wbs
WARWICK BUSINESS SCHOOL
THE UNIVERSITY OF WARWICK

For the Change Makers

Advanced Programming for Data Science

Week 8: Data Analysis and Modeling
Information Systems and Management
Warwick Business School

Predictive Analytics with Machine Learning

What Is Machine Learning?

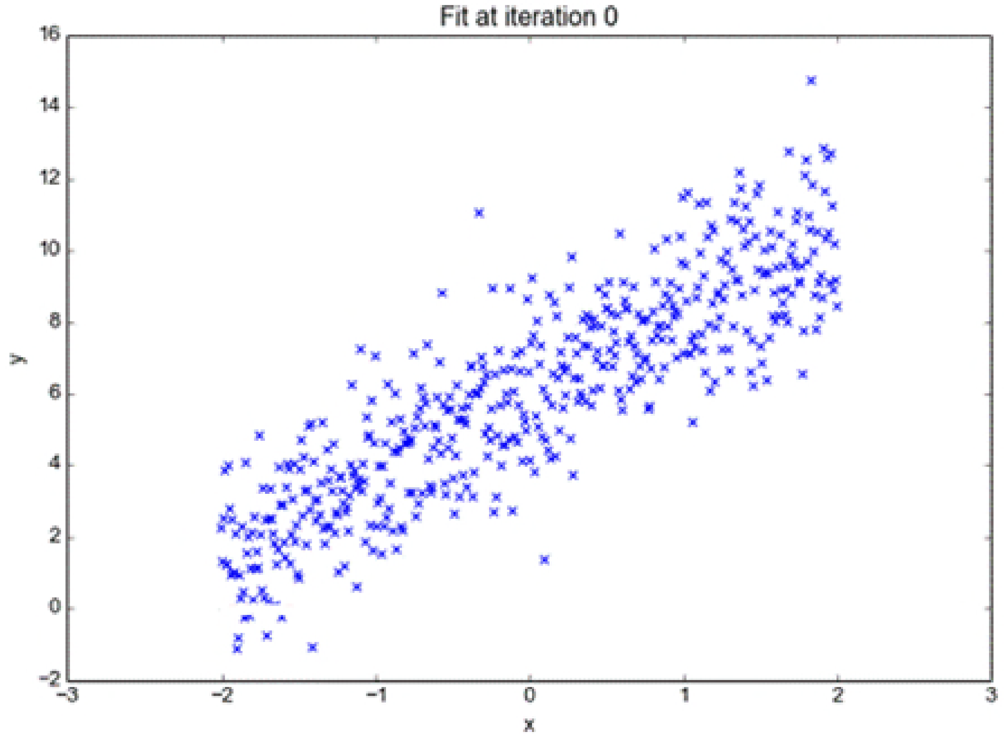
https://www.geeksforgeeks.org/machine-learning/

What Is Machine Learning?

"A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."

Statistic Analysis vs. Machine Learning

Simple Linear Regression: y = ax + b



Statistic Analysis vs. Machine Learning

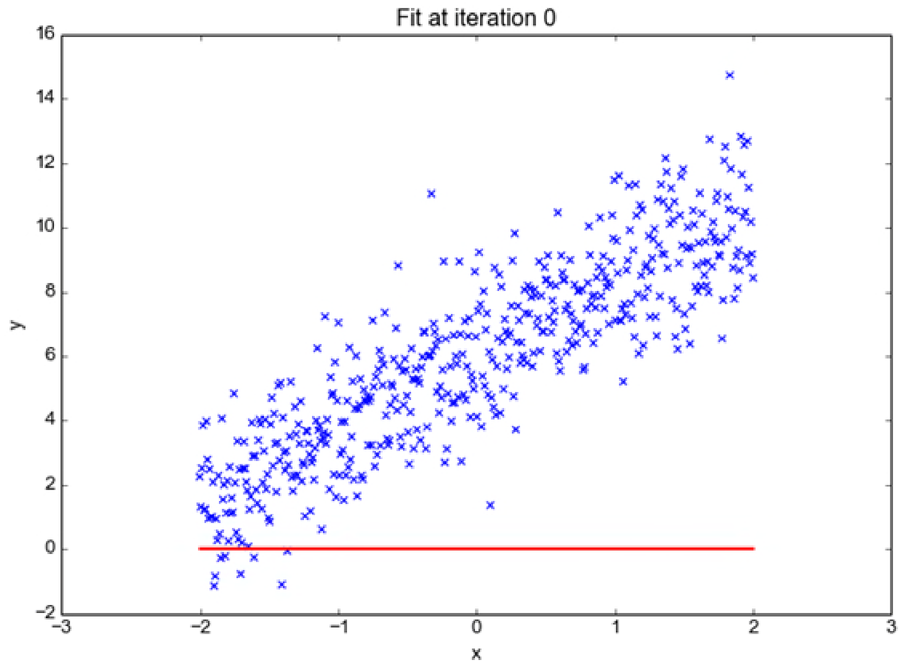
Simple Linear Regression: y = ax + b

- How to get a?
- Statistics:

Calculate $a = \frac{\sum[(x_i - \bar{x}) \cdot (y_i - \bar{y})]}{\Sigma[(x_i - \bar{x})^2]}$

- Machine Learning:

Minizine the loss function: $MSE = \frac{1}{n} \sum_{j=1}^{n} (yj - \hat{y}_j)^2$



Linear regression

- There are four assumptions associated with a linear regression model:
  - **Linearity**: The relationship between X and the mean of Y is linear.
  - **Homoscedasticity**: The variance of residual is the same for any value of X.
  - **Independence**: Observations are independent of each other.
  - **Normality**: For any fixed value of X, Y is normally distributed.
  - ***Collinearity**: Predictors should not be highly collinear.

  - Residual analysis ~ linearity, homoscedasticity, and independence
  - Normal Quantile ~ normality
  - VIF ~ collinearity

Warwick Business School wbs.ac.uk

nature methods

Published: 03 April 2018

Points of Significance

Statistics versus machine learning

Danilo Bzdok, Naomi Altman & Martin Krzywinski

Nature Methods **15**, 233–234(2018) | Cite this article

Statistics draws population inferences from a sample, and machine learning finds generalizable predictive patterns.

Two major goals in the study of biological systems are inference and prediction. Inference creates a mathematical model of the data-generation process to formalize understanding or test a hypothesis about how the system behaves. Prediction aims at forecasting unobserved outcomes or future behavior, such as whether a mouse with a given gene expression pattern has a disease. Prediction makes it possible to identify best courses of action (e.g., treatment choice) without requiring understanding of the underlying mechanisms. In a typical research project, both inference and prediction can be of value—we want to know how

Statistical Analysis vs. Machine Learning

- Objectives: inference vs. prediction
- Assumptions: strict vs. loose
- Dataset: low dimension vs. high dimension; small vs. big
- Relationships and interaction: simple vs. complex

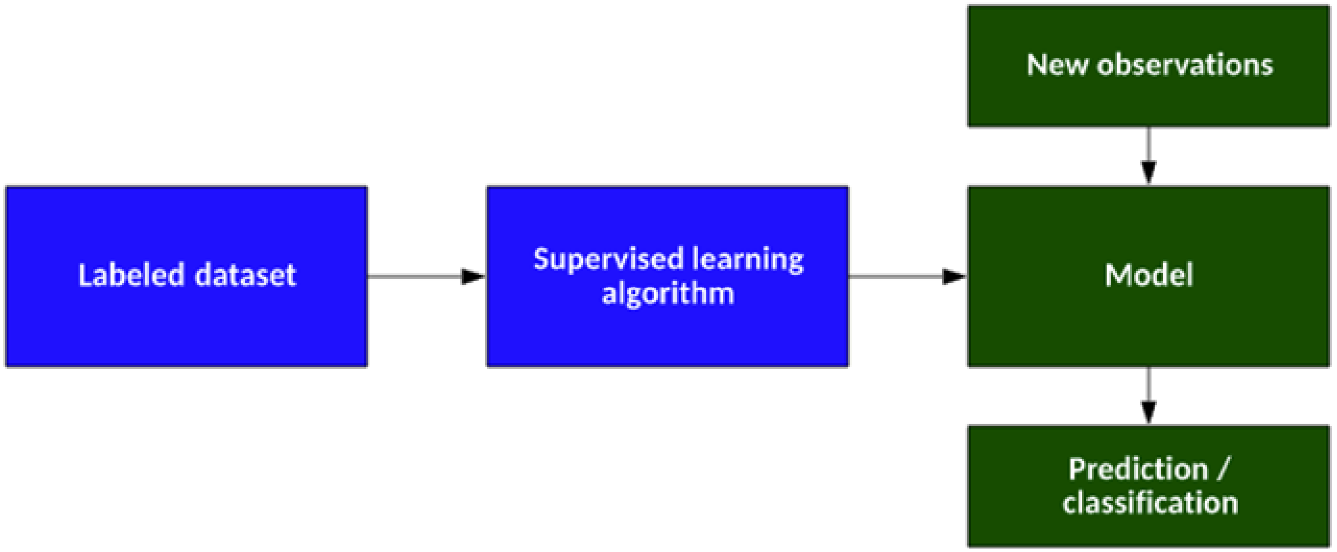
Categories of machine learning

- Supervised Learning
  - ➢Classification
  - ➢Regression
- Unsupervised Learning
  - ➢Clustering
  - ➢Dimensionality reduction

Supervised learning

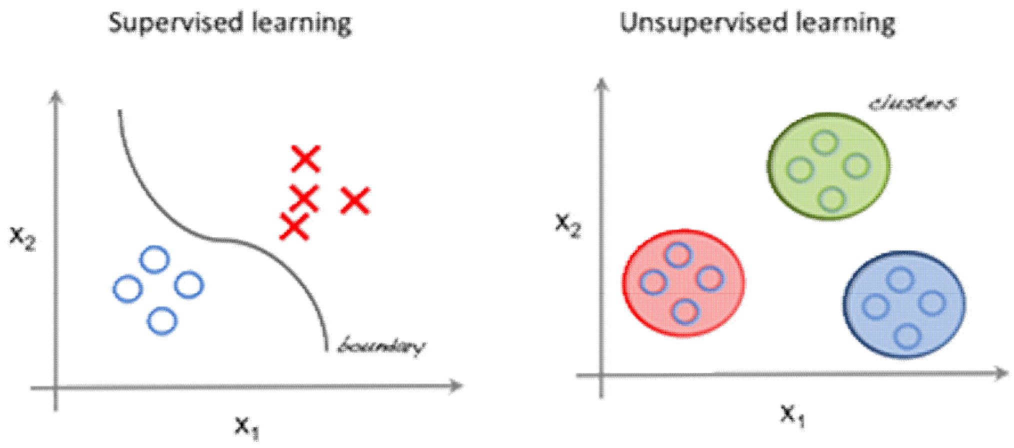
- Model training process that takes in data samples and associated outputs (known as labels or responses) to learn the relationship or mapping between inputs x (features) and corresponding outputs y (labels).
- This learned knowledge (model) can then be used in the future to predict an output y' for any new input data sample x'.
- So called "supervised" as the model learns on data samples where the desired output responses/labels are already known beforehand in the training phase. The learning is done through adjustment from supervision.
- Best for predictive tasks.

X	Y
...	...
...	...
...	...
...	...
...	...
...	...
...	...

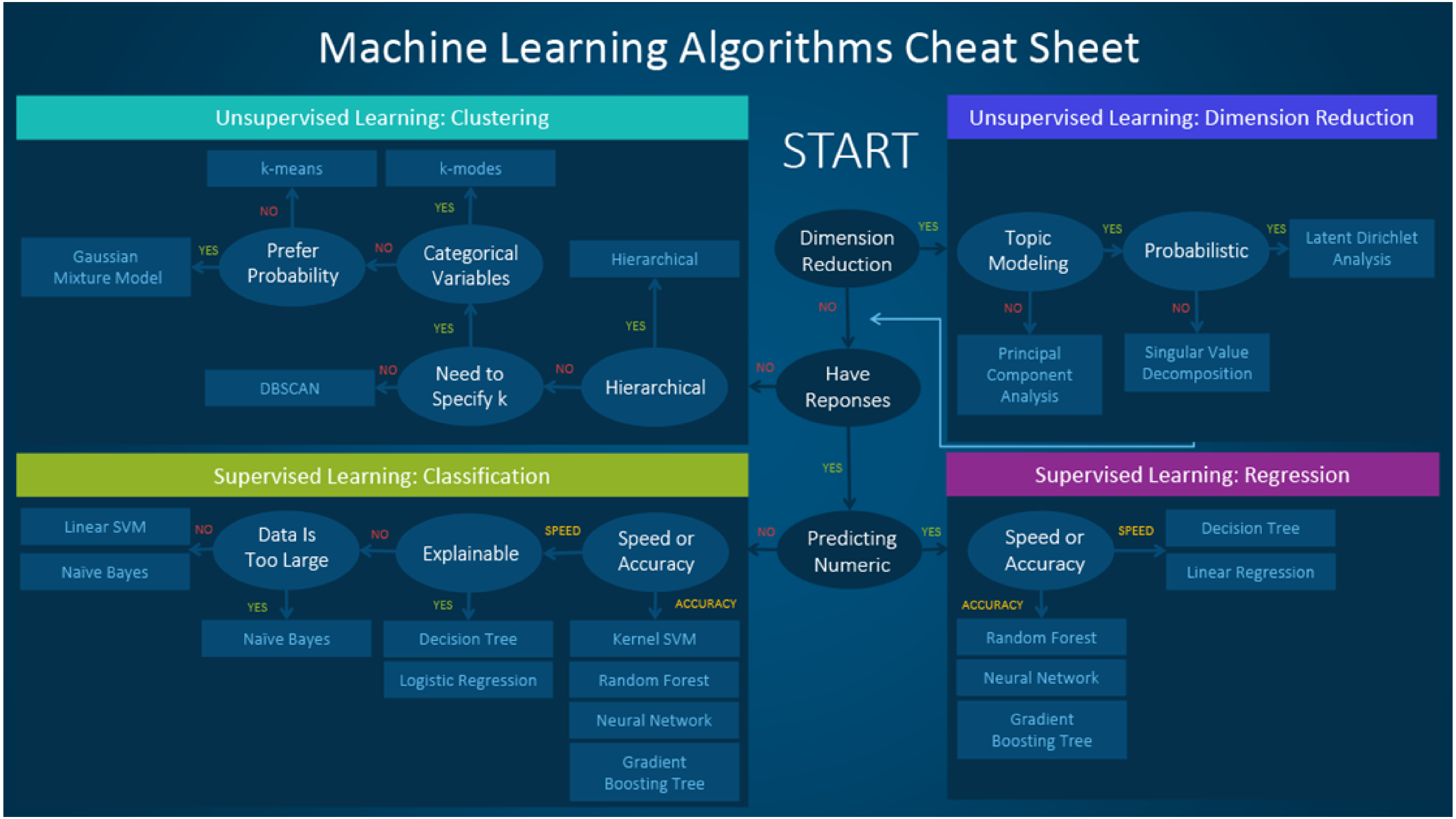

New observations
Labeled dataset
Supervised learning algorithm
Model
Prediction / classification

Unsupervised learning

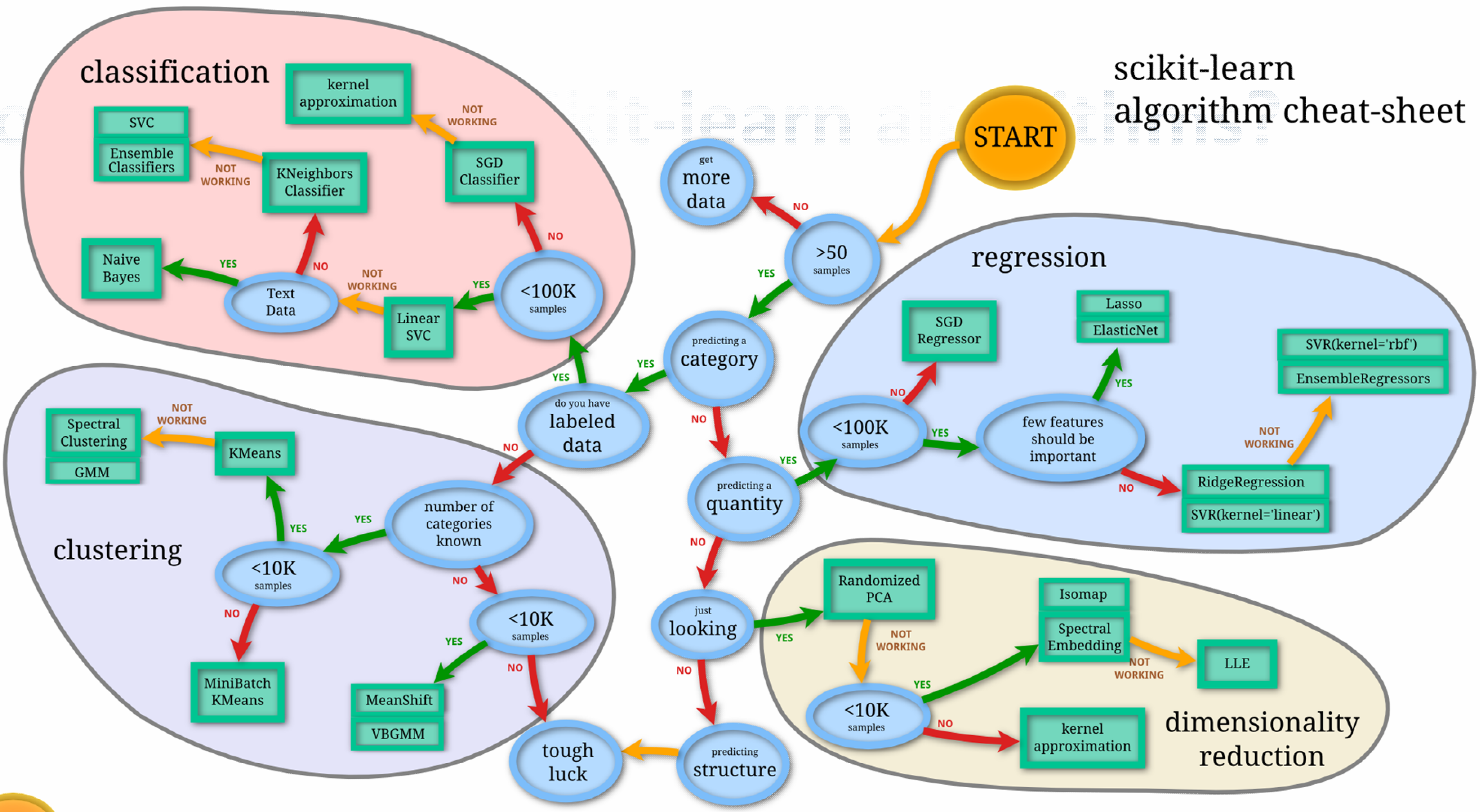
- There is no labeled training data to learn from: only the input variables(X) are given with no corresponding output variables (Y).
- The machine tries to infer the hidden structure in the dataset.



https://towardsdatascience.com/unsupervised-learning-with-python-173c51dc7f03



https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/



https://scikit-learn.org/stable/tutorial/machine_learning_map/

scikit-learn

- Simple and efficient tools for data mining and data analysis
- Built on NumPy, SciPy, and matplotlib
    1. Classification
    2. Regression
    3. Clustering
    4. Dimensionality reduction.
    5. Model selection
    6. Preprocessing

Preprocessing in sklearn.

- The **sklearn.preprocessing** module provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.
- While Pandas provides many similar functionalities as sklearn.preprocessing , you are advised to use sklearn for machine learning tasks and Pandas for simpler analysis.

Univariate Imputation of missing values

- The SimpleImputer class provides basic strategies for imputing missing values.
- Missing values can be imputed with a provided constant value, or using the statistics (mean, median or most frequent)
- Key parameters:
  1. missing_values : number, string, np.nan (default) or None
  2. strategy : string, such as "mean" (default), "median", "most_frequent", "constant", optional
  3. fill_value : string or numerical value, optional (default=None)

Two-step imputation

- Step 1: create the imputation transformer

```
from sklearn.impute import SimpleImputer # import the imputer class
imp = SimpleImputer(strategy='mean') # set the imputer
```

- Step 2: apply the transformer using method `fit_transform()`, the data needs to be a dataframe/array

```
df_titan['Age'] = imp.fit_transform(df_titan[['Age']])
df_titan = pd.DataFrame(imp.fit_transform(df_titan))
```

Note: Sklearn would transfer DataFrame into ndarray if you do global transformation.

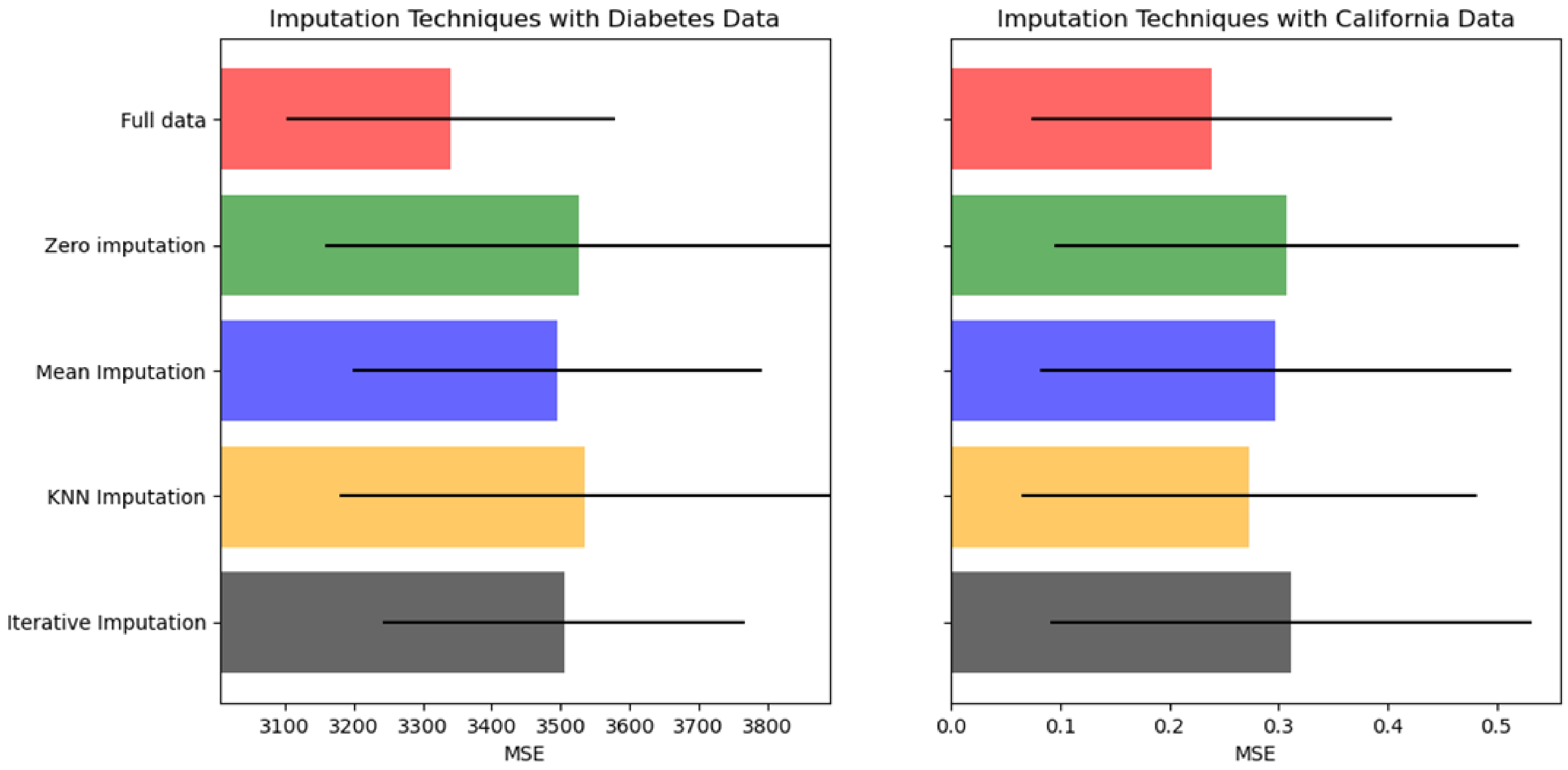
Other Imputation of missing values

- The IterativeImputer class estimates missing values as a function of other features, and uses that estimate for imputation.
  - Iteratively estimates **EACH** feature from all the others.
  - Linear regression by default but can be changed to other estimators.

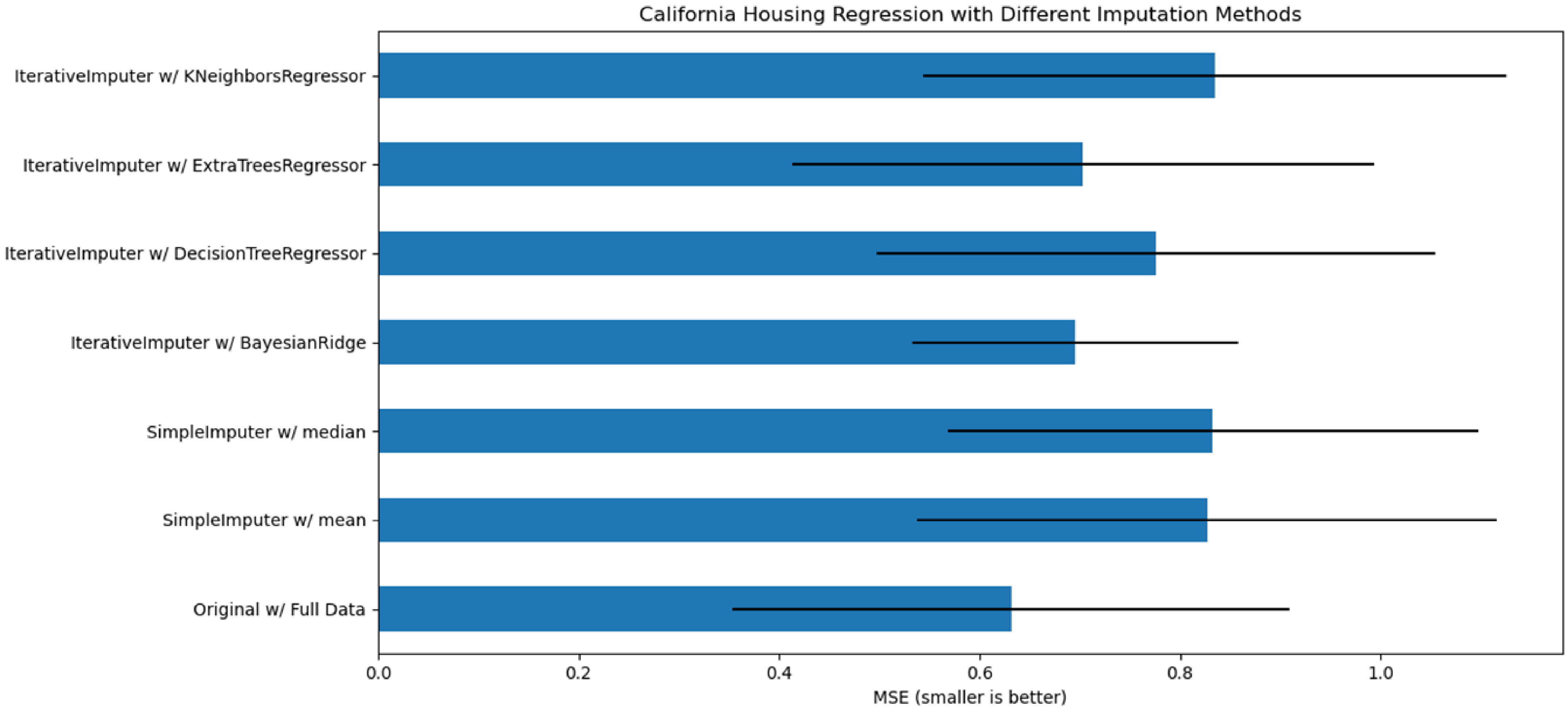
```
imp = IterativeImputer(n_nearest_features=2,
estimator=BayesianRidge())
```

- The KNNImputer class estimates missing values from nearest neighbors that have a value for the feature.
  - The values from neighbors will be averaged uniformly or weighted.

```
imp = KNNImputer(n_neighbors=2, weights="uniform")
```



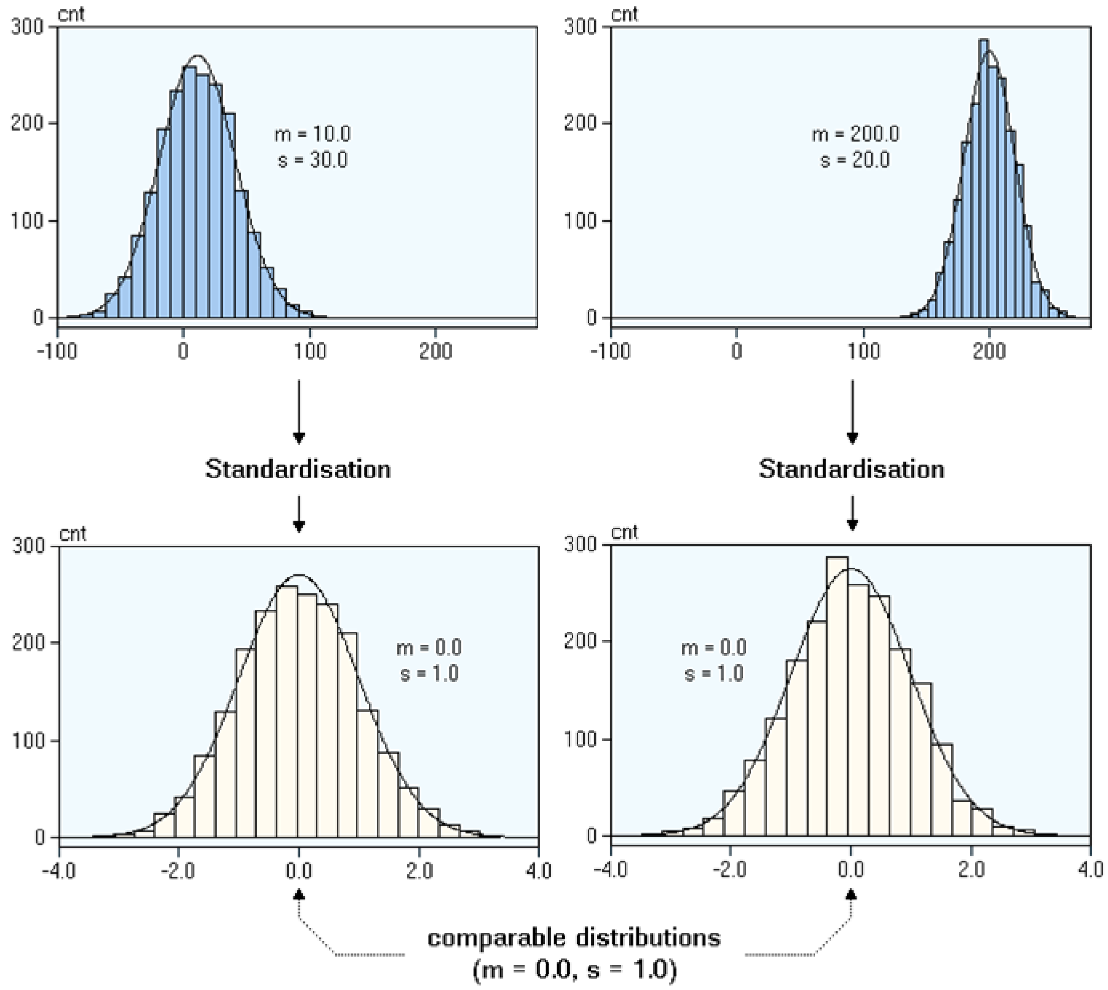
Imputation Techniques with Diabetes Data
Full data
Zero imputation
Mean Imputation
KNN Imputation
Iterative Imputation
3100 3200 3300 3400 3500 3600 3700 3800
MSE
Imputation Techniques with California Data
0.0 0.1 0.2 0.3 0.4 0.5
MSE


California Housing Regression with Different Imputation Methods
IterativeImputer w/ KNeighborsRegressor
IterativeImputer w/ ExtraTreesRegressor
IterativeImputer w/ DecisionTreeRegressor
IterativeImputer w/ BayesianRidge
SimpleImputer w/ median
SimpleImputer w/ mean
Original w/ Full Data
0.0
0.2
0.4
0.6
0.8
1.0
MSE (smaller is better)

- https://scikit-learn.org/stable/auto_examples/impute/plot_iterative_imputer_variants_comparison.html#sphx-glr-auto-examples-impute-plot-iterative-imputer-variants-comparison-py
- https://scikit-learn.org/stable/auto_examples/impute/plot_missing_values.html#sphx-glr-auto-examples-impute-plot-missing-values-py

Deal with features in different scales

- Standardization is a statistic approach to transform your data so that they'll have the properties of a Gaussian distribution with mean of 0 and standard deviation of 1. It is in fact normalize your data.
- Scaling or rescaling (normalization) is a algebra approach to transforms your data into a range between 0 and 1 (or -1 to 1). It is in fact standardize your data.
- They both aim to solve the issue of difference scales in multivariate problems. e.g. age and fare in Titanic case.


cnt
m = 10.0
s = 30.0
cnt
m = 200.0
s = 20.0
Standardisation
Standardisation
cnt
m = 0.0
s = 1.0
cnt
m = 0.0
s = 1.0
comparable distributions
(m = 0.0, s = 1.0)

Standardization

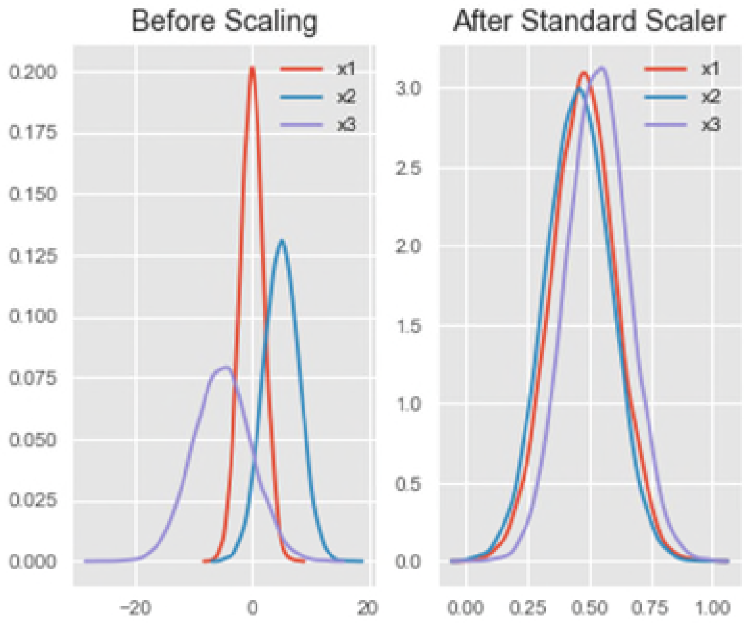
- Step 1: create the transformer StandardScaler

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler() # initialize the standardization transformer.
```

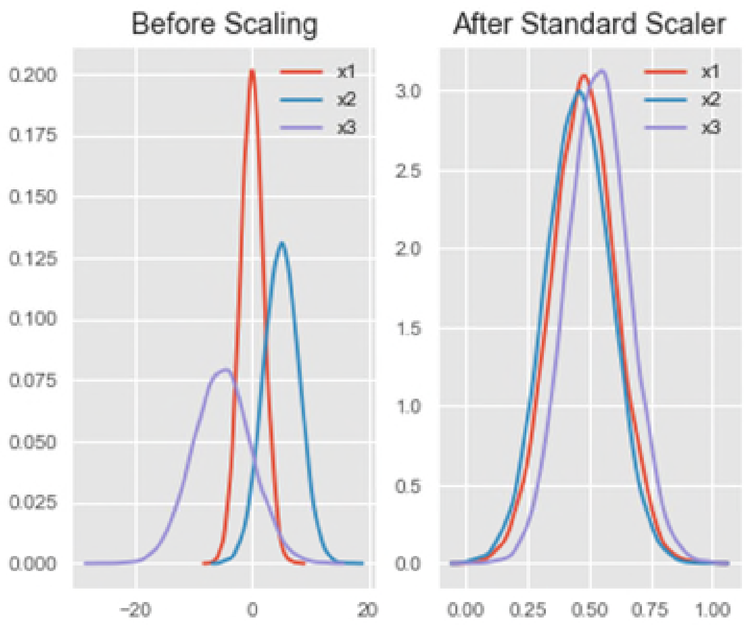
- Step 2: apply the scaler to the data using method fit_transform()

```
df_titan['Age'] = scaler.fit_transform(df_titan[['Age']])
df_titan = scaler.fit_transform(df_titan)# if all numeric
```

- StandardScaler therefore cannot guarantee balanced feature scales in the presence of outliers.



- StandardScaler therefore cannot guarantee balanced feature scales in the presence of outliers.



MinMaxScaler transformation

- Step 1: create the transformer MinMaxScaler

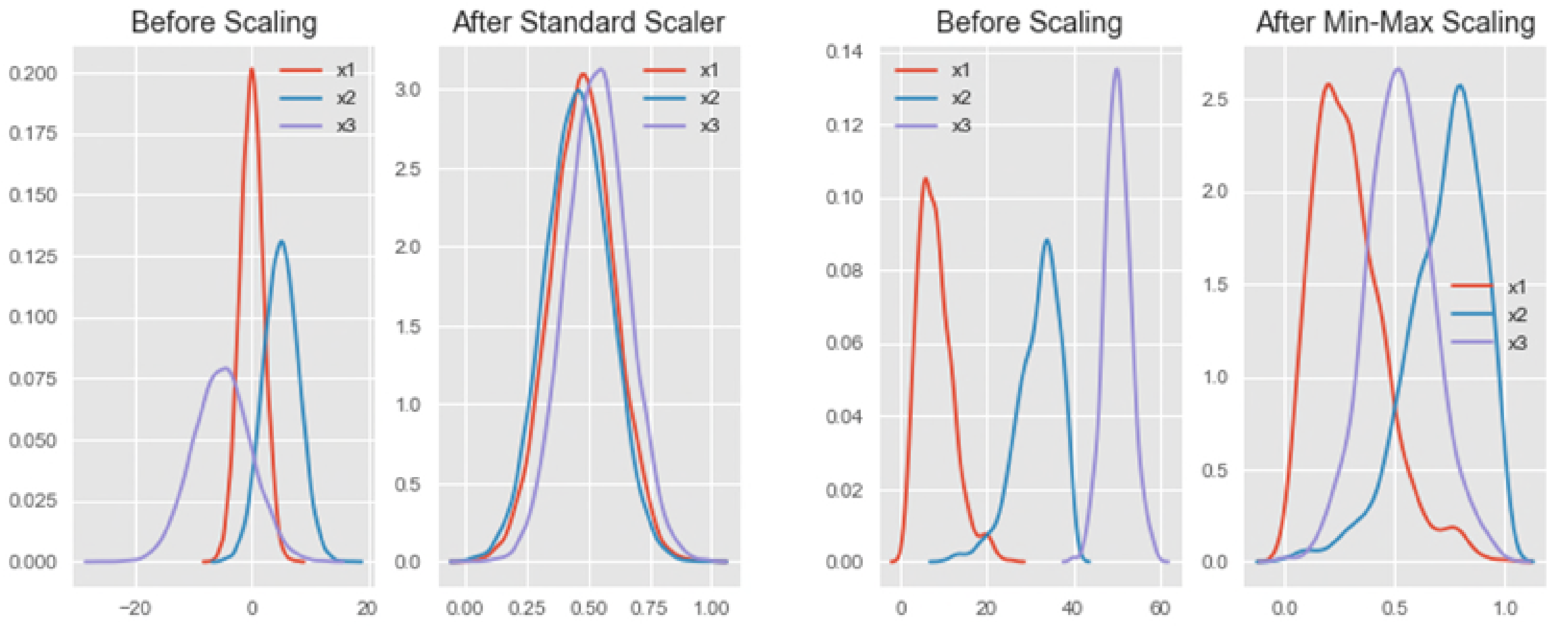
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler() # set the standardization transformer.
```

- Step 2: apply the imputer to the data using method fit_transform()

```
df_titan['Age'] = scaler.fit_transform(df_titan[['Age']])
df_titan = scaler.fit_transform(df_titan)# if all numeric
```

- The same process can be applied with MaxAbsScaler and RobustScaler.

StandardScaler vs MinMaxScaler



https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

Handling categorical features

- Most machine learning algorithms only work well with numeric input. Therefore, we need to convert strings (categorical features) into numbers.
- Convert each value in a column to a number.

	Course	Mark
1	Management	72
2	Accounting	68
3	Finance	78
4	Accounting	65
5	Management	65

	Course	Mark
1	0	72
2	1	68
3	2	78
4	1	65
5	0	65

	Management	Accounting	Finance	Mark
1	1	0	0	72
2	0	1	0	68
3	0	0	1	78
4	0	1	0	65
5	1	0	0	65

Ordinal Encoding (Label Encoding)

- Step 1: create the transformer OrdinalEncoder (LabelEncoder)

```
from sklearn.preprocessing import OrdinalEncoder
enc = OrdinalEncoder () # set the standardization transformer.
```

- Step 2: apply the imputer to the data using method fit_transform()

```
df_titan['Sex'] = enc.fit_transform(df_titan[['Sex']])
df_titan = scaler.fit_transform(df_titan)# if all numeric
```

One-Hot Encoding

- Step 1: create the transformer OneHotEncoder

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder () # set the standardization transformer.
```

- Step 2: apply the imputer to the data using method fit_transform()

```
df_titan['Sex'] = enc.fit_transform(df_titan[['Sex']])
df_titan = enc.fit_transform(df_titan)# cannot handel missing value and returns
a numpy array instead of pandas dataframe.
df_titan = pd.DataFrame(enc.fit_transform(df_titan))
```

get_dummies vs. OneHotEncoder

- Unknown category
- *string
- Pipeline