Exploratory Data Analysis

Understanding Your Data

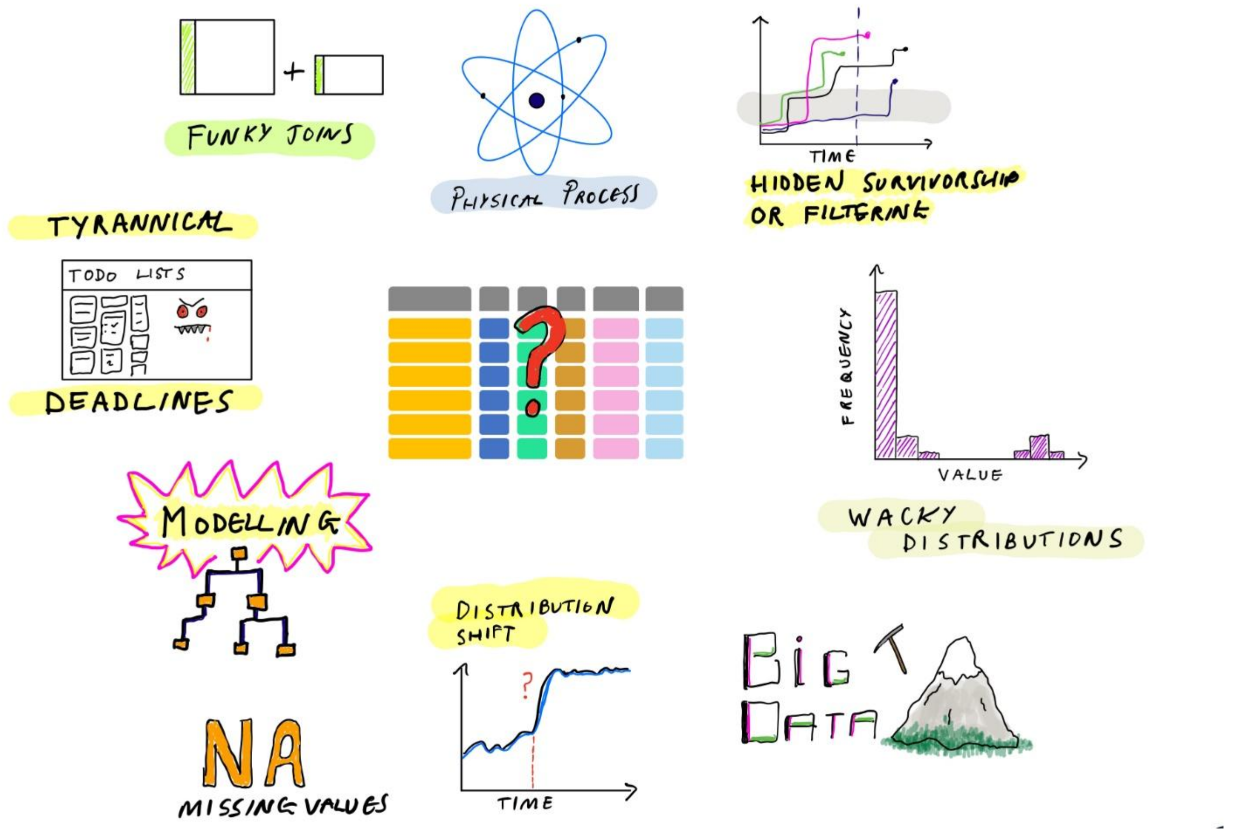
TS. Đỗ Như Tài
dntai@sgu.edu.vn

Week Overview

- Exploratory Data Analysis (EDA) Introduction
  - ➢ What is EDA
  - ➢ Why need EDA, what can do with EDA
  - ➢ How to do EDA
- EDA Techniques
  - ➢ Groups of EDA Methods
  - ➢ Visualisation Methods
- EDA Tools
  - ➢ Python Modules for EDA
- EDA Case Studies

Thinking

- How to choose the most suitable algorithms for your dataset?

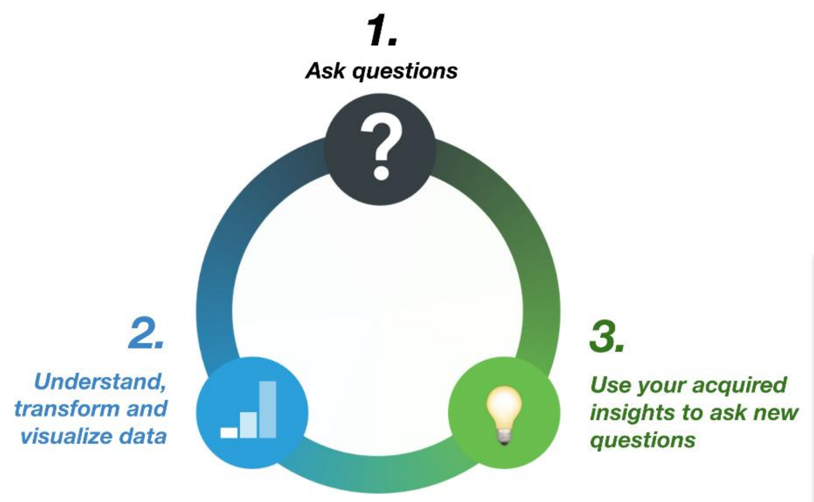


https://alastairrushworth.github.io/exploring_eda/EDA.html#1

- How to ensure you are ready to use machine learning techniques in a new project?
- Answer: Exploratory Data Analysis (EDA) helps to answer

Exploratory Data Analysis

- Exploratory Data Analysis (EDA) is a process for summarising, visualising, and becoming intimately familiar with the important characteristics of the data
- EDA is an iterative cycle:
  - Generate questions about your data:
  - Search for answers by visualising, transforming, and modelling your data
  - Use what you learn to refine your questions and/or generate new questions


1.
Ask questions
2.
Understand, transform and visualize data
3.
Use your acquired insights to ask new questions

https://duo.com/labs/research/gamifying-data-science-education

- What type of variation occurs within my variables?
- What type of covariation occurs between my variables?

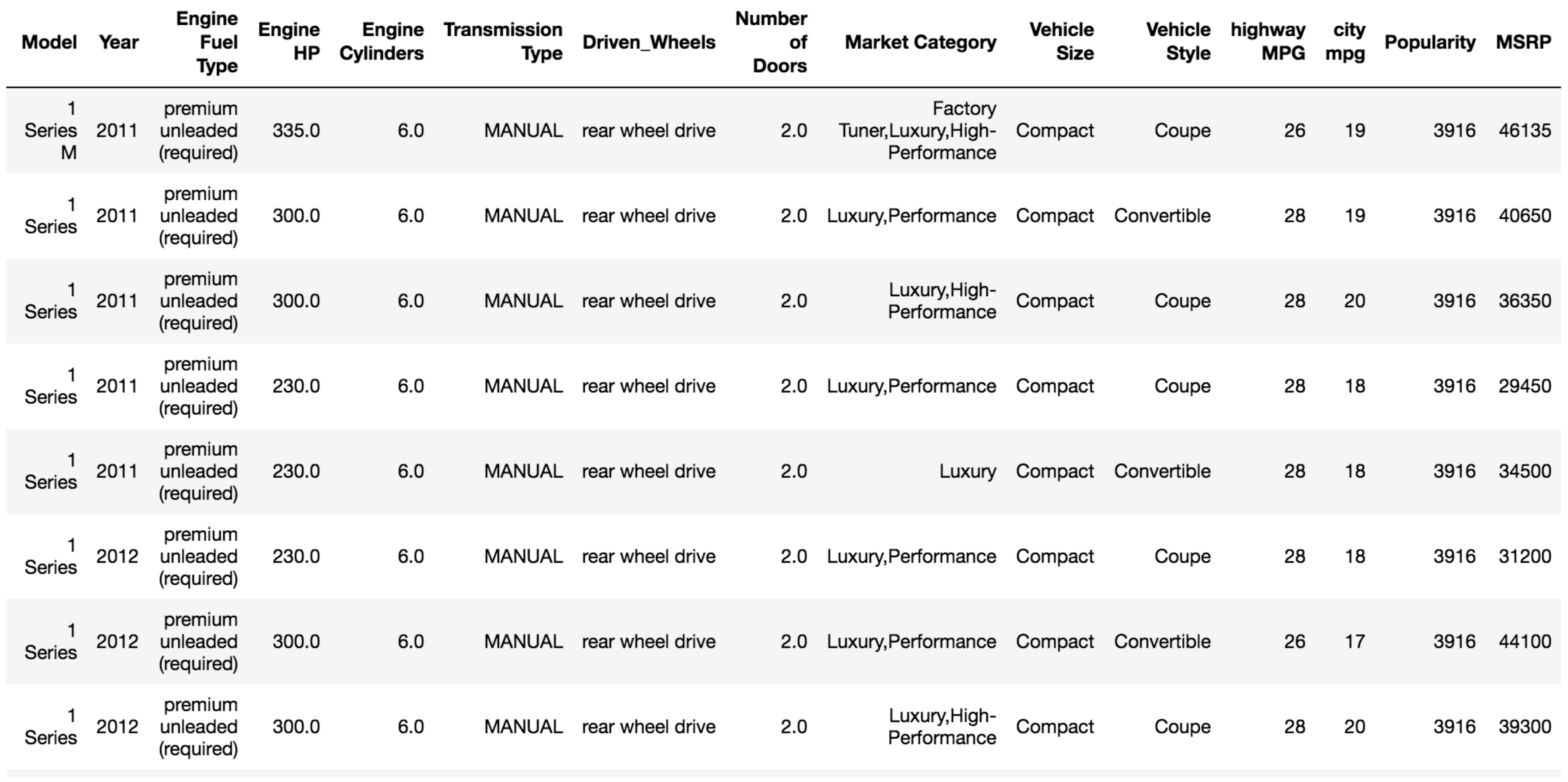
- EDA is *statisticians' way of story telling* where you explore data, find patterns and tells insights

Key Concepts of Exploratory Data Analysis

- 4 Objectives of EDA
  - Discover Patterns
  - Spot Anomalies
  - Frame Hypothesis
  - Check Assumptions
- Stuff done during EDA
  - Measures of central tendency: mean, median, mode
  - Spread measurement : standard deviation, variance
  - Shape of distribution: distribution, trends
  - Outlier
  - Correlations
  - Visual Exploration

Making Sense of Data – Distinguish Types of Attributes

- input takes the form of instances and attributes/features
  - information to a machine learning learner takes the form of a set of instances
  - each instance is described by a fixed predefined set of features or attributes:
  - Types: Numerical (Discrete and Continuous) and Nominal (Categorical)



Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3916	46135
1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3916	40650
1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	36350
1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	29450
1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3916	34500
1 Series	2012	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	31200
1 Series	2012	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	26	17	3916	44100
1 Series	2012	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	39300

Types of EDA Methods

- EDA methods: generally classified into two ways
  - graphical or non-graphical/quantitative : summarising data in a visual way or calculation of summary statistics
  - univariate or multivariate: summary statistics for each feature/attribute or find relationship between features

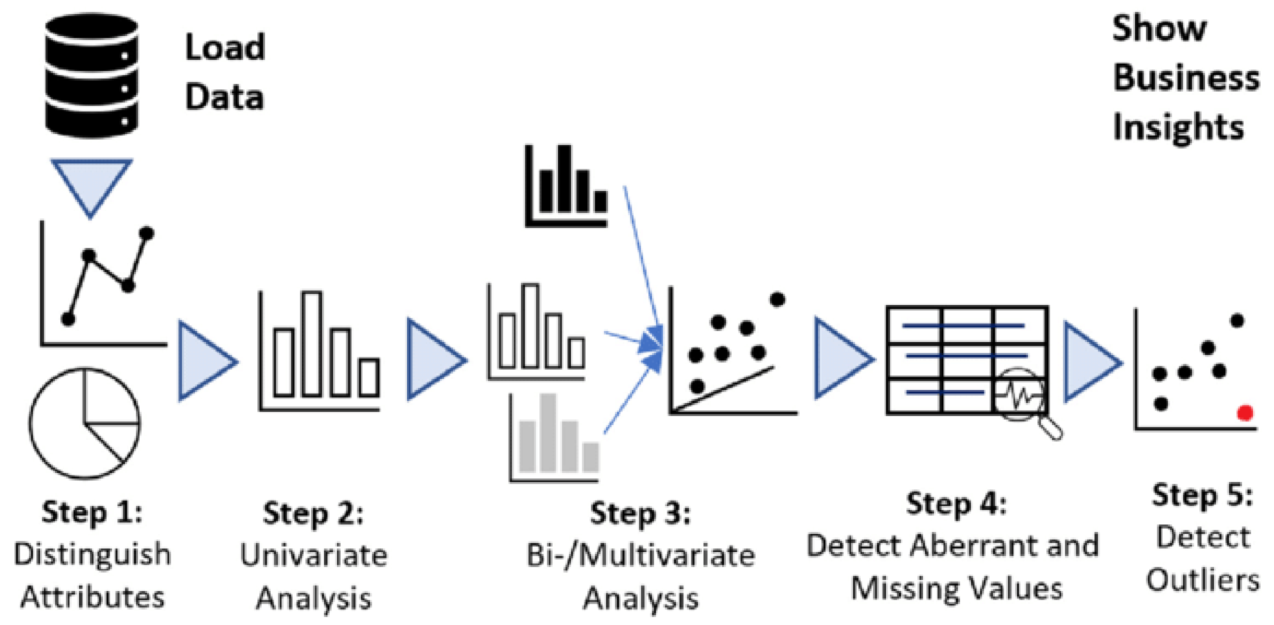
	Univariate	Multivariate
Non-Graphical	**Categorical Variable**: tabular representation of frequency **Quantitative Variable:** • Location (mean, median) • Shape and Spread • Modality • Outliers …	**One Categorical Variable and One Quantitative Variable**: Standard univariable non-graphical statistics for the quantitative variable separately for each level of the categorial variable **Two and more Quantitative Variable:** • Correlation, • Covariance, • …
Graphical	**Categorical Variable**: Bar Chart **Quantitative Variable:** • Histogram • Boxplot • …	**One Categorical Variable and One Quantitative Variable**: • Side-by-side Boxplots **Two and more Categorical Variable**: • Grouped Bar Chart **Two and more Quantitative Variable:** • Scatter plot, Correlation Heatmap, Pairplot **…**

How to do EDA

Steps/activities involved in EDA:

- identification of variables and data types
- non-graphical and graphical univariate analysis
- bi-/multivariate analysis, correlation analysis
- detect missing values and anomalies
- detect outliers

A typical example:

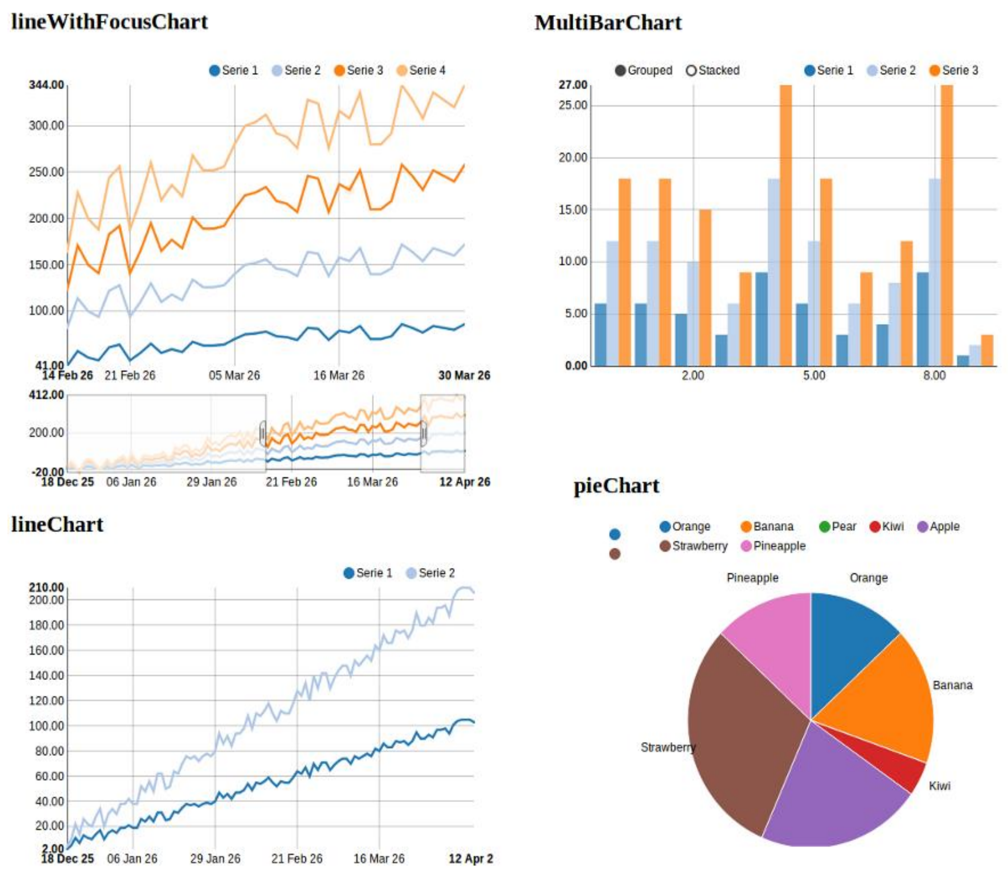

Load Data
Show Business Insights
Step 1: Distinguish Attributes
Step 2: Univariate Analysis
Step 3: Bi-/Multivariate Analysis
Step 4: Detect Aberrant and Missing Values
Step 5: Detect Outliers

https://www.researchgate.net/publication/342282008_Exploratory_Data_Analysis_and_Data_Envelopment_Analysis_of_Construction_and_Demolition_Waste_Management_in_the_European_Economic_Area/figures?lo=1

Visualisation

There are four basic presentation types

- composition
- comparison
- distribution
- relationship



https://raw.githubusercontent.com/areski/python-nvd3
https://www.tatvic.com/blog/7-visualizations-learn-r/

To determine which is best suited

- How many variables in a single chart?
- How many data points display for each variable?
- Will you display values over a period of time, or among items or groups

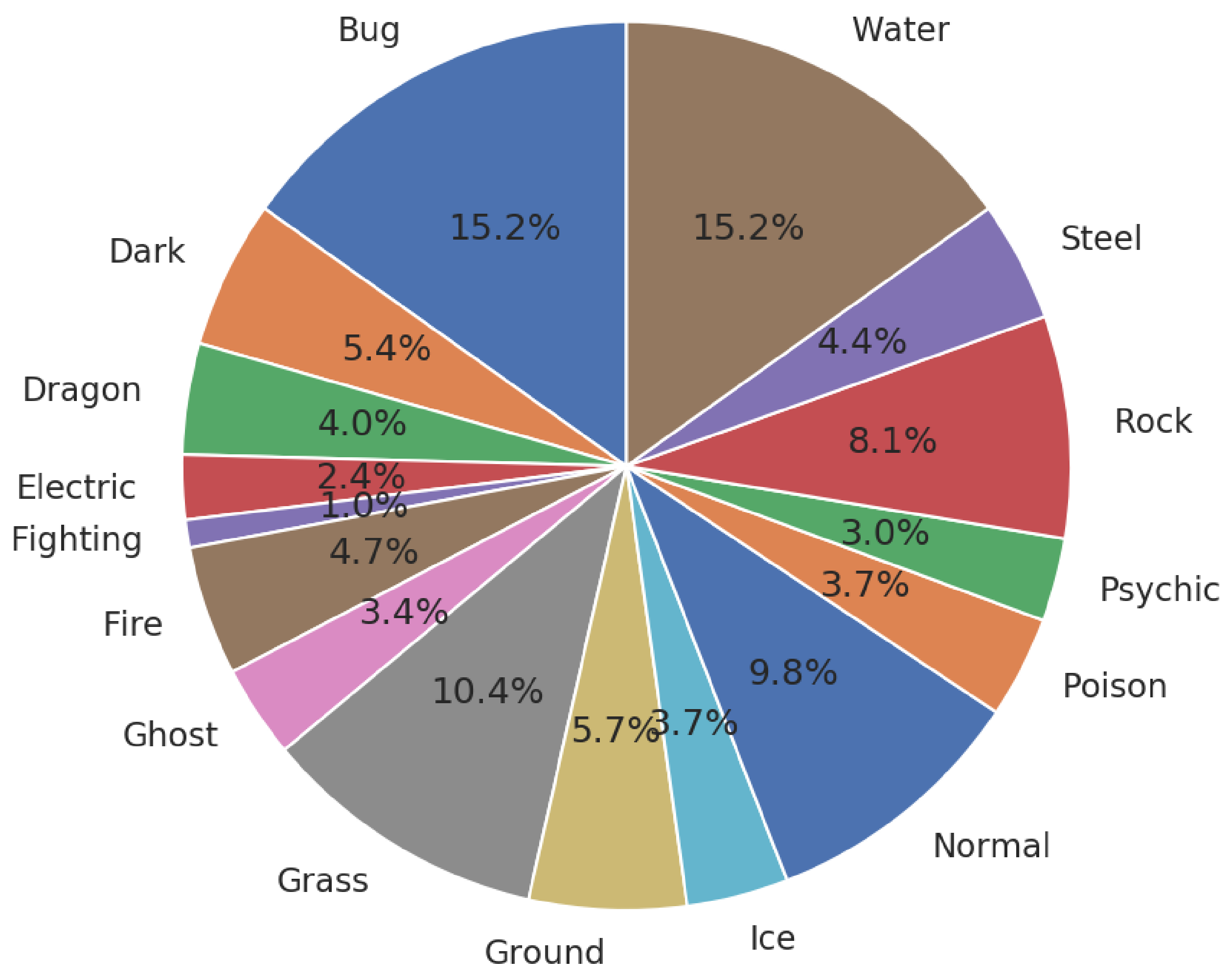
Visual Aids

Common charts in EDA:

- Pie chart
- Histogram
- Bar & Stack Bar Chart
- Box Plot & Violin plot
- Area Chart
- Scatter Plot
- Correlogram
- Heatmap

Pie chart

- Pie chart: circle divided to sectors, to communicate proportions
- a common method for representing categorical variables

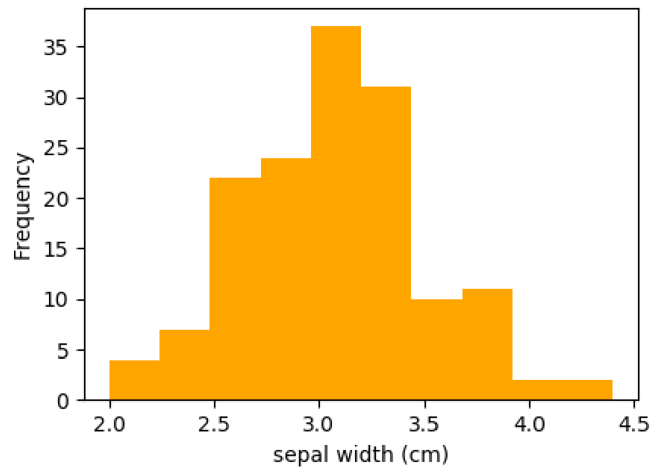


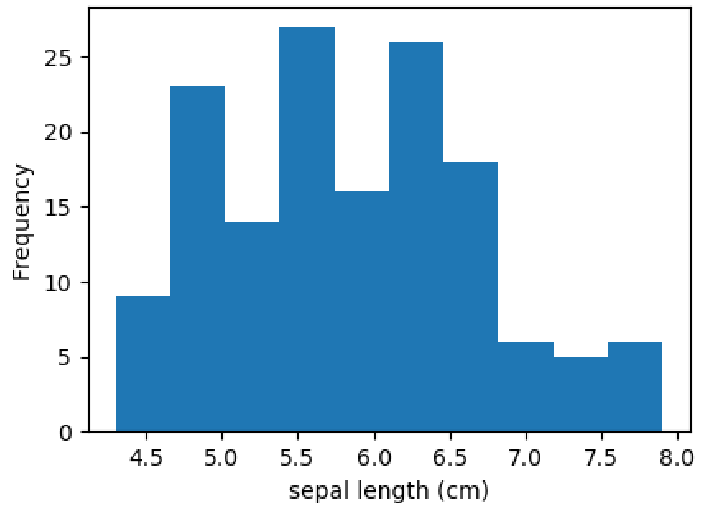
✓ simple and easy-to-understand
✓ understand information quickly

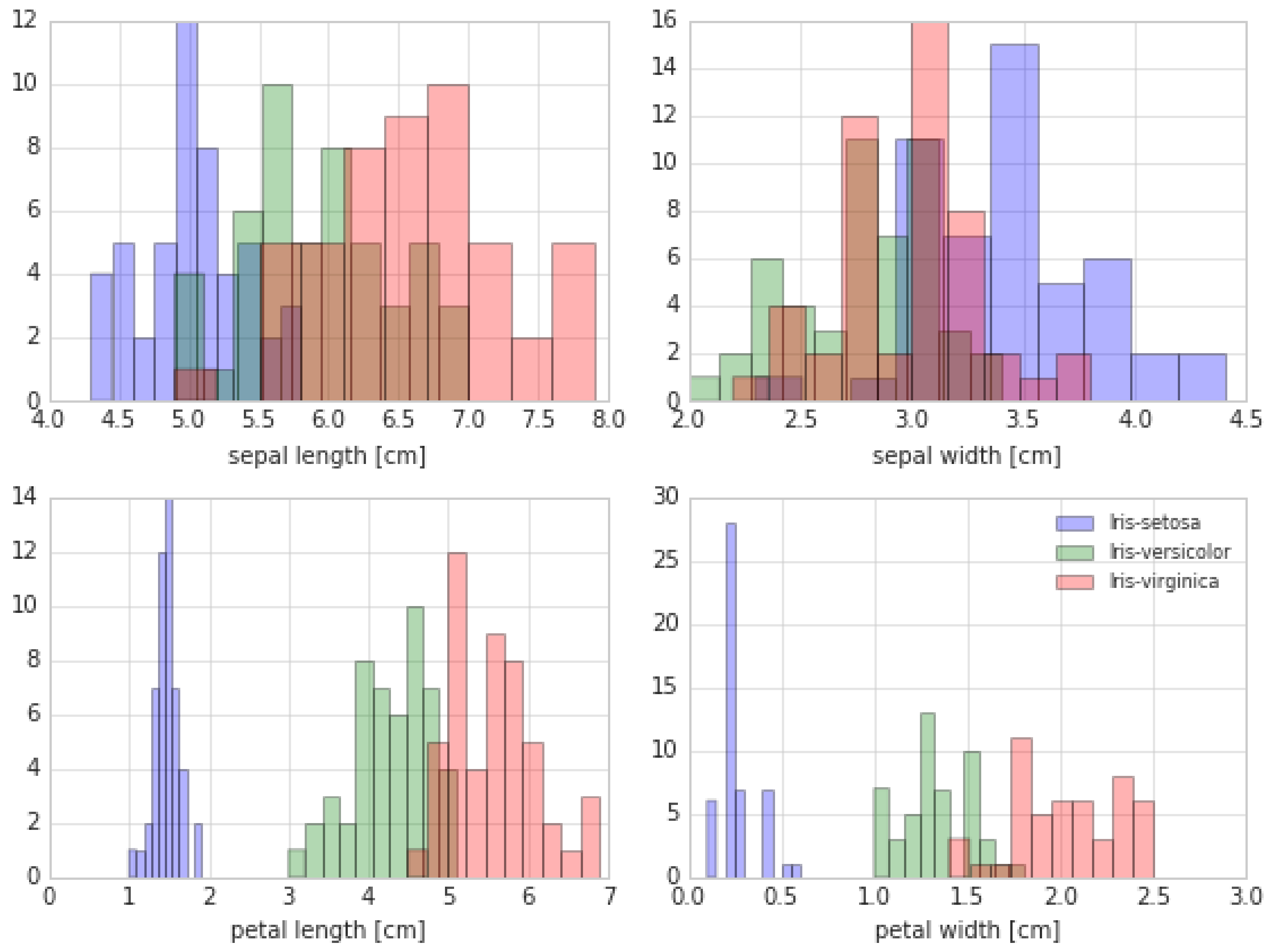
○ difficult to compare a few pieces
○ unhelpful when observing trends over time

Histogram

- Histogram: a plot of the frequency distribution of numeric variable by splitting values to small equal-sized bins, provide a visual summary of central tendency, spread, and shape.



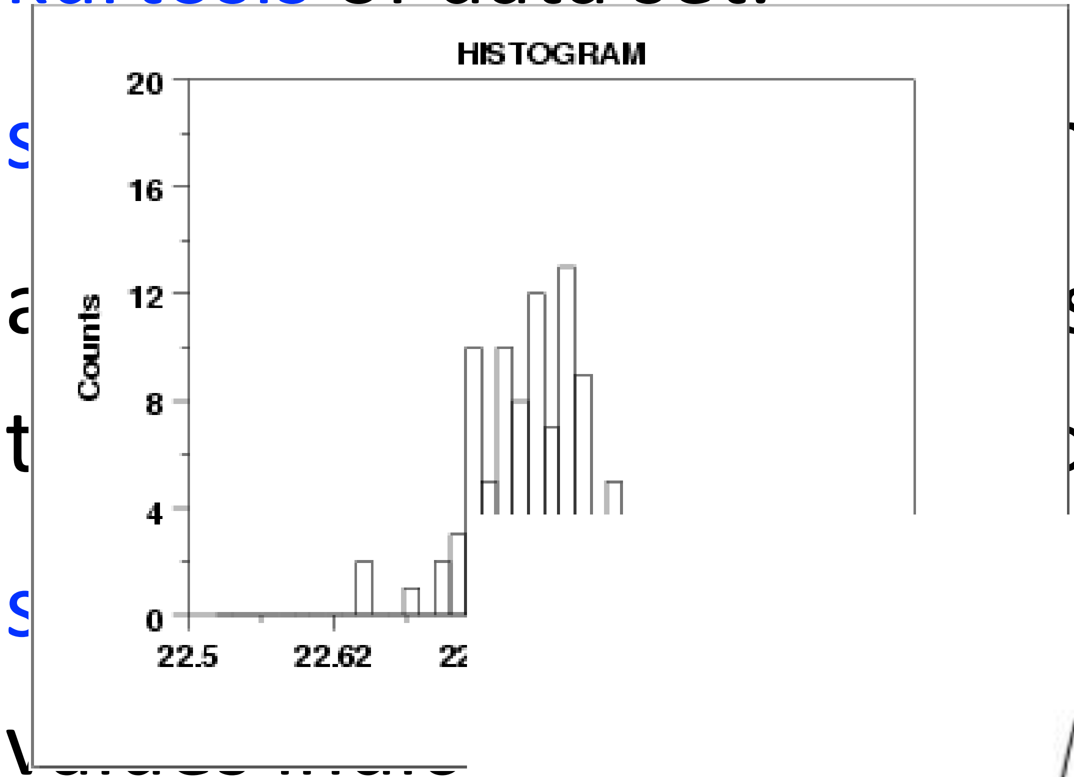


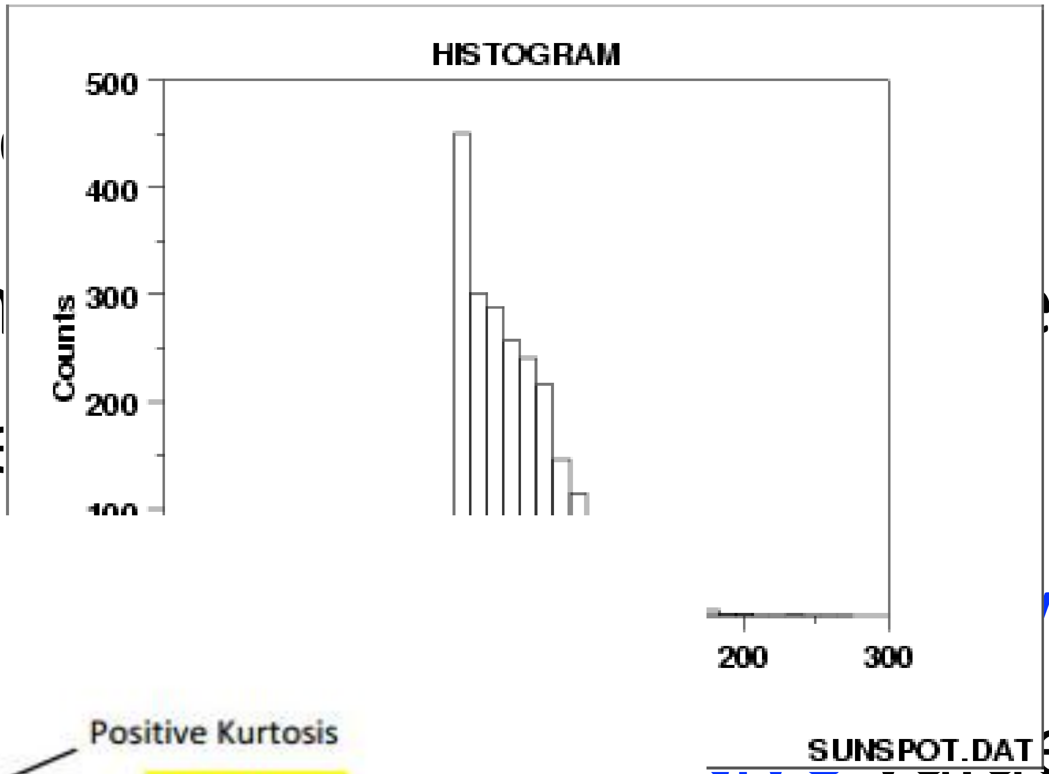


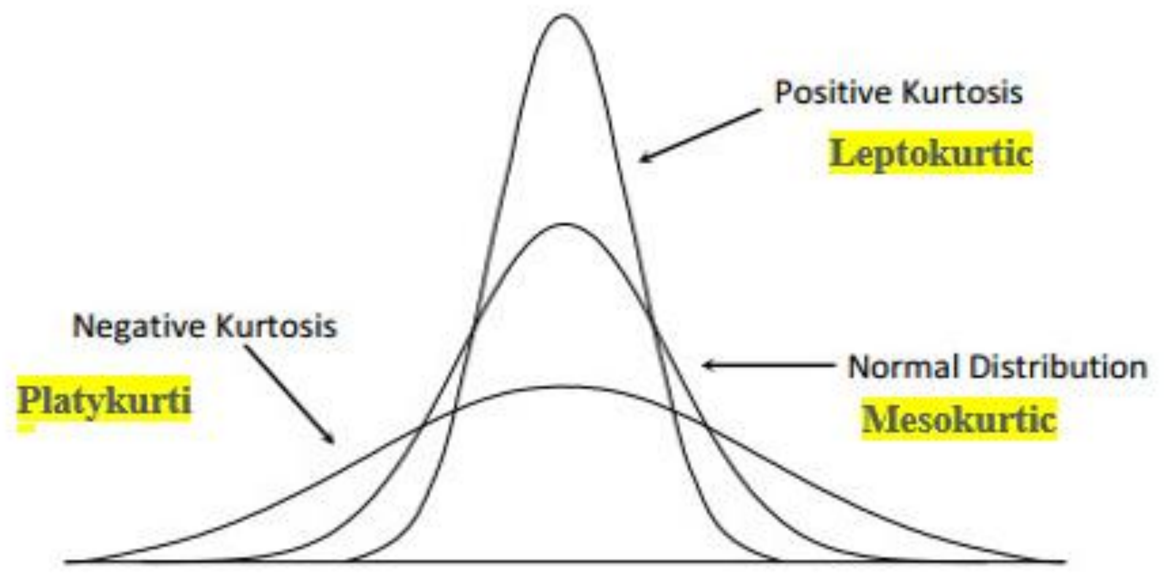
https://stackoverflow.com/questions/37911731/seaborn-histogram-with-4-panels-2-x-2-in-python

Shape of Data - Skewness and Kurtosis

- histogram is effective for showing both the skewness and kurtosis of data set.
- s he la
- a s syn e
  t cente
- s ve
  v es
  indicate ske
- the type of ized by a
  measureme

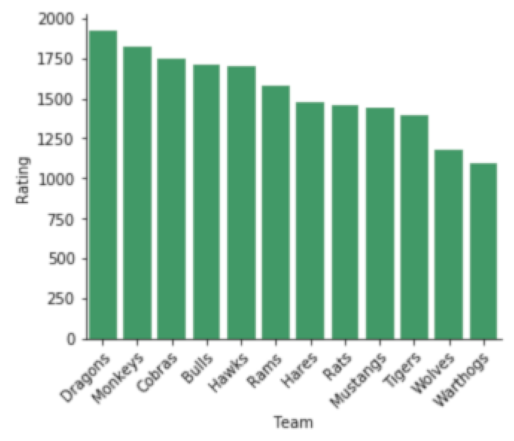


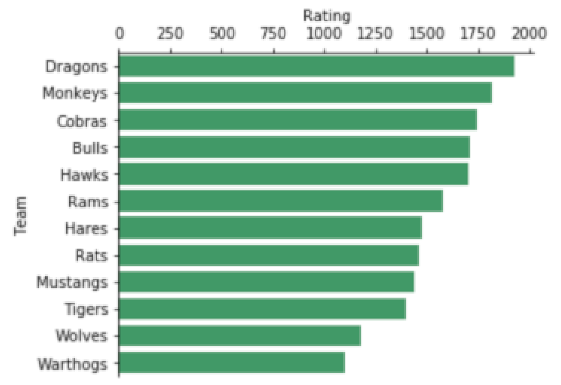


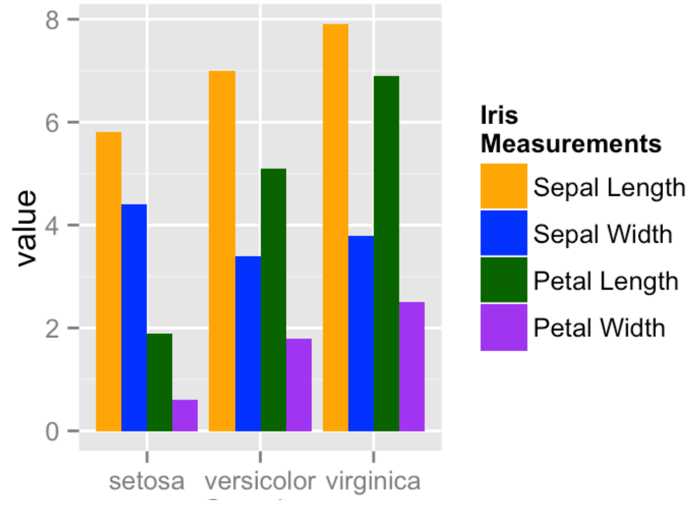


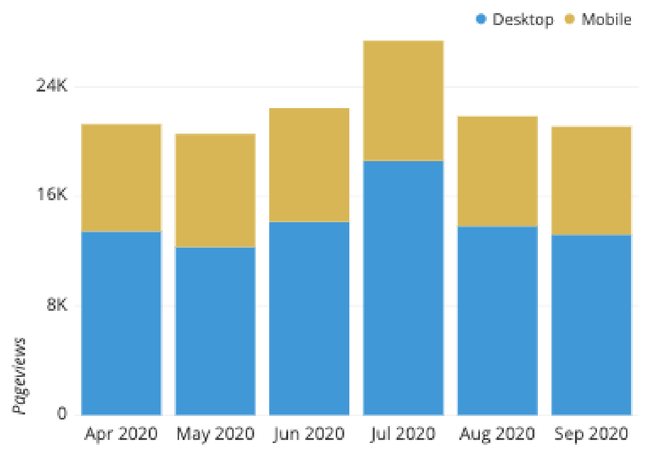
Bar & Stacked Bar Chart

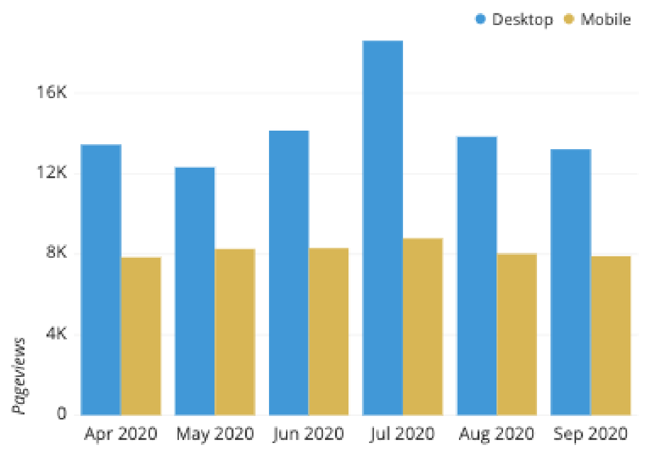
- Bar charts: a way of summarizing a set of categorical data, displays data using bars, each representing a particular category, the height is proportional to a specific aggregation
- Bars can be horizontal or vertical







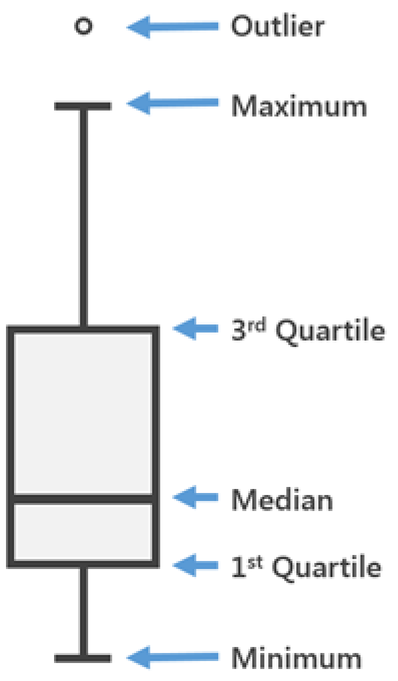


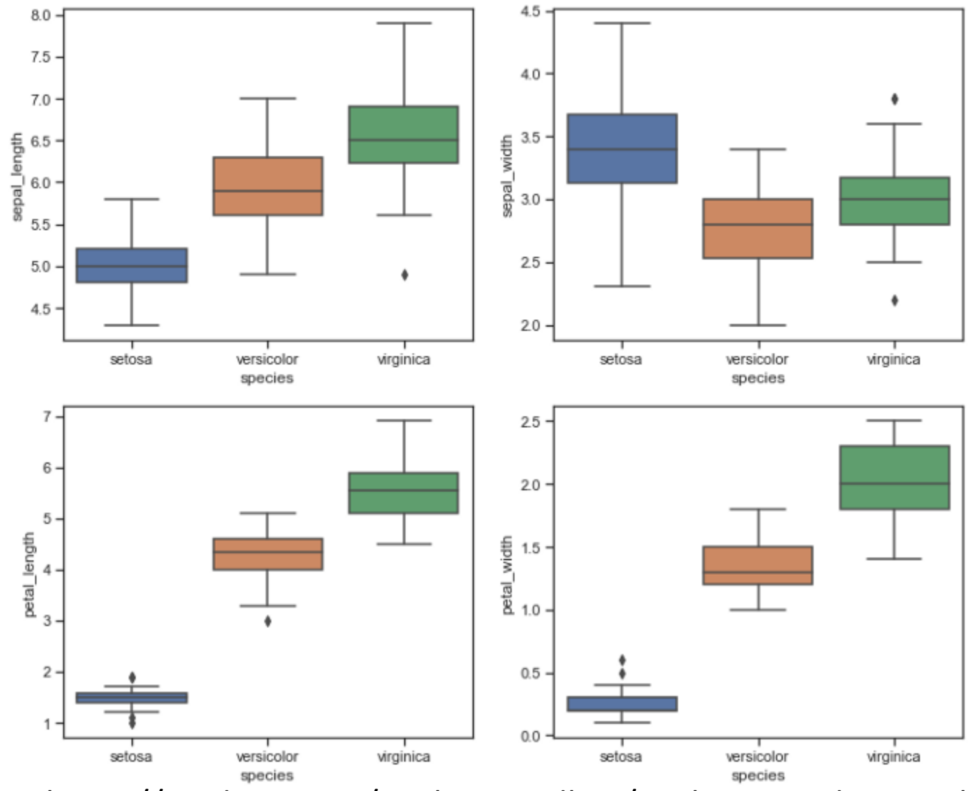


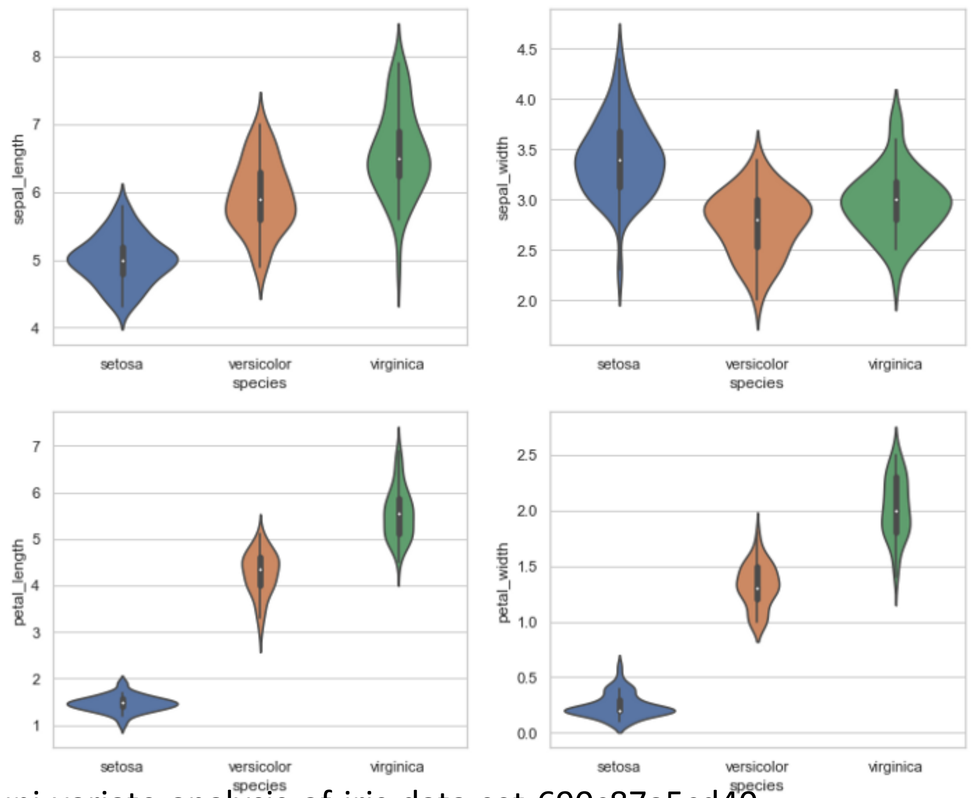
https://chartio.com/learn/charts/bar-chart-complete-guide/

Box Plot & Violin Plot

- Box plot: box and whisker plot, displays a summary of a large amount of data in five numbers , a good indication of how the values in the data are spread out with in groups
- plot a combination of categorical and continuous variables
- Violin plot: similar as box plot, additionally shows the kernel density estimation of the underlying distribution



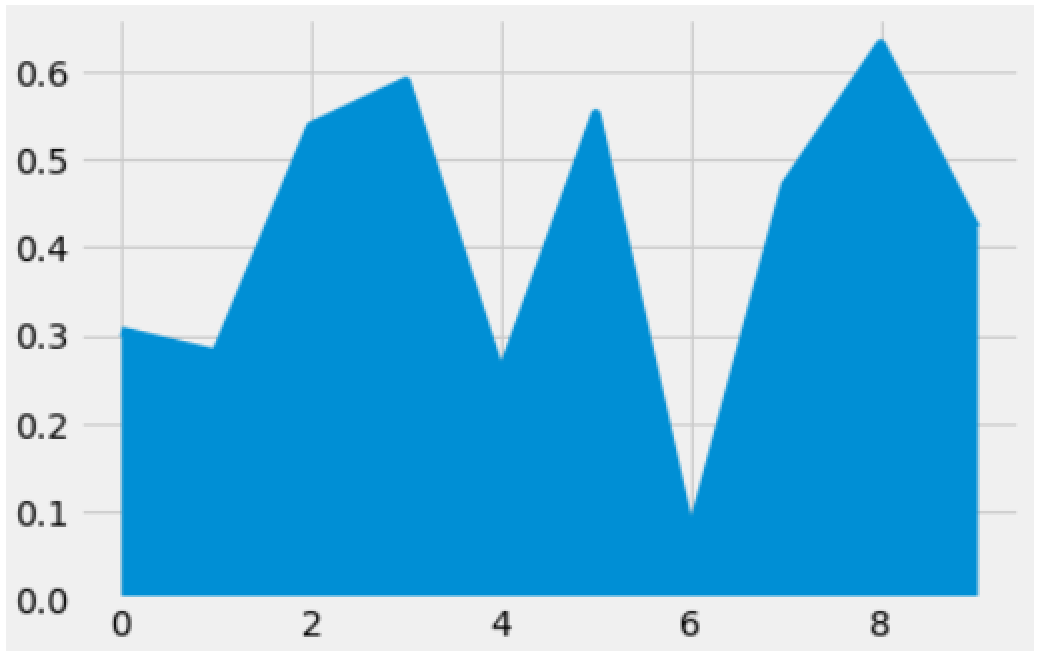


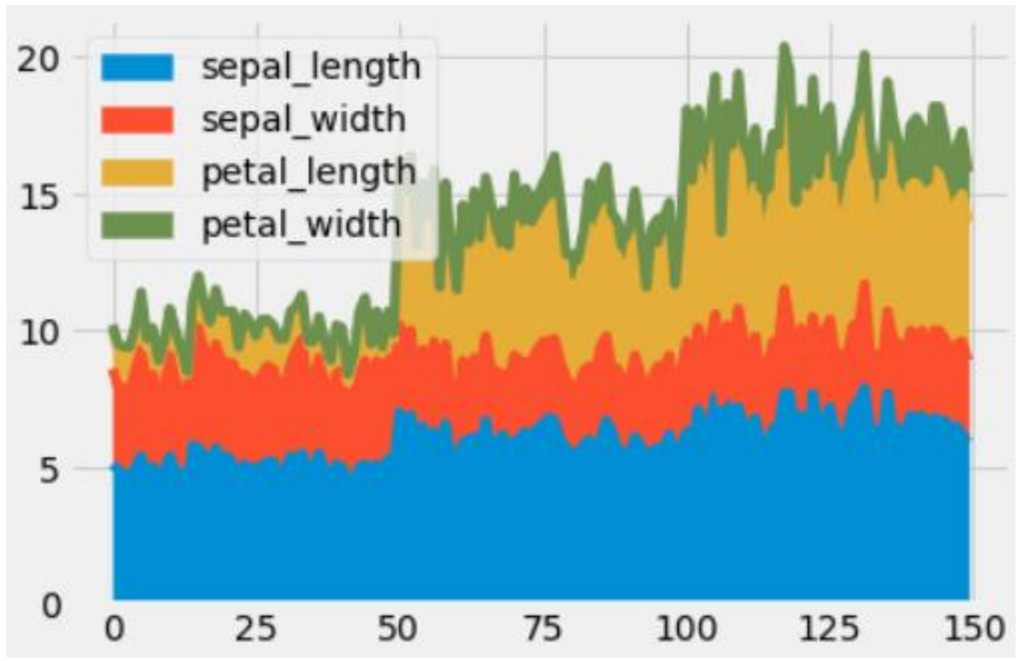


https://medium.com/analytics-vidhya/exploratory-data-analysis-uni-variate-analysis-of-iris-data-set-690c87a5cd40

Area Chart/Stacked Chart

- base on the line chart, areas between axis and line are commonly emphasized with colors
- share features with bar charts and line charts, compare two or more quantities, work better for large difference and multiple values over time

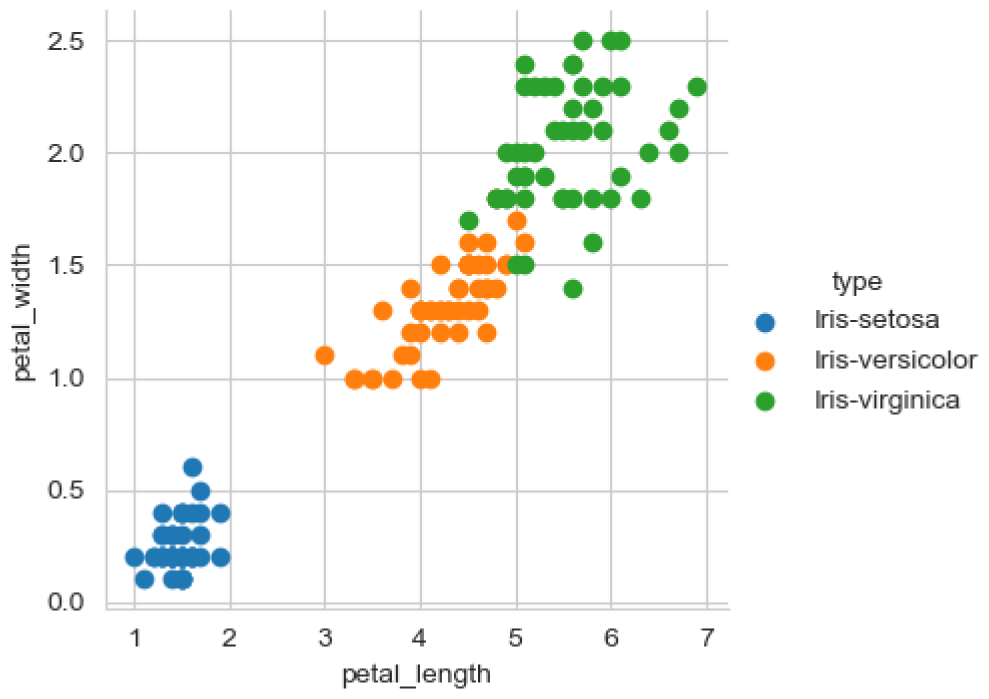


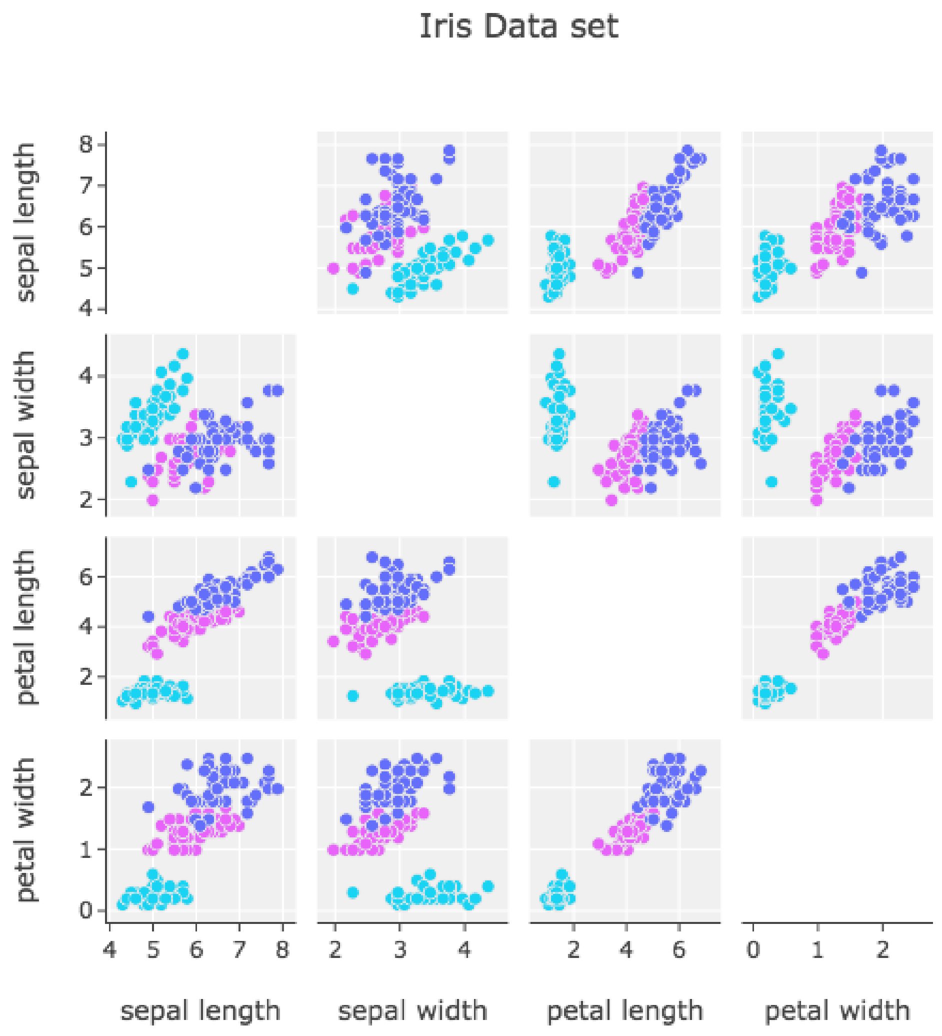


https://levelup.gitconnected.com/data-visualization-with-pandas-in-action-part-2-2cc8674da1d0

Scatter Plot

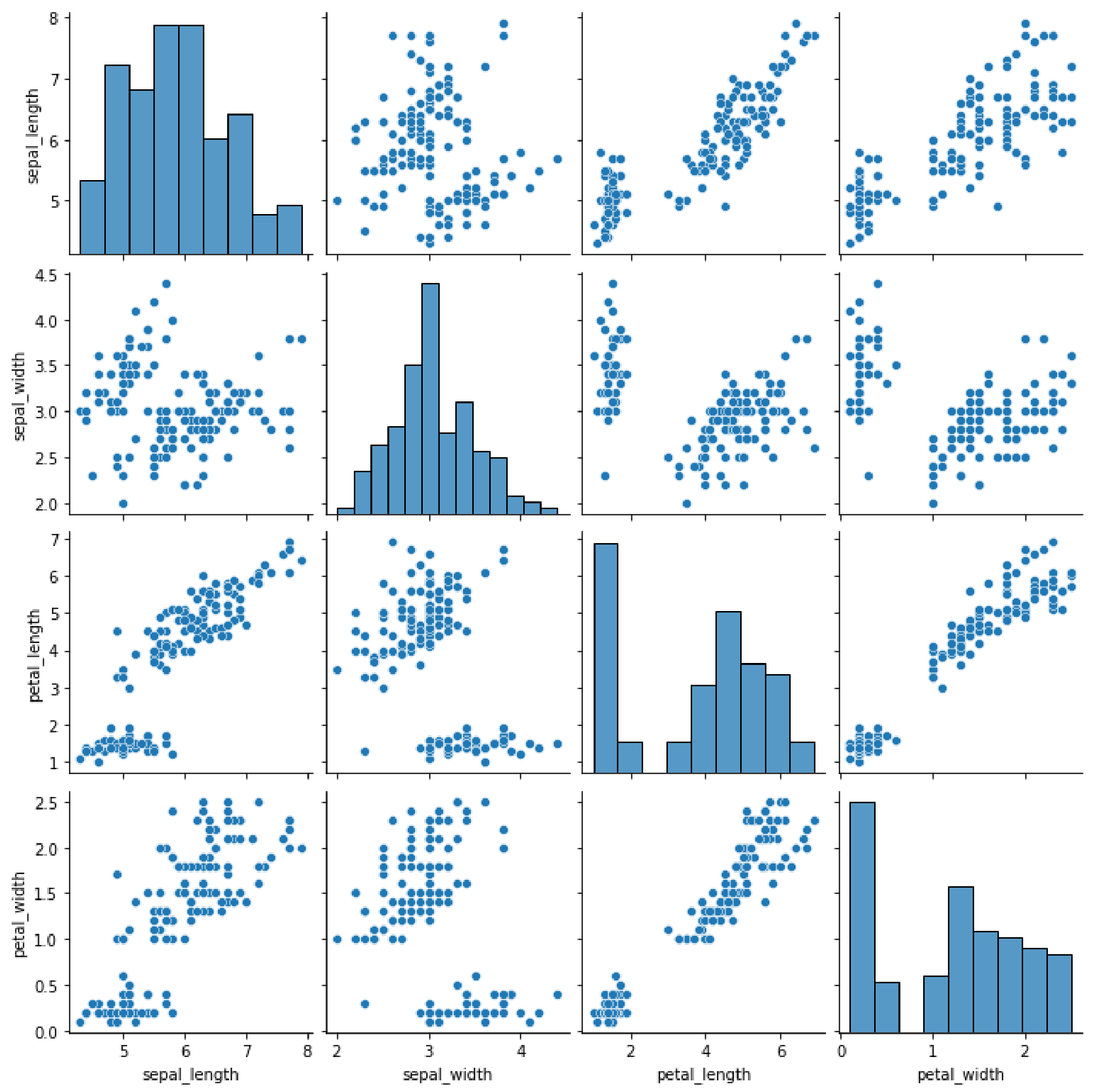
- use a Cartesian coordinates system to display values of two variables for a set of data
- show the relationship between two variables, referred to as correlation plots

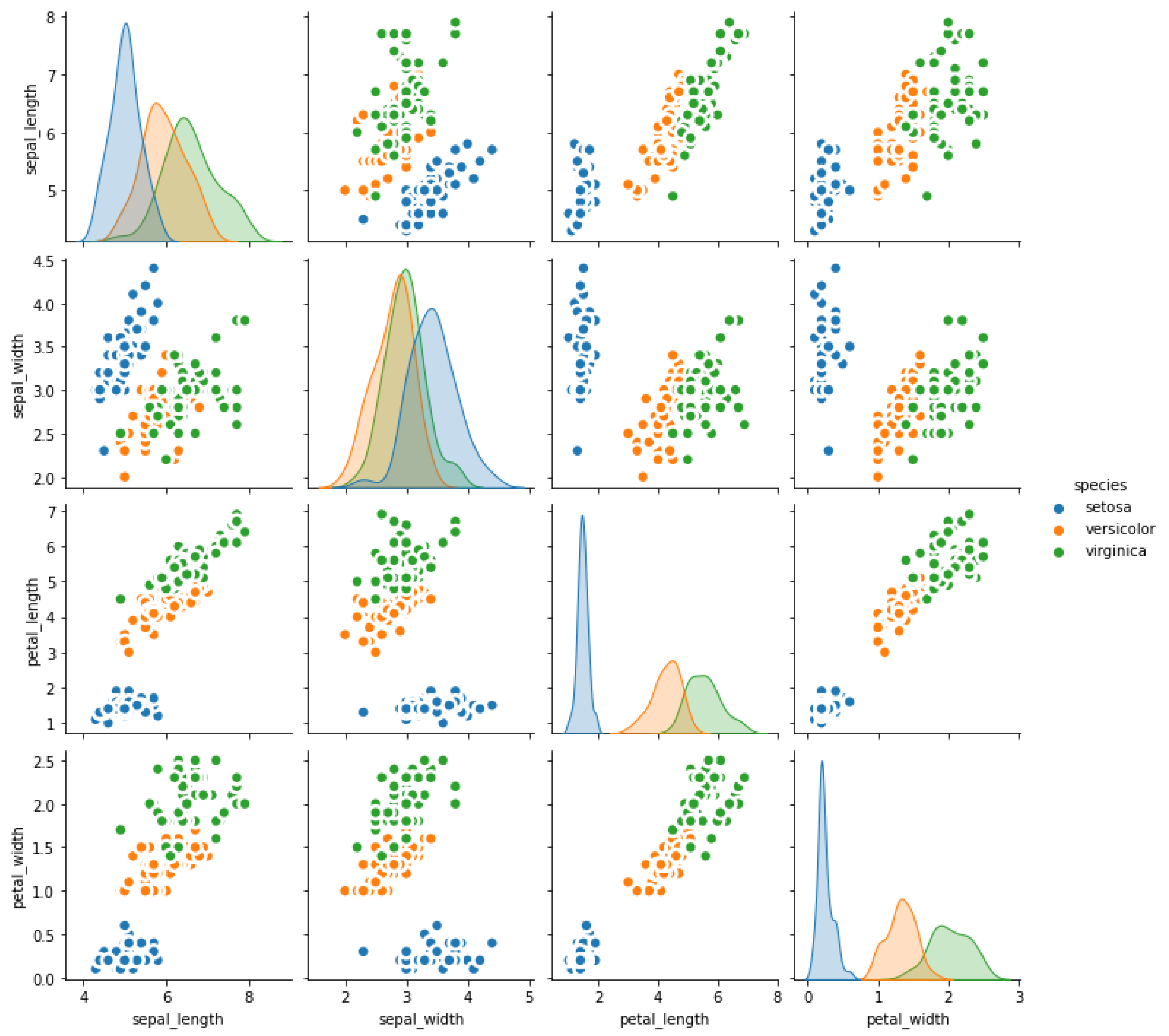




Correlogram

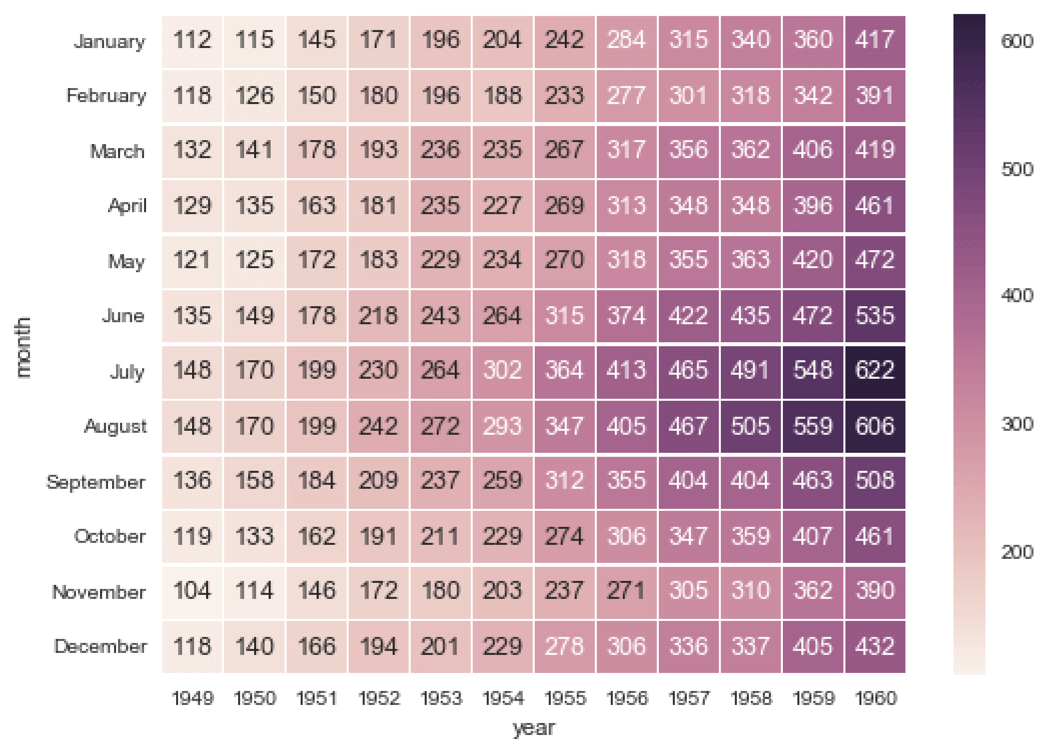
- Correlogram: AKS correlation matrix, to analyse the relationship between each pair of numeric variables



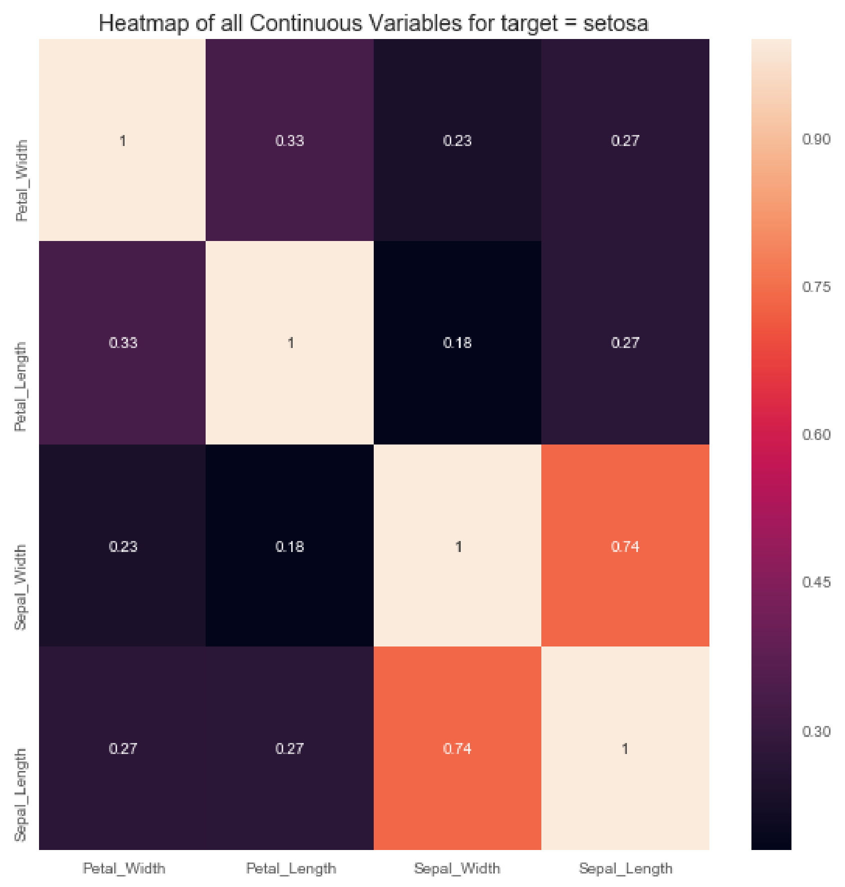


HeatMap

- Heatmap: a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colors
- useful to see which intersections of the categorical values, have higher concentration of the data compared to the others

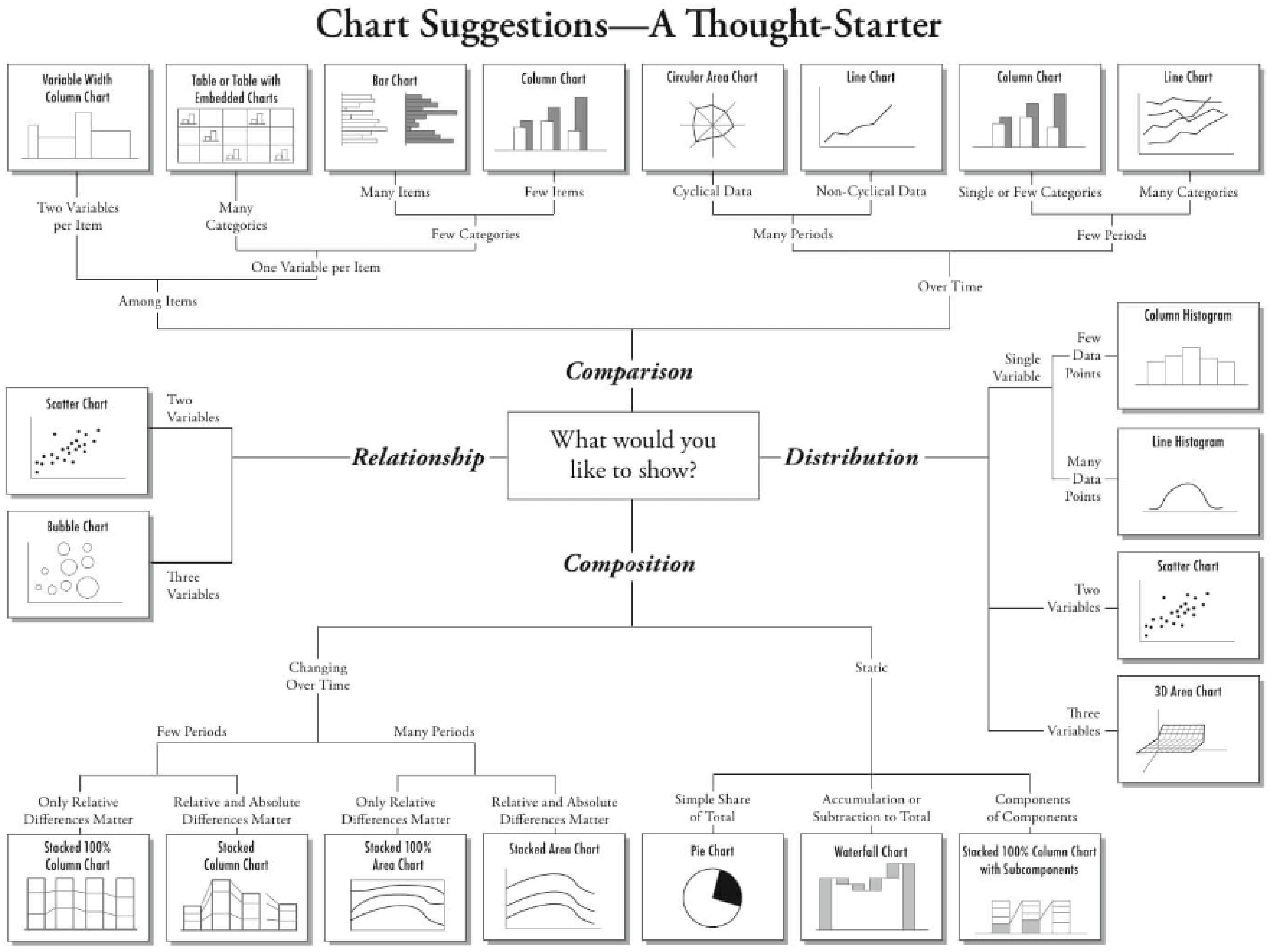


month	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
January	112	115	145	171	196	204	242	284	315	340	360	417
February	118	126	150	180	196	188	233	277	301	318	342	391
March	132	141	178	193	236	235	267	317	356	362	406	419
April	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
June	135	149	178	218	243	264	315	374	422	435	472	535
July	148	170	199	230	264	302	364	413	465	491	548	622
August	148	170	199	242	272	293	347	405	467	505	559	606
September	136	158	184	209	237	259	312	355	404	404	463	508
October	119	133	162	191	211	229	274	306	347	359	407	461
November	104	114	146	172	180	203	237	271	305	310	362	390
December	118	140	166	194	201	229	278	306	336	337	405	432



	Petal_Width	Petal_Length	Sepal_Width	Sepal_Length
Petal_Width	1	0.33	0.23	0.27
Petal_Length	0.33	1	0.18	0.27
Sepal_Width	0.23	0.18	1	0.74
Sepal_Length	0.27	0.27	0.74	1

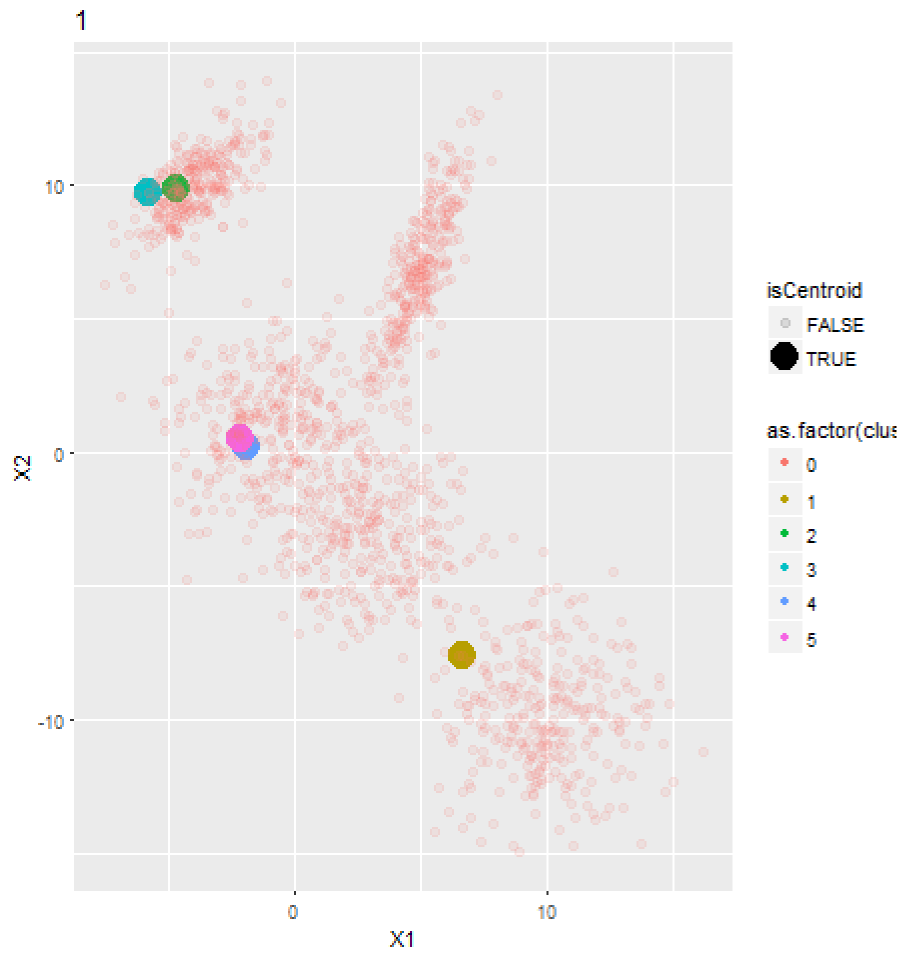
Choose the Most Suitable Plots

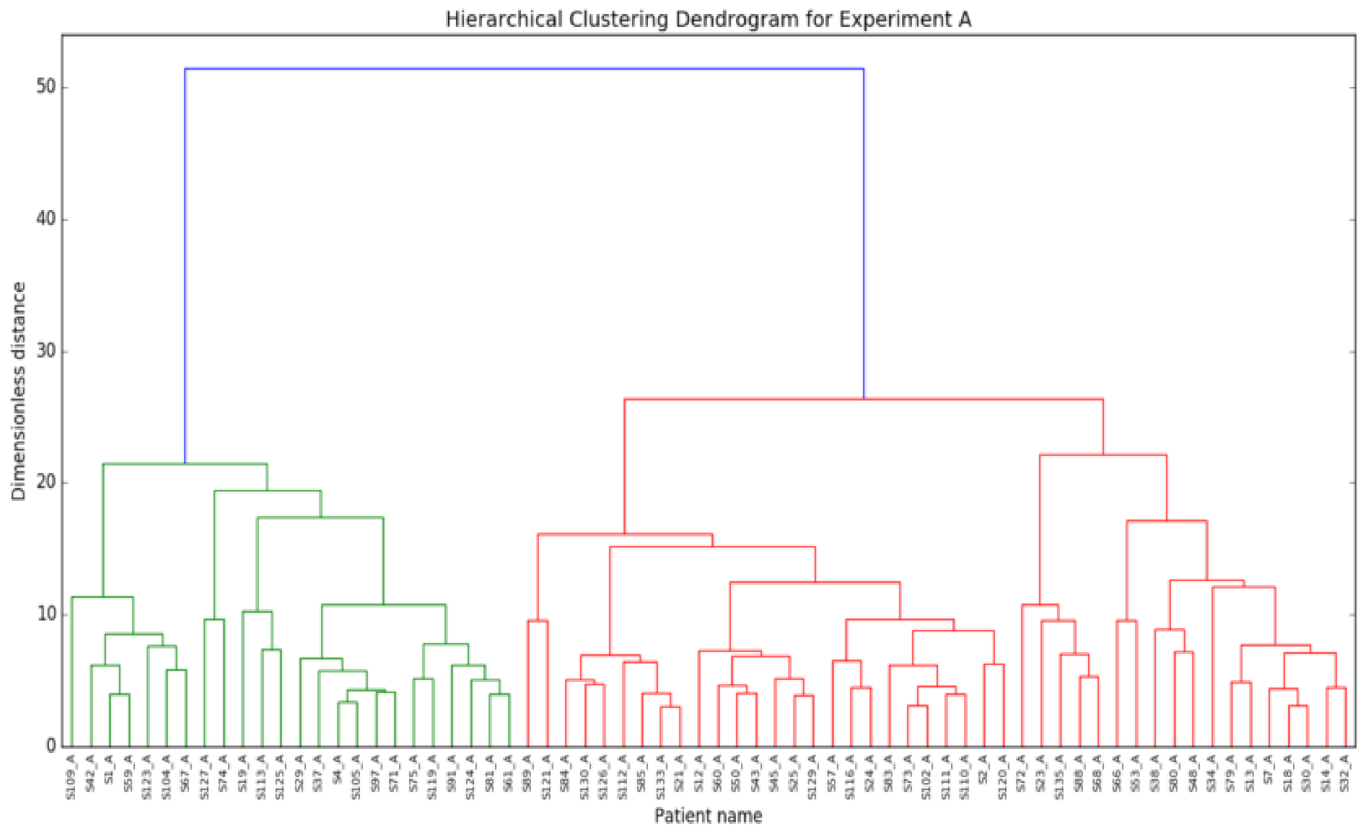


https://www.tatvic.com/blog/7-visualizations-learn-r/

Clustering Analysis for EDA

- Clustering in EDA to find new insights





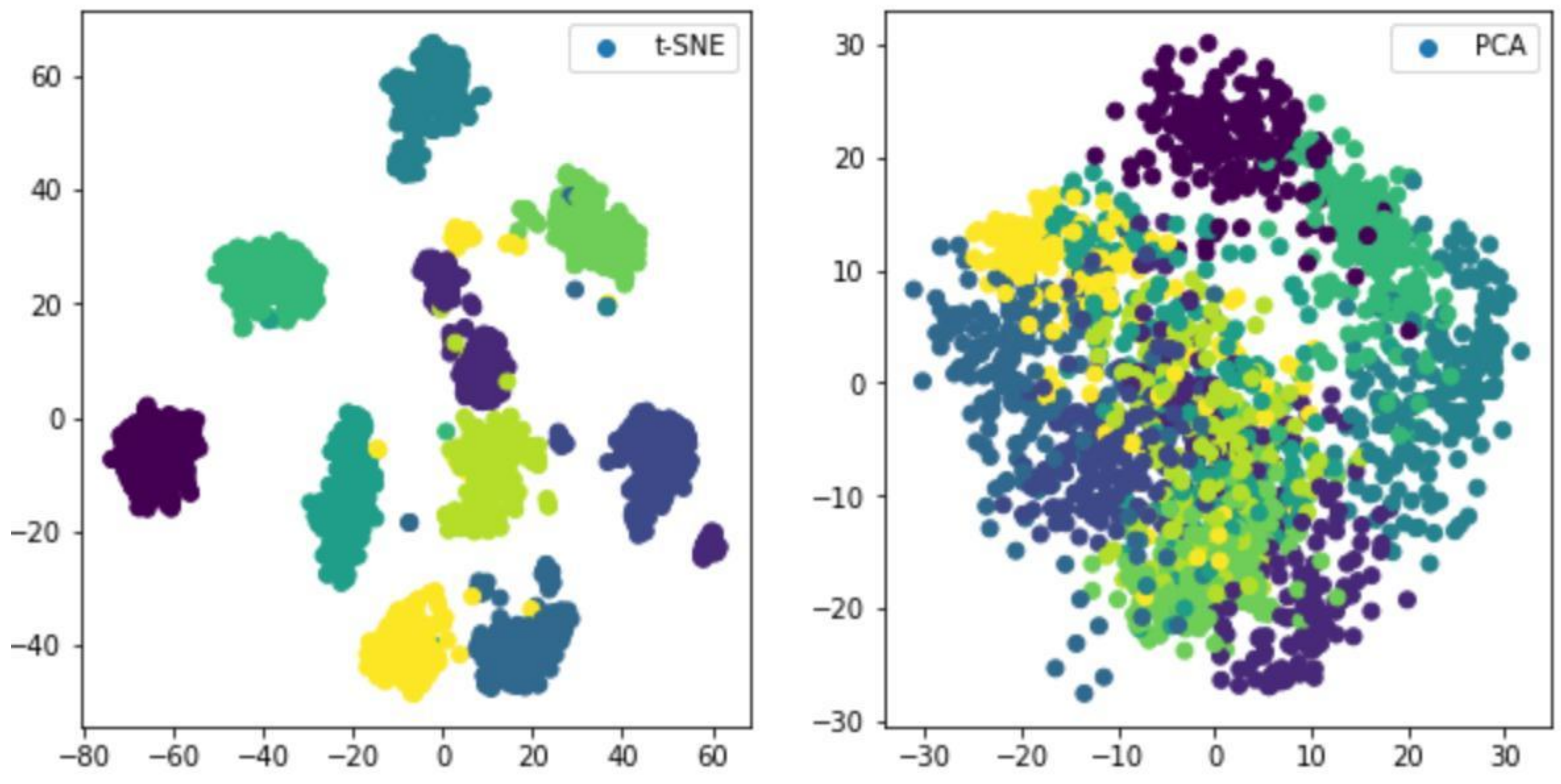
K-Means clustering
can be used to
detect possible outliers

Hierarchical clustering
can be used to
find underlying connectivity properties

https://www.neuroelectrics.com/blog/2018/07/06/clustering-methods-in-exploratory-analysis/

Dimensionality Reduction for EDA

- Reduce the dimensions of the data into fewer dimensions would help describing the relationship between variables



T-distributed stochastic neighbor embedding (T-SNE) and Principal Component Analysis (PCA)

https://www.programmersought.com/article/92363395092/

What to look for in your plots?

- Turn the information into useful questions
    - Which values are the most common? Why?
    - Which values are rare? Why?
    - Can you see any unusual patterns? What might explain them?

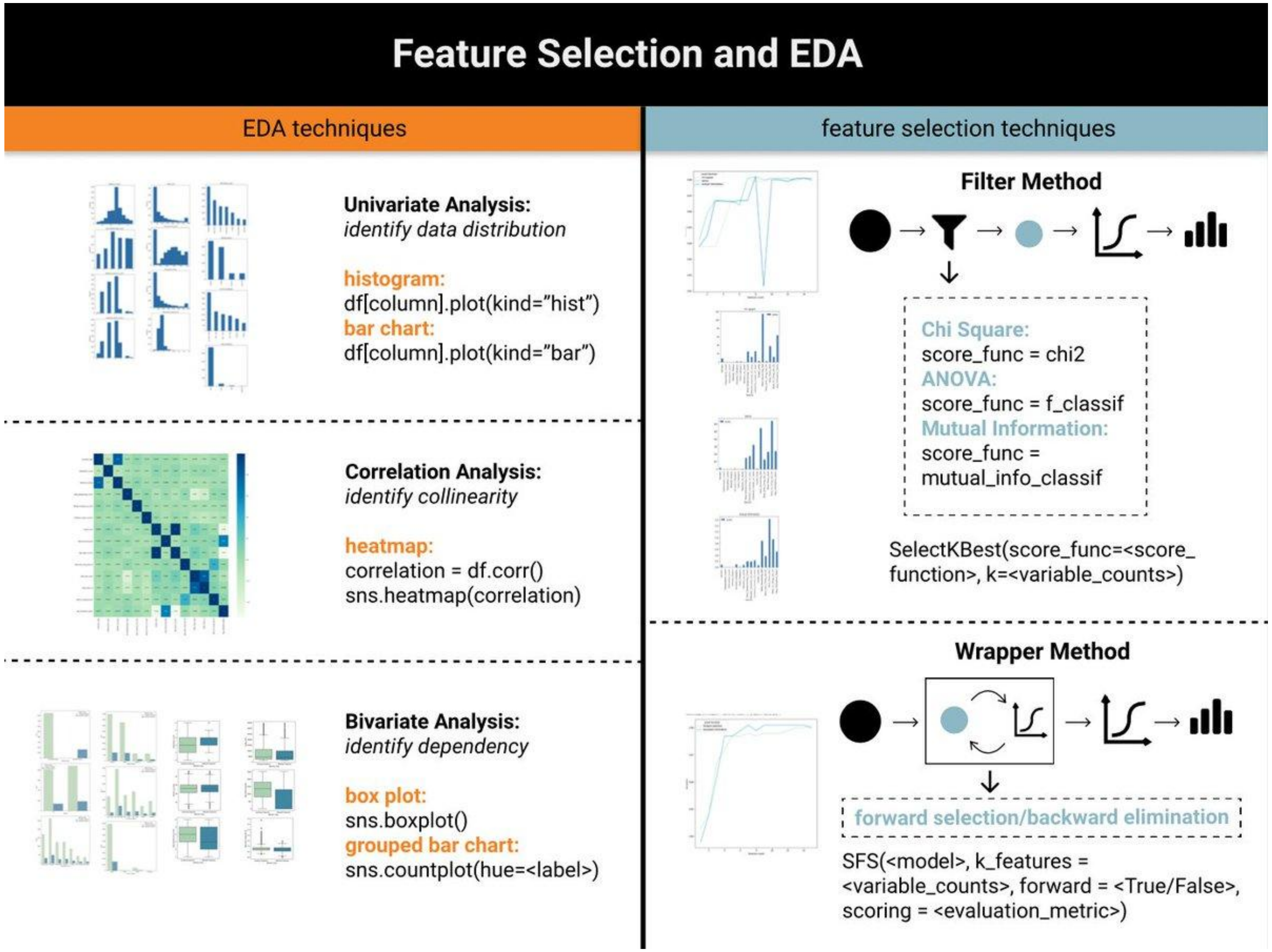
- Clusters suggest that subgroups exist in your data.
    - How can you explain or describe the clusters?
    - How are the observations within each cluster similar to each other?
    - How are the observations in separate clusters different from each other?

EDA and Data Preprocessing

EDA helps us to prepare it for the upcoming tasks,
e.g. data preprocessing and modelling

EDA	***Data Preprocessing***
Duplicate Data-Points?	Delete Duplicate Data-Points
Missing Values?	Imputation
Outlier?	Exclude outliers
Highly Correlated Features?	Handling Highly Correlated Features
Low-Variance Features?	Handling Low-Variance Features
Imbalanced Data?	Oversampling, Undersampling
Features vary in their scale?	Feature Scaling
High-dimensional data?	Dimensionality Reduction, Feature selection

EDA and Feature Selection



Feature Selection and EDA

EDA techniques

Univariate Analysis:
identify data distribution

histogram:
df[column].plot(kind="hist")
bar chart:
df[column].plot(kind="bar")

Correlation Analysis:
identify collinearity

heatmap:
correlation = df.corr()
sns.heatmap(correlation)

Bivariate Analysis:
identify dependency

box plot:
sns.boxplot()
grouped bar chart:
sns.countplot(hue=<label>)

feature selection techniques

Filter Method

Chi Square:
score_func = chi2
ANOVA:
score_func = f_classif
Mutual Information:
score_func =
mutual_info_classif

SelectKBest(score_func=<score_function>, k=<variable_counts>)

Wrapper Method

forward selection/backward elimination

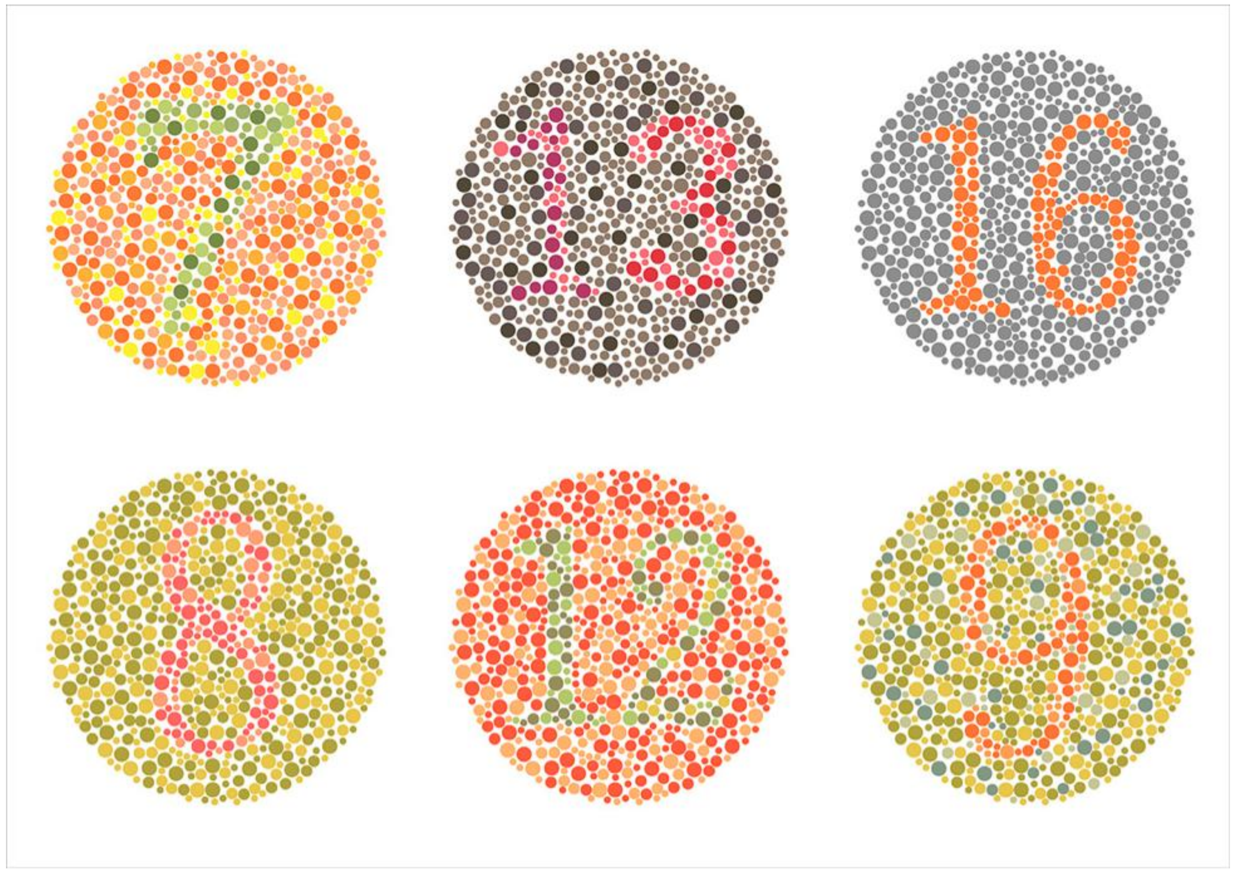
SFS(<model>, k_features =
<variable_counts>, forward = <True/False>,
scoring = <evaluation_metric>)

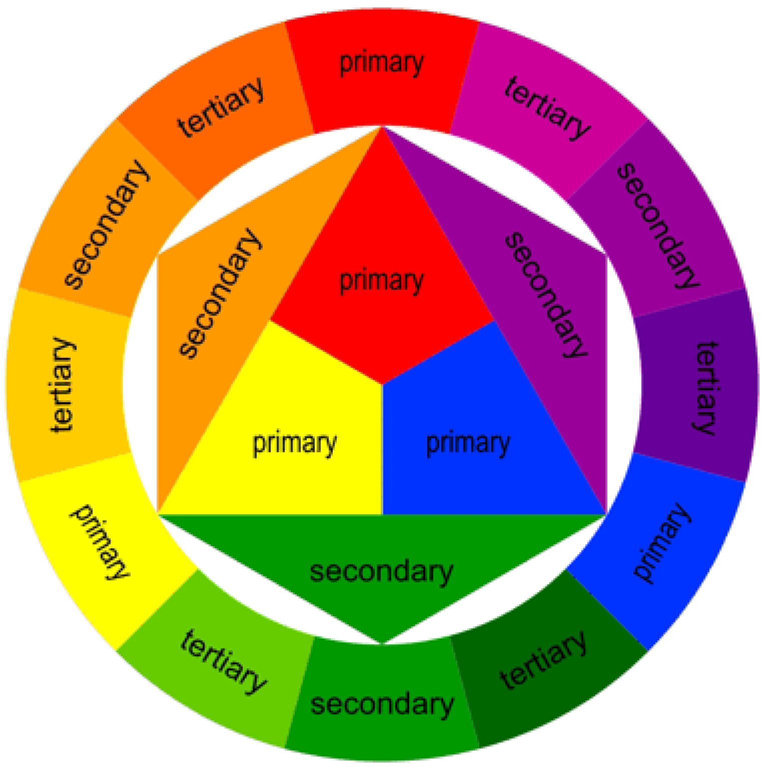
https://www.visual-design.net/post/feature-selection-and-eda-in-machine-learning

Be Colorblind-friendly - Make Your Charts Accessible

- Stop using red and green, blue and yellow together
- If must, leverage light vs. dark, offer alternate methods to distinguishing data
- Stop using the stoplight palette
- Use a colorblind-friendly palette:
  https://colorbrewer2.org/#type=diverging&scheme=BrBG&n=5

  Color wheel: https://graf1x.com/the-color-wheel-chart-poster/





https://www.tableau.com/about/blog/examining-data-viz-rules-dont-use-red-green-together

Lesson 03 - 02

Exploratory Data Analysis Tools

Python Modules

Week Overview

- Exploratory Data Analysis (EDA) Introduction
  - ➢ What is EDA
  - ➢ Why need EDA, what can do with EDA
  - ➢ How to do EDA
- EDA Techniques
  - ➢ Groups of EDA Methods
  - ➢ Visualisation Methods
- EDA Tools
  - ➢ Python Modules for EDA
- EDA Case Studies

EDA Tools

Some of the open source tools to facilitate EDA

- Python: This is an open source programming language widely used in data analysis, data mining, and data science
- R programming language: an open source programming language that is widely utilized in statistical computation and graphical data analysis, provides packages like ggplot2 for data visualisation
- Weka: This is an open source data mining package that involves several EDA tools and algorithms
- Orange: This is an open source and workbench-style tool for data analysis

Python Essential Modules for EDA

- Brief Overview
  - **Numerical Python (Numpy)**: matrix / numerical analysis layer, the fundamental library , powerful tool: ndarray
  - **Scientific Python (Scipy)**: scientific computing utilities/functions (linalg, mathematical approximation, etc…)
  - **Scikit-learn (sklearn)**: machine learning toolbox
  - **Matplotlib**: plotting and visualization
  - **Pandas**: data manipulation and analysis, powerful data structures including data frame and series

  Many others:
    - OpenCV: computer vision
    - Statsmodels: Statistics in Python
    - Seaborn, Bokeh, Plotly: Plotting and Visualisation
    - NLTK, Gensim: NLP tools
    - Theano, Caffe, Pytorch, Lasagne: Deep Learning
    - Pybrain, Pylearn2: Machine learning
    - DEAP, NEAT-python: Evolutionary Computation

Software - Anaconda

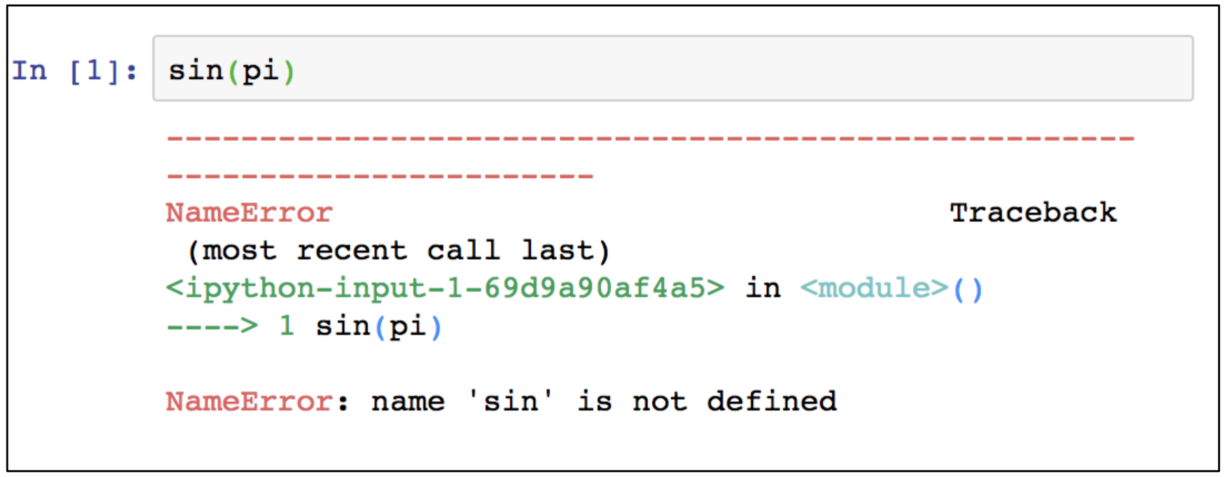
- What is Anaconda (miniconda)?
  - An essential large (~400 mb) Python installation
  - It contains almost everything your need for basic machine learning and data analysis
- Why to use Anaconda?
  - gives the user ability to make an easy install of the version of python
  - gives high performance computing with Anaconda Accelerate and several other components
  - *removes bottlenecks* involved in installing the right packages while taking into considerations their compatibility
  - no risk of messing up required system libraries
  - over *7500 open source packages*, manyof which are *not* in the pip repo

Software - IDEs

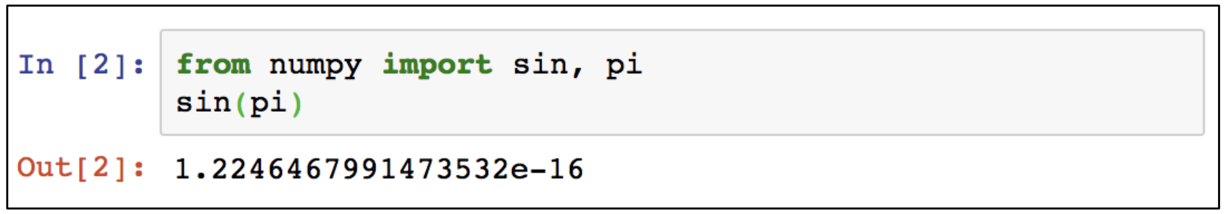
- Recommend IDEs:
  - Ipython (Jupyter Notebook)
    - Ipython stands for "Interactive Python"
    - Writing *markdown* at the same time
    - More reasons can be found: http://pythonforengineers.com/why-ipython-is-the-best-thing-since-sliced-bread/
  - PyCharm EDU
    - A real powerful IDE
    - Many good features (file browser, intelligent auto-completion, run in IDE with input & console windows for trial and error development)
    - free to download here: https://www.jetbrains.com/pycharm-edu/
  - RStudio: an integrated development environment for R

Using Python Modules

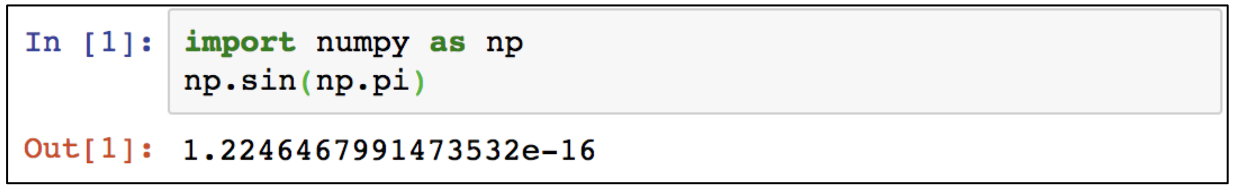
- Python libraries are called modules
- Each module needs to be imported before use.
- Three common alternatives:
  - Import the full module:
    - import numpy
  - Import selected functions from the module:
    - from numpy import array, sin, cos
  - Import all functions from the module: (Not Recommended)
    - from numpy import *
- All modules support shortcuts:
  - *e.g.,* import numpy as np



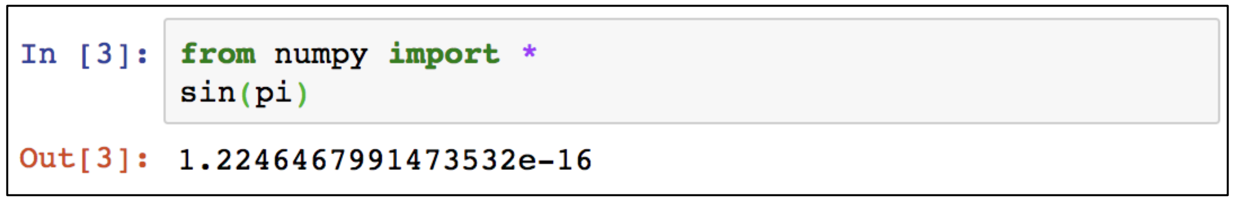
```
In [1]: sin(pi)

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-69d9a90af4a5> in <module>()
----> 1 sin(pi)

NameError: name 'sin' is not defined
```



```
In [2]: from numpy import sin, pi
        sin(pi)

Out[2]: 1.2246467991473532e-16
```



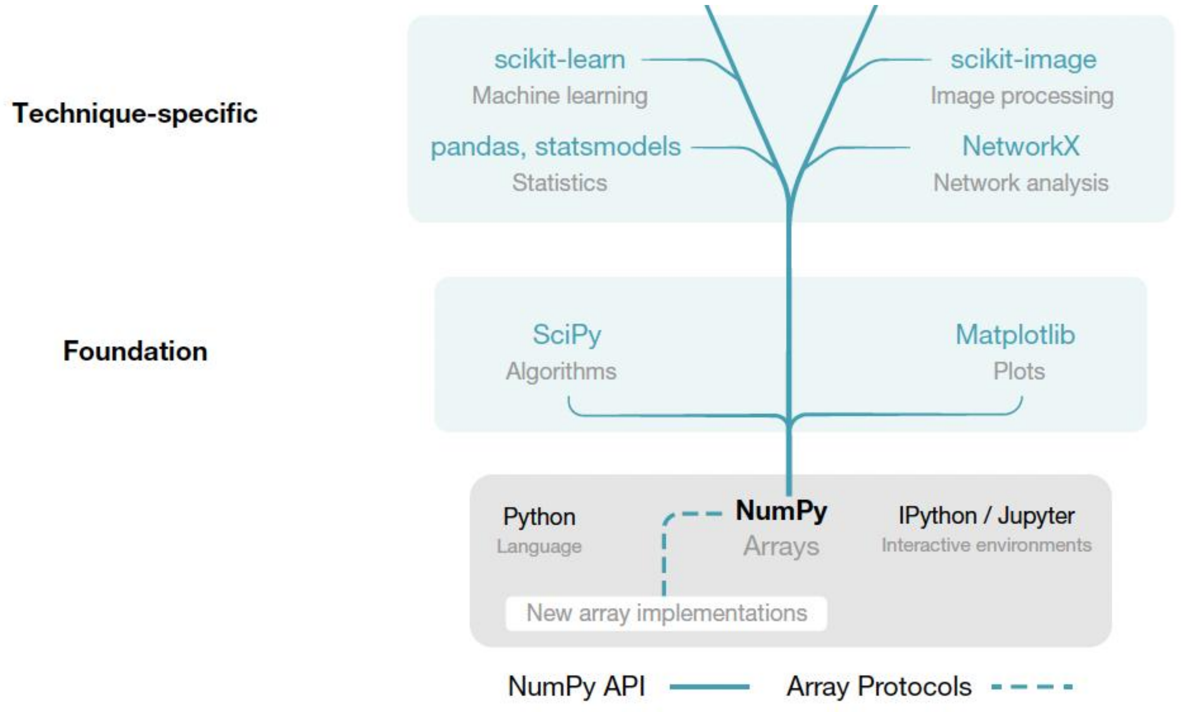
```
In [1]: import numpy as np
        np.sin(np.pi)

Out[1]: 1.2246467991473532e-16
```



```
In [3]: from numpy import *
        sin(pi)

Out[3]: 1.2246467991473532e-16
```

Numpy

- Numpy is the foundation for scientific computing in Python
- It has many features:
  - powerful for *large, multi-dimensional arrays* object
  - basic linear algebra functions
  - basic Fourier transforms
  - sophisticated random number capabilities
  - tools for integrating C/C++ code, Fortran code



- More information at the Numpy web page: http://www.numpy.org/

Numpy for EDA

- Data manipulation: NumPy provides powerful array objects that allow efficiently manipulate and process data, e.g., filtering, slicing, indexing, and reshaping data, making it easier to extract relevant information from your dataset.

- Mathematical operations: NumPy supports various mathematical operations, enabling you to perform element-wise arithmetic, logarithmic, trigonometric, and other mathematical functions on arrays of data

- Descriptive statistics: NumPy offers statistical functions to compute descriptive statistics on your data, e.g., mean, median, standard deviation, variance, min, max, percentiles, etc.

- Data visualization: plays an essential role in data preparation for visualization libraries like Matplotlib and Seaborn

SciPy and SciPy for EDA

SciPy stands for Scientific Python, is a collection of mathematical algorithms and convenience functions built upon Numpy

- more functional
- a powerful tool for advanced scientific computing and data analysis
- widely used in various scientific and engineering disciplines, including physics, chemistry, biology, economics, and data science

SciPy for EDA

- statistical tests: e.g., t-tests, chi-square tests, ANOVA, and correlation tests, to assess relationships between variables and identify significant differences in data, and for hypothesis testing to validate assumptions and draw conclusions about the data.
- outlier detection (‘scipy.stats.zscore’),
- distance calculations (scipy.spatial.distance), …

Pandas

- Pandas (Python Data Analysis Library), a python library that provides easy-to-use data structures and data analysis tools for data manipulation, data cleaning, data exploration, and data preparation tasks in data science and data analysis
- built on Numpy, Scipy, and Matplotlib (to some extent).
- many features:
  - support for CSV, Excel, JSON, SQL, HDF5, …
  - powerful tools: DataFrame and Series
  - data cleansing
  - re-shape & merge data (joins & merge) & pivoting
  - data visualisation
  - database-like operations: filtering, sorting, grouping, aggregating, merging, and joining
- more information at the Pandas web page: https://pandas.pydata.org/

Pandas for EDA

- Data Inspection and Summary: Pandas provides methods like head(), tail(), info(), describe(), .. and shape to quickly inspect the data and get a summary of the dataset, including the data types, missing values, and basic statistics

- filter and select data based on conditions using boolean indexing, methods like loc[], iloc[], and boolean expressions to extract relevant subsets of data

- powerful grouping and aggregation functions using groupby(). You can group data based on one or more columns and apply aggregate functions like sum(), mean(), count(), etc., to obtain insights and summaries

- reshape and pivot data using methods like pivot(), melt(), stack(), and unstack()

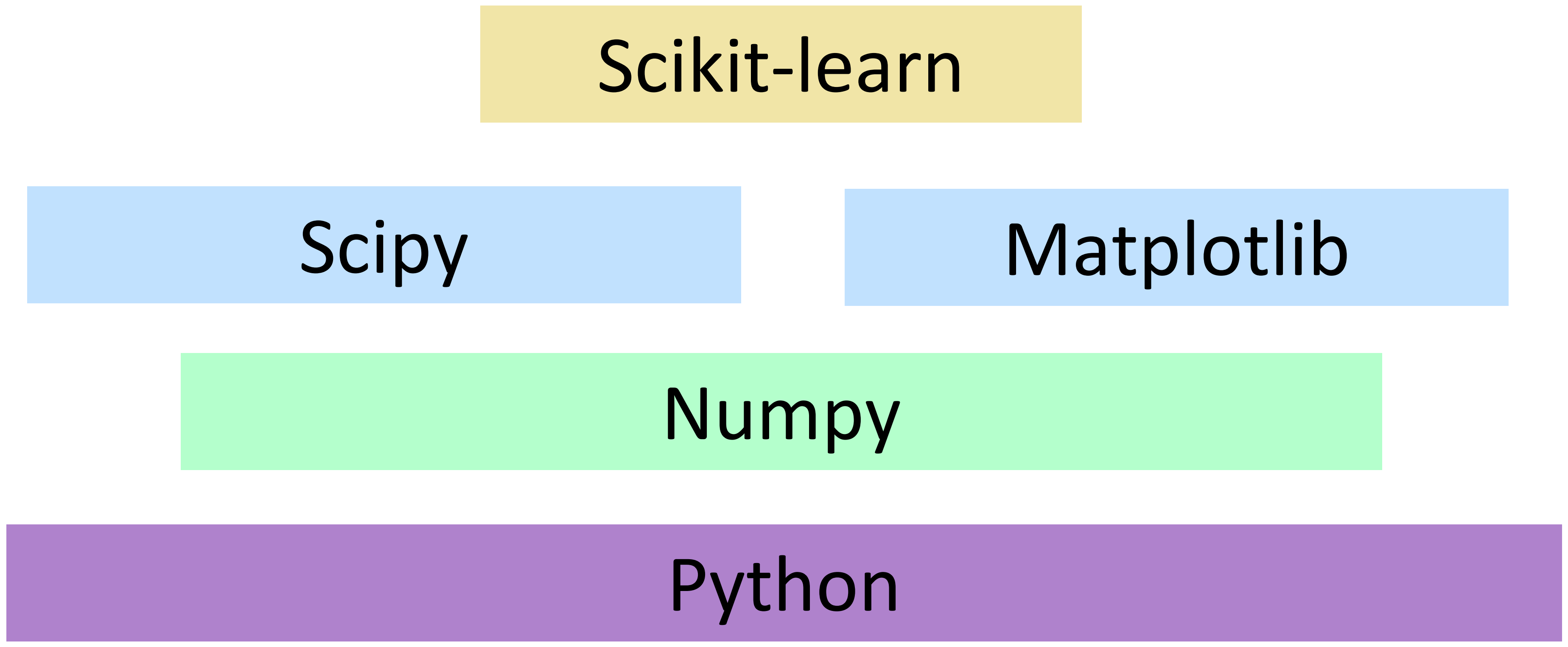
- provide a simple interface for creating basic visualizations directly from DataFrames and Series using the '.plot()' method

Matplotlib and Seaborn

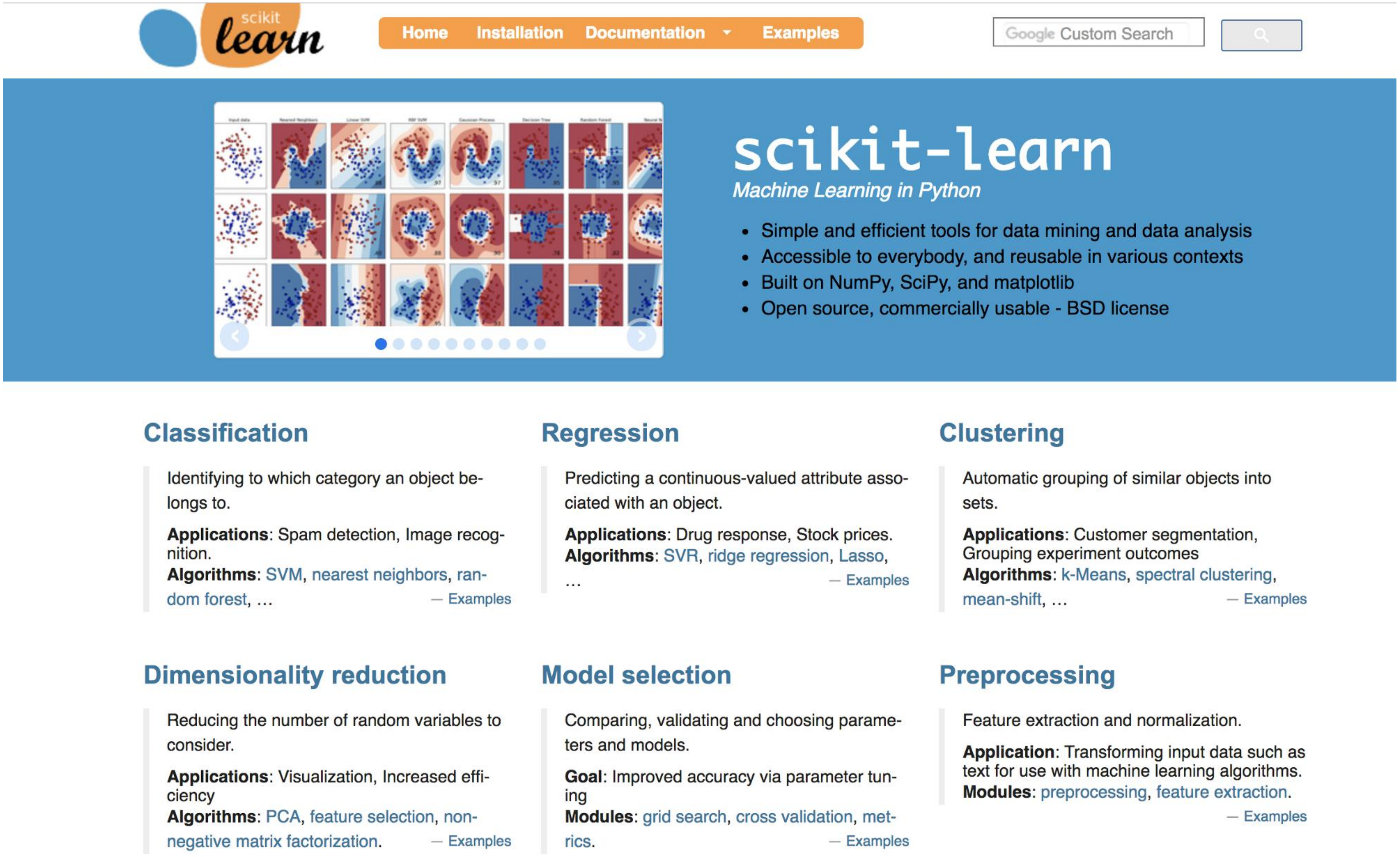
- Matplotlib is a comprehensive graphics library for generating scientific figures
  - high-quality output in many formats, including PNG, PDF, SVG, EPS, and PGF
  - efficiently with data frames and arrays
  - GUI for interactively exploring figures but supports figure generation without the GUI
  - integrates well with Jupyter notebooks
  - support 3D
- More information at the Matplotlib web page: http://matplotlib.org/
- Seaborn: another Python data visualization library based on matplotlib
  - have a polished default style, a better looking figure
  - more convenient when using a pandas DataFrame
  - 3D is not supported
- For basic plots, use Matplotlib, for more advanced statistical visualizations, use Seaborn

Scikit-learn

- What is Scikit-learn in Python?
  - **Sci**Py Tool**Kit**
  - Scikit-learn (sklearn): an open source library that provides a consistent API for using traditional state-of-the-art ML algorithms/methods in Python.
  - majorly written in Python, some of sklearn's internal algorithms are written in ***Cython*** using third party bindings.
  - Sklearn's *major Python* dependencies are scipy, numpy and matplotlib.



Scikit-learn - Overview



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

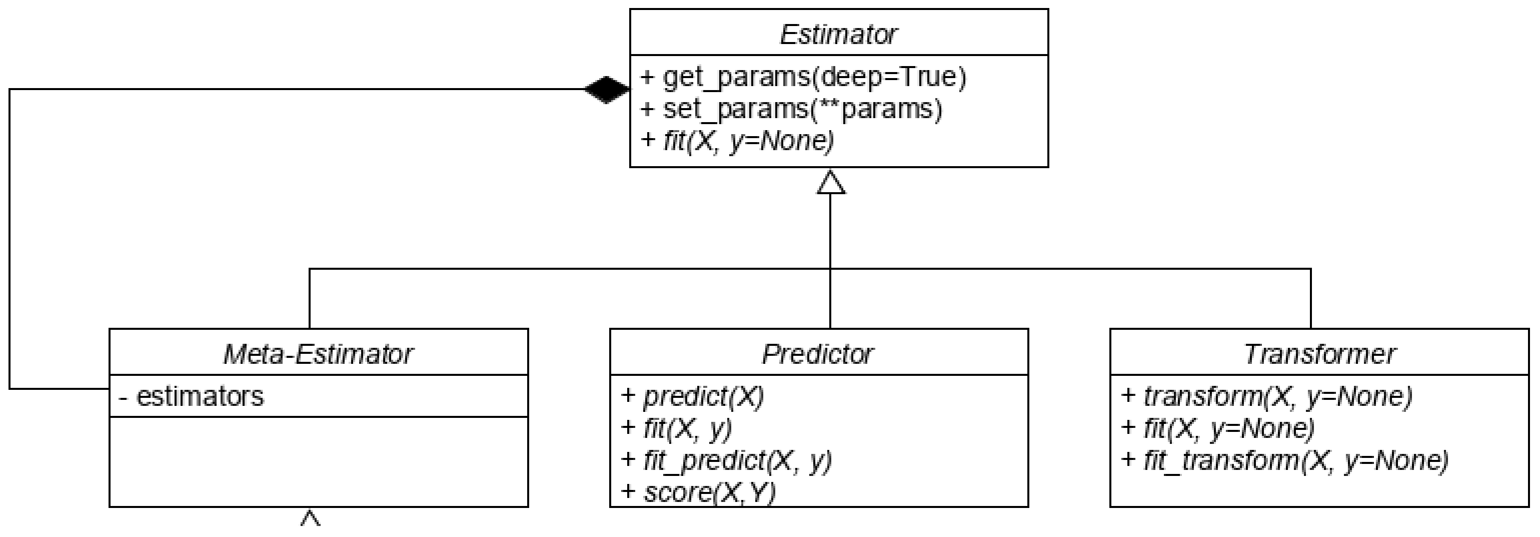
Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.
— Examples

http://scikit-learn.org/stable/index.html

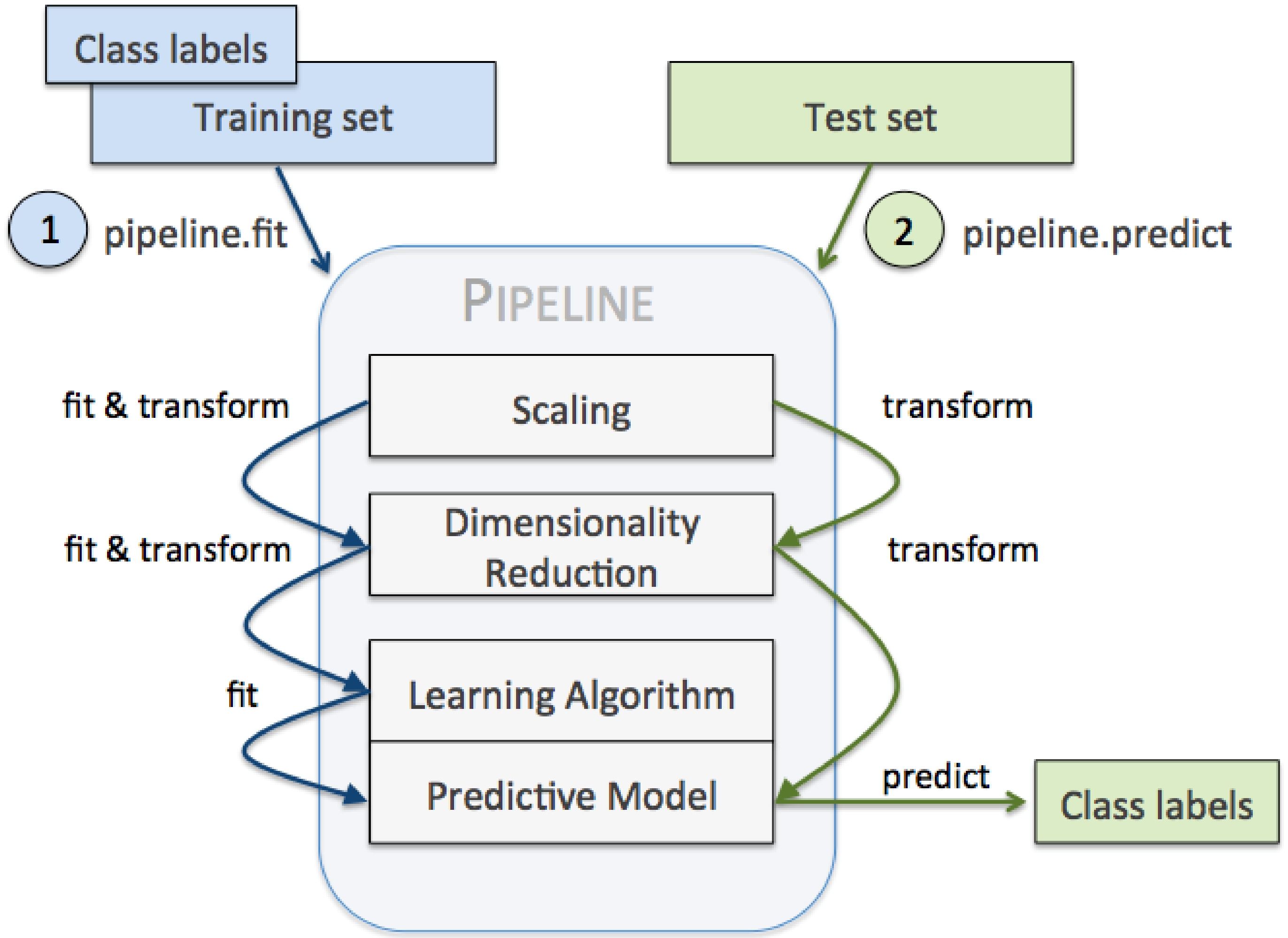
Scikit-learn – APIs

- Estimators and Meta-estimators:
  - Estimators: define instantiation mechanism of objects and expose a ***fit*** method for learning a model from training data, E.g., *LinearRegression*
  - Meta-estimators : combine one or more estimators into a single estimator E.g. *ensemble, RF*
- Predictors: a ***predict*** method that takes an array "X test" and produces predictions for "X test", based on the learned parameters of the estimator.
- Transformers
  - modify or filter data before feeding, a transformer interface which defines a transform method, *E.g. StandardScaler()*



Scikit-learn – APIs

- Pipelines and Feature Unions
  - A *distinguishing* feature of the Scikit-learn API is its ability to compose new estimators from several base estimators. Two ways:
    - Pipeline objects chain multiple estimators into a single one.
    - FeatureUnion objects combine multiple transformers into a single one that concatenates their outputs
- Pipelines:

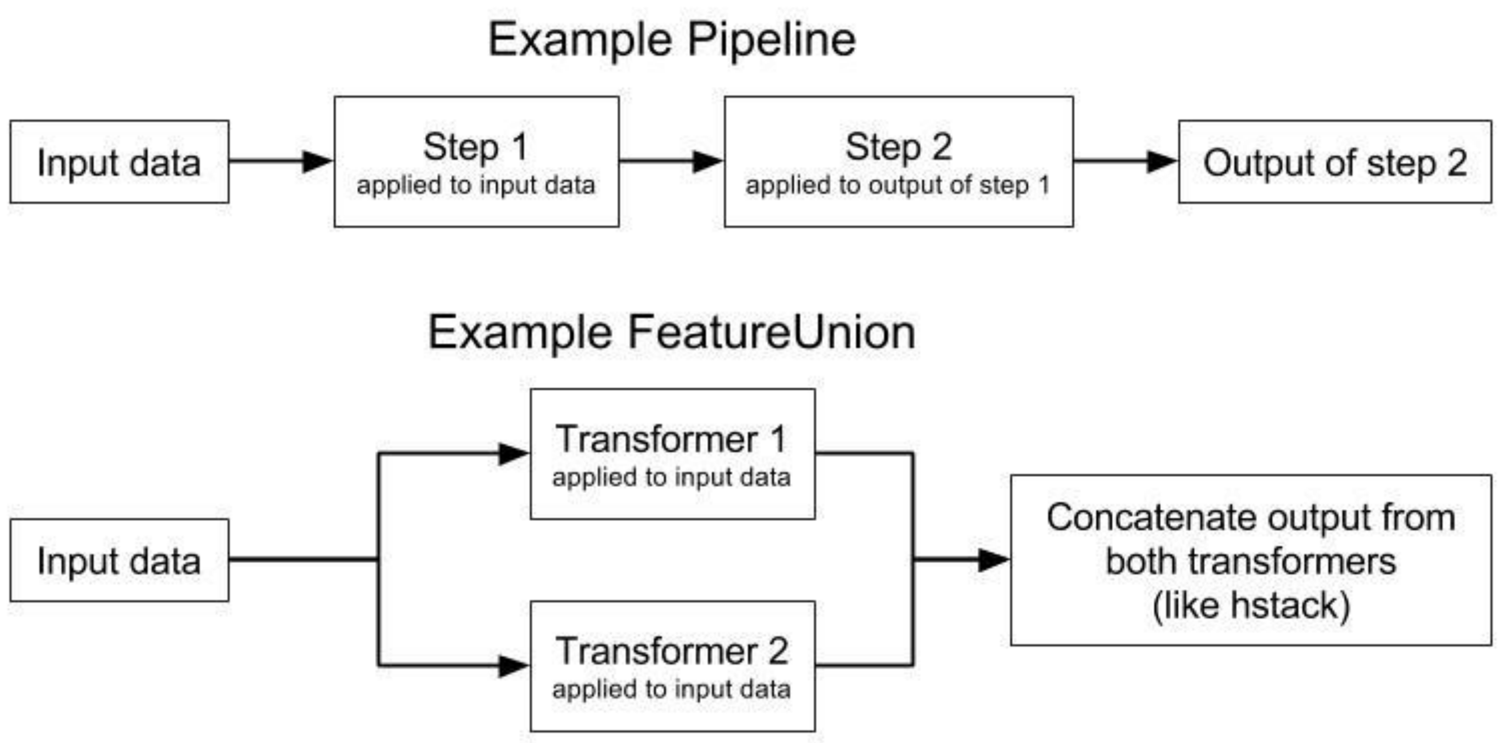


Scikit-learn – Feature Union

- apply different transformers on the same input data in parallel and concatenate the outputs
- a FeatureUnion takes as input a list of transformers. Calling *fit* on the union is the same as calling *fit* independently on each of the transformers and then joining their outputs

For example:

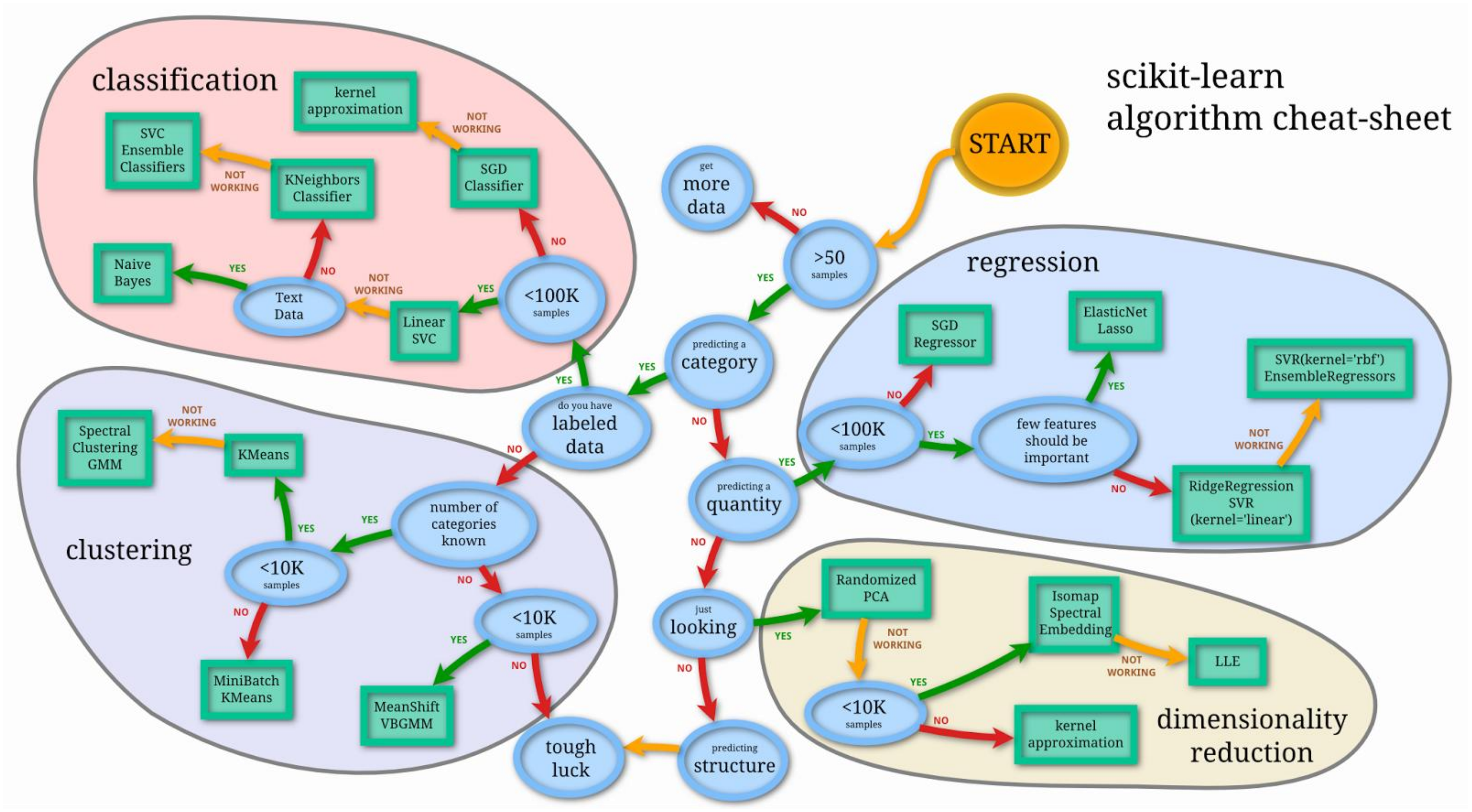
- *Join two feature transformers (i.e., PCA and UnivariateFeatureSelection) to construct combined features*



Scikit-learn for EDA

- Data Preprocessing: Scikit-learn provides preprocessing techniques, e.g., scaling, normalization, handling missing values, and encoding categorical variables, to ensure data is in a suitable format for analysis.
- Dimensionality Reduction: Scikit-learn offers techniques like Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) for dimensionality reduction.
- Feature Selection: identifying the most relevant features for modeling or understanding the relationships between variables.
- Unsupervised Learning: clustering algorithms (e.g., KMeans) for discovering patterns and structures.

Scikit-learn Cheat Sheet



https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

Scikit-Learn Cheat Sheet

Python For Data Science *Cheat Sheet*

Scikit-Learn

Learn Python for data science **Interactively** at www.DataCamp.com

Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross_validation import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

Training And Test Data

```
>>> from sklearn.cross_validation import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        random_state=0)
```

Create Your Model

Supervised Learning Estimators

Linear Regression

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

Support Vector Machines (SVM)

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

Naive Bayes

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

KNN

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

Unsupervised Learning Estimators

Principal Component Analysis (PCA)

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

K Means

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

Model Fitting

Supervised learning	
`>>> lr.fit(X, y)`	Fit the model to the data
`>>> knn.fit(X_train, y_train)`	
`>>> svc.fit(X_train, y_train)`	
Unsupervised Learning	
`>>> k_means.fit(X_train)`	Fit the model to the data
`>>> pca_model = pca.fit_transform(X_train)`	Fit to data, then transform it

Prediction

Supervised Estimators	
`>>> y_pred = svc.predict(np.random.random((2,5)))`	Predict labels
`>>> y_pred = lr.predict(X_test)`	Predict labels
`>>> y_pred = knn.predict_proba(X_test)`	Estimate probability of a label
Unsupervised Estimators	
`>>> y_pred = k_means.predict(X_test)`	Predict labels in clustering algos

Preprocessing The Data

Standardization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```

Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

Encoding Categorical Features

```
>>> from sklearn.preprocessing import LabelEncoder
>>> enc = LabelEncoder()
>>> y = enc.fit_transform(y)
```

Imputing Missing Values

```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```

Generating Polynomial Features

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

Evaluate Your Model's Performance

Classification Metrics

Accuracy Score

`>>> knn.score(X_test, y_test)`	Estimator score method

`>>> from sklearn.metrics import accuracy_score`
`>>> accuracy_score(y_test, y_pred)`
	Metric scoring functions

Classification Report

`>>> from sklearn.metrics import classification_report`
`>>> print(classification_report(y_test, y_pred))`
	Precision, recall, f1-score and support

Confusion Matrix

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

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

R^2 Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

Clustering Metrics

Adjusted Rand Index

```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

Homogeneity

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

V-measure

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

Tune Your Model

Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
              "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
                        param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5),
              "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
                                 param_distributions=params,
                                 cv=4,
                                 n_iter=8,
                                 random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```