Missing Values

How to fill the missing values

Statistics based :-

⇒ median

⇒ mean

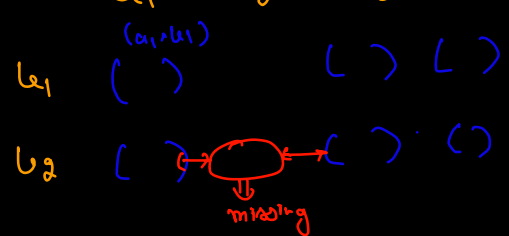
mode [categorical]

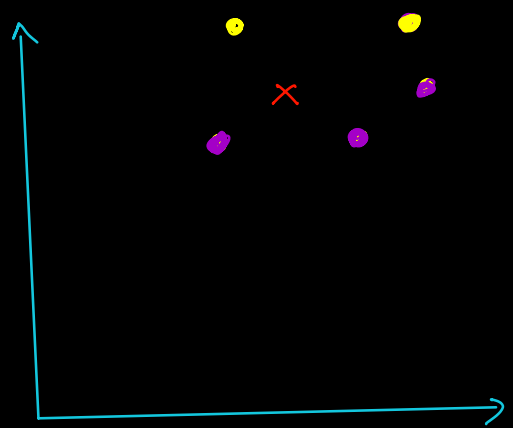
Distance Based :-

⇒ K-nn values

K-nn based Imputation

Euclidean or manhattan





$$\text{Euclidean} = \sqrt[2]{(y_2 - y_1)^2 + (x_2 - x_1)^2}$$

$$\text{Manhattan} = |x_2 - x_1| + |y_2 - y_1|$$

⇒ mean for numerical
mode for categorical

⇒ K should be odd for categorical

Framework

⇒ Create a small subset of data with complete observation.

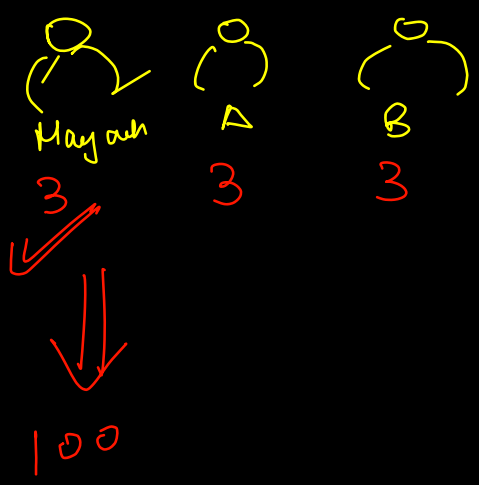
⇒ Delete some values manully

⇒ Use multiple method to fill.

⇒ See where are they failing / error calculation

⇒ Use the best method

	Person	Gender	Company	Price each
1)	350	—	—	0
2)	200	—	—	0
⇒ 3)	NaN	—	—	0
4)	200	—	—	0
⇒ 5)	NaN	—	—	0

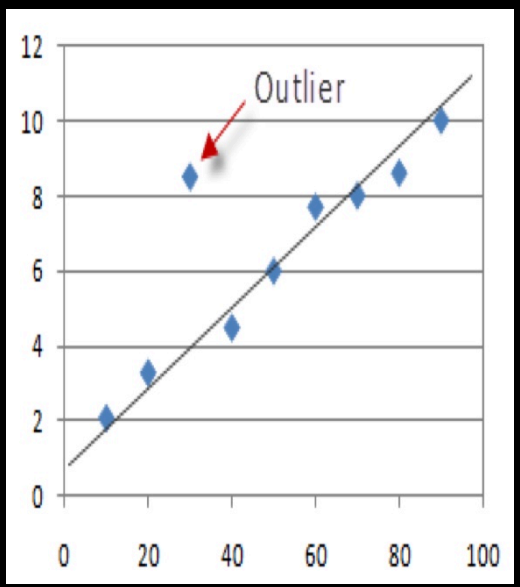


100 lectures

(original value - calculated value)^2

Outlier Analysis

Observation which is inconsistent with rest of dataset.

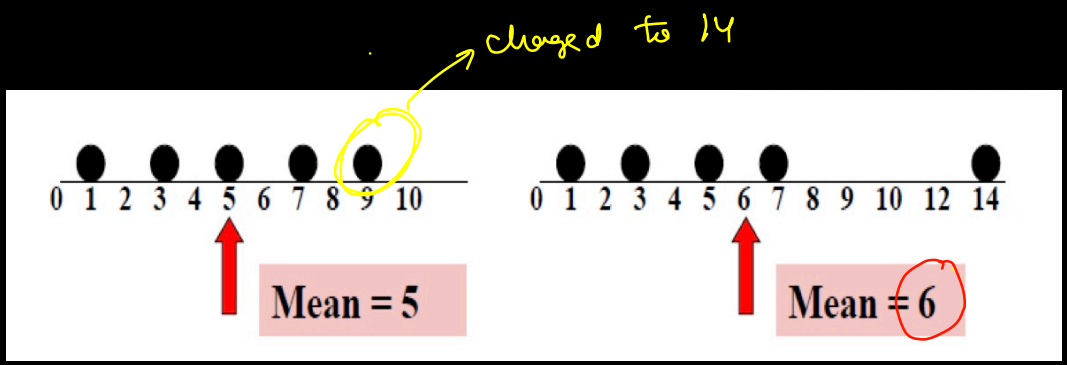


Cause of Outliers

⇒ Poor data quality / Contamination

⇒ Low quality / malfunctioning equipment

⇒ Correct but exceptional data



IQR , Box plot

Upper fence

Lower fence

Applications

⇒ Fraud Detection
⇒ Medical Treatment
⇒ Income / buying habits

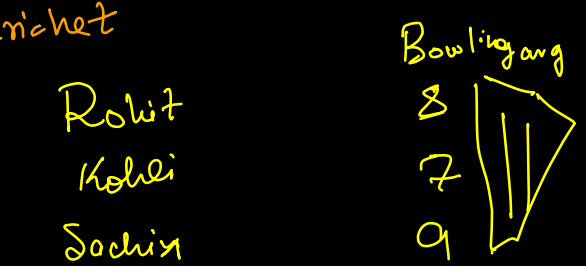
Variable Importance / feature selection

⇒ subset of relevant features to be used in model training / construction

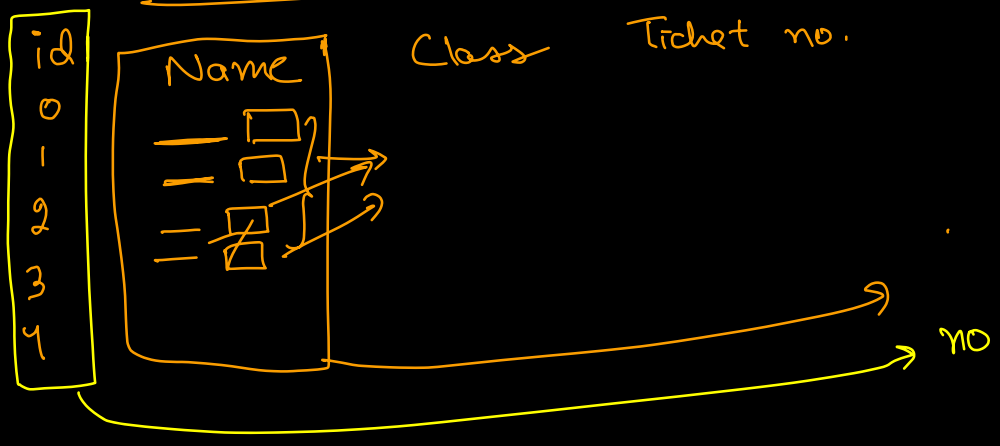
Real life

Cricket

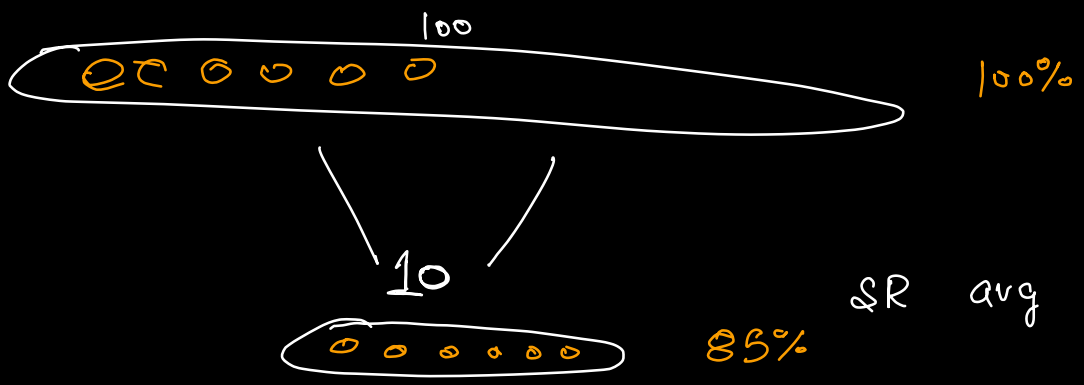
	Bowling avg	minutes played	Seconds played ≤ 60
Rohit	8		
Kohli	7		
Sachin	9		



Titanic Dataset



Dimensionality Reduction



non-linear model

speed

Methods

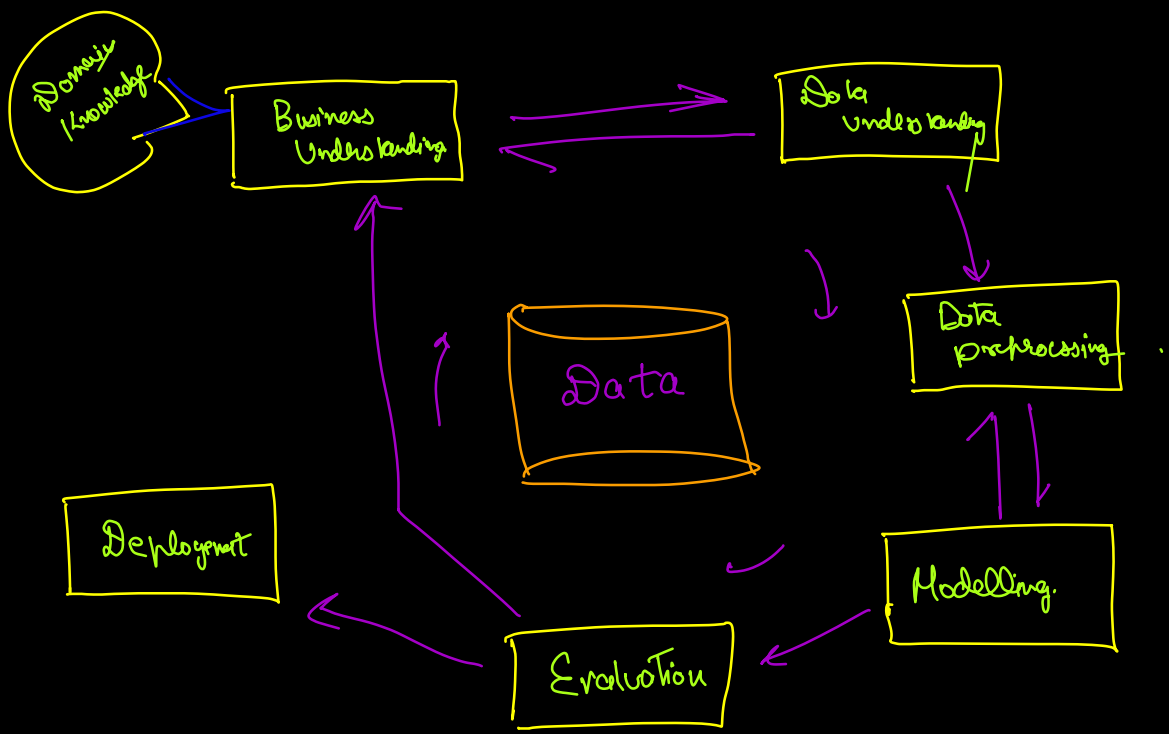
- Filter
- Wrapper
- Embedded

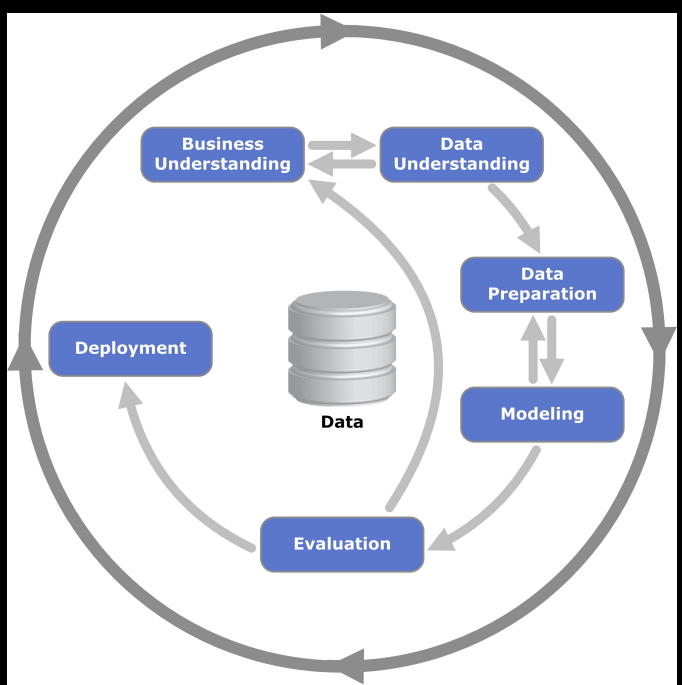
Dimensionality Reduction

- PCA
- RF
- Co-relation
- SVD

Normalization / Feature Scaling

CRISP - DM


Domain Knowledge
Business Understanding
Data Understanding
Data
Data preprocessing
Deployment
Modelling
Evaluation


Business Understanding
Data Understanding
Data Preparation
Deployment
Data
Modeling
Evaluation

Data Science Libraries

Numpy ⇒ numerical python

Fast computations on n-dimensional arrays.

- Based on only one data structure
  nd array

np. Command

array like ⇒ super fast

list in python but they are very slow to process.

50x faster than your traditional python list

Why should we use numpy

list
tuple
dict
set

⟹ heterogenous

- np array are homogenous

Create a numpy array.

np.array(seq)

⟹ creates an array

Scalar ⇒ 10, 20

Vector [10, 20, 30 ...]

Matrix [ ]

Why numpy array faster than list

1. Locality of reference ⇒ Things are stored in numpy array in continuous blocks of memory