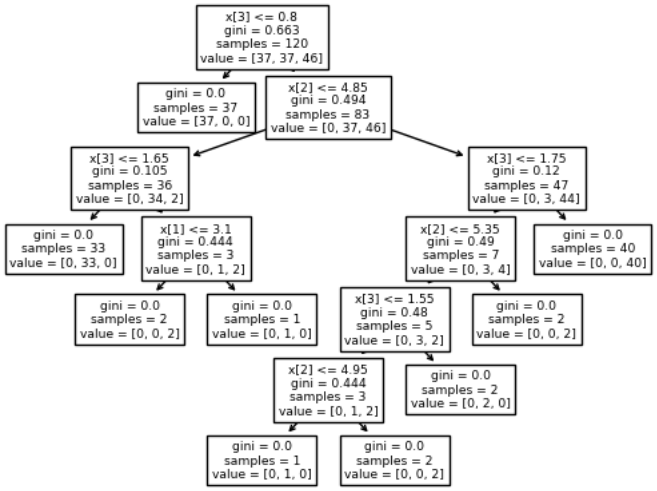
```
y_test
```

```
array([1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 2, 1, 1, 0, 1, 1, 0, 2, 0,
       0, 0, 2, 0, 1, 0, 1, 2])
```

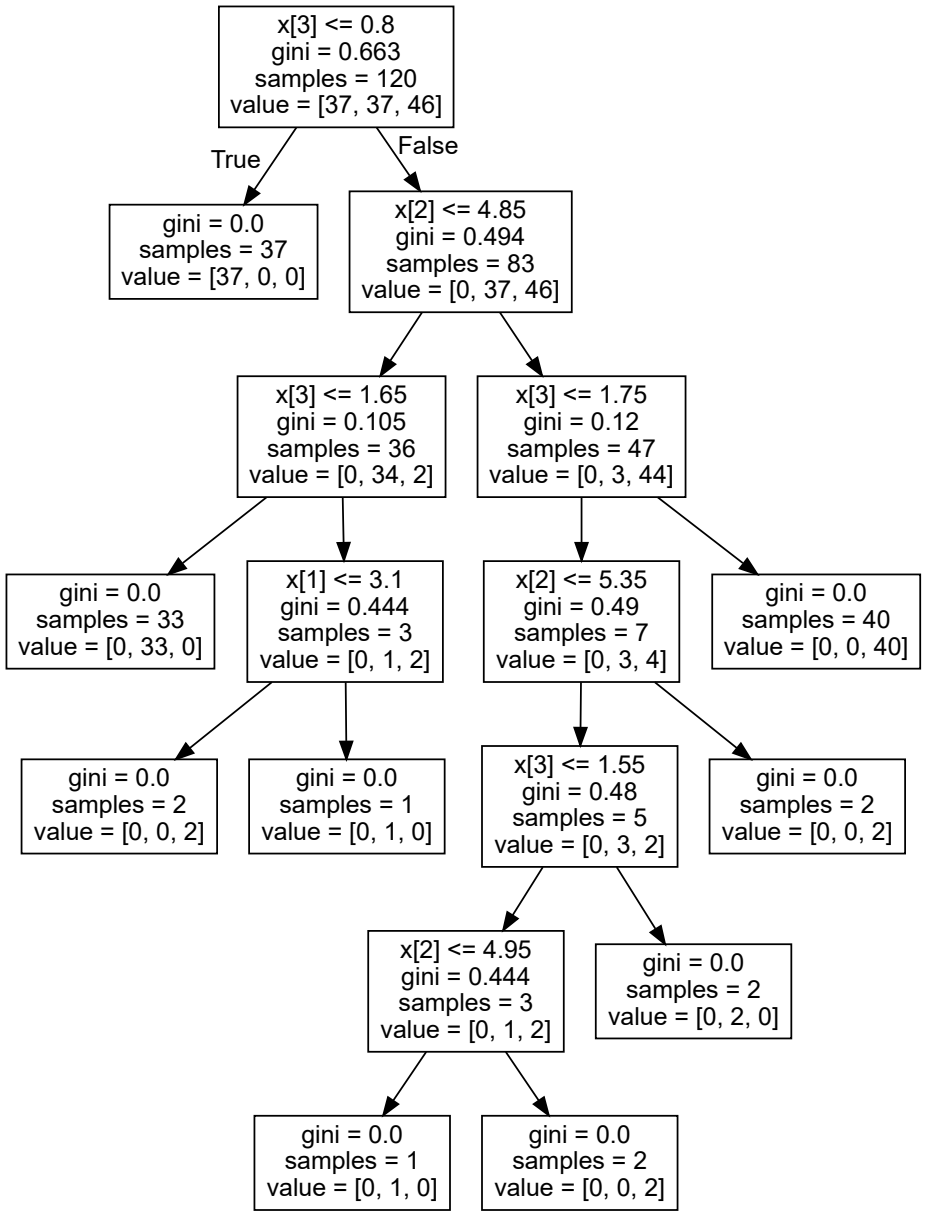
```
tree.plot_tree(clf)
```

```
[Text(0.4, 0.9285714285714286, 'x[3] <= 0.8\ngini = 0.663\nsamples = 120\nvalue = [37, 37, 46]'),
 Text(0.3, 0.7857142857142857, 'gini = 0.0\nsamples = 37\nvalue = [37, 0, 0]'),
 Text(0.5, 0.7857142857142857, 'x[2] <= 4.85\ngini = 0.494\nsamples = 83\nvalue = [0, 37, 46]'),
 Text(0.2, 0.6428571428571429, 'x[3] <= 1.65\ngini = 0.105\nsamples = 36\nvalue = [0, 34, 2]'),
 Text(0.1, 0.5, 'gini = 0.0\nsamples = 33\nvalue = [0, 33, 0]'),
 Text(0.3, 0.5, 'x[1] <= 3.1\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(0.2, 0.35714285714285715, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.4, 0.35714285714285715, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.8, 0.6428571428571429, 'x[3] <= 1.75\ngini = 0.12\nsamples = 47\nvalue = [0, 3, 44]'),
 Text(0.7, 0.5, 'x[2] <= 5.35\ngini = 0.49\nsamples = 7\nvalue = [0, 3, 4]'),
 Text(0.6, 0.35714285714285715, 'x[3] <= 1.55\ngini = 0.48\nsamples = 5\nvalue = [0, 3, 2]'),
 Text(0.5, 0.21428571428571427, 'x[2] <= 4.95\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(0.4, 0.07142857142857142, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.6, 0.07142857142857142, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.7, 0.21428571428571427, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
 Text(0.8, 0.35714285714285715, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.9, 0.5, 'gini = 0.0\nsamples = 40\nvalue = [0, 0, 40]')]
```

Follow link (ctrl + click)

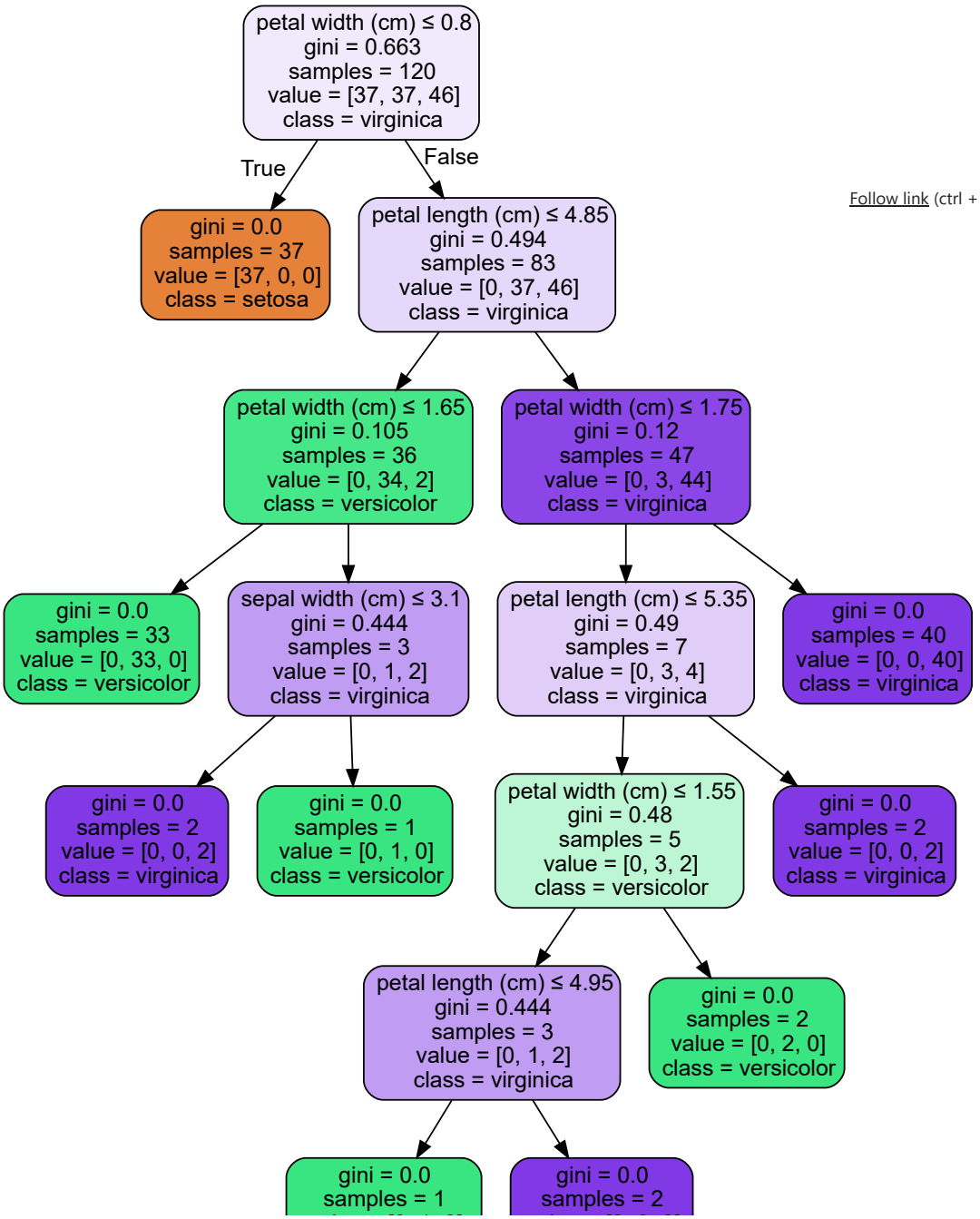



```
import graphviz


dot_data = tree.export_graphviz(clf)
graph = graphviz.Source(dot_data)
graph
```



Follow link (ctrl + click)

```
dot_data = tree.export_graphviz(clf, out_file=None,
                      feature_names=iris.feature_names,
                      class_names=iris.target_names,
                      filled=True, rounded=True,
                      special_characters=True)
graph = graphviz.Source(dot_data)
graph
```



Follow link (ctrl + click)

```
from sklearn import tree
X = [[0, 0], [1, 1]]
Y = [0, 1]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, Y)
```

```
clf.predict([[2., 2.]])
```

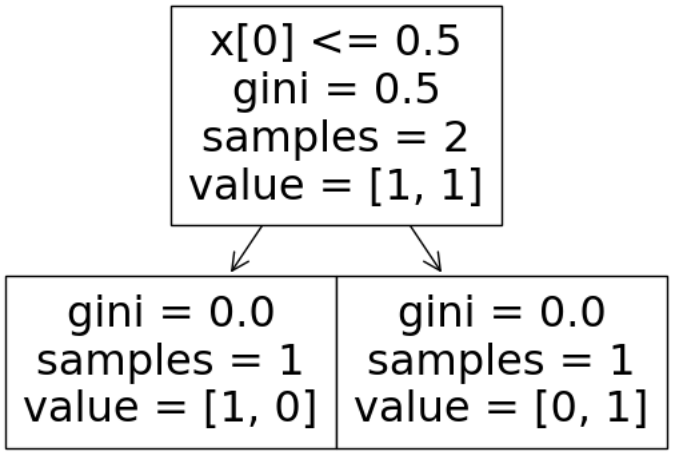
```
array([1])
```

```
clf.predict_proba([[2., 2.]])
```

```
array([[0., 1.]])
```

```
tree.plot_tree(clf)
```

```
[Text(0.5, 0.75, 'x[0] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.25, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.75, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]')]
```



Follow link (ctrl + click)

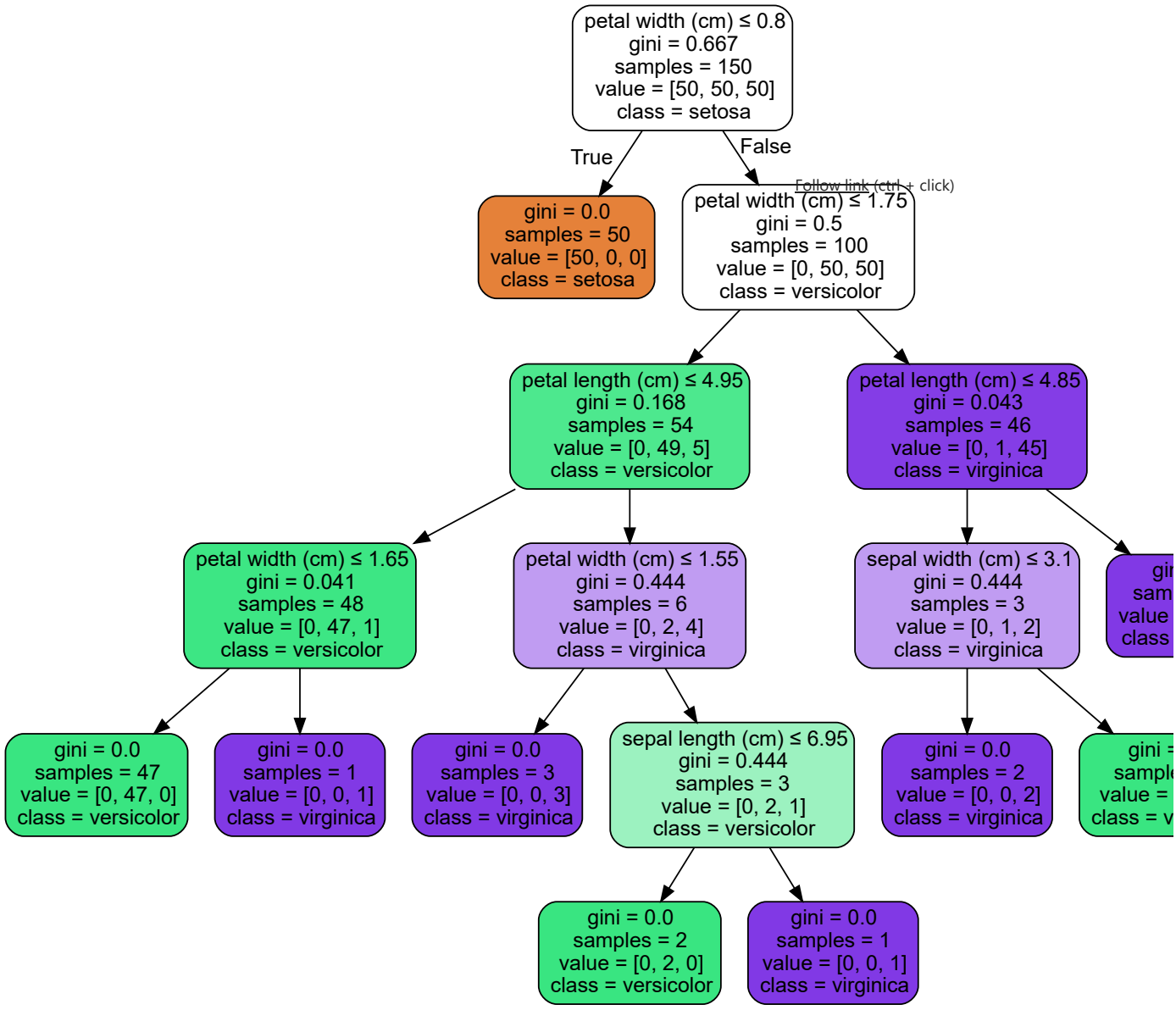
```
from sklearn.datasets import load_iris
from sklearn import tree
```

```
from sklearn.datasets import load_iris
from sklearn import tree
iris = load_iris()
X, y = iris.data, iris.target
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, y)
```

```
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("iris")
```

```
'iris.pdf'
```

```
dot_data = tree.export_graphviz(clf, out_file=None,
                      feature_names=iris.feature_names,
                      class_names=iris.target_names,
                      filled=True, rounded=True,
                      special_characters=True)
graph = graphviz.Source(dot_data)
graph
```



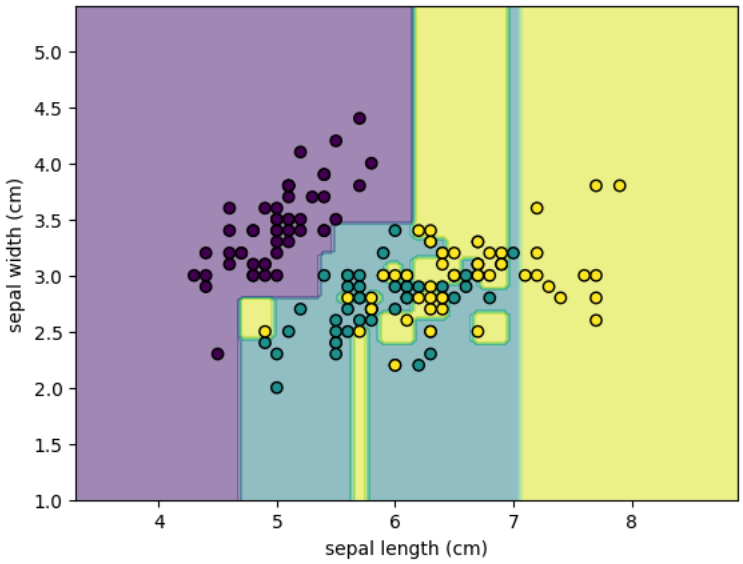

```
x, y = iris.data, iris.target
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=123)
```

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text
iris = load_iris()
decision_tree = DecisionTreeClassifier(random_state=0, max_depth=2)
decision_tree = decision_tree.fit(iris.data, iris.target)
r = export_text(decision_tree, feature_names=iris['feature_names'])
print(r)
```

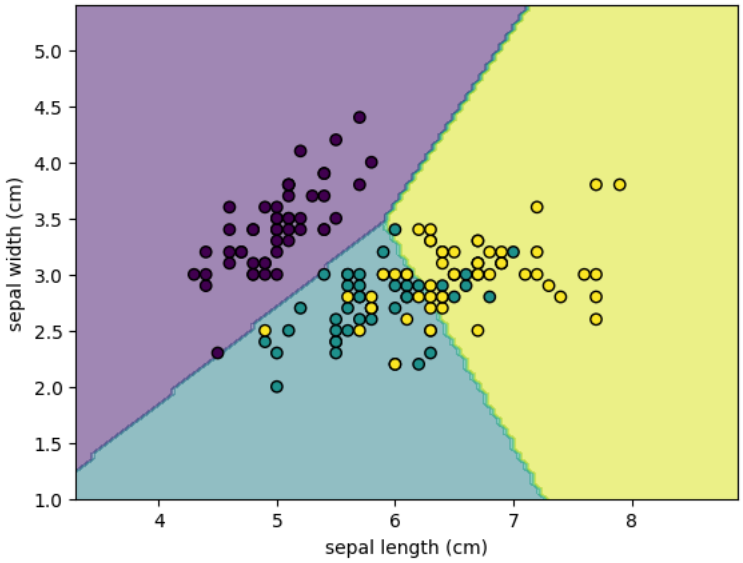
```
|--- petal width (cm) <= 0.80
|   |--- class: 0
|--- petal width (cm) >  0.80
|   |--- petal width (cm) <= 1.75
|   |   |--- class: 1
|   |--- petal width (cm) >  1.75
|   |   |--- class: 2
```

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.inspection import DecisionBoundaryDisplay
iris = load_iris()
X = iris.data[:, :2]
classifier = DecisionTreeClassifier().fit(X, iris.target)
disp = DecisionBoundaryDisplay.from_estimator(
    classifier, X, response_method="predict",
    xlabel=iris.feature_names[0], ylabel=iris.feature_names[1],
    alpha=0.5,
)
disp.ax_.scatter(X[:, 0], X[:, 1], c=iris.target, edgecolor="k")
plt.show()
```

Follow link (ctrl + click)



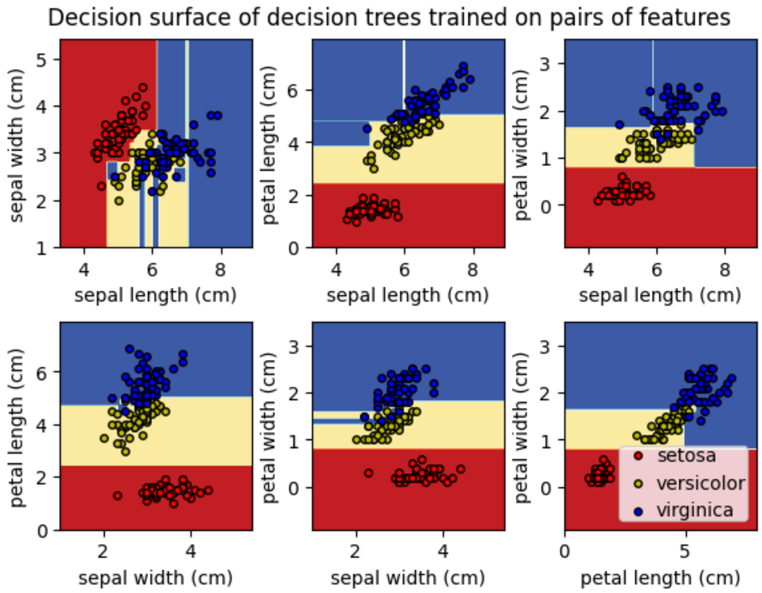

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.inspection import DecisionBoundaryDisplay
iris = load_iris()
X = iris.data[:, :2]
classifier = LogisticRegression().fit(X, iris.target)
disp = DecisionBoundaryDisplay.from_estimator(
    classifier, X, response_method="predict",
    xlabel=iris.feature_names[0], ylabel=iris.feature_names[1],
    alpha=0.5,
)
disp.ax_.scatter(X[:, 0], X[:, 1], c=iris.target, edgecolor="k")
plt.show()
```



Follow link (ctrl + click)

```
import matplotlib.pyplot as plt
import numpy as np

from sklearn.datasets import load_iris
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.tree import DecisionTreeClassifier

# Parameters
n_classes = 3
plot_colors = "ryb"
plot_step = 0.02


for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]):
    # We only take the two corresponding features
    X = iris.data[:, pair]
    y = iris.target

    # Train
    clf = DecisionTreeClassifier().fit(X, y)

    # Plot the decision boundary
    ax = plt.subplot(2, 3, pairidx + 1)
    plt.tight_layout(h_pad=0.5, w_pad=0.5, pad=2.5)
    DecisionBoundaryDisplay.from_estimator(
        clf,
        X,
        cmap=plt.cm.RdYlBu,
        response_method="predict",
        ax=ax,
        xlabel=iris.feature_names[pair[0]],
        ylabel=iris.feature_names[pair[1]],
    )

    # Plot the training points
    for i, color in zip(range(n_classes), plot_colors):
        idx = np.where(y == i)
        plt.scatter(
            X[idx, 0],
            X[idx, 1],
            c=color,
            label=iris.target_names[i],
            cmap=plt.cm.RdYlBu,
            edgecolor="black",
            s=15,
        )

plt.suptitle("Decision surface of decision trees trained on pairs of features")
plt.legend(loc="lower right", borderpad=0, handletextpad=0)
_ = plt.axis("tight")
```

```
<ipython-input-18-aebfe965615f>:38: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(
<ipython-input-18-aebfe965615f>:38: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(
<ipython-input-18-aebfe965615f>:38: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(
<ipython-input-18-aebfe965615f>:38: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(
<ipython-input-18-aebfe965615f>:38: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(                                                                      Follow link (ctrl + click)
<ipython-input-18-aebfe965615f>:38: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(
```



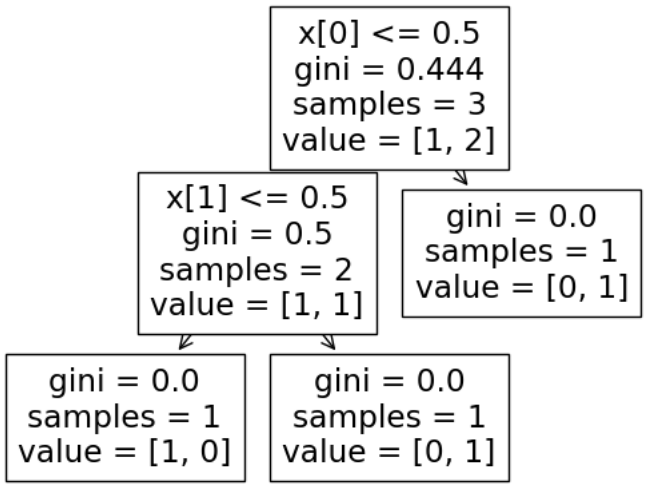
Group B

```
from sklearn import tree


X = [[0,0], [0,1],[1,0]]
y = [0,1,1]


clf = tree.DecisionTreeClassifier().fit(X,y)
clf
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
tree.plot_tree(clf)
```

```
[Text(0.6, 0.8333333333333334, 'x[0] <= 0.5\ngini = 0.444\nsamples = 3\nvalue = [1, 2]'),
 Text(0.4, 0.5, 'x[1] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.2, 0.16666666666666666, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.6, 0.16666666666666666, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.8, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]')]
```



Follow link (ctrl + click)

```
clf.predict([[1,1]])
```

```
array([1])
```

```
clf.predict([[0,0]])
```

```
array([0])
```

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn import tree
```

```
iris  = load_iris()
```

```
X = iris.data
X.shape
```

```
(150, 4)
```

```
y = iris.target
y.shape
```

```
(150,)
```

```
x_train,x_test,y_train, y_test = train_test_split(X,y,test_size=0.3)
```

```
x_train.shape
```

```
(105, 4)
```

```
y_train.shape
```

```
(105,)
```

```
clf = tree.DecisionTreeClassifier().fit(x_train,y_train)
clf
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred = clf.predict(x_test)
y_pred
```

```
array([2, 0, 1, 0, 0, 1, 1, 2, 0, 0, 0, 1, 0, 2, 2, 1, 1, 0, 0, 1, 0, 1,
       2, 1, 2, 1, 1, 2, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 2, 1, 2, 2,
       1])
```

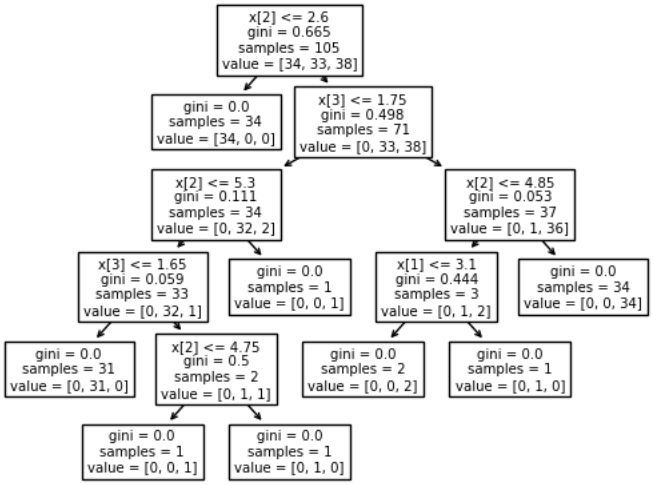
```
y_test
```

Follow link (ctrl + click)

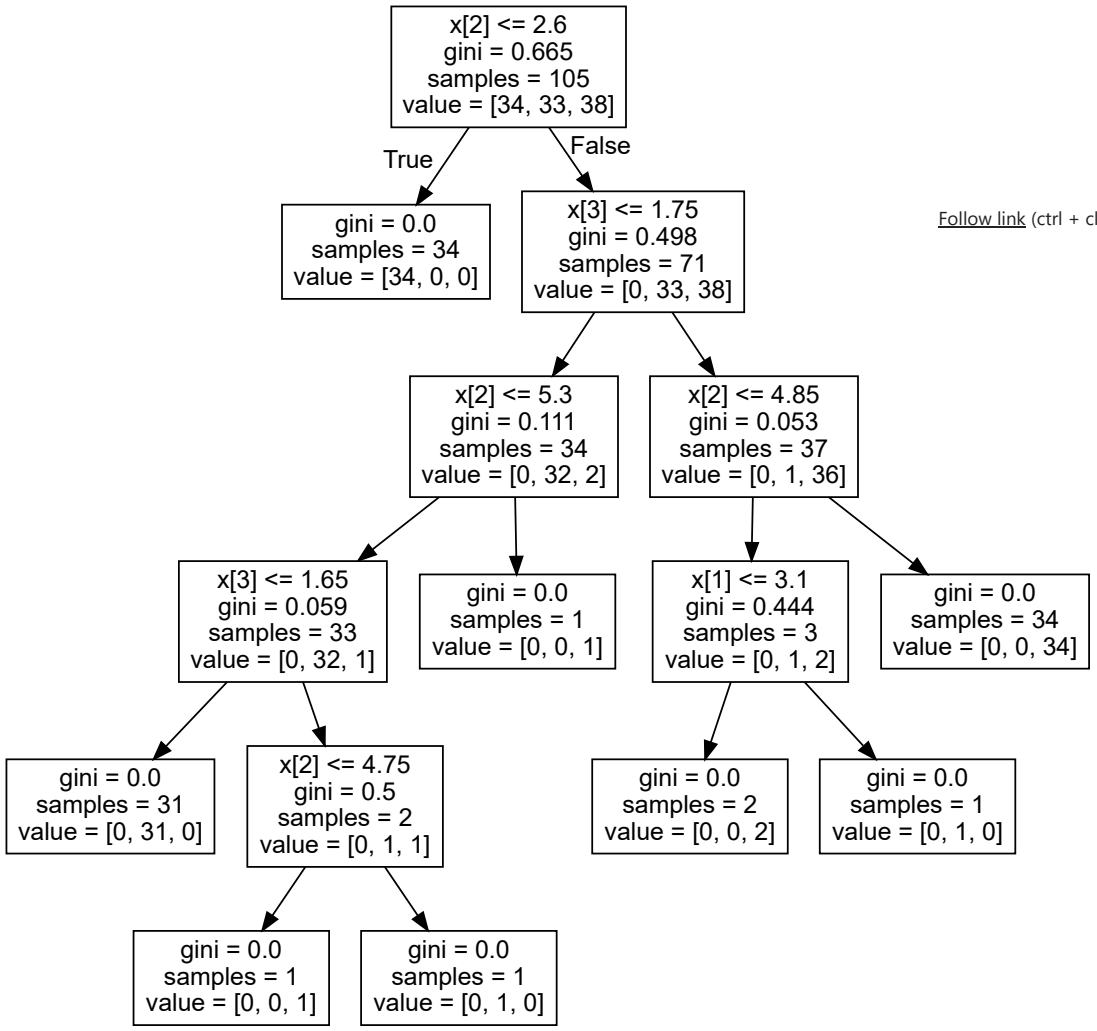
```
array([2, 0, 1, 0, 0, 1, 1, 2, 0, 0, 0, 1, 0, 2, 2, 1, 2, 0, 0, 1, 0, 1,
       2, 1, 2, 1, 2, 2, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 2, 1, 2, 2,
       1])
```

```
tree.plot_tree(clf)
```

```
[Text(0.4444444444444444, 0.9166666666666666, 'x[2] <= 2.6\ngini = 0.665\nsamples = 105\nvalue = [34, 33, 38]'),
 Text(0.3333333333333333, 0.75, 'gini = 0.0\nsamples = 34\nvalue = [34, 0, 0]'),
 Text(0.5555555555555556, 0.75, 'x[3] <= 1.75\ngini = 0.498\nsamples = 71\nvalue = [0, 33, 38]'),
 Text(0.3333333333333333, 0.5833333333333334, 'x[2] <= 5.3\ngini = 0.111\nsamples = 34\nvalue = [0, 32, 2]'),
 Text(0.2222222222222222, 0.4166666666666667, 'x[3] <= 1.65\ngini = 0.059\nsamples = 33\nvalue = [0, 32, 1]'),
 Text(0.1111111111111111, 0.25, 'gini = 0.0\nsamples = 31\nvalue = [0, 31, 0]'),
 Text(0.3333333333333333, 0.25, 'x[2] <= 4.75\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]'),
 Text(0.2222222222222222, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.4444444444444444, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.4444444444444444, 0.4166666666666667, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.7777777777777778, 0.5833333333333334, 'x[2] <= 4.85\ngini = 0.053\nsamples = 37\nvalue = [0, 1, 36]'),
 Text(0.6666666666666666, 0.4166666666666667, 'x[1] <= 3.1\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(0.5555555555555556, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.7777777777777778, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.8888888888888888, 0.4166666666666667, 'gini = 0.0\nsamples = 34\nvalue = [0, 0, 34]')]
```

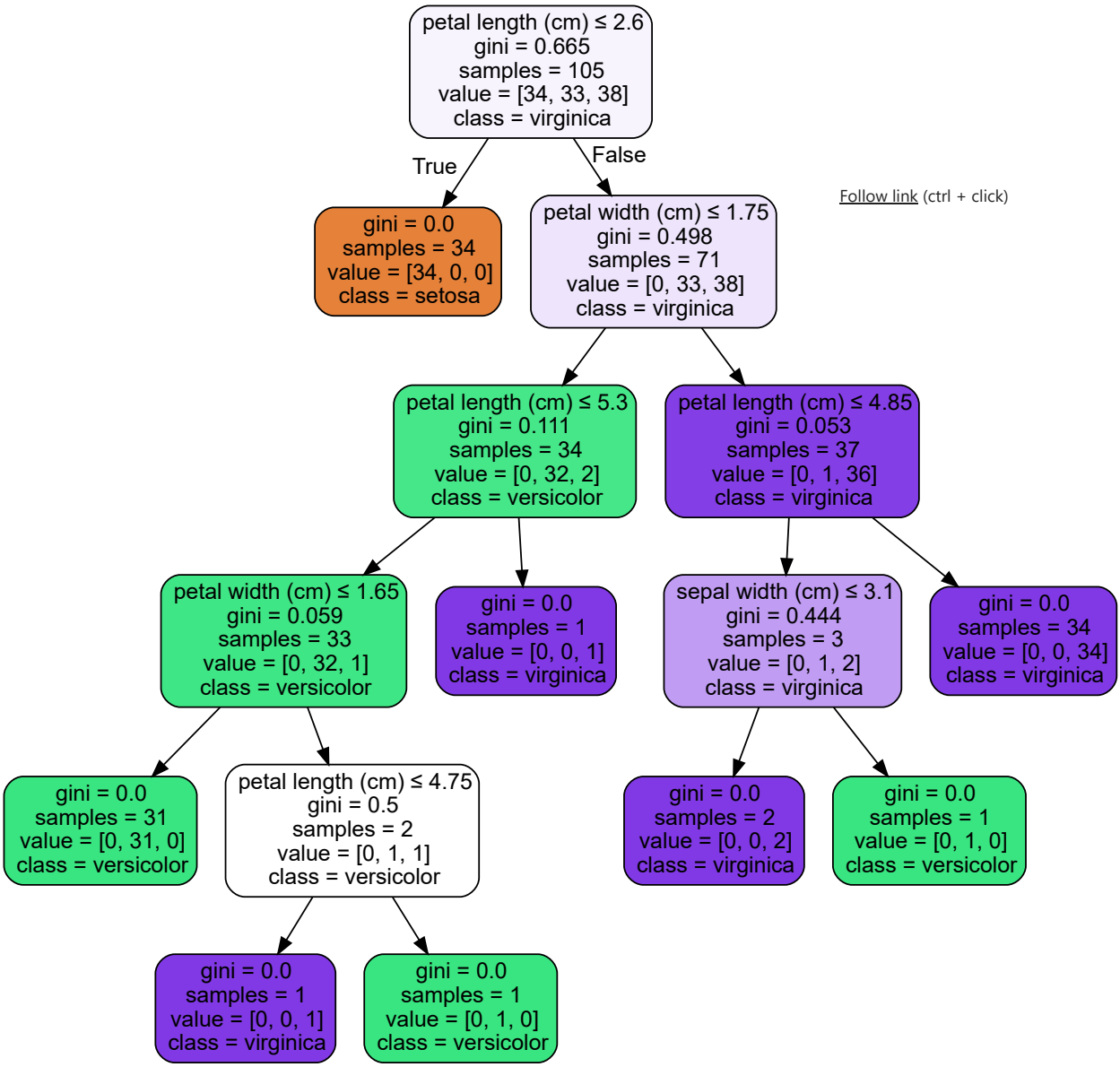



```
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph
```



Follow link (ctrl + click)

```
dot_data = tree.export_graphviz(clf, out_file=None,
                      feature_names=iris.feature_names,
                      class_names=iris.target_names,
                      filled=True, rounded=True,
                      special_characters=True)
graph = graphviz.Source(dot_data)
graph
```



Follow link (ctrl + click)

Double-click (or enter) to edit

Lab 8. ML Pipeline Implementation

1. Adult Dataset - https://www.kaggle.com/code/jieyima/income-classification-model
2. Pinguin Dataset - https://github.com/ketanmakde/DT-RF_Penguin-Data-Antarctica/blob/main/Penguin%20Data%20Antarctica.ipynb

```
# Importing necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Load Iris dataset
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target

X = iris_df.drop(columns=['target'])
y = iris_df['target']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Training the Decision Tree Classifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)

# Predicting on the test set
y_pred = classifier.predict(X_test)

# Calculating accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Calculating correlation
correlation = y_test.corr(pd.Series(y_pred))
print("Correlation:", correlation)

# Creating confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)

# Plotting heatmap of confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()

# Plotting scatter plot for test and predicted values
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(X_test.iloc[:, 0], X_test.iloc[:, 1], c=y_test, cmap='viridis')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Test Data')
plt.colorbar(label='Target')

plt.subplot(1, 2, 2)
plt.scatter(X_test.iloc[:, 0], X_test.iloc[:, 1], c=y_pred, cmap='viridis')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Predicted Data')
plt.colorbar(label='Predicted Target')

plt.tight_layout()
plt.show()

# Classification report
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))

# Model Evaluation with Cross-Validation
cv_scores = cross_val_score(classifier, X, y, cv=5)
print("Cross-validation Scores:", cv_scores)
print("Mean CV Accuracy:", cv_scores.mean())
```

Follow link (ctrl + click)

1. Adult Dataset
2. Pinguins Dataset

Lab Steps

1. Get a dataset ready
2. Make a data preprocessing (cleaning, encoding)
3. Make a EDA
4. Prepare a machine learning model to make predictions
5. Fit the model to the data and make a prediction

Follow link (ctrl + click)

1. Getting a dataset ready

```
# Import the  dataset and save it to a variable
# using pandas and read_csv()
# Hint: You can directly pass the URL of a csv to read_csv()

#https://raw.githubusercontent.com/saravrajavelu/Adult-Income-Analysis/master/adult.csv

import pandas as pd
url = "https://raw.githubusercontent.com/Yorko/mlcourse.ai/main/data/adult.data.csv"
df = pd.read_csv(url)

# First ten rows
df.head(10)
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States

```
# Shape of training data (num_rows, num_columns)
df.shape
```

```
(32561, 15)
```

```
# List column names, types and other properties
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64 
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64 
 11  capital-loss    32561 non-null  int64 
 12  hours-per-week  32561 non-null  int64 
 13  native-country  32561 non-null  object
 14  salary          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Follow link (ctrl + click)

```
# Show statistical characterisctics of data
df.describe()
```

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

2. Make a data preprocessing (cleaning, encoding)

```
# Identifying missing values


# Identifying numerical columns


# Identifying categorical columns


# Imputer for numerical data
num_imputer = SimpleImputer(strategy='median')
# Imputer for categorical data
cat_imputer = SimpleImputer(strategy='most_frequent')


# Show unique values and counts in categorical columns




# Replace categorical values with numerical values
#Label encoding
```

3. EDA

```
# Distribution of the 'income' column
```

```
# Count plots for 'marital-status' and 'income'


# Distribution of the 'age' column

# Distribution of the 'Workclass'column

# Distribution of the 'Occupation'column

# Distribution of the 'Marital-status' column

# Pair plots of entire dataset

# Correlation matrix heatmap


Start coding or generate with AI.
```

Follow link (ctrl + click)

4. Prepare ML model

In essence, the target column is our target variable (also called y or labels) and the rest of the other columns are our independent variables (also called data or X). let's create X and y by splitting our dataframe up.

```
#
X = df.drop('income', axis=1)
y = df['income']

# Split data into training and test sets


Start coding or generate with AI.
```

5. Fit the model to the data and make a prediction

```
# Initialize the model

# Train the model

# Predict on the test set
```

Lab9. Model Evaluation

1. https://www.datacamp.com/blog/machine-learning-models-explained
2. https://www.datacamp.com/blog/classification-machine-learning
3. https://github.com/rasbt/stat451-machine-learning-fs20/tree/master/L12/code
4. https://www.evidentlyai.com/classification-metrics/multi-class-metrics

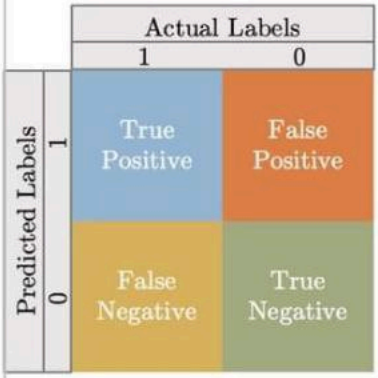
Cheat Sheet – Imbalanced Data in Classification

Classifier that always predicts label blue yields prediction accuracy of 90%

Accuracy doesn't always give the correct insight about your trained model

Accuracy: %age correct prediction	Correct prediction over total predictions	One value for entire network
Precision: Exactness of model	From the detected cats, how many were actually cats	Each class/label has a value
Recall: Completeness of model	Correctly detected cats over total cats	Each class/label has a value
F1 Score: Combines Precision/Recall	Harmonic mean of Precision and Recall	Each class/label has a value

Performance metrics associated with Class 1



(Is your prediction correct?) (What did you predict)

True Negative

(Your prediction is correct) (You predicted 0)

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{F1 score} = 2x\frac{(\text{Prec x Rec})}{(\text{Prec} + \text{Rec})}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{False +ve rate} = \frac{FP}{TN + FP}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

$$\text{Recall, Sensitivity} = \text{True +ve rate} = \frac{TP}{TP + FN}$$

Possible solutions

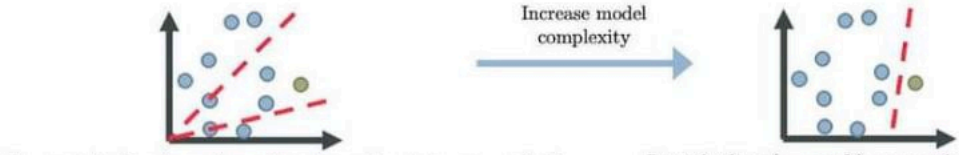
1. **Data Replication:** Replicate the available data until the number of samples are comparable

2. **Synthetic Data:** Images: Rotate, dilate, crop, add noise to existing input images and create new data

3. **Modified Loss:** Modify the loss to reflect greater error when misclassifying smaller sample set

$$loss = a * loss_{green} + b * loss_{blue} \quad a > b$$

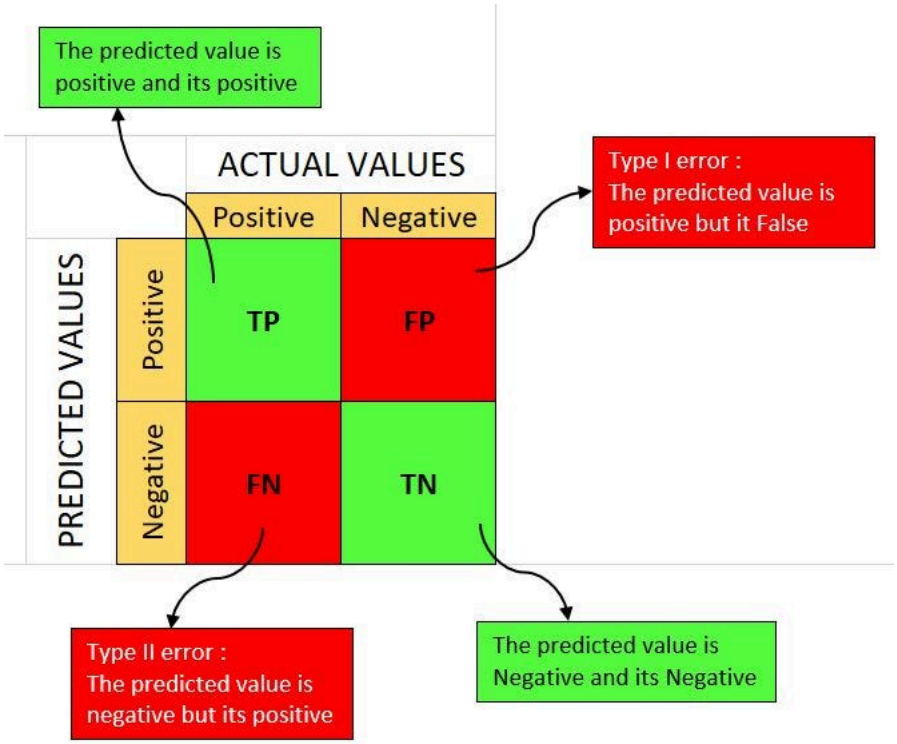
4. **Change the algorithm:** Increase the model/algorithm complexity so that the two classes are perfectly separable (Con: Overfitting)



No straight line (y=ax) passing through origin can perfectly separate data. **Best solution:** line y=0, predict all labels blue

Straight line (y=ax+b) can perfectly separate data. Green class will no longer be predicted as blue

Source: https://www.cheatsheets.aqeel-anwar.com



Follow link (ctrl + click)

A bit of context

Imagine you are a healthcare startup, and want an AI assistant able to predict whether a given patient has a heart disease or not based on its health record. This is a binary classification problem where the model will predict

Follow link (ctrl + click)

- 1, True or Yes if the patient has heart disease
- 0, False or No otherwise

1 Confusion matrix

A 2X2 matrix that nicely summarizes the number of correct predictions of the model. It also helps in computing different other performance metrics.

Predicti / Reality	Yes	No
Yes	True Positives (TP)	False Negatives (FN)
No	False Positives(FP)	True Negatives (TN)

Type I Error

Type II Error

Type I & II Errors can be used interchangeably when referring to False Positives and False negatives respectively

2 Accuracy

We get *accuracy* by answering this question: **"out of the predictions made by the model, what percentage is correct?"**

$$Accuracy = \frac{TP + TN}{Total\ number\ observation}$$

Zoumana KEITA

3 Precision

We get *precision* by answering this question: **"out of all the YES predictions, how many of them were correct?"**

$$Precision = \frac{TP}{TP + FP}$$

Follow link (ctrl + click)

4 Recall / Sensitivity

It aims to answer this question: **"how good was the model at predicting real Yes events?"**, which can be considered as the flip of the precision.

$$Sensitivity = \frac{TP}{TP + FN}$$

5 Recall / Specificity

It aims to answer this question: **"how good was the model at predicting real No events?".**

$$Specificity = \frac{TN}{TN + FP}$$

6 F1 Score

Sometimes used when dealing with imbalanced data set, meaning that there are more of one class/label than there are of the other. It corresponds to the harmonic mean of the precision and recall.

$$F1\ Score = 2 . \frac{Precision\ x\ Recall}{Precision + Recall}$$

Zoumana KEITA

7 AUC – ROC Curve

AUC– ROC generates probability values instead of binary 0/1 values. It should be used when your data set is roughly balanced.

Using ROC for imbalanced data sets lead to incorrect interpretation. Follow link (ctrl + click)

ROC curves provide good overview of trade-off between the TP rate and FP rate for binary classifier using different probability thresholds.

- A value below 0.5 indicates a poor classifier
- A value of 0.5 means random classifier
- Value over 0.7 corresponds to a good classifier
- 0.8 indicates a strong classifier
- We have 1 when the classifier perfectly predicts everything.

Strategies to choose the right metric

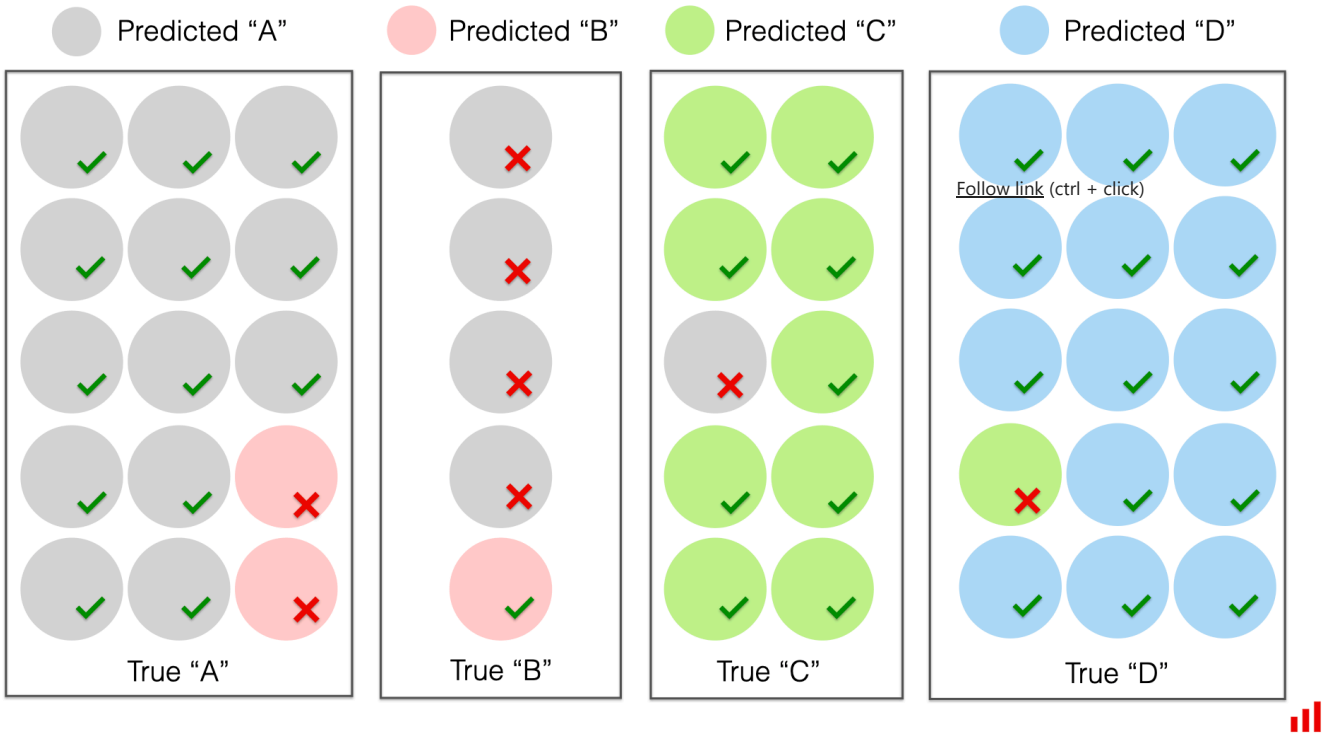
Choose accuracy
- The cost of FP and FN are roughly equal.
- The benefit of TP and TN are roughly equal.

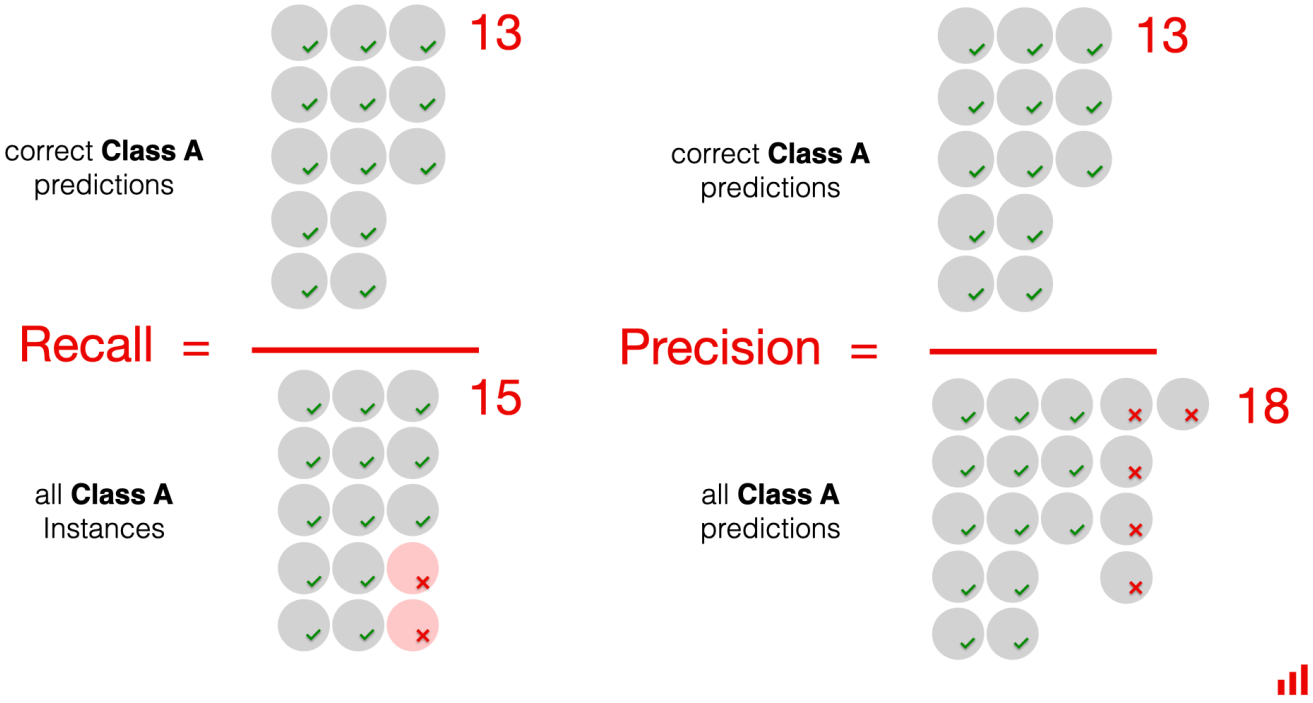
Choose Precision
- The cost of FP is much higher than a FN.
- The benefit of a TP is much higher than a TN.

Choose recall
- The cost of FN is much higher than a FP.
- The cost of a TN is much higher than a TP.

ROC AUC & Precision – Recall curves
- Use ROC when the dealing with balanced data sets.
- Use precision-recall for imbalanced data sets.

Zoumana KEITA






```
#https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic
import pandas as pd
df = pd.read_csv('https://archive.ics.uci.edu/ml/'
                 'machine-learning-databases'
                 '/breast-cancer-wisconsin/wdbc.data', header=None)
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	22	23	24	25	26	27	28	
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.26
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.18
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.24
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	14.91	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.25
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.16

5 rows × 32 columns

Follow link (ctrl + click)

```
from sklearn.preprocessing import LabelEncoder

X = df.loc[:, 2:].values
y = df.loc[:, 1].values
le = LabelEncoder()
y = le.fit_transform(y)
le.classes_
```

```
array(['B', 'M'], dtype=object)
```

```
le.transform(['M', 'B'])
```

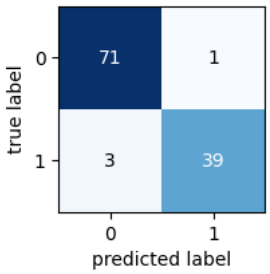
```
array([1, 0])
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = \
    train_test_split(X, y,
                     test_size=0.20,
                     stratify=y,
                     random_state=1)
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline

from mlxtend.evaluate import confusion_matrix
#or
#from sklearn.metrics import confusion_matrix



pipe_knn = make_pipeline(StandardScaler(),
                         KNeighborsClassifier(n_neighbors=5))

pipe_knn.fit(X_train, y_train)

y_pred = pipe_knn.predict(X_test)

confmat = confusion_matrix(y_test, y_pred)

print(confmat)
```

```
[[71  1]
 [ 3 39]]
```

```
from mlxtend.plotting import plot_confusion_matrix
import matplotlib.pyplot as plt

fig, ax = plot_confusion_matrix(conf_mat=confmat, figsize=(2, 2))
plt.show()
```



Follow link (ctrl + click)

```
from sklearn.metrics import  accuracy_score, precision_score, recall_score, f1_score
print("ACC", accuracy_score(y_true=y_test, y_pred=y_pred))
print("Precision", precision_score(y_true=y_test, y_pred=y_pred))
print("Recall", recall_score(y_true=y_test, y_pred=y_pred))
print("F1 score", f1_score(y_true=y_test, y_pred=y_pred))
```

```
ACC 0.9649122807017544
Precision 0.975
Recall 0.9285714285714286
F1 score 0.951219512195122
```

```
print(confusion_matrix(y_test,y_pred))
```

```
[[71  1]
 [ 3 39]]
```

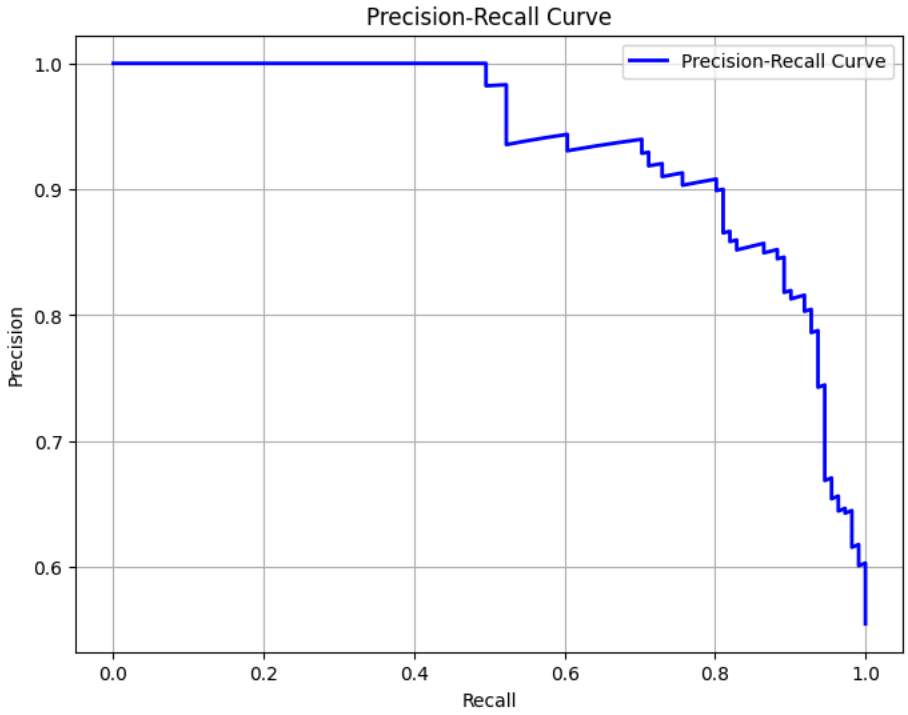
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.96      0.99      0.97        72
               1       0.97      0.93      0.95        42

        accuracy                           0.96       114
       macro avg       0.97      0.96      0.96       114
    weighted avg       0.97      0.96      0.96       114
```

```
# Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, classification_report, precision_recall_curve, f1_score

# Generate a synthetic dataset for demonstration
X, y = make_classification(n_samples=1000, n_features=10, n_classes=2, random_state=42)

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define a function to evaluate the model and print relevant metrics
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:")
    print(cm)
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    # Calculate precision, recall, and F1 score
    precision = cm[1, 1] / (cm[1, 1] + cm[0, 1])
    recall = cm[1, 1] / (cm[1, 1] + cm[1, 0])
    f1 = 2 * (precision * recall) / (precision + recall)
    print(f"\nPrecision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1 Score: {f1:.2f}")

    # Plot Precision-Recall Curve
    precision, recall, _ = precision_recall_curve(y_test, model.predict_proba(X_test)[:, 1])
    plt.figure(figsize=(8, 6))
    plt.plot(recall, precision, color='blue', lw=2, label='Precision-Recall Curve')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.title('Precision-Recall Curve')
    plt.legend(loc='best')
    plt.grid(True)
    plt.show()

# Example model evaluation
# Replace this with your actual model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)

# Evaluate the model
evaluate_model(model, X_test, y_test)
```

Follow link (ctrl + click)

```
Confusion Matrix:
[[75 14]
 [20 91]]

Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.84      0.82        89
           1       0.87      0.82      0.84       111

    accuracy                           0.83       200
   macro avg       0.83      0.83      0.83       200
weighted avg       0.83      0.83      0.83       200


Precision: 0.87
Recall: 0.82
F1 Score: 0.84
```



Follow link (ctrl + click)

```
#https://www.datacamp.com/tutorial/precision-recall-curve-tutorial
```

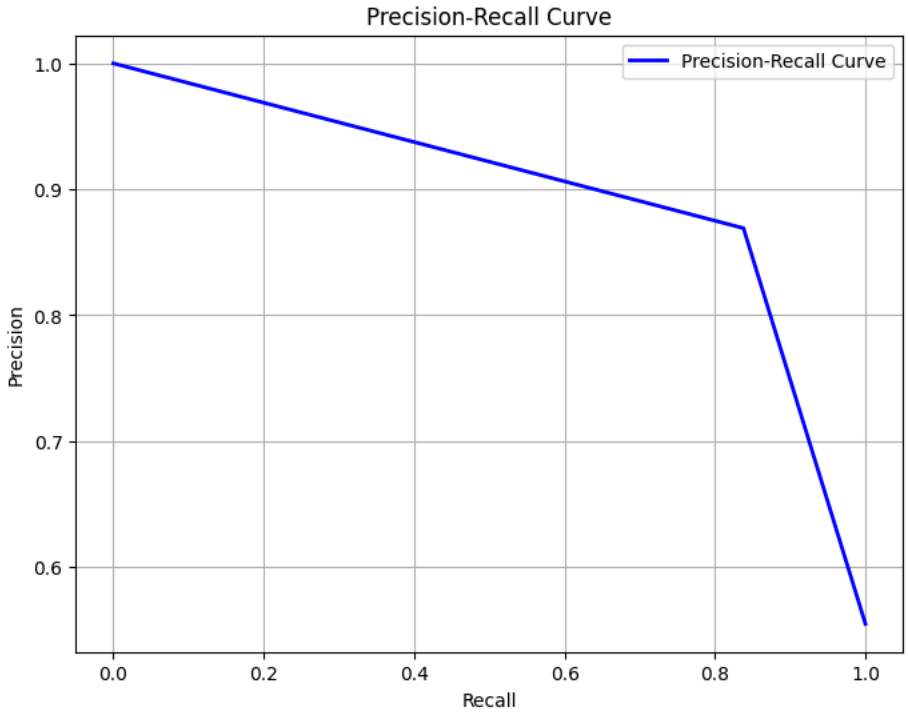
```
# Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report, precision_recall_curve
```

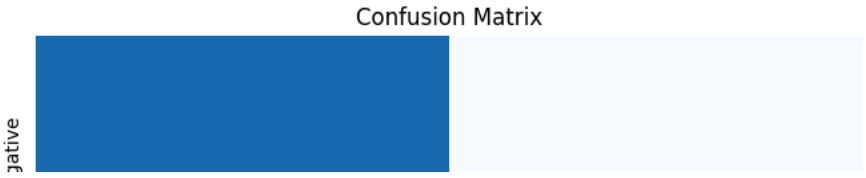
Follow link (ctrl + click)

```
# Generate a synthetic dataset for demonstration
X, y = make_classification(n_samples=1000, n_features=10, n_classes=2, random_state=42)

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Example model evaluation with Decision Tree
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Define a function to evaluate the model and print relevant metrics
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:")
    print(cm)
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    # Calculate precision, recall, and F1 score
    precision = cm[1, 1] / (cm[1, 1] + cm[0, 1])
    recall = cm[1, 1] / (cm[1, 1] + cm[1, 0])
    f1 = 2 * (precision * recall) / (precision + recall)
    print(f"\nPrecision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1 Score: {f1:.2f}")

    # Plot Precision-Recall Curve
    precision, recall, _ = precision_recall_curve(y_test, model.predict_proba(X_test)[:, 1])
    plt.figure(figsize=(8, 6))
    plt.plot(recall, precision, color='blue', lw=2, label='Precision-Recall Curve')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.title('Precision-Recall Curve')
    plt.legend(loc='best')
    plt.grid(True)
    plt.show()

    # Plot Confusion Matrix
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
                xticklabels=['Predicted Negative', 'Predicted Positive'],
                yticklabels=['Actual Negative', 'Actual Positive'])
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.title('Confusion Matrix')
    plt.show()

    # Plot Predicted vs True Labels
    plt.figure(figsize=(8, 6))
    plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap='coolwarm', marker='o', label='True Labels')
    plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, cmap='coolwarm', marker='x', label='Predicted Labels')
    plt.title('Predicted vs True Labels')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.colorbar(label='Class')
    plt.legend()
    plt.grid(True)
    plt.show()

# Evaluate the model
evaluate_model(model, X_test, y_test)
```

```
Confusion Matrix:
[[75 14]
 [18 93]]

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.84      0.82        89
           1       0.87      0.84      0.85       111

    accuracy                           0.84       200
   macro avg       0.84      0.84      0.84       200
weighted avg       0.84      0.84      0.84       200


Precision: 0.87
Recall: 0.84
F1 Score: 0.85
```

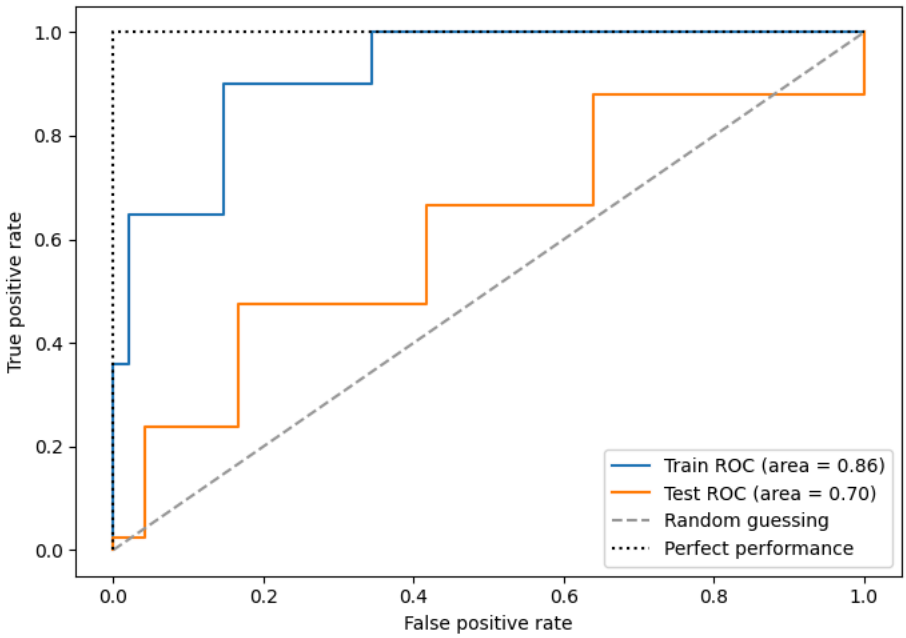
Follow link (ctrl + click)





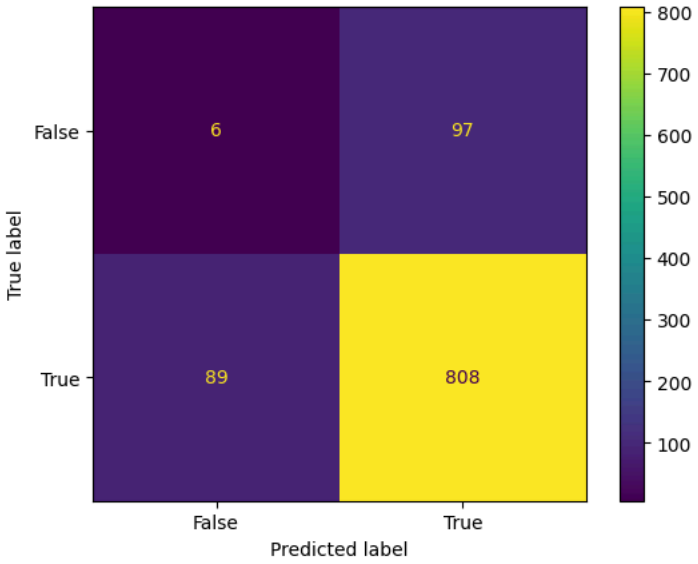
```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline
import matplotlib.pyplot as plt

from sklearn.metrics import roc_curve, auc
import numpy as np


# smaller training set to make the curve more interesting
X_train2 = X_train[:, [4, 14]]

pipe_knn = make_pipeline(StandardScaler(),
                         KNeighborsClassifier())


fig = plt.figure(figsize=(7, 5))


#############################################################
### TRAINING ROC CURVE
train_probas = pipe_knn.fit(X_train2,
                            y_train).predict_proba(X_train2)

fpr, tpr, thresholds = roc_curve(y_train,
                                 train_probas[:, 1],
                                 pos_label=1)
roc_auc = auc(fpr, tpr)

plt.step(fpr,
         tpr,
         label='Train ROC (area = %0.2f)'
               % (roc_auc))
#############################################################


#############################################################
### TEST ROC CURVE
test_probas = pipe_knn.predict_proba(X_test[:, [4, 14]])

fpr, tpr, thresholds = roc_curve(y_test,
                                 test_probas[:, 1],
                                 pos_label=1)
roc_auc = auc(fpr, tpr)

plt.step(fpr,
         tpr,
         where='post',
         label='Test ROC (area = %0.2f)'
               % (roc_auc))
#############################################################

plt.plot([0, 1],
         [0, 1],
         linestyle='--',
         color=(0.6, 0.6, 0.6),
         label='Random guessing')

plt.plot([0, 0, 1],
         [0, 1, 1],
         linestyle=':',
         color='black',
         label='Perfect performance')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.legend(loc="lower right")

plt.tight_layout()
plt.show()
```

Follow link (ctrl + click)



Follow link (ctrl + click)

```
import matplotlib.pyplot as plt
import numpy
from sklearn import metrics

actual = numpy.random.binomial(1,.9,size = 1000)
predicted = numpy.random.binomial(1,.9,size = 1000)

confusion_matrix = metrics.confusion_matrix(actual, predicted)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```



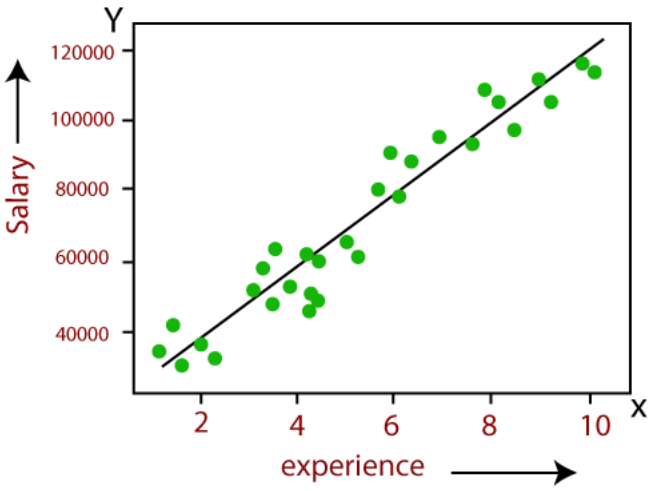
K-Means-ի իրականացում

Double-click (or enter) to edit

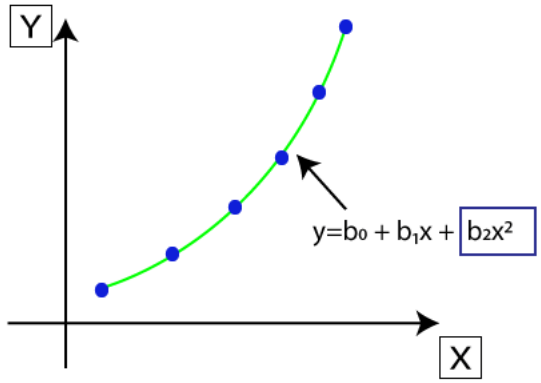
Lab 10. Regression Models

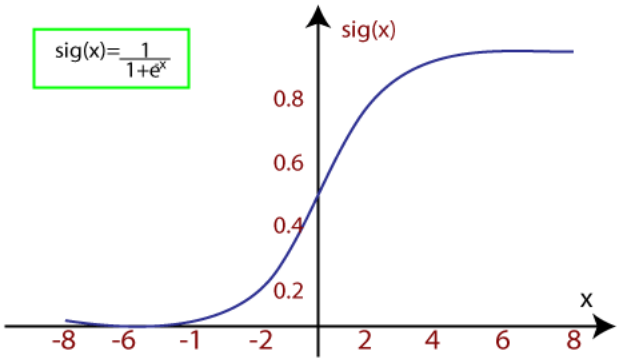
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

- List item



Follow link (ctrl + click)






```
#import numpy as np
#import matplotlib.pyplot as plt
#noise = np.random.normal(0,1,50)
#from sklearn.linear_model import LinearRegression
```

```
#import numpy as np
#generate x np array with 50 items, range (0,10)
#use np.linspace function
import numpy as np
x = np.linspace(0,10,50)
x
```

```
array([ 0.        ,  0.20408163,  0.40816327,  0.6122449 ,  0.81632653,
        1.02040816,  1.2244898 ,  1.42857143,  1.63265306,  1.83673469,
```

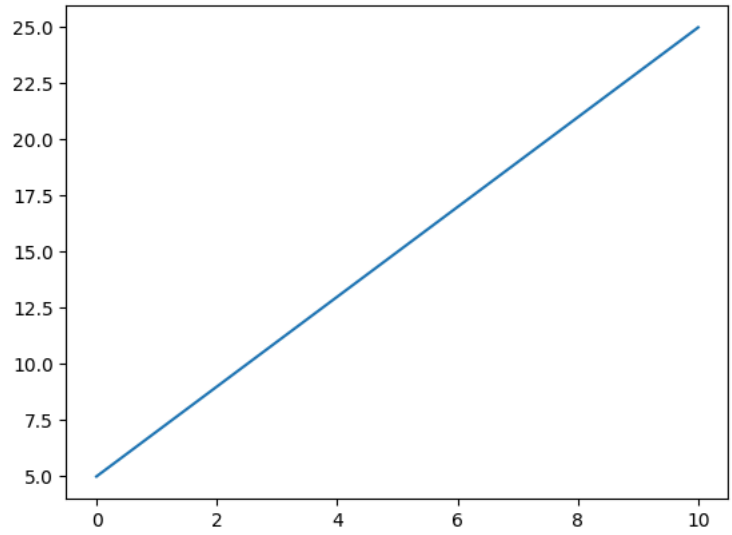
```
       2.04081633,  2.24489796,  2.44897959,  2.65306122,  2.85714286,
       3.06122449,  3.26530612,  3.46938776,  3.67346939,  3.87755102,
       4.08163265,  4.28571429,  4.48979592,  4.69387755,  4.89795918,
       5.10204082,  5.30612245,  5.51020408,  5.71428571,  5.91836735,
       6.12244898,  6.32653061,  6.53061224,  6.73469388,  6.93877551,
       7.14285714,  7.34693878,  7.55102041,  7.75510204,  7.95918367,
       8.16326531,  8.36734694,  8.57142857,  8.7755102 ,  8.97959184,
       9.18367347,  9.3877551 ,  9.59183673,  9.79591837, 10.        ])
```

Follow link (ctrl + click)

```
#Linear function
y_l = 2*x+5
y_p = 2*x**2+1
```

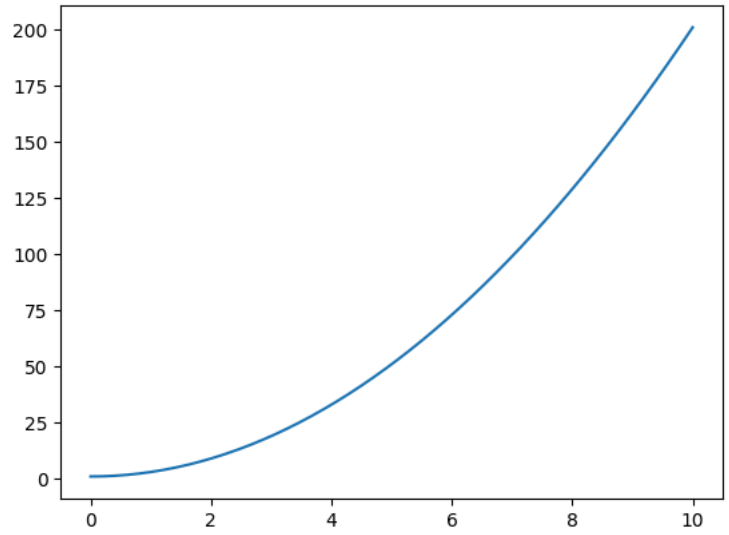
```
#Visualize functions
import matplotlib.pyplot as plt
plt.plot(x,y_l)
```

```
[<matplotlib.lines.Line2D at 0x7a3269b47790>]
```




```
#Visualize functions
plt.plot(x,y_p)
```

```
[<matplotlib.lines.Line2D at 0x7a3269951ed0>]
```




```
noise = np.random.normal(0,2,50)
noise
```

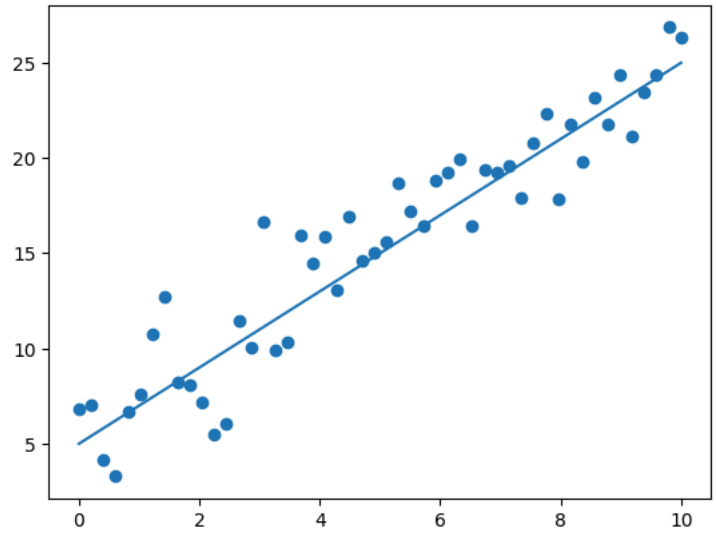
```
array([ 1.79547616,  1.59378962, -1.66543361, -2.94475593,  0.04130455,
        0.54226051,  3.28660217,  4.8296459 , -0.03732696, -0.58691974,
       -1.87680131, -3.9659396 , -3.82679181,  1.15710474, -0.6375356 ,
        5.49348791, -1.64441136, -1.57849122,  3.57651556,  1.71077493,
        2.70514358, -0.51034516,  2.95367541,  0.21979741,  0.24682601,
```

```
       0.40079053,  3.09002759,  1.20325383,  0.04097001,  2.01411716,
       1.99585326,  2.29754534, -1.63278322,  0.92037596,  0.36847416,
       0.33456527, -1.80330447,  0.66406458,  1.81812973, -3.10898007,
       0.42694391, -1.93753198,  1.04114722, -0.79990414,  1.41223338,
      -2.23379824, -0.31952979,  0.18964401,  2.26971547,  1.34976151])
```

```
#data points visualization
y_noise = y_l+noise
plt.plot(x,y_l)
plt.scatter(x,y_noise)
```

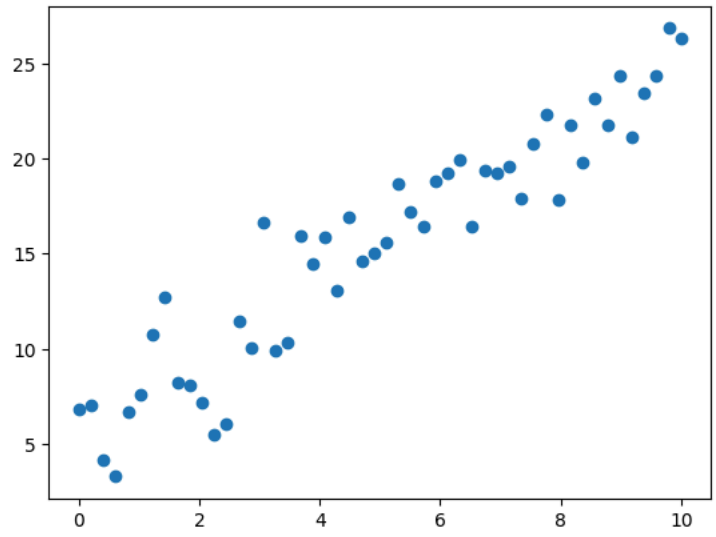
Follow link (ctrl + click)

```
<matplotlib.collections.PathCollection at 0x7a32699e6530>
```




```
plt.scatter(x,y_noise)
```

```
<matplotlib.collections.PathCollection at 0x7a3269655270>
```




```
X = x.reshape(-1,1)
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression().fit(X,y_noise)
```

```
lr
```

```
▾ LinearRegression
LinearRegression()
```

```
lr.coef_
```

```
array([1.98740531])
```

```
lr.intercept_
```

```
5.480562106155217
```

```
y_reg = x*lr.coef_[0]+lr.intercept_
y_reg
```

```
array([ 5.48056211,  5.88615503,  6.29174795,  6.69734087,  7.10293379,
        7.50852671,  7.91411963,  8.31971255,  8.72530547,  9.13089839,
        9.53649131,  9.94208423, 10.34767715, 10.75327008, 11.158863  ,
       11.56445592, 11.97004884, 12.37564176, 12.78123468, 13.1868276 ,
       13.59242052, 13.99801344, 14.40360636, 14.80919928, 15.2147922 ,
       15.62038512, 16.02597804, 16.43157097, 16.83716389, 17.24275681,
       17.64834973, 18.05394265, 18.45953557, 18.86512849, 19.27072141,
       19.67631433, 20.08190725, 20.48750017, 20.89309309, 21.29868601,
       21.70427893, 22.10987186, 22.51546478, 22.9210577 , 23.32665062,
       23.73224354, 24.13783646, 24.54342938, 24.9490223 , 25.35461522])
```

Follow link (ctrl + click)

```
y_pred = lr.predict(X)
y_pred
```

```
array([ 5.48056211,  5.88615503,  6.29174795,  6.69734087,  7.10293379,
        7.50852671,  7.91411963,  8.31971255,  8.72530547,  9.13089839,
        9.53649131,  9.94208423, 10.34767715, 10.75327008, 11.158863  ,
       11.56445592, 11.97004884, 12.37564176, 12.78123468, 13.1868276 ,
       13.59242052, 13.99801344, 14.40360636, 14.80919928, 15.2147922 ,
       15.62038512, 16.02597804, 16.43157097, 16.83716389, 17.24275681,
       17.64834973, 18.05394265, 18.45953557, 18.86512849, 19.27072141,
       19.67631433, 20.08190725, 20.48750017, 20.89309309, 21.29868601,
       21.70427893, 22.10987186, 22.51546478, 22.9210577 , 23.32665062,
       23.73224354, 24.13783646, 24.54342938, 24.9490223 , 25.35461522])
```

Start coding or generate with AI.

```
x.shape
```

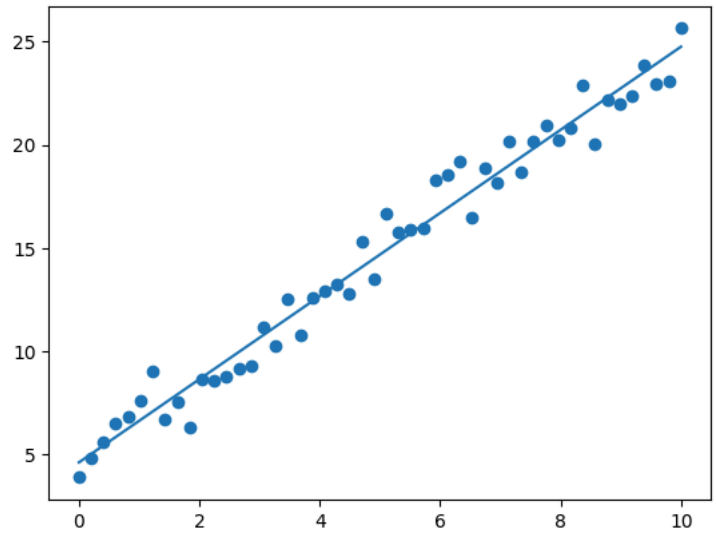
```
(50, 1)
```

```
from sklearn.linear_model import LinearRegression
```

```
lr = LinearRegression().fit(x,y_noise)
```

```
plt.scatter(x,y_noise)
plt.plot(x,y)
```

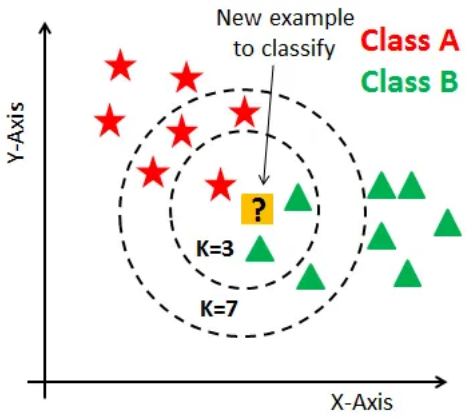
```
[<matplotlib.lines.Line2D at 0x7f40ef698670>]
```



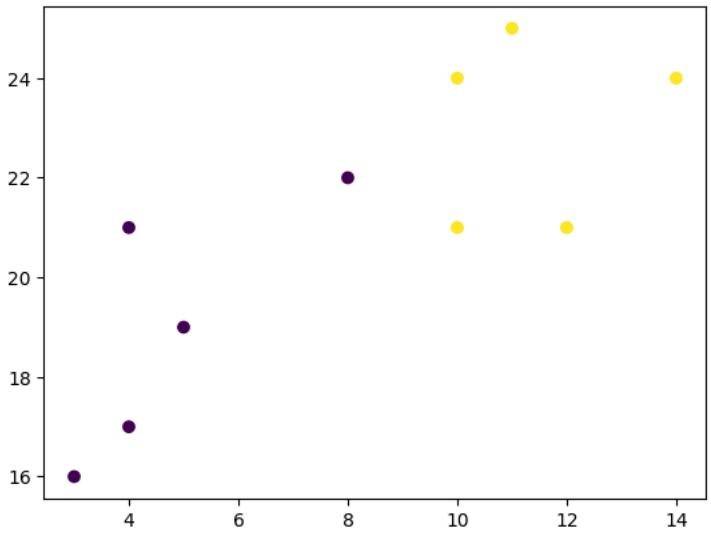
Start coding or generate with AI.

Lab 11. KNN

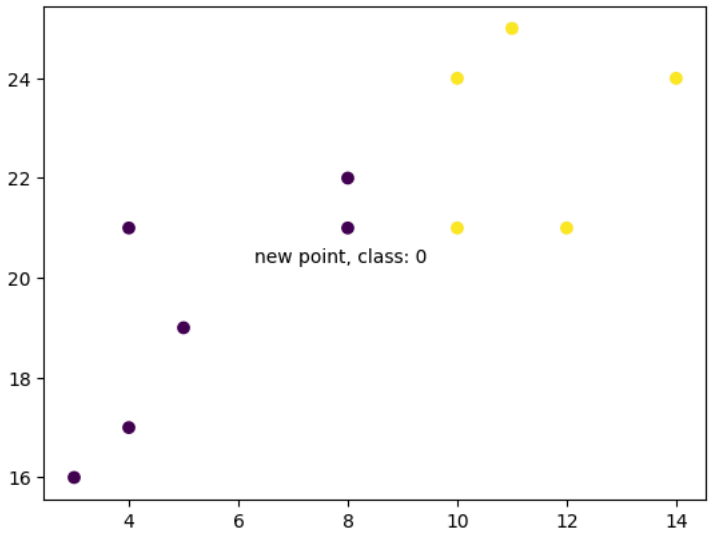
1. https://www.ibm.com/topics/knn#:~:text=the%20next%20step-,K%2DNearest%20Neighbors%20Algorithm,of%20an%20individual%20data%20point.
2. https://www.studocu.com/in/document/jawaharlal-nehru-technological-university-anantapur/computer-science-engineering/k-nearest-neighbor-its-really-helpful-for-the-learners/5402239
3. https://raw.githubusercontent.com/meuwebsite/T-shirtSizes--Classification/master/Tshirt_Sizing_Dataset.csv Follow link (ctrl + click)
4. https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn
5. https://towardsdatascience.com/getting-acquainted-with-k-nearest-neighbors-ba0a9ecf354f



Start coding or generate with AI.




```
from sklearn.neighbors import KNeighborsClassifier


#X =[[]]
knn = KNeighborsClassifier().fit(X, y)
knn.predict(X_new)
```

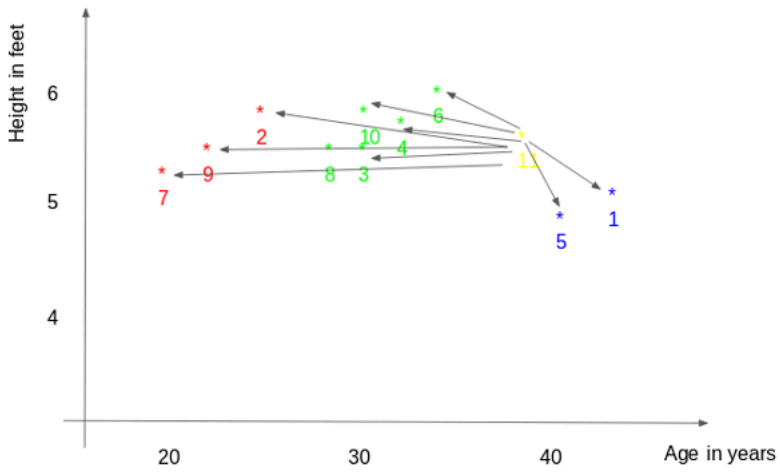


Follow link (ctrl + click)

Start coding or generate with AI.

Lab 12. KNN from scratch

- Distance Metrics- https://medium.com/mlearning-ai/a-brief-introduction-to-distance-measures-ac89cbd2298
- https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/




```
import numpy as np

def euclidean_distance(a, b):
  """
  Calculates the Euclidean distance between two points.

  Args:
    a: A numpy array representing the first point.
    b: A numpy array representing the second point.

  Returns:
    The Euclidean distance between the two points.
  """
  distance = ((b[0] - a[0])**2 + (b[1] - a[1])**2)**0.5
  return distance

def distances(a, points):
  distance = ((points[:,0] - a[0])**2 + (points[:,1] - a[1])**2)**0.5
  return distance
```

Start coding or generate with AI.

Lab 13.Naive Bayes Classification

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Follow link (ctrl + click)

- P(h): the probability of hypothesis h being true (regardless of the data). This is known as the prior probability of h.
- P(D): the probability of the data (regardless of the hypothesis). This is known as the prior probability.
- P(h|D): the probability of hypothesis h given the data D. This is known as posterior probability.
- P(D|h): the probability of data d given that the hypothesis h was true. This is known as posterior probability.

Naive Bayes classifier calculates the probability of an event in the following steps:

- **Step 1**: Calculate the prior probability for given class labels
- **Step 2**: Find Likelihood probability with each attribute for each class
- **Step 3**: Put these value in Bayes Formula and calculate posterior probability.
- **Step 4**: See which class has a higher probability, given the input belongs to the higher probability class.

Whether	Play
Sunny	No
Sunny	No
Overcast	Yes
Rainy	Yes
Rainy	Yes
Rainy	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table

Whether	No	Yes
Overcast		4
Sunny	2	3
Rainy	3	2
Total	5	9

Likelihood Table 1

Whether	No	Yes		
Overcast		4	=4/14	0.29
Sunny	2	3	=5/14	0.36
Rainy	3	2	=5/14	0.36
Total	5	9		
	=5/14	=9/14		
	0.36	0.64		

Likelihood Table 2

Whether	No	Yes	Posterior Probability for No	Posterior Probability for Yes
Overcast		4	0/5=0	4/9=0.44
Sunny	2	3	2/5=0.4	3/9=0.33
Rainy	3	2	3/5=0.6	2/9=0.22
Total	5	9		

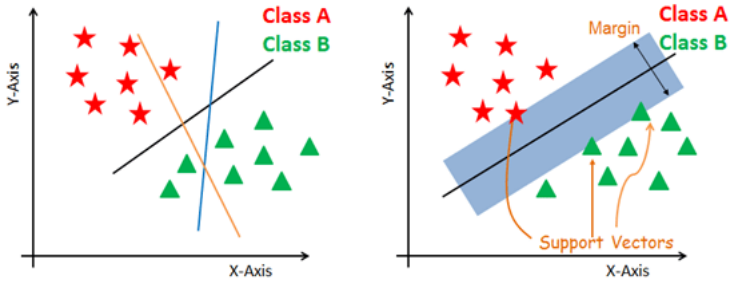
```
from sklearn.naive_bayes import GaussianNB
#make _blobs
```

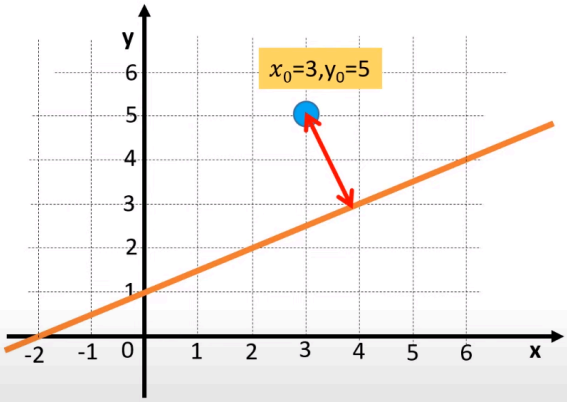
Start coding or generate with AI.

Lab 14. Support Vector Machines

Follow link (ctrl + click)

1. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/
2. https://www.geeksforgeeks.org/support-vector-machine-algorithm/
3. https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python
4. https://youtu.be/_YPScrckx28
5. https://www.youtube.com/watch?v=1NxnPkZM9bc
6. https://www.youtube.com/watch?v=gUzEN2TxnxE



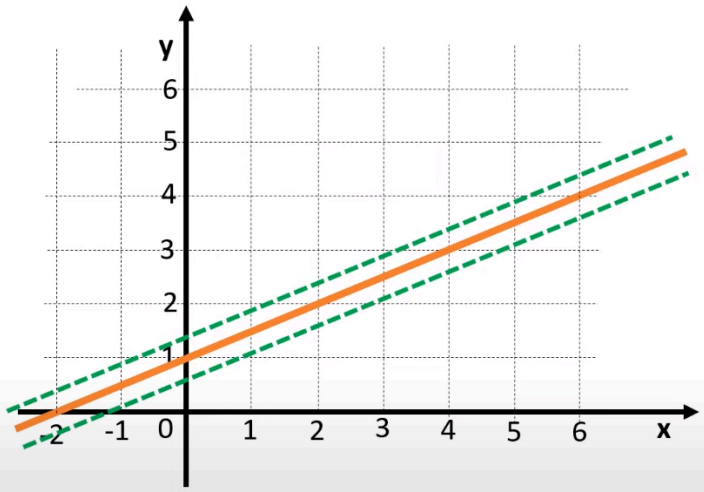


$$Ax + By + C = 0$$

$$-2x + 4y - 4 = 0$$

$$d = \frac{|Ax_0 + By_o + C|}{\sqrt{A^2 + B^2}}$$

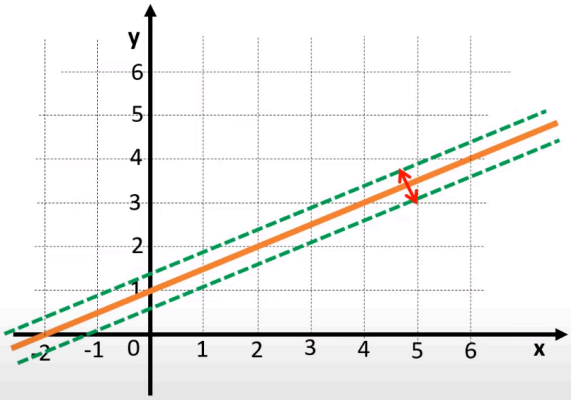
$$d = \frac{|(-2) \cdot 3 + 4 \cdot 5 + (-4)|}{\sqrt{(-2)^2 + 4^2}} = \frac{10}{\sqrt{20}} = 2.236$$



$$Ax + By + C = 0$$

$$-2x + 4y - 4 = 1$$
$$-2x + 4y - 4 = 0$$
$$-2x + 4y - 4 = -1$$

Follow link (ctrl + click)

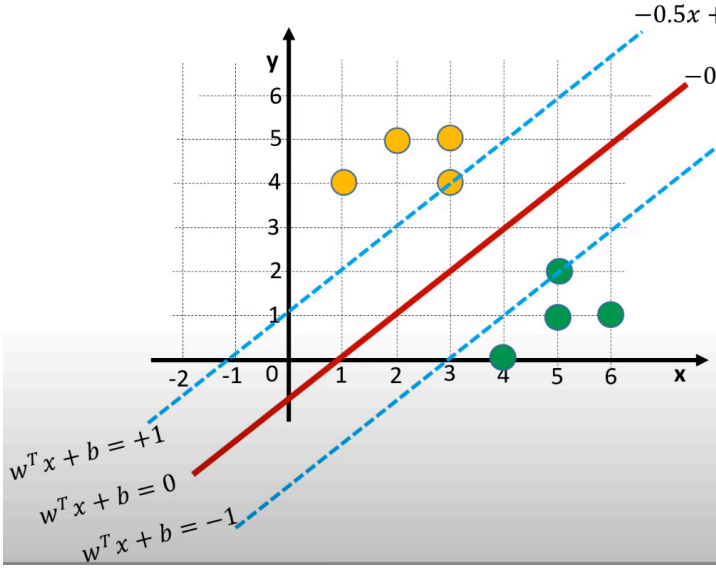


$$Ax + By + C = 0$$

$$-2x + 4y - 4 = 1$$
$$-2x + 4y - 4 = 0$$
$$-2x + 4y - 4 = -1$$

$$d = \frac{|C_2 - C_1|}{\sqrt{A^2 + B^2}}$$

$$d = \frac{|(-3) - (-5)|}{\sqrt{(-2)^2 + 4^2}} = \frac{2}{\sqrt{20}} = 0.447$$

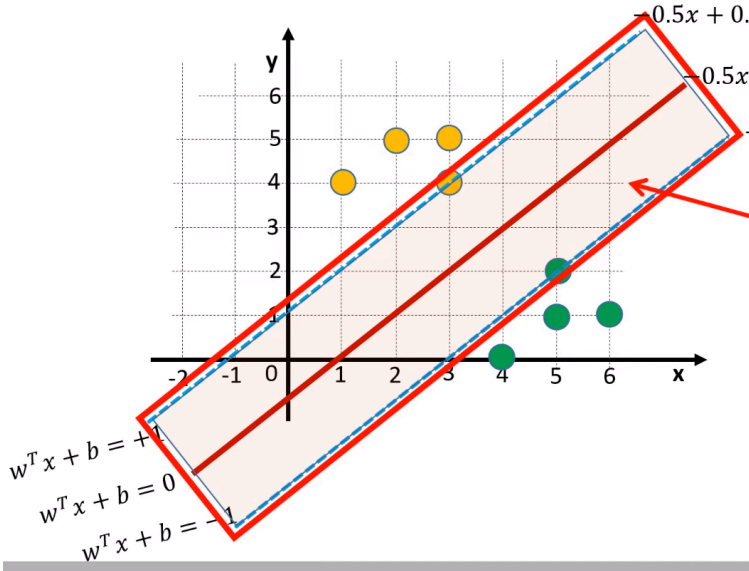


$$-0.5x + 0.5y + b = 1$$
$$-0.5x + 0.5y + b = 0$$
$$-0.5x + 0.5y + b = -1$$

$$b = 0.5$$

Group (Y_i)	x	y
A (+1)	1	4
A (+1)	2	5
A (+1)	3	5
A (+1)	3	4
B (-1)	6	1
B (-1)	4	0
B (-1)	5	2
B (-1)	5	1

$$Y \begin{cases} +1 \text{ if } -0.5x + 0.5y + b \geq 0 \\ -1 \text{ if } -0.5x + 0.5y + b < 0 \end{cases}$$



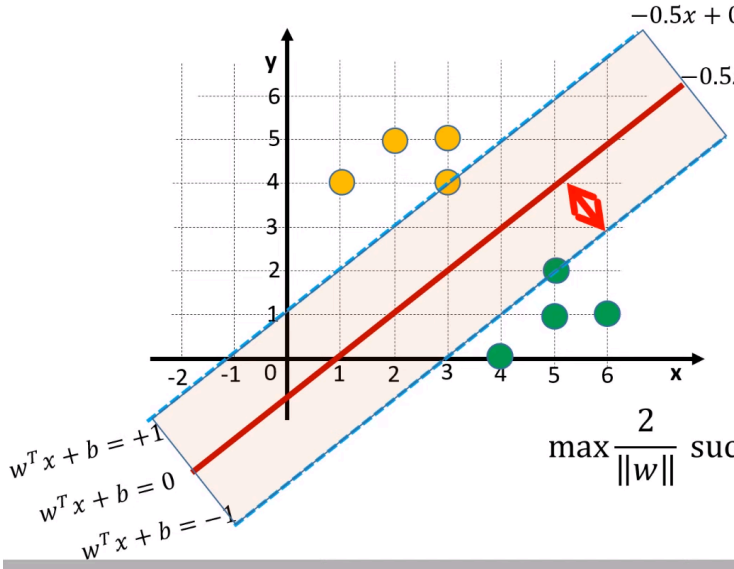
$b = 0.5$

Group (Y_i)	x	y
A (+1)	1	4
A (+1)	2	5
A (+1)	3	5
A (+1)	3	4
B (-1)	6	1
B (-1)	4	0
B (-1)	5	2
B (-1)	5	1

Follow link (ctrl + click)

$$d = \frac{2}{\sqrt{A^2 + B^2}} = \frac{2}{\|w\|}$$

The area between the two blue lines is called the margin. The SVM therefore tries to find a hyperplane that maximizes the width of this margin.



$b = 0.5$

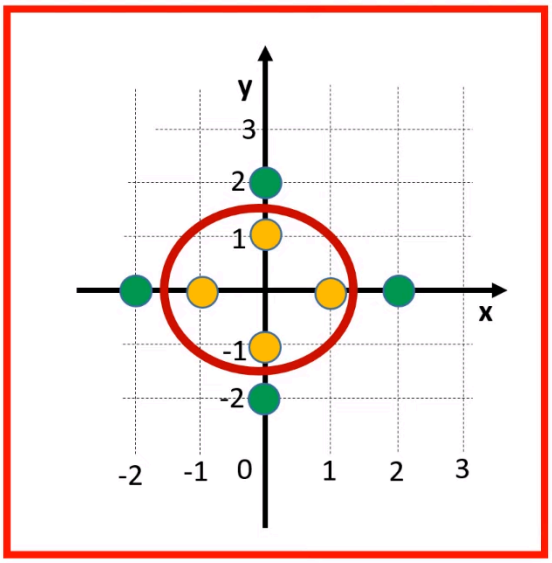
Group (Y_i)	x	y
A (+1)	1	4
A (+1)	2	5
A (+1)	3	5
A (+1)	3	4
B (-1)	6	1
B (-1)	4	0
B (-1)	5	2
B (-1)	5	1

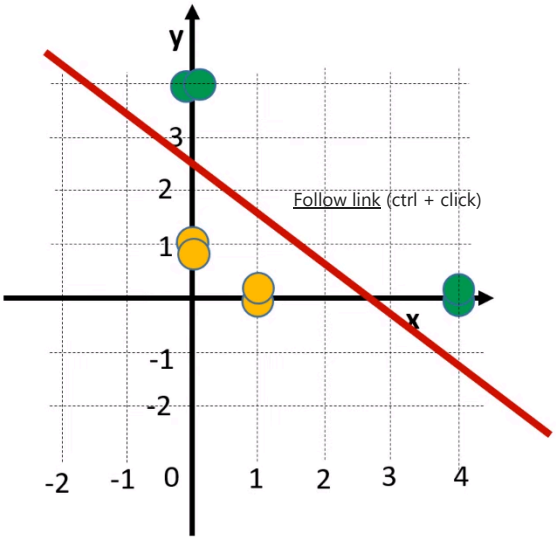
$$-0.5x + 0.5y + 0.5 = 0$$

$$-0.5 \cdot 5 + 0.5 \cdot 2 + 0.5 = -1$$

$$\max \frac{2}{\|w\|} \text{ such that } w^T x_i + b \begin{matrix} \geq 1 & \text{if } Y_i = +1 \\ \leq -1 & \text{if } Y_i = -1 \end{matrix}$$

The width of the margin is therefore constrained so that it cannot span beyond the support vector if the hyperplane can separate the two groups completely.





Lab 14. K-means

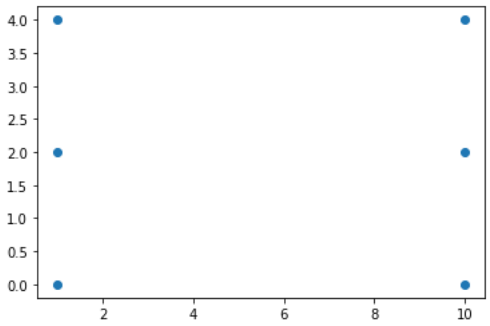
1. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1
2. https://medium.com/swlh/how-to-choose-the-right-number-of-clusters-in-the-k-means-algorithm-9160c57ec760
3. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
4. https://www.naftaliharris.com/blog/visualizing-k-means-clustering/
5. https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html#sphx-glr-auto-examples-cluster-plot-kmeans-digits-py
6. https://realpython.com/k-means-clustering-python/

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, init= "k-means++",random_state=0)
```

```
kmeans.fit(X)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
```

```
KMeans(n_clusters=2, random_state=0)
```



Double-click (or enter) to edit

Lab 15- Miniproject

https://github.com/mrdbourke/zero-to-mastery-ml/blob/master/section-3-structured-data-projects/end-to-end-heart-disease-classification.ipynb

Start coding or generate with AI.

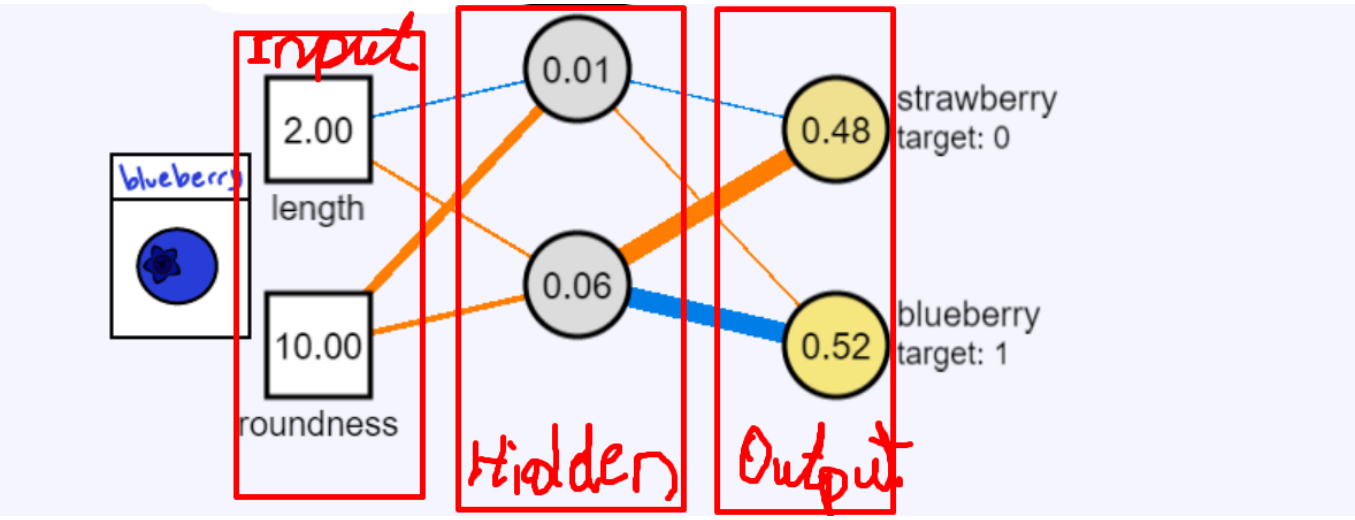
Follow link (ctrl + click)

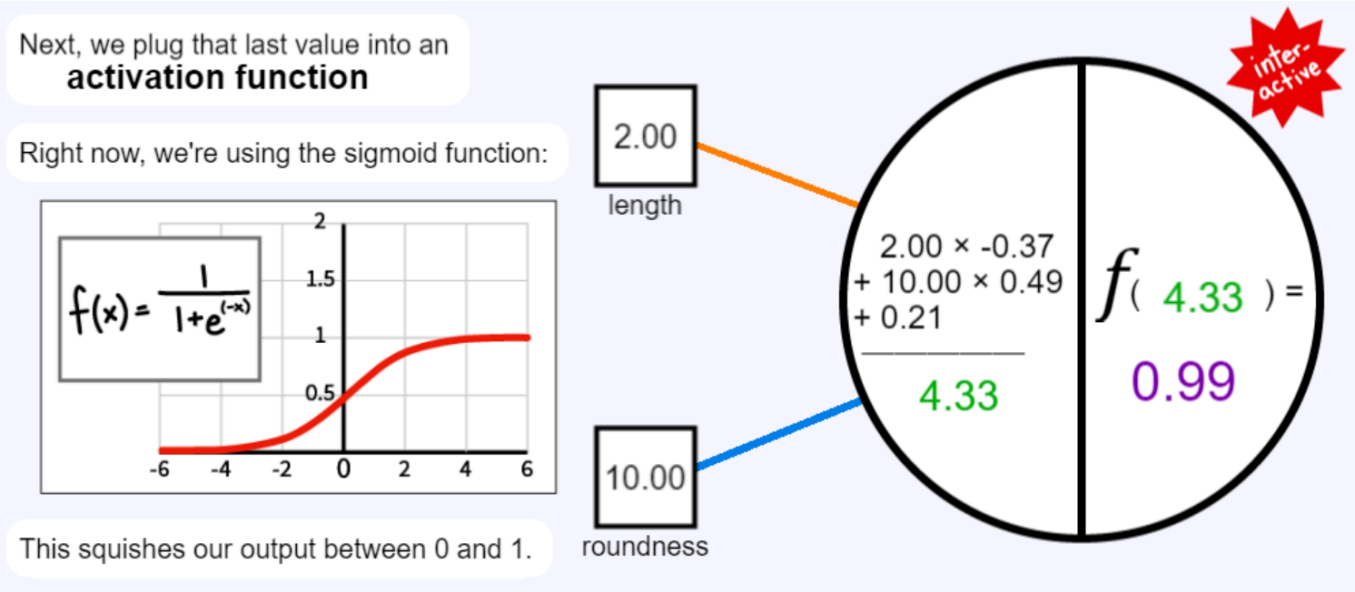
Lab 16. Neural Network Implementation

- https://aegeorge42.github.io/
- https://edu.anarcho-copy.org/Algorithm/grokking-deep-learning.pdf

Start coding or generate with AI.

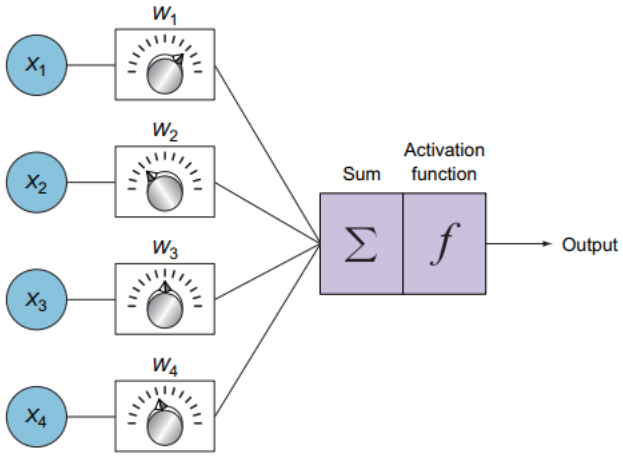
LAB-Local Search,Gradient Descent and Neural Network

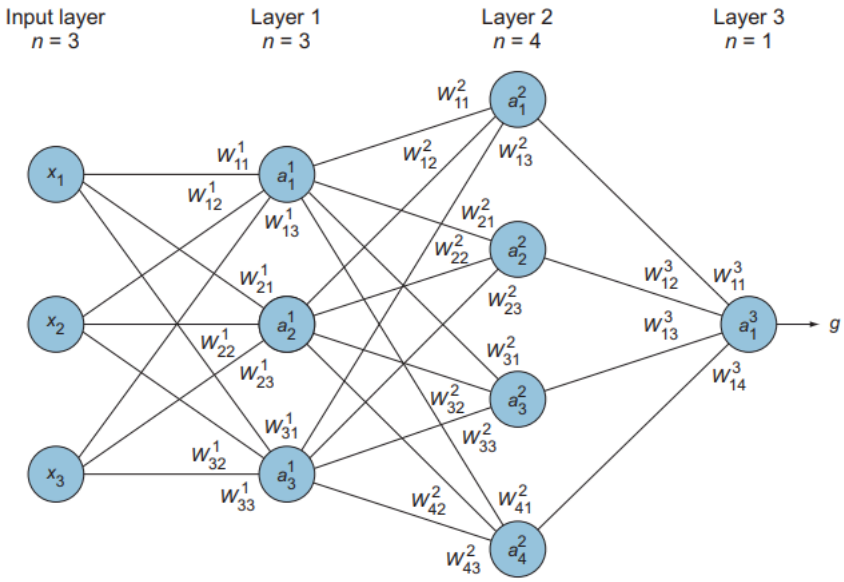




```
import numpy as np
#Sigmoid function as activation function
def sigmoid(x):
  return 1/(1+np.exp(-x))
#Derivative of sigmoid
def sigmoid_derivative(x):
  return x*(1-x)
```

Follow link (ctrl + click)





We have all we need to start the feedforward calculations:

$$a_1^{(1)} = \sigma(w_{11}^{(1)}\, x_1 + w_{21}^{(1)}\, x_2 + w_{31}^{(1)}\, x_3)$$

$$a_2^{(1)} = \sigma(w_{12}^{(1)}\, x_1 + w_{22}^{(1)}\, x_2 + w_{32}^{(1)}\, x_3)$$

$$a_3^{(1)} = \sigma(w_{13}^{(1)}\, x_1 + w_{23}^{(1)}\, x_2 + w_{33}^{(1)}\, x_3)$$

Follow link (ctrl + click)

Then we do the same calculations for layer 2

$$a_1^{(2)}, a_2^{(2)}, a_3^{(2)}, \text{and } a_4^{(2)}$$

all the way to the output prediction in layer 3:

$$\hat{y} = a_1^{(2)} = \sigma\,(w_{11}^{(3)}\, a_1^{(2)} + w_{12}^{(3)}\, a_2^{(2)} + w_{13}^{(3)}\, a_3^{(2)} + w_{14}^{(3)}\, a_4^{(2)})$$

The feedforward process

$$\hat{y} = \sigma \begin{bmatrix} w_{11}^3 & w_{12}^3 & w_{13}^3 & w_{14}^3 \end{bmatrix} \cdot \sigma \begin{bmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \\ w_{31}^2 & w_{32}^2 & w_{33}^2 \\ w_{41}^2 & w_{42}^2 & w_{43}^2 \end{bmatrix} \cdot \sigma \begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \\ w_{31}^1 & w_{32}^1 & w_{33}^1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$W^{(3)}$ $W^{(2)}$ $W^{(1)}$

Layer 3 Layer 2 Layer 1 Input vector

The number we'll use to adjust the weight

$$\Delta W = \sum_{i=1}^{n} \frac{\partial J}{\partial W}(x_i, y_i)$$

$$\frac{\partial J}{\partial W}(x_i, y_i) = \frac{1}{n} \sum_{i=1}^{n} x_i(\hat{y}_i - y_i)$$

Derivative of MSE


```
def sigmoid(x):
  return 1/(1+np.exp(-x))
```

```
class NeuralNetwork:
  def __init__(self, input_size, hidden_size,output_size):
    pass
  def forward(self,inputs):
    pass
  def backward(self,targets, learning_rate):
    pass
  def train(self, inputs, targets, learning_rate, epoch)
    pass
```

Follow link (ctrl + click)

Lab 17. Grid Search

- https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/
- https://www.datacamp.com/tutorial/parameter-optimization-machine-learning-models
- https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/

```
from sklearn import metrics
sorted(metrics.SCORERS.keys())
```

```
['accuracy',
 'adjusted_mutual_info_score',
 'adjusted_rand_score',
 'average_precision',
 'balanced_accuracy',
 'completeness_score',
 'explained_variance',
 'f1',
 'f1_macro',
 'f1_micro',
 'f1_samples',
 'f1_weighted',
 'fowlkes_mallows_score',
 'homogeneity_score',
 'jaccard',
 'jaccard_macro',
 'jaccard_micro',
 'jaccard_samples',
 'jaccard_weighted',
 'matthews_corrcoef',
 'max_error',
 'mutual_info_score',
 'neg_brier_score',
 'neg_log_loss',
 'neg_mean_absolute_error',
 'neg_mean_absolute_percentage_error',
 'neg_mean_gamma_deviance',
 'neg_mean_poisson_deviance',
 'neg_mean_squared_error',
 'neg_mean_squared_log_error',
 'neg_median_absolute_error',
 'neg_negative_likelihood_ratio',
 'neg_root_mean_squared_error',
 'normalized_mutual_info_score',
 'positive_likelihood_ratio',
 'precision',
 'precision_macro',
 'precision_micro',
 'precision_samples',
 'precision_weighted',
 'r2',
 'rand_score',
 'recall',
 'recall_macro',
 'recall_micro',
 'recall_samples',
 'recall_weighted',
 'roc_auc',
 'roc_auc_ovo',
 'roc_auc_ovo_weighted',
 'roc_auc_ovr',
 'roc_auc_ovr_weighted',
 'top_k_accuracy',
 'v_measure_score']
```

```
data = pd.read_csv("https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv",header=None)
print((data[[1,2,3,4,5]] == 0).sum())
```

```
#Model #1:
 GridSearchCV(
    estimator = RandomForestClassifier(),
    param_grid = {'max_depth': [2, 4, 8, 15], 'max_features': ['auto', 'sqrt']},
    scoring='roc_auc',
    n_jobs=4,
    cv=5,
    refit=True, return_train_score=True)
```

Follow link (ctrl + click)

```
#Model #2:
 GridSearchCV(
    estimator = KNeighborsClassifier(),
    param_grid = {'n_neighbors': [5, 10, 20], 'algorithm': ['ball_tree', 'brute']},
    scoring='accuracy',
    n_jobs=8,
    cv=10,
    refit=False)
```

```
#Model #3:
 GridSearchCV(
    estimator = GradientBoostingClassifier(),
    param_grid = {'number_attempts': [2, 4, 6], 'max_depth': [3, 6, 9, 12]},
    scoring='accuracy',
    n_jobs=2,
    cv=7,
    refit=True)
```

```
============================================================
```

Others

Data Formats(CSV, JSON, XML)

```
print("Data Engineering")
```

```
     Data Engineering
```

```
import pandas as pd
df = pd.read_json("/content/sample_data/anscombe.json")
df
```

	Series	X	Y
0	I	10	8.04
1	I	8	6.95
2	I	13	7.58
3	I	9	8.81
4	I	11	8.33
5	I	14	9.96
6	I	6	7.24
7	I	4	4.26
8	I	12	10.84
9	I	7	4.81
10	I	5	5.68
11	II	10	9.14
12	II	8	8.14
13	II	13	8.74
14	II	9	8.77
15	II	11	9.26
16	II	14	8.10
17	II	6	6.13
18	II	4	3.10
19	II	12	9.13
20	II	7	7.26
21	II	5	4.74
22	III	10	7.46
23	III	8	6.77
24	III	13	12.74
25	III	9	7.11
26	III	11	7.81
27	III	14	8.84
28	III	6	6.08
29	III	4	5.39
30	III	12	8.15

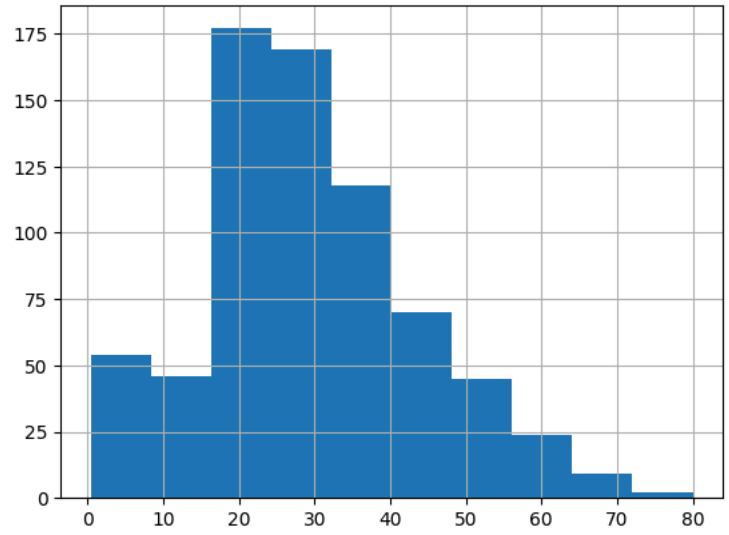
Follow link (ctrl + click)

```
df_tit = pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv")
df_tit.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S

```
df_tit['Age'].hist()
```

```
<Axes: >
```



Follow link (ctrl + click)

```
df_xml=pd.read_xml("https://www.w3schools.com/xml/simple.xml")
df_xml
```

	name	price	description	calories
0	Belgian Waffles	$5.95	Two of our famous Belgian Waffles with plenty ...	650
1	Strawberry Belgian Waffles	$7.95	Light Belgian waffles covered with strawberrie...	900
2	Berry-Berry Belgian Waffles	$8.95	Light Belgian waffles covered with an assortme...	900
3	French Toast	$4.50	Thick slices made from our homemade sourdough ...	600
4	Homestyle Breakfast	$6.95	Two eggs, bacon or sausage, toast, and our eve...	950

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

JSON Parsing

```
import json
student = {
    "name":"John",
    "age":25,
    "email": "john@student.com"
}
#Write content to file
with open("student.json","w") as write_file:
  json.dump(student,write_file)
#Read content from file
f = open("/content/student.json")
json_content = json.load(f)
json_content
```

```
{'name': 'John', 'age': 25, 'email': 'john@student.com'}
```

API

```
import requests

url = "https://ai-weather-by-meteosource.p.rapidapi.com/find_places"

querystring = {"text":"fishermans wharf","language":"en"}

headers = {
    "X-RapidAPI-Key": "90259ec543mshc0489c0f1fd8a95p1c6e5djsnfa79bbdbc717",
    "X-RapidAPI-Host": "ai-weather-by-meteosource.p.rapidapi.com"
}

response = requests.get(url, headers=headers, params=querystring)

print(response.json())
```

Follow link (ctrl + click)

```
[{'name': 'Fishermans Wharf', 'place_id': 'fishermans-wharf-6930485', 'adm_area1': 'California', 'adm_area2': 'Contra Costa', 'country'
```

```
import requests
url = "https://gist.githubusercontent.com/rominirani/8235702/raw/a50f7c449c41b6dc8eb87d8d393eeff62121b392/employees.json"
response = requests.get(url)
response
```

```
<Response [200]>
```

```
print(response.status_code)
```

```
200
```

```
#print(response.content)
```

```
print(response.json())
```

```
{'Employees': [{'userId': 'rirani', 'jobTitleName': 'Developer', 'firstName': 'Romin', 'lastName': 'Irani', 'preferredFullName': 'Romin
```

```
type(response.json())
```

```
dict
```

```
response.json()["Employees"]
```

```
[{'userId': 'rirani',
  'jobTitleName': 'Developer',
  'firstName': 'Romin',
  'lastName': 'Irani',
  'preferredFullName': 'Romin Irani',
  'employeeCode': 'E1',
  'region': 'CA',
  'phoneNumber': '408-1234567',
  'emailAddress': 'romin.k.irani@gmail.com'},
 {'userId': 'nirani',
  'jobTitleName': 'Developer',
  'firstName': 'Neil',
  'lastName': 'Irani',
  'preferredFullName': 'Neil Irani',
  'employeeCode': 'E2',
  'region': 'CA',
  'phoneNumber': '408-1111111',
  'emailAddress': 'neilrirani@gmail.com'},
 {'userId': 'thanks',
  'jobTitleName': 'Program Directory',
  'firstName': 'Tom',
  'lastName': 'Hanks',
  'preferredFullName': 'Tom Hanks',
  'employeeCode': 'E3',
  'region': 'CA',
  'phoneNumber': '408-2222222',
  'emailAddress': 'tomhanks@gmail.com'}]
```

```
for item in response.json()["Employees"]:
  print(item)
```

```
{'userId': 'rirani', 'jobTitleName': 'Developer', 'firstName': 'Romin', 'lastName': 'Irani', 'preferredFullName': 'Romin Irani', 'emplo
{'userId': 'nirani', 'jobTitleName': 'Developer', 'firstName': 'Neil', 'lastName': 'Irani', 'preferredFullName': 'Neil Irani', 'employe
```

```
{'userId': 'thanks', 'jobTitleName': 'Program Directory', 'firstName': 'Tom', 'lastName': 'Hanks', 'preferredFullName': 'Tom Hanks', 'e
```

```
import requests
url = "https://youthgrowth.in/dikw-data-information-knowledge-and-wisdom/"
response = requests.get(url)
response
```

```
<Response [200]>
```

Follow link (ctrl + click)

```
print(response.content)
```

```
b'<!DOCTYPE html><html dir="ltr" lang="en-US" prefix="og: https://ogp.me/ns#"><head><meta charset="UTF-8"><link rel="profile" href="htt
```

```
# import urllib library
from urllib.request import urlopen

# import json
import json
# store the URL in url as
# parameter for urlopen
url = "https://www.omdbapi.com/?i=tt3896198&apikey=31a5073f&t=gladiator"

# store the response of URL
response = urlopen(url)

# storing the JSON response
# from url in data
data_json = json.loads(response.read())

# print the json response
print(json.dumps(data_json, indent=2, sort_keys=True))
```


```
{
  "Actors": "Russell Crowe, Joaquin Phoenix, Connie Nielsen",
  "Awards": "Won 5 Oscars. 60 wins & 104 nominations total",
  "BoxOffice": "$187,705,427",
  "Country": "United States, United Kingdom, Malta, Morocco",
  "DVD": "15 Jun 2011",
  "Director": "Ridley Scott",
  "Genre": "Action, Adventure, Drama",
  "Language": "English",
  "Metascore": "67",
  "Plot": "A former Roman General sets out to exact vengeance against the corrupt emperor who murdered his family and sent him into sla
  "Poster": "https://m.media-amazon.com/images/M/MV5BMDliMmNhNDEtODUyOS00MjNlLTgxODEtN2U3NzIxMGVkZTA1L2ltYWdlXkEyXkFqcGdeQXVyNjU0OTQ0OT
  "Production": "N/A",
  "Rated": "R",
  "Ratings": [
    {
      "Source": "Internet Movie Database",
      "Value": "8.5/10"
    },
    {
      "Source": "Rotten Tomatoes",
      "Value": "80%"
    },
    {
      "Source": "Metacritic",
      "Value": "67/100"
    }
  ],
  "Released": "05 May 2000",
  "Response": "True",
  "Runtime": "155 min",
  "Title": "Gladiator",
  "Type": "movie",
  "Website": "N/A",
  "Writer": "David Franzoni, John Logan, William Nicholson",
  "Year": "2000",
  "imdbID": "tt0172495",
  "imdbRating": "8.5",
  "imdbVotes": "1,566,334"
}
```


Start coding or generate with AI.

Web Scraping (Beautiful Soup)

```
import requests
url = "https://realpython.github.io/fake-jobs/"
page = requests.get(url)
print(page)
```

Follow link (ctrl + click)

```
<Response [200]>
```

```
resonse = page.text
print(resonse)
```

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(resonse, "html.parser")
results = soup.find(id = "ResultsContainer")
```

```
print(results.prettify())
```

```
job_elements= results.find_all("div", class_="card-content")
```

```
print(len(job_elements))
```

```
100
```

```
for job in job_elements:
  print(job,end="\n"*4)
```

```
<div class="card-content">
<div class="media">
<div class="media-left">
<figure class="image is-48x48">
<img alt="Real Python Logo" src="https://files.realpython.com/media/real-python-logo-thumbnail.7f0db70c2ed2.jpg?__no_cf_polish=1"/>
</figure>
</div>
<div class="media-content">
<h2 class="title is-5">Senior Python Developer</h2>
<h3 class="subtitle is-6 company">Payne, Roberts and Davis</h3>
</div>
</div>
<div class="content">
<p class="location">
        Stewartbury, AA
      </p>
<p class="is-small has-text-grey">
<time datetime="2021-04-08">2021-04-08</time>
</p>
</div>
<footer class="card-footer">
<a class="card-footer-item" href="https://www.realpython.com" target="_blank">Learn</a>
<a class="card-footer-item" href="https://realpython.github.io/fake-jobs/jobs/senior-python-developer-0.html" target="_blank">Apply<
</footer>
</div>



<div class="card-content">
<div class="media">
<div class="media-left">
<figure class="image is-48x48">
<img alt="Real Python Logo" src="https://files.realpython.com/media/real-python-logo-thumbnail.7f0db70c2ed2.jpg?__no_cf_polish=1"/>
</figure>
</div>
<div class="media-content">
<h2 class="title is-5">Energy engineer</h2>
<h3 class="subtitle is-6 company">Vasquez-Davidson</h3>
</div>
</div>
<div class="content">
<p class="location">
        Christopherville, AA
      </p>
<p class="is-small has-text-grey">
<time datetime="2021-04-08">2021-04-08</time>
</p>
```

```
</div>
<footer class="card-footer">
<a class="card-footer-item" href="https://www.realpython.com" target="_blank">Learn</a>
<a class="card-footer-item" href="https://realpython.github.io/fake-jobs/jobs/energy-engineer-1.html" target="_blank">Apply</a>
</footer>
</div>
```

Follow link (ctrl + click)

```
d = {'job_title': [], 'job_company': [], 'job_location': [], 'job_date': []}

for job_element in job_elements:
  job_title=job_element.find("h2", class_="title is-5").text
  job_company=job_element.find("h3", class_="company").text
  job_location=job_element.find("p", class_="location").text.replace('\n', '')
  job_date=job_element.find("time").text
  d['job_title'].append(job_title)
  d['job_company'].append(job_company)
  d['job_location'].append(job_location)
  d['job_date'].append(job_date)

import pandas as pd
df = pd.DataFrame(data=d)
df
```

	job_title	job_company	job_location	job_date
0	Senior Python Developer	Payne, Roberts and Davis	Stewartbury, AA	2021-04-08
1	Energy engineer	Vasquez-Davidson	Christopherville, AA	2021-04-08
2	Legal executive	Jackson, Chambers and Levy	Port Ericaburgh, AA	2021-04-08
3	Fitness centre manager	Savage-Bradley	East Seanview, AP	2021-04-08
4	Product manager	Ramirez Inc	North Jamieview, AP	2021-04-08
...	...	...	...	...
95	Museum/gallery exhibitions officer	Nguyen, Yoder and Petty	Lake Abigail, AE	2021-04-08
96	Radiographer, diagnostic	Holder LLC	Jacobshire, AP	2021-04-08
97	Database administrator	Yates-Ferguson	Port Susan, AE	2021-04-08
98	Furniture designer	Ortega-Lawrence	North Tiffany, AA	2021-04-08
99	Ship broker	Fuentes, Walls and Castro	Michelleville, AP	2021-04-08

100 rows × 4 columns

Data Preprocessing (cleaning)

```
dirty_data_url = "https://raw.githubusercontent.com/sharadkhare/Complete-Python-Pandas-Tutorial-in-Hindi-With-Notes-/main/dirtydata.csv"

import pandas as pd
df = pd.read_csv(dirty_data_url)
df.head()
```

	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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
```

```
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  32 non-null     int64
 1   Date      31 non-null     object
 2   Pulse     32 non-null     int64
 3   Maxpulse  32 non-null     int64
 4   Calories  30 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1.4+ KB
```

Follow link (ctrl + click)

```
df.head(32)
```

	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	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3

```
df_new = df.dropna()
```

```
df_new.head(33)
```

	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
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
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	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

Follow link (ctrl + click)

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  32 non-null     int64  
 1   Date      31 non-null     object 
 2   Pulse     32 non-null     int64  
 3   Maxpulse  32 non-null     int64  
 4   Calories  30 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1.4+ KB
```

```
df.iloc[28]
```

```
Duration              60
Date        '2020/12/28'
Pulse                103
Maxpulse             132
Calories             NaN
Name: 28, dtype: object
```

```
df_new.iloc[28]
```

```
Duration              60
Date        '2020/12/31'
Pulse                 92
Maxpulse             115
Calories           243.0
Name: 31, dtype: object
```

```
df_new.info()
```

Follow link (ctrl + click)

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29 entries, 0 to 31
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  29 non-null     int64
 1   Date      29 non-null     object
 2   Pulse     29 non-null     int64
 3   Maxpulse  29 non-null     int64
 4   Calories  29 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1.4+ KB
```

```
df.dropna(inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29 entries, 0 to 31
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  29 non-null     int64
 1   Date      29 non-null     object
 2   Pulse     29 non-null     int64
 3   Maxpulse  29 non-null     int64
 4   Calories  29 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1.4+ KB
```

```
df = pd.read_csv(dirty_data_url)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  32 non-null     int64
 1   Date      31 non-null     object
 2   Pulse     32 non-null     int64
 3   Maxpulse  32 non-null     int64
 4   Calories  30 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1.4+ KB
```

```
df.fillna(130)
```

	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	130.0
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	130	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	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	130.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3

Follow link (ctrl + click)

```
#Replace null values


print(df)
```

```
    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      20201226    100       120     250.0
27        60  '2020/12/27'     92       118     241.0
28        60  '2020/12/28'    103       132       NaN
29        60  '2020/12/29'    100       132     280.0
30        60  '2020/12/30'    102       129     380.3
31        60  '2020/12/31'     92       115     243.0
```

Follow link (ctrl + click)

```
df['Date'].fillna('2020/12/27',inplace = True)
df.iloc[22]
```

```
Duration            45
Date        2020/12/27
Pulse              100
Maxpulse           119
Calories         282.0
Name: 22, dtype: object
```

Replace using mean, median and mode

```
x = df['Calories']
print(x)
```

```
0     409.1
1     479.0
2     340.0
3     282.4
4     406.0
5     300.0
6     374.0
7     253.3
8     195.1
9     269.0
10    329.3
11    250.7
12    250.7
13    345.3
14    379.3
15    275.0
16    215.2
17    300.0
18      NaN
19    323.0
20    243.0
21    364.2
22    282.0
23    300.0
24    246.0
25    334.5
26    250.0
27    241.0
28      NaN
29    280.0
30    380.3
31    243.0
Name: Calories, dtype: float64
```

```
cal_mean = df['Calories'].mean()
df['Calories'].fillna(cal_mean, inplace=True)
print(df['Calories'])
```

```
0     409.10
1     479.00
2     340.00
3     282.40
4     406.00
5     300.00
6     374.00
7     253.30
8     195.10
9     269.00
10    329.30
11    250.70
12    250.70
```

```
13    345.30
14    379.30
15    275.00
16    215.20
17    300.00
18    304.68
19    323.00
20    243.00
21    364.20
22    282.00
23    300.00
24    246.00
25    334.50
26    250.00
27    241.00
28    304.68
29    280.00
30    380.30
31    243.00
Name: Calories, dtype: float64
```

Follow link (ctrl + click)

```
df = pd.read_csv(dirty_data_url)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  32 non-null     int64
 1   Date      31 non-null     object
 2   Pulse     32 non-null     int64
 3   Maxpulse  32 non-null     int64
 4   Calories  30 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1.4+ KB
```

```
df['Calories'].isnull()
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
```