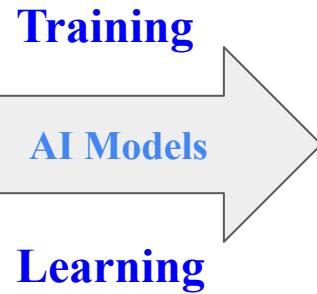
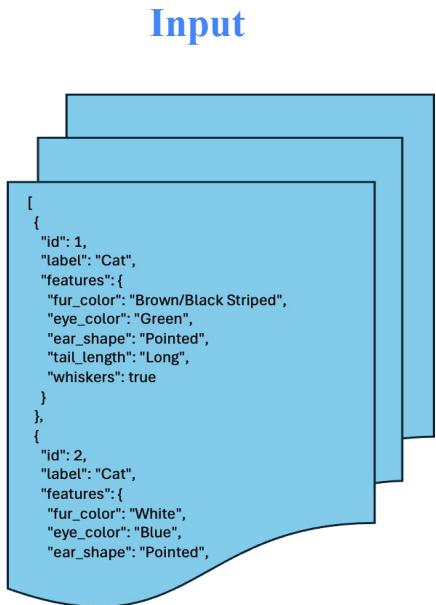


Input: dataset, features

Content

- Machine Learning Workflow
- Data Mining Overview
- Data Preprocessing
- Data Cleaning Techniques
- Data Reduction
- Data Transformation
- Association Rules
- Predictors, Features
- Feature Engineering and Selection
- Handling Imbalanced Data

Machine Learning Workflow



Output



Cats' Features:

- Fur Color
- Eye Color
- Ear Shape
- Tail Length
- Whiskers



Doraemon's Features:

- Blue Body
- Bell Around Neck
- Pocket on Stomach
- Round Face
- Propeller Hat



Lucky Cats' Features:

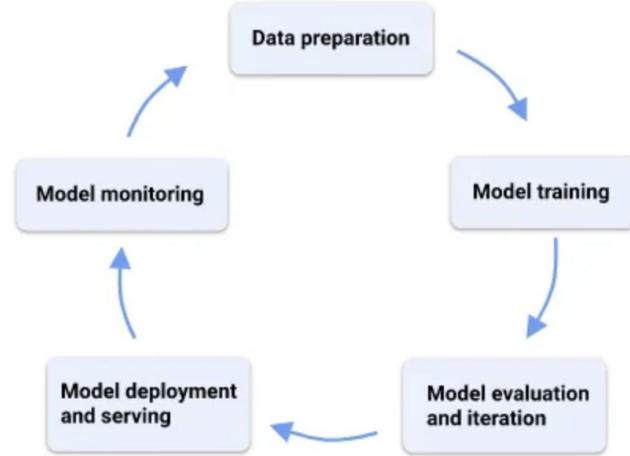
- Raised Paw
- Coin or Fish
- Base or Platform
- Expression
- Color

Machine Learning Workflow

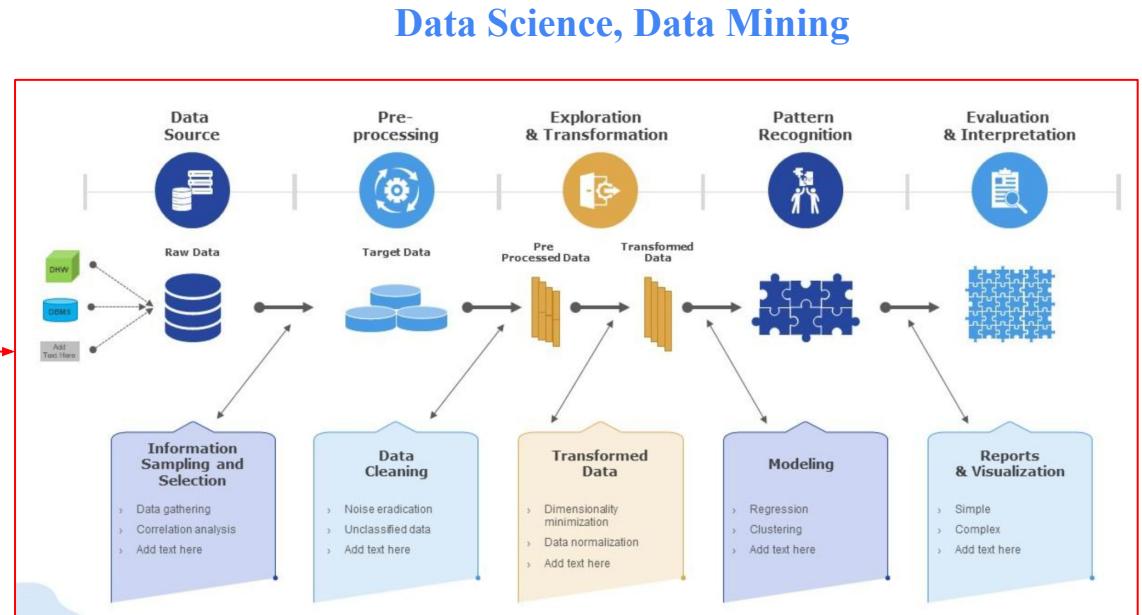
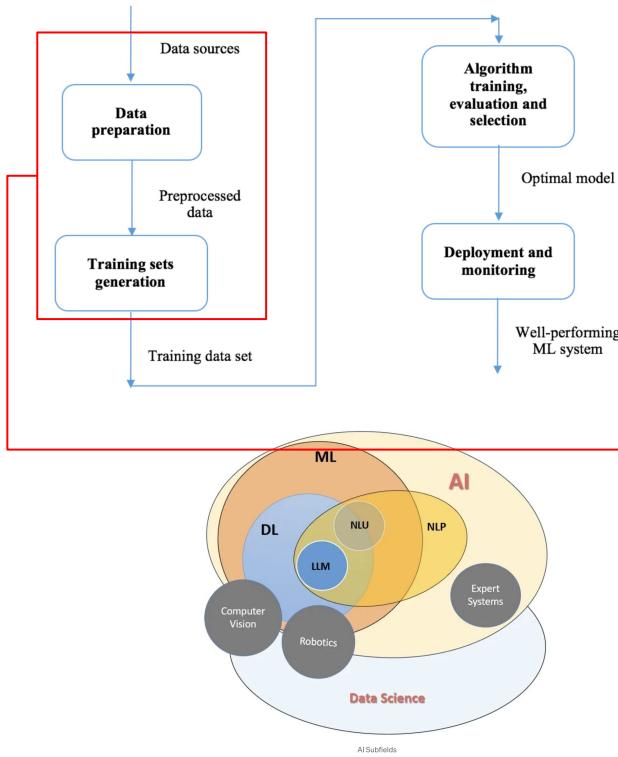
Input → Model → Output

- **Input:** These are the *features or attributes* provided to the model. They can be numerical, categorical, or other types of data.
- **Model:** The model processes the input data using a specific *algorithm or architecture*. It learns patterns and relationships from the input to make predictions.
- **Output:** The model produces an output based on the input. For example, in a *classification* task, the output could be a class label (e.g., spam or not spam), while in *regression*, it could be a numerical value (e.g., predicting house prices).

Machine learning workflow



Data Mining Overview



Data Mining Overview

1. Data Source:

- **Definition:** Data sources refer to the origin of data used for analysis. They can be internal (e.g., company databases, logs) or external (e.g., public datasets, APIs).
- **Importance:** High-quality data is essential for accurate insights. Ensure data relevance, accuracy, and completeness.
- **Examples:** Customer transaction records, sensor data, social media posts.

2. Data Processing:

- **Purpose:** Data processing prepares raw data for analysis.
- **Steps:**
 - *Data Cleaning:* Remove inconsistencies, handle missing values, and correct errors.
 - *Data Transformation:* Normalize, scale, or encode features.
- **Quality Assurance:** Validate data quality to avoid biased results.

3. Exploration and Transformation:

- **Exploratory Data Analysis (EDA):**
 - Visualize data (histograms, scatter plots).
 - Calculate summary statistics (mean, variance).
- **Feature Engineering:**
 - Create new features from existing ones (e.g., age groups from birthdates).
 - Feature selection (choosing relevant features).

Data Mining Overview

4. Pattern Recognition:

- **Machine Learning Models:**
 - Classification (predicting categories).
 - Regression (predicting continuous values).
 - Clustering (grouping similar data points).
- **Model Training and Evaluation:**
 - Train models using historical data.
 - Evaluate performance (accuracy, precision, recall).

5. Evaluation and Interpretation:

- **Metrics:**
 - Accuracy: Correct predictions divided by total predictions.
 - Precision: True positives divided by true positives plus false positives.
 - Recall: True positives divided by true positives plus false negatives.
- **Interpretation:**
 - Understand model decisions (feature importance, coefficients).
 - Iterate based on insights.

Data Mining Overview

Q: why do we need data mining?

- **Knowledge Discovery:**
 - Data mining helps uncover hidden patterns, relationships, and trends within data.
 - By analyzing historical data, we gain new knowledge that informs decision-making.
- **Business Intelligence:**
 - Organizations use data mining to enhance business strategies.
 - It aids in understanding customer behavior, optimizing processes, and identifying growth opportunities.
- **Predictive Modeling:**
 - Data mining builds predictive models based on historical data.
 - These models help forecast future outcomes, such as sales, stock prices, or disease outbreaks.
- **Risk Assessment:**
 - Data mining assesses risks by analyzing historical incidents.
 - For instance, credit scoring models predict creditworthiness based on past behavior.
- **Personalization:**
 - Companies personalize recommendations (e.g., Amazon product suggestions) using data mining.
 - It tailors experiences to individual preferences.
- **Scientific Research:**
 - Researchers analyze data to validate hypotheses and discover new scientific insights.
 - Fields like genomics, climate science, and social sciences benefit from data mining.

Data Preprocessing

- **What is Data Preprocessing?**

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

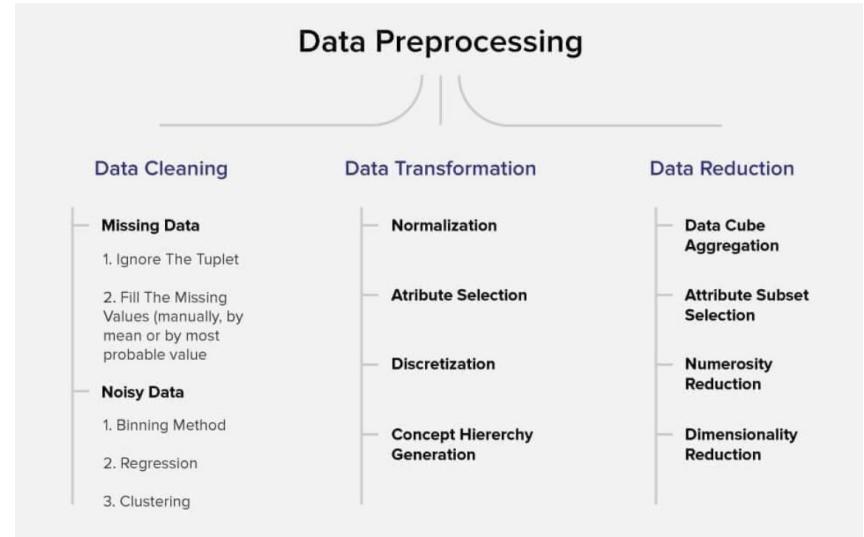
- **Why is Data Preprocessing Important?**

Preprocessing of data is mainly to check the data quality. The quality can be checked by the following:

- Accuracy: To check whether the data entered is correct or not.
- Completeness: To check whether the data is available or not recorded.
- Consistency: To check whether the same data is kept in all the places that do or do not match.
- Timeliness: The data should be updated correctly.
- Believability: The data should be trustable.
- Interpretability: The understandability of the data.

Data Preprocessing

- Major Tasks in Data Preprocessing
 - Data Cleaning
 - Missing Data
 - Ignore The Tuple
 - Fill The Missing Values (Manually, by mean, or by most probable value)
 - Noisy Data
 - Binning Method
 - Regression
 - Clustering
 - Data Transformation
 - Normalization
 - Attribute Selection
 - Concept Hierarchy Generation
 - Data Reduction
 - Data Cube Aggregation
 - Attribute Subset Selection
 - Numerosity Reduction
 - Dimensionality Reduction



Data Cleaning Techniques

Data cleaning is the process of removing incorrect data, incomplete data, and inaccurate data from the datasets, and it also replaces the missing values. Here are some techniques for data cleaning:

Handling Missing Values

- Standard values like “Not Available” or “NA” can be used to replace the missing values.
- Missing values can also be filled manually, but it is not recommended when that dataset is big.
- The attribute’s mean value can be used to replace the missing value when the data is normally distributed wherein in the case of non-normal distribution median value of the attribute can be used.
- While using regression or decision tree algorithms, the missing value can be replaced by the most probable value.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	Braund, Mr. Owen Harris	male	22.0	1	0.0	A/5 21171	7.2500		S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0	1	0.0		71.2833	C85	C
2	3	1		female		0	0.0	STON/O2. 3101282	7.9250		S
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0.0	113803	53.1000	C123	S
4	5	0	Allen, Mr. William Henry	male	35.0	0	0.0	373450	8.0500		S
...	--	--	--	--	--	--	--	--	--	--	--
886	887	0	Montvila, Rev. Juozas	male	27.0	0	0.0	211536	13.0000		S
887	888	1	Graham, Miss. Margaret Edith	female	19.0	0	0.0	112053	30.0000	B42	S
888	889	0	Johnston, Miss. Catherine Helen "Carrie"	female		1	2.0	W/C. 6607	23.4500		S
889	890	1	Behr, Mr. Karl Howell	male	26.0	0	0.0	111369	30.0000	C148	C
890	891	0	Dooley, Mr. Patrick	male	32.0	0	0.0	370376	7.7500		Q

891 rows × 12 columns

Data Cleaning Techniques

- Replacing with previous value – Forward fill

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny



	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	32.0	9.0	Sunny
2	1/5/2017	28.0	9.0	Snow
3	1/6/2017	28.0	7.0	Snow
4	1/7/2017	32.0	7.0	Rain
5	1/8/2017	32.0	7.0	Sunny
6	1/9/2017	32.0	7.0	Sunny
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

Data Cleaning Techniques

- Replacing with next value – Backward fill

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny



	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	28.0	9.0	Sunny
2	1/5/2017	28.0	7.0	Snow
3	1/6/2017	32.0	7.0	Rain
4	1/7/2017	32.0	8.0	Rain
5	1/8/2017	34.0	8.0	Sunny
6	1/9/2017	34.0	8.0	Cloudy
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

Example: Data Cleaning in Python

1. Reading the data

```
import pandas as pd  
  
df = pd.read_csv("train.csv", usecols=['Age', 'Fare', 'Survived'])  
df
```

	Survived	Age	Fare
0	1.0	22.0	7.250
1	NaN	NaN	NaN
2	NaN	26.0	7.925
3	NaN	NaN	53.100
4	0.0	35.0	8.050
...
886	0.0	27.0	13.000
887	1.0	19.0	30.000
888	0.0	NaN	23.450
889	1.0	26.0	30.000
890	0.0	32.0	7.750

891 rows × 3 columns

Example: Data Cleaning in Python

2. Checking if there are missing values

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

Output:

Survived 4

Age 179

Fare 2

dtype: int64

3. Filling missing values with 0

```
new_df = df.fillna(0)
```

```
new_df
```

	Survived	Age	Fare
0	1.0	22.0	7.250
1	0.0	0.0	0.000
2	0.0	26.0	7.925
3	0.0	0.0	53.100
4	0.0	35.0	8.050
...
886	0.0	27.0	13.000
887	1.0	19.0	30.000
888	0.0	0.0	23.450
889	1.0	26.0	30.000
890	0.0	32.0	7.750

891 rows × 3 columns

Example: Data Cleaning in Python

4. Filling NaN values with forward fill value

```
new_df = df.fillna(method="ffill")  
new_df
```

	Survived	Age	Fare
0	1.0	22.0	7.250
1	1.0	22.0	7.250
2	1.0	26.0	7.925
3	1.0	26.0	53.100
4	0.0	35.0	8.050
...
886	0.0	27.0	13.000
887	1.0	19.0	30.000
888	0.0	19.0	23.450
889	1.0	26.0	30.000
890	0.0	32.0	7.750
891 rows × 3 columns			

5. Setting forward fill limit to 1

```
new_df = df.fillna(method="ffill",limit=1)  
new_df
```

	Survived	Age	Fare
0	1.0	22.0	7.250
1	1.0	22.0	7.250
2	NaN	26.0	7.925
3	NaN	26.0	53.100
4	0.0	35.0	8.050
...
886	0.0	27.0	13.000
887	1.0	19.0	30.000
888	0.0	19.0	23.450
889	1.0	26.0	30.000
890	0.0	32.0	7.750

Example: Data Cleaning in Python

6. Filling NaN values in Backward Direction

```
new_df = df.fillna(method="bfill")new_df
```

	Survived	Age	Fare
0	1.0	22.0	7.250
1	0.0	26.0	7.925
2	0.0	26.0	7.925
3	0.0	35.0	53.100
4	0.0	35.0	8.050
...
886	0.0	27.0	13.000
887	1.0	19.0	30.000
888	0.0	26.0	23.450
889	1.0	26.0	30.000
890	0.0	32.0	7.750

7. Interpolate of missing values

```
new_df = df.interpolate()  
df
```

	Survived	Age	Fare
0	1.00	22.0	7.2500
1	0.75	24.0	7.5875
2	0.50	26.0	7.9250
3	0.25	30.5	53.1000
4	0.00	35.0	8.0500
...
886	0.00	27.0	13.0000
887	1.00	19.0	30.0000
888	0.00	22.5	23.4500
889	1.00	26.0	30.0000
890	0.00	32.0	7.7500

891 rows × 3 columns

Example: Data Cleaning in Python

8. Dropna()

```
new_df = df.dropna()
```

```
new_df
```

	Survived	Age	Fare
0	1.0	22.0	7.2500
4	0.0	35.0	8.0500
7	0.0	2.0	21.0750
8	1.0	27.0	11.1333
9	1.0	14.0	30.0708
...
885	0.0	39.0	29.1250
886	0.0	27.0	13.0000
887	1.0	19.0	30.0000
889	1.0	26.0	30.0000
890	0.0	32.0	7.7500
710 rows × 3 columns			

9. Deleting the rows having all NaN values

```
new_df = df.dropna(how='all')
```

```
new_df
```

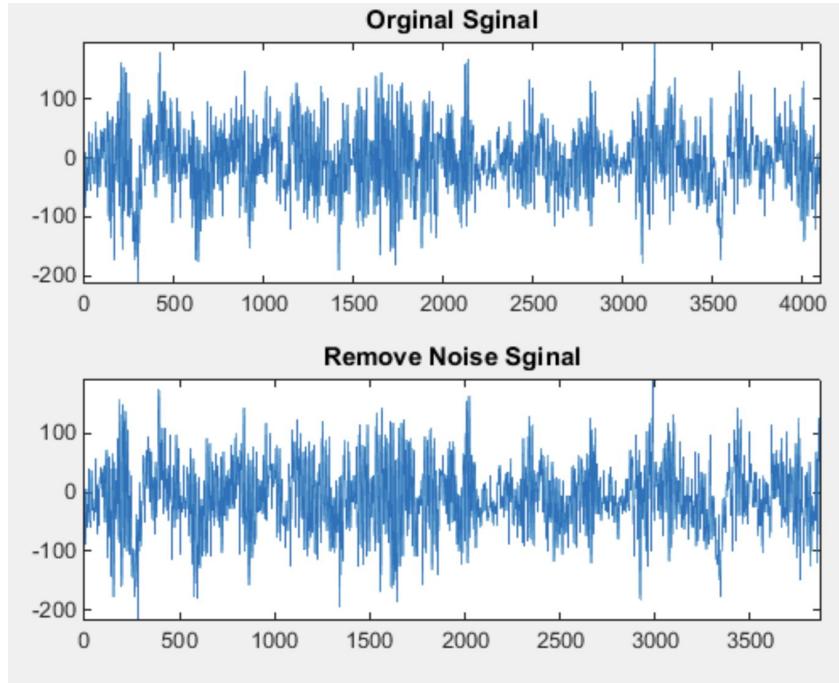
Those rows in which all the values are NaN values will be deleted.

If the row even has one value even then it will not be dropped.

Data Cleaning Techniques

Handling Noisy Data

Noisy generally means random error or containing unnecessary data points. Handling noisy data is one of the most important steps as it leads to the optimization of the model we are using. Here are some of the methods to handle noisy data.



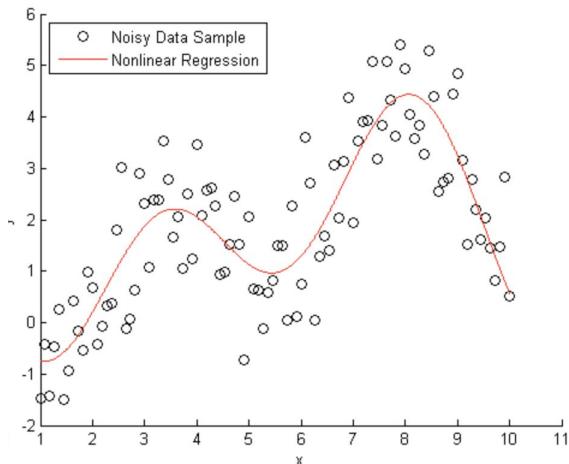
Data Cleaning Techniques

- **Binning:** This method is to smooth or handle noisy data. First, the data is sorted then, and then the sorted values are separated and stored in the form of bins. There are three methods for smoothing data in the bin. Smoothing by bin mean method: In this method, the values in the bin are replaced by the mean value of the bin; Smoothing by bin median: In this method, the values in the bin are replaced by the median value; Smoothing by bin boundary: In this method, the using minimum and maximum values of the bin values are taken, and the closest boundary value replaces the values.

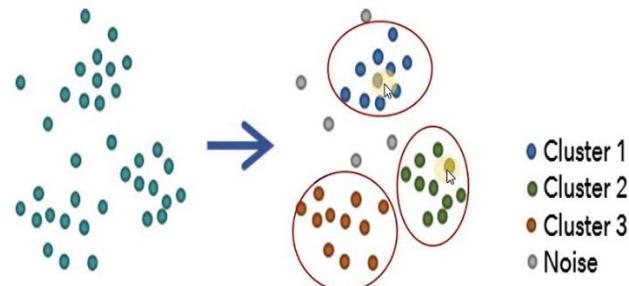


Data Cleaning Techniques

- **Regression:** This is used to smooth the data and will help to handle data when unnecessary data is present. For the analysis, purpose regression helps to decide the variable which is suitable for our analysis.
- **Clustering:** This is used for finding the outliers and also in grouping the data. Clustering is generally used in unsupervised learning.



Regression

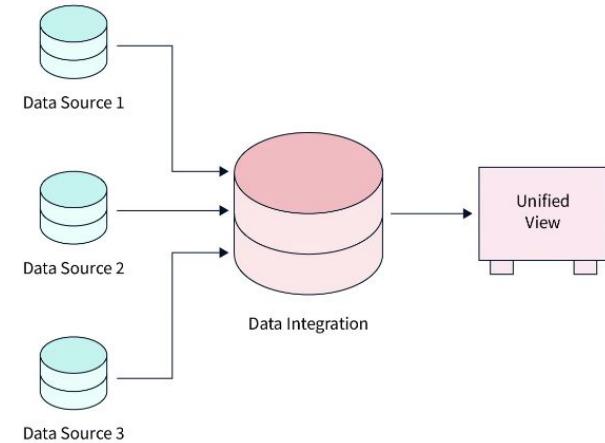


Clustering

Data Integration

The process of combining multiple sources into a single dataset. The Data integration process is one of the main components of data management. There are some problems to be considered during data integration.

- **Schema integration:** Integrates metadata (a set of data that describes other data) from different sources.
- **Entity identification problem:** Identifying entities from multiple databases.
 - For example, the system or the user should know the student *id* of *one database* and *studentname* of another database belonging to the same entity.
- **Detecting and resolving data value concepts:** The data taken from different databases while merging may differ. The attribute values from one database may differ from another database.
 - For example, the date format may differ, like “MM/DD/YYYY” or “DD/MM/YYYY”.



Data Reduction

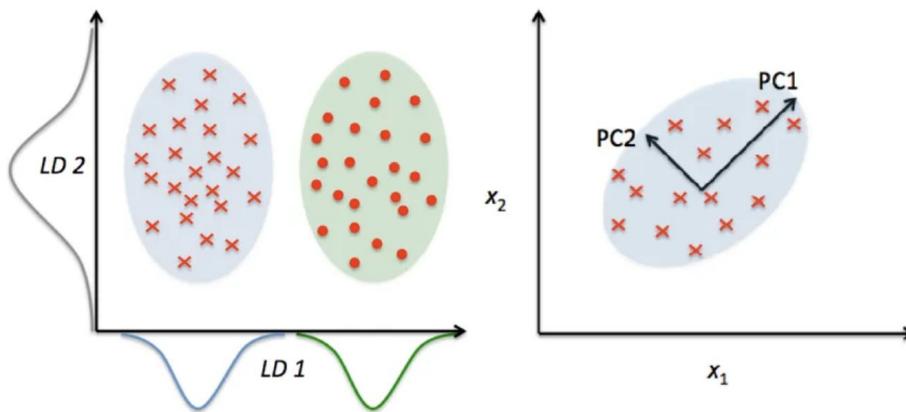
- Dimensionality reduction:

Linear Dimensionality Reduction Methods

1. **PCA** (Principal Component Analysis) : Popularly used for dimensionality reduction in continuous data, PCA rotates and projects data along the direction of increasing variance. The features with the maximum variance are the principal components.
2. **Factor Analysis** : a technique that is used to reduce a large number of variables into fewer numbers of factors. The values of observed data are expressed as functions of a number of possible causes in order to find which are the most important. The observations are assumed to be caused by a linear transformation of lower dimensional latent factors and added Gaussian noise.
3. **LDA** (Linear Discriminant Analysis): projects data in a way that the class separability is maximised. Examples from same class are put closely together by the projection. Examples from different classes are placed far apart by the projection

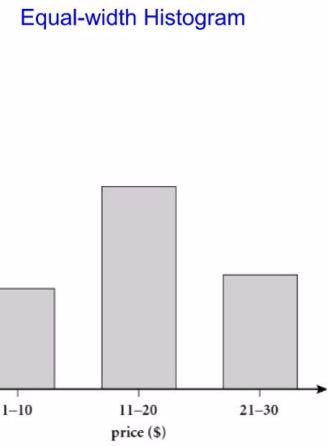
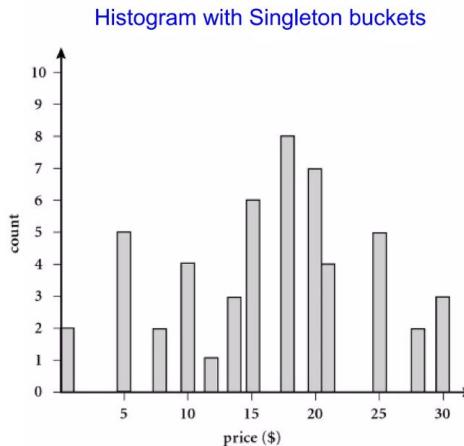
Data Reduction

PCA orients data along the direction of the component with maximum variance whereas LDA projects the data to signify the class separability



Data Reduction

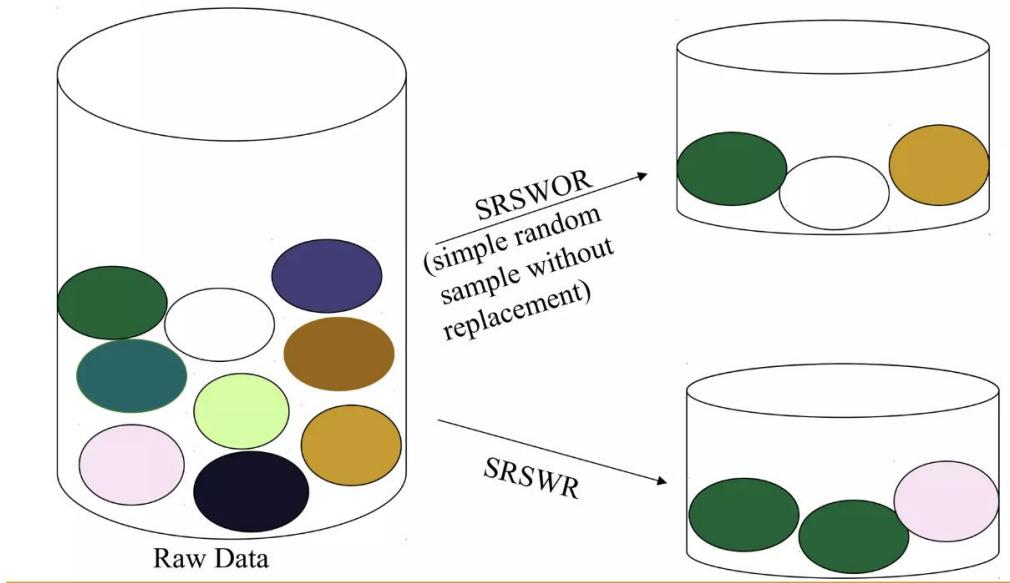
- Numerosity Reduction:
 - Reduce data volume by choosing alternative, smaller forms of data representation.
 - Parametric methods
 - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
 - Example: Log-linear models, Regression
 - Non-parametric methods
 - Do not assume models
 - Major families, histograms, clustering, sampling



List of prices: 1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 10, 12, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 18, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 25, 25, 25, 25, 25, 28, 28, 30, 30,

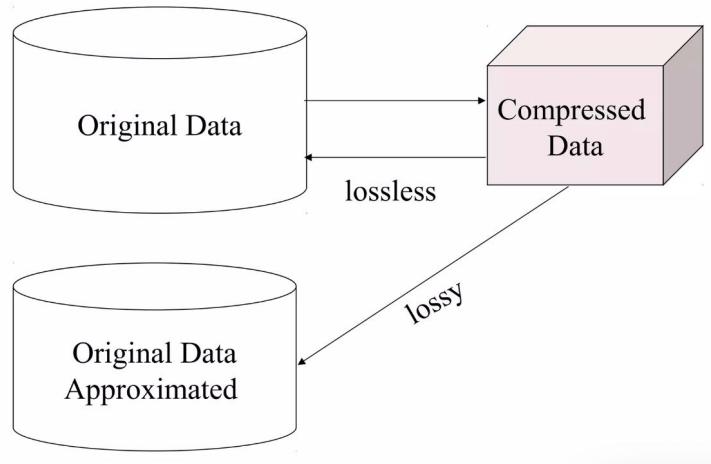
Data Reduction

Sampling: with or without replacement



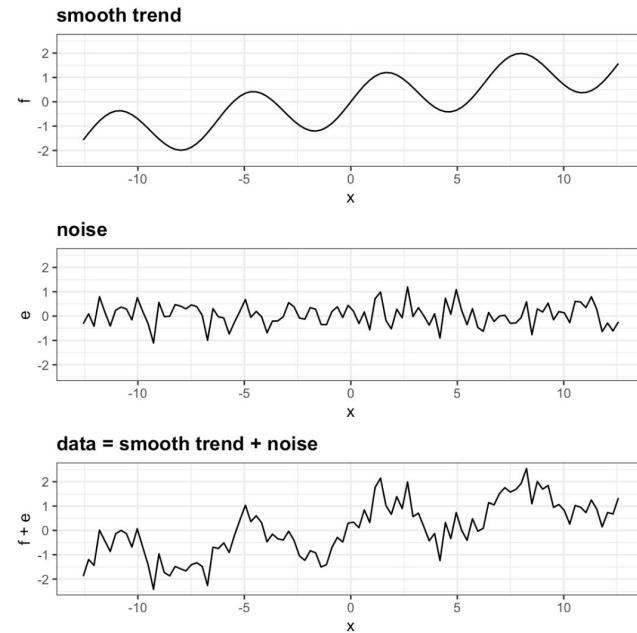
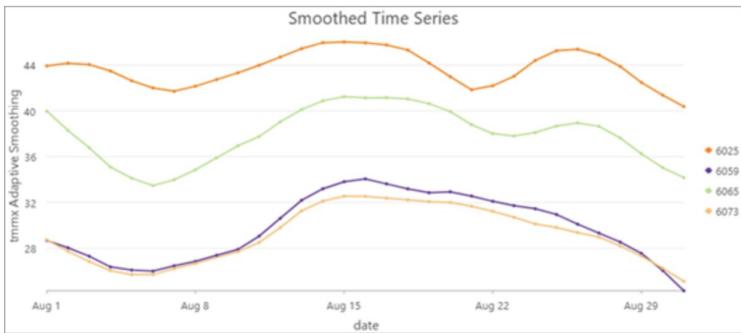
Data Reduction

- **Data compression:**
 - The compressed form of data is called data compression. This compression can be lossless or lossy.
 - When there is no loss of information during compression, it is called lossless compression.
 - Whereas lossy compression reduces information, but it removes only the unnecessary information.
 - Application:
 - String compression
 - Audio/video compression



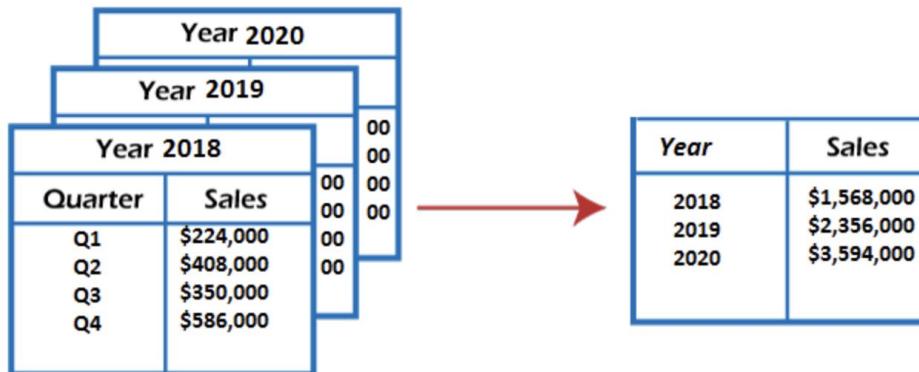
Data Transformation

- **Smoothing:**
 - Remove noise from the dataset, which helps in knowing the important features of the dataset.
 - By smoothing, we can find even a simple change that helps in prediction.
 - **Functions (How)**



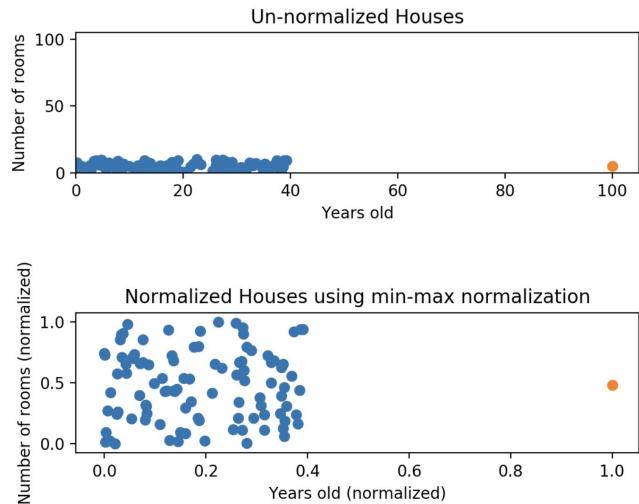
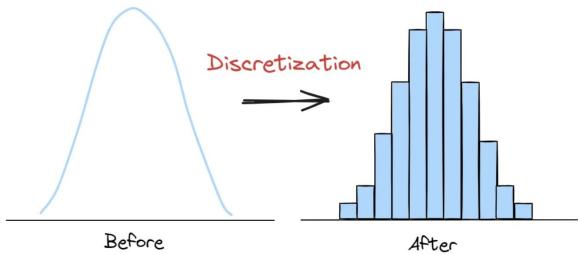
Data Transformation

- **Aggregation:**
 - Data Aggregation is the process of compiling large volumes of data and transforming it into an organized and summarized format that is more consumable and comprehensive.
 - Data Aggregation can enable the capability to forecast future trends and aid in predictive analysis.
 - For example, a company may look at monthly sales data of a product instead of raw sales data to understand its performance better and forecast future sales.



Data Transformation

- **Discretization:**
 - The continuous data here is split into intervals.
Discretization reduces the data size.
 - For example, rather than specifying the class time, we can set an interval like (3 pm-5 pm, or 6 pm-8 pm).
- **Normalization:**
 - It is the method of scaling the data so that it can be represented in a smaller range. For example, ranging from -1.0 to 1.0.



Data Preprocessing Steps in Machine Learning

Step 1: Importing Libraries and the Dataset

Python Code:

```
import pandas as pd
import numpy as np
dataset = pd.read_csv('Data.csv')
print(dataset)
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	0
1	Spain	27.0	48000.0	1
2	Germany	30.0	54000.0	0
3	Spain	38.0	61000.0	0
4	Germany	40.0	NaN	1
5	France	35.0	58000.0	1
6	Spain	NaN	52000.0	0
7	France	48.0	79000.0	1
8	Germany	50.0	83000.0	0
9	France	37.0	67000.0	1

Data Preprocessing Steps in Machine Learning

Step 2: The Independent Variable

```
x= data_set.iloc[:, :-1].values  
x  
  
array([['France', 44.0, 72000.0],  
       ['Spain ', 27.0, 48000.0],  
       ['Germany', 30.0, 54000.0],  
       ['Spain ', 38.0, 61000.0],  
       ['Germany', 40.0, nan],  
       ['France', 35.0, 58000.0],  
       ['Spain ', nan, 52000.0],  
       ['France', 48.0, 79000.0],  
       ['Germany', 50.0, 83000.0],  
       ['France', 37.0, 67000.0]], dtype=object)
```

Data Preprocessing Steps in Machine Learning

Step 3: Extracting the Dependent Variable

```
y= data_set.iloc[:,3].values  
y  
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1], dtype=int64)
```

Step 4: Filling the Dataset with the Mean Value of the Attribute

```
from sklearn.preprocessing import Imputer  
imputer= Imputer(missing_values ='NaN', strategy='mean', axis = 0)  
imputerimputer= imputer.fit(x[:, 1:3])  
x[:, 1:3]= imputer.transform(x[:, 1:3])  
x
```

```
array([[ 'France', 44.0, 72000.0],  
       [ 'Spain ', 27.0, 48000.0],  
       [ 'Germany', 30.0, 54000.0],  
       [ 'Spain ', 38.0, 61000.0],  
       [ 'Germany', 40.0, 63777.77777777778],  
       [ 'France', 35.0, 58000.0],  
       [ 'Spain ', 38.777777777777778, 52000.0],  
       [ 'France', 48.0, 79000.0],  
       [ 'Germany', 50.0, 83000.0],  
       [ 'France', 37.0, 67000.0]], dtype=object)
```

Data Preprocessing Steps in Machine Learning

Step 5: Encoding the Country Variable

The machine learning models use mathematical equations. So categorical data is not accepted, so we convert it into numerical form.

```
from sklearn.preprocessing import LabelEncoder  
label_encoder_x= LabelEncoder()  
x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
```

```
array([[0, 44.0, 72000.0],  
       [2, 27.0, 48000.0],  
       [1, 30.0, 54000.0],  
       [2, 38.0, 61000.0],  
       [1, 40.0, 63777.7777777778],  
       [0, 35.0, 58000.0],  
       [2, 38.77777777777778, 52000.0],  
       [0, 48.0, 79000.0],  
       [1, 50.0, 83000.0],  
       [0, 37.0, 67000.0]], dtype=object)
```

Data Preprocessing Steps in Machine Learning

Step 6: Dummy Encoding

These dummy variables replace the categorical data as 0 and 1 in the absence or the presence of the specific categorical data.

Encoding for Purchased Variable

```
labelencoder_y= LabelEncoder()
y= labelencoder_y.fit_transform(y)
```

```
labelencoder_y= LabelEncoder()
y= labelencoder_y.fit_transform(y)
y
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1], dtype=int64)
```

Data Preprocessing Steps in Machine Learning

Step 7: Splitting the Dataset into Training and Test Sets

```
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=0)
```

Step 8: Feature Scaling

```
from sklearn.preprocessing import StandardScaler
```

```
st_x= StandardScaler()  
x_train= st_x.fit_transform(x_train)
```

```
array([[ 0.13483997,  0.26306757,  0.12381479],  
       [-0.94387981, -0.25350148,  0.46175632],  
       [ 1.21355975, -1.97539832, -1.53893341],  
       [ 1.21355975,  0.05261351, -1.11141978],  
       [-0.94387981,  1.64058505,  1.7202972 ],  
       [ 1.21355975, -0.0813118 , -0.16751412],  
       [-0.94387981,  0.95182631,  0.98614835],  
       [-0.94387981, -0.59788085, -0.48214934]])
```

```
x_test= st_x.transform(x_test)
```

```
array([[1.0e+00, 3.0e+01, 5.4e+04],  
       [1.0e+00, 5.0e+01, 8.3e+04]])
```

Association Rules - Association Rule Discovery

Supermarket shelf management –

Market-basket model:

- **Goal:** Identify items that are bought together by sufficiently many customers
- **Approach:** Process the sales data collected with barcode scanners to find dependencies among items
- **A classic rule:**
 - If someone buys diaper and milk, then he/she is likely to buy beer
 - Don't be surprised if you find six-packs next to diapers!

Association Rules - The Market-Basket Model

- A large set of items
 - e.g., things sold in a supermarket
- A large set of baskets
 - Each basket is a small subset of items
 - e.g., the things one customer buys on one day
- Discover association rules:

People who bought {x,y,z} tend to buy {v,w}

 - Example application: Amazon

Input:

Basket	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Output:

Rules Discovered:

{Milk} --> {Coke}
{Diaper, Milk} --> {Beer}

Association Rules - Frequent Itemsets

- **Simplest question:** Find sets of items that appear together “frequently” in baskets
- **Support** for itemset I : Number of baskets containing all items in I
 - (Often expressed as a fraction of the total number of baskets)
- Given a **support threshold s** , then sets of items that appear in at least s baskets are called **frequent itemsets**

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Support of
 $\{\text{Beer, Bread}\} = 2$

Association Rules - Frequent Itemsets Example

- Items = {milk, coke, pepsi, beer, juice}
- Support threshold = 3 baskets

$$B_1 = \{m, c, b\}$$

$$B_3 = \{m, b\}$$

$$B_5 = \{m, p, b\}$$

$$B_7 = \{c, b, j\}$$

$$B_2 = \{m, p, j\}$$

$$B_4 = \{c, j\}$$

$$B_6 = \{m, c, b, j\}$$

$$B_8 = \{b, c\}$$

- Frequent itemsets: $\{m\}$, $\{c\}$, $\{b\}$, $\{j\}$,
 $\{m, b\}$, $\{b, c\}$, $\{c, j\}$.

Association Rules - Association Rules

- **Define: Association Rules:**

If-then rules about the contents of baskets

- $\{i_1, i_2, \dots, i_k\} \rightarrow j$ means: “if a basket contains all of i_1, \dots, i_k then it is *likely* to contain j ”
- **In practice there are many rules, want to find significant/interesting ones!**
- **Confidence** of association rule is the probability of j given $I = \{i_1, \dots, i_k\}$

$$\text{conf}(I \rightarrow j) = \frac{\text{support}(I \cup j)}{\text{support}(I)}$$
$$\begin{aligned}\text{conf}(I \rightarrow j) &= \\ &= P(j|I) = \frac{P(I, j)}{P(I)}\end{aligned}$$

Association Rules - Association Rules

- Not all high-confidence rules are interesting
 - The rule $X \rightarrow \text{milk}$ may have high confidence for many itemsets X , because milk is just purchased very often (independent of X)
- Interest of an association rule $I \rightarrow j$:
abs. difference between its confidence and the fraction of baskets that contain j

$$\text{Interest}(I \rightarrow j) = |\text{conf}(I \rightarrow j) - P[j]| = |P(j|I) - P(j)|$$

- Interesting rules: those with high interest values (usually above 0.5)
- Why absolute value? Want to capture both *positive* and *negative* associations between itemsets and items

Example: Confidence and Interest

$$B_1 = \{m, c, b\}$$

$$B_2 = \{m, p, j\}$$

$$B_3 = \{m, b\}$$

$$B_4 = \{c, j\}$$

$$B_5 = \{m, p, b\}$$

$$B_6 = \{m, c, b, j\}$$

$$B_7 = \{c, b, j\}$$

$$B_8 = \{b, c\}$$

- Association rule: $\{m, b\} \rightarrow c$

- Support = 2
- Confidence = $2/4 = 0.5$
- Interest = $|0.5 - 5/8| = 1/8$
 - Item c appears in $5/8$ of the baskets
 - The rule is not very interesting!

Association Rule Mining

- **Problem:** Find all association rules with support $\geq s$ and confidence $\geq c$

- **Note:** Support of an association rule is the support of the entire set of items in the rule (left side + right side)
- **Hard part: Finding the frequent itemsets!**
 - If $\{i_1, i_2, \dots, i_k\} \rightarrow \{j\}$ has high support and confidence, then both $\{i_1, i_2, \dots, i_k\}$ and $\{i_1, i_2, \dots, i_k, j\}$ will be “frequent”

$$\text{conf}(I \rightarrow j) = \frac{\text{support}(I \cup j)}{\text{support}(I)}$$

Mining Association Rules

$$\text{conf}(I \rightarrow j) = \frac{\text{support}(I \cup j)}{\text{support}(I)}$$

- **Step 1:** Find all frequent itemsets I
 - (we will explain this next)
- **Step 2: Rule generation**
 - For every subset A of I , generate a rule $A \rightarrow I \setminus A$
 - Since I is frequent, A is also frequent
 - **Variant 1:** Single pass to compute the rule confidence
 - $\text{confidence}(A, B \rightarrow C, D) = \text{support}(A, B, C, D) / \text{support}(A, B)$
 - **Variant 2:**
 - **Observation:** If $A, B, C \rightarrow D$ is below confidence, then so is $A, B \rightarrow C, D$
 - Can generate “bigger” rules from smaller ones!
 - **Output the rules above the confidence threshold**

Example: Mining Association Rules

$$B_1 = \{m, c, b\} \quad B_2 = \{m, p, j\}$$

$$B_3 = \{m, c, b, n\} \quad B_4 = \{c, j\}$$

$$B_5 = \{m, p, b\} \quad B_6 = \{m, c, b, j\}$$

$$B_7 = \{c, b, j\} \quad B_8 = \{b, c\}$$

■ Support threshold $s = 3$, confidence $c = 0.75$

■ Step 1) Find frequent itemsets:

- $\{b, m\}$ $\{b, c\}$ $\{c, m\}$ $\{c, j\}$ $\{m, c, b\}$

■ Step 2) Generate rules:

- ~~$b \rightarrow m: c=4/6$~~ $b \rightarrow c: c=5/6$ ~~$b, c \rightarrow m: c=3/5$~~

- $m \rightarrow b: c=4/5$... $b, m \rightarrow c: c=3/4$

~~$b \rightarrow c, m: c=3/6$~~

Compacting the Output

- To reduce the number of rules, we can post-process them and only output:
 - **Maximal frequent itemsets:**
No immediate superset is frequent
 - Gives more pruning
- or
- **Closed itemsets:**
No immediate superset has the same support (> 0)
 - Stores not only frequent information, but exact supports/counts

Example: Maximal/Closed

Support Maximal(s=3) Closed			
A	4	No	No
B	5	No	Yes
C	3	No	No
AB	4	Yes	Yes
AC	2	No	No
BC	3	Yes	Yes
ABC	2	No	Yes

Frequent, but superset BC also frequent.

Frequent, and its only superset, ABC, not freq.

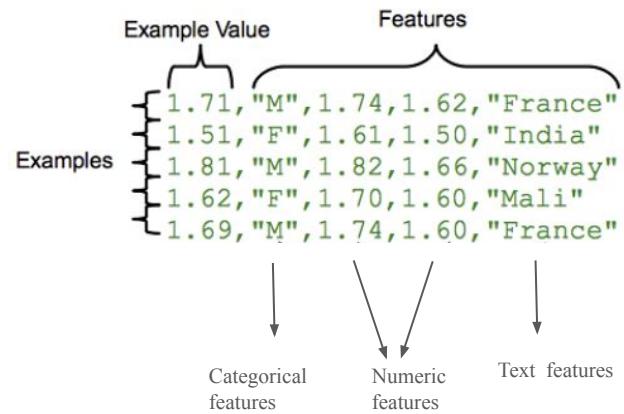
Superset BC has same support.

Its only superset, ABC, has smaller support.

Predictors, Features

Predictors, also known as **features** or independent variables, are essential components in machine learning models.

- Machine learning algorithms process input data, which is structured in rows (instances) and columns (attributes known as features).
- These features represent the input data that the model uses to make predictions.
- Examples of predictors include:
 - Numeric features (e.g., age, temperature, salary)
 - Categorical features (e.g., gender, product category, country)
 - Text features (e.g., product descriptions, customer reviews)
- Features are crucial attributes that impact or are useful for solving a problem. The importance of features outweighs everything else in determining the result, and no algorithm alone can replace the benefits of proper feature engineering.
- Properly selecting and preprocessing predictors significantly impact model performance.



Predictors, Features - Example

Can you determine the Numeric, Categorical, and Text features?

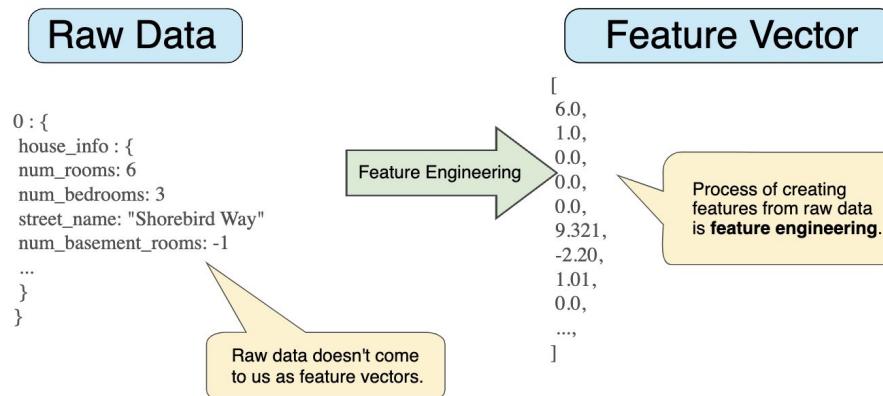
- Dataset: [World Happiness Report 2021](#)
- Examples of predictors include:
 - Numeric features
 - Categorical features
 - Text features

	Country name	Regional indicator	Ladder score	Standard error of ladder score	upperwhisker	lowerwhisker	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
0	Finland	Western Europe	7.842	0.032	7.904	7.780	10.775	0.954	72.0	0.949
1	Denmark	Western Europe	7.620	0.035	7.687	7.552	10.933	0.954	72.7	0.946
2	Switzerland	Western Europe	7.571	0.036	7.643	7.500	11.117	0.942	74.4	0.919
3	Iceland	Western Europe	7.554	0.059	7.670	7.438	10.878	0.983	73.0	0.955
4	Netherlands	Western Europe	7.464	0.027	7.518	7.410	10.932	0.942	72.4	0.913

Feature Engineering and Selection

- **What is feature engineering?**

- Feature engineering is the pre-processing step of machine learning, which extracts features from raw data.
- It helps to represent an underlying problem to predictive models in a better way, which as a result, improve the accuracy of the model for unseen data.
- The predictive model contains predictor variables and an outcome variable, and while the feature engineering process selects the most useful predictor variables for the model.



Feature Engineering and Selection

- Why use feature engineering?

- Better features mean flexibility

In machine learning, we always try to choose the optimal model to get good results. However, sometimes after choosing the wrong model, still, we can get better predictions, and this is because of better features. The flexibility in features will enable you to select the less complex models. Because less complex models are faster to run, easier to understand and maintain, which is always desirable.

- Better features mean simpler models

If we input the well-engineered features to our model, then even after selecting the wrong parameters (not much optimal), we can have good outcomes. After feature engineering, it is not necessary to do hard for picking the right model with the most optimized parameters. If we have good features, we can better represent the complete data and use it to best characterize the given problem.

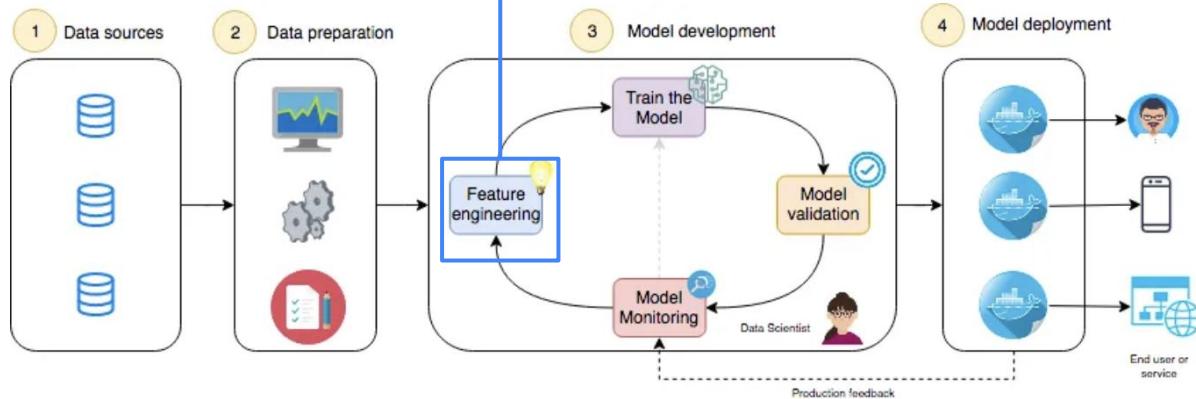
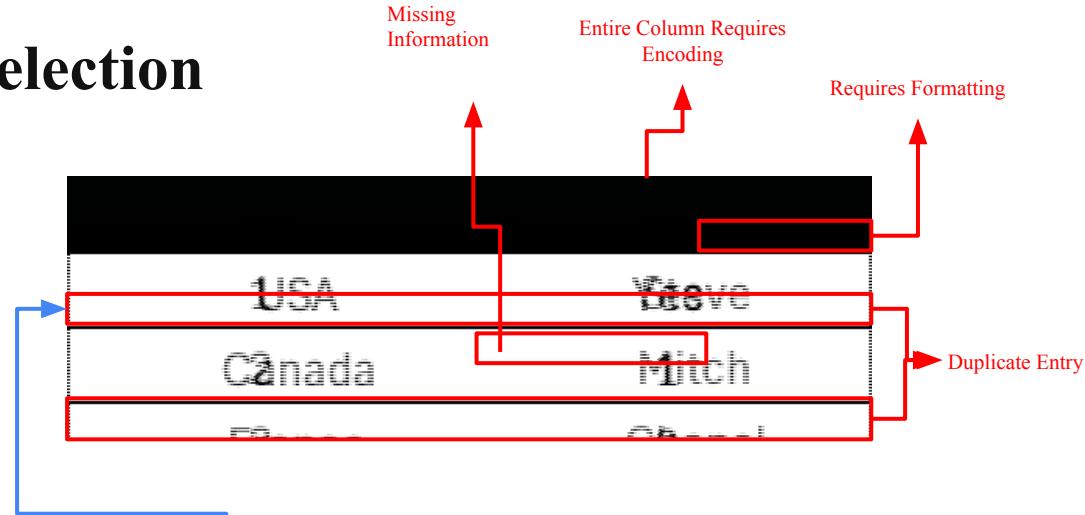
- Better features mean better results

As already discussed, in machine learning, as data we will provide will get the same output. So, to obtain better results, we must need to use better features.

Feature Engineering and Selection

- **5 Steps to Feature Engineering**

1. Data Cleansing
2. Data Transformation
3. Feature Extraction
4. Feature Selection
5. Feature Iteration



Feature Engineering and Selection

- **5 Steps to Feature Engineering**

1. **Data Cleansing**

Data cleansing is the process of dealing with errors or inconsistencies in the data. This step involves identifying incorrect data, missing data, duplicated data, and irrelevant data. Moreover, Data cleansing is the process of deleting, replacing, or modifying data to remove outliers and incorrect values.

Before Data Cleansing					After Data Cleansing				
Car Make and Model	Value USD	Passenger Capacity	Passenger Doors	Fuel Economy	Car Make and Model	Value USD	Passenger Capacity	Passenger Doors	Fuel Economy
Acura RDX	43600	5	4	N/A	Acura RDX	43600	5	4	0
Audi A5	51200	4	2	27	Audi A5	51200	4	2	27
Audi TTS	51900				Audi TTS	51900			
BMW 2-Series	32850	4	2	N/A	BMW 2-Series	32850	4	2	0
Chevrolet Corvette	55495	2		19	Chevrolet Corvette	55495	2	2	19

Feature Engineering and Selection

- **5 Steps to Feature Engineering**

- 2. **Data Transformation**

Data transformation is the process of transforming the data from one layout to another. Transformation needs to occur in a way that does not change the meaning of the original data. There are several techniques to transform the data depending on the desired outcome:

- Transformation
- Standardization
- Data Encoding

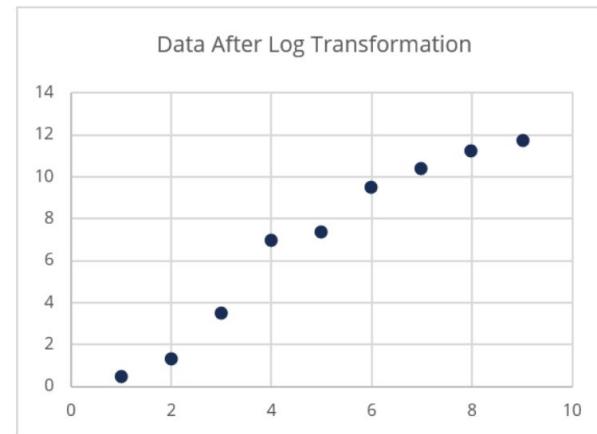
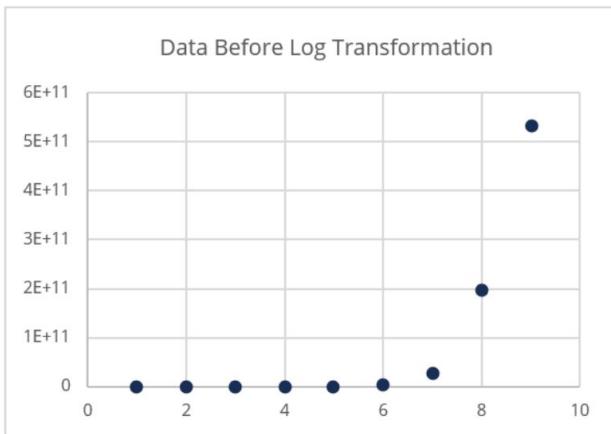
Feature Engineering and Selection

- **5 Steps to Feature Engineering**

2. Data Transformation

- Transformation

Transformation refers to the application of a mathematical function to every data point. Transformation is a great way to handle highly skewed data.



Feature Engineering and Selection

- **5 Steps to Feature Engineering**

2. Data Transformation

- Standardization

Standardization refers to the process of converting the data into a uniform format. Data standardization is a great way of handling data with different units.



Feature Engineering and Selection

- 5 Steps to Feature Engineering

2. Data Transformation

- Data Encoding

Encoding refers to the process of converting categorical variables to numerical variables. Data encoding is a great way of handling nominal and ordinal variables.

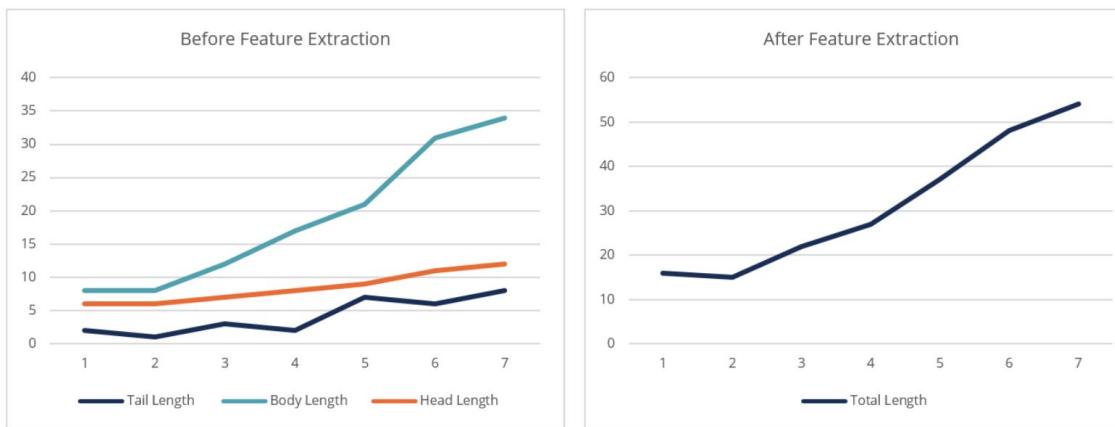
Data Before Encoding	Data After Encoding
Age Group (< 18)	1
Age Group (18-25)	2
Age Group (26-50)	3
Age Group (> 50)	4

Feature Engineering and Selection

- **5 Steps to Feature Engineering**

3. Feature Extraction

Feature extraction is the process of extracting new features from the existing attributes. This process is primarily concerned with reducing the number of features in the model. Feature extraction can be a lengthy process that requires the use of advanced analytics techniques (e.g., Principal Component Analysis).

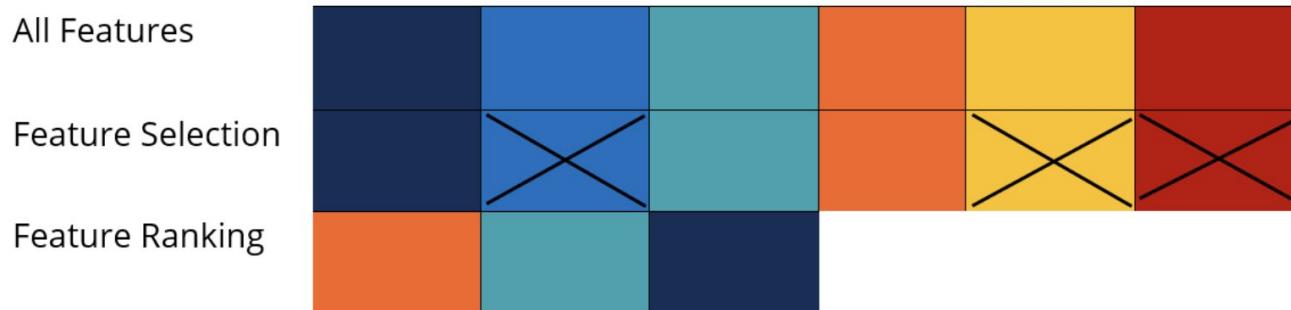


Feature Engineering and Selection

- **5 Steps to Feature Engineering**

4. **Feature Selection**

Feature selection is the process of selecting the correct subset of features to ensure that the most relationship with the target variable is captured. It consists of eliminating features that do not explain the behavior of the target variable.

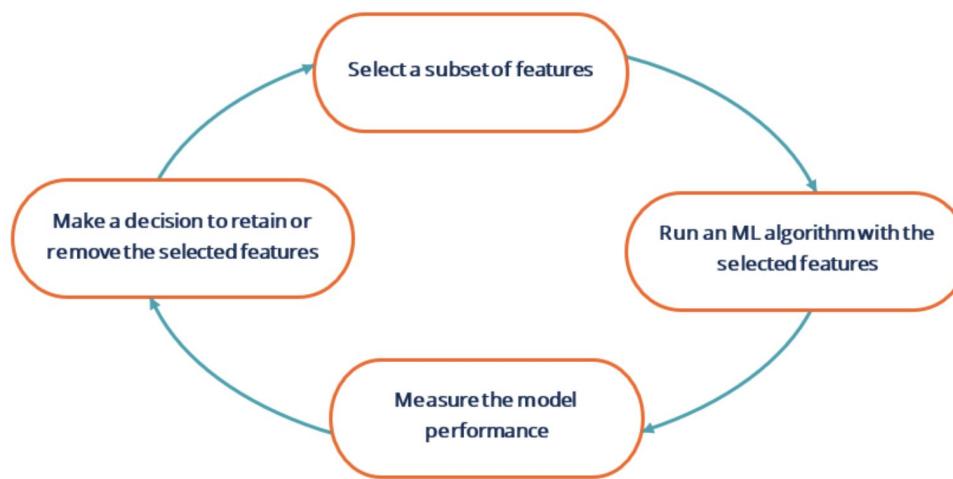


Feature Engineering and Selection

- **5 Steps to Feature Engineering**

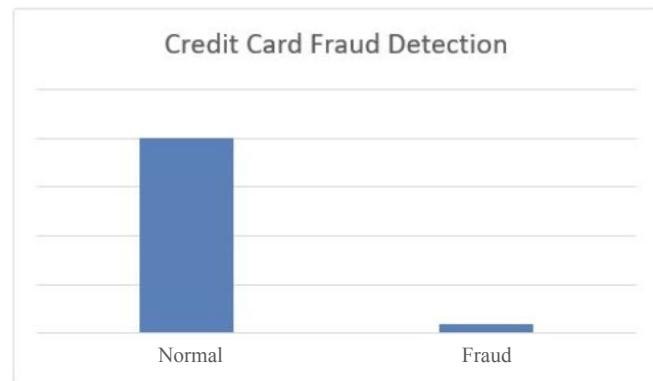
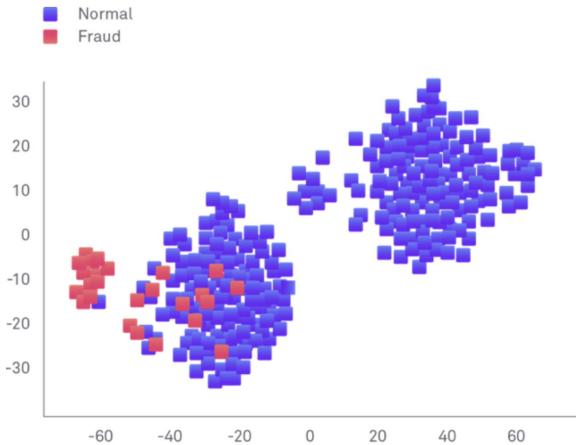
5. Feature Iteration

Feature iteration, also known as the wrapper method of feature selection, is the final step in feature engineering. It is an iterative process involving the four steps below:



Handling Imbalanced Data

Imbalanced data refers to datasets where the target class has an uneven distribution of observations, i.e., one class label has a very high number of observations, and the other has a deficient number of observations.



Handling Imbalanced Data

- **Why is Imbalanced Data a Problem?**

- **Skewed Class Distribution:** Imbalanced dataset occurs when one class (the minority class) is significantly underrepresented compared to another class (the majority class) in a classification problem. This can skew the model's learning process because it may prioritize the majority class, leading to poor performance on the minority class.
- **Biased Model Training:** Machine learning models aim to minimize errors, often measured by metrics like accuracy. In imbalanced datasets, a model can achieve high accuracy by simply predicting the majority class for all instances, ignoring the minority class completely. As a result, the model is biased towards the majority class and fails to capture patterns in the minority class accurately.
- **Poor Generalization:** Imbalanced data can result in models that generalize poorly to new, unseen data, especially for the minority class. Since the model hasn't learned enough about the minority class due to its scarcity in the training data, it may struggle to make accurate predictions for instances belonging to that class in real-world scenarios.
- **Costly Errors:** In many real-world applications, misclassifying instances from the minority class can be more costly or have higher consequences than misclassifying instances from the majority class. Imbalanced data exacerbates this issue because the model tends to make more errors on the minority class, potentially leading to significant negative impacts.
- **Evaluation Metrics Misleading:** Traditional evaluation metrics like accuracy can be misleading in imbalanced datasets. For instance, a model achieving high accuracy may perform poorly on the minority class, which is often the class of interest. Using metrics like precision, recall, F1-score, or area under the ROC curve (AUC-ROC) can provide a more nuanced understanding of the model's performance across different classes.

Handling Imbalanced Data

- How to handle Imbalanced Data?

- Choose Proper Evaluation Metric

- For an imbalanced class dataset, the F1 score is a more appropriate metric.
- Precision recall

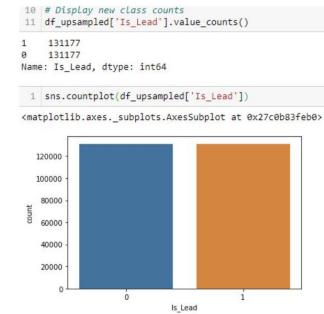
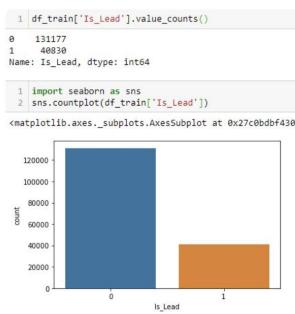
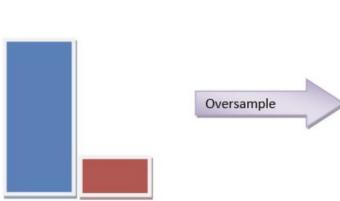
$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

- Resampling

- Oversampling
- Undersampling



```
from sklearn.utils import resample
#create two different dataframes of majority and minority class
df_majority = df_train[(df_train['Is_Lead']==0)]
df_minority = df_train[(df_train['Is_Lead']==1)]
# upsample minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,      # sample with replacement
                                 n_samples= 131177, # to match majority class
                                 random_state=42)  # reproducible results
# Combine majority class with upsampled minority class
df_upsampled = pd.concat([df_minority_upsampled, df_majority])
```



Handling Imbalanced Data

- How to handle Imbalanced Data?

- SMOTE (Synthetic Minority Oversampling Technique)
 - Oversample the minority class
 - In SMOTE new instances are synthesized from the existing data.

```
from imblearn.over_sampling import SMOTE
# Resampling the minority class. The strategy can be changed as required.
sm = SMOTE(sampling_strategy='minority', random_state=42)
# Fit the model to generate the data.
oversampled_X, oversampled_Y = sm.fit_sample(df_train.drop('Is_Lead', axis=1), df_train['Is_Lead'])
oversampled = pd.concat([pd.DataFrame(oversampled_Y), pd.DataFrame(oversampled_X)], axis=1)
```

```
10 oversampled['Is_Lead'].value_counts()
1    131177
0    131177
Name: Is_Lead, dtype: int64
```

Handling Imbalanced Data

- How to handle Imbalanced Data?

- BalancedBaggingClassifier
 - The same as a sklearn classifier but with additional balancing.
 - This classifier takes two special parameters, “sampling_strategy” and “replacement”. The sampling_strategy decides the type of resampling required (e.g., ‘majority’ – resample only the majority class, ‘all’ – resample all classes, etc.), and replacement decides whether it is going to be a sample with replacement or not.

```
from imblearn.ensemble import BalancedBaggingClassifier
from sklearn.tree import DecisionTreeClassifier
#Create an instance
classifier = BalancedBaggingClassifier(base_estimator=DecisionTreeClassifier(),
                                         sampling_strategy='not majority',
                                         replacement=False,
                                         random_state=42)
classifier.fit(X_train, y_train)
preds = classifier.predict(X_test)
```

Handling Imbalanced Data

- How to handle Imbalanced Data?

- Threshold Moving

- Threshold for Classifiers:
 - Classifiers predict probabilities of class membership.
 - Default threshold: 0.5 (above belongs to one class, below to the other).
 - Imbalanced classes may require adjusting the threshold.
 - Optimal Threshold:
 - For imbalanced data, find an optimal threshold.
 - Use ROC Curves and Precision-Recall Curves.
 - Grid search or explore a range of values.

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier()
rf_model.fit(X_train,y_train)
rf_model.predict_proba(X_test) #probability of the class label
Output:

array([[0.97, 0.03],
       [0.94, 0.06],
       [0.78, 0.22],
       ...,
       [0.95, 0.05],
       [0.11, 0.89],
       [0.72, 0.28]])
```

After getting the probability we can check for the optimum value.

```
step_factor = 0.05
threshold_value = 0.2
roc_score=0
predicted_proba = rf_model.predict_proba(X_test) #probability of prediction
while threshold_value <=0.8: #continue to check best threshold upto probability 0.8
    temp_thresh = threshold_value
    predicted = (predicted_proba[:,1] >= temp_thresh).astype('int') #change the class boundary
    print('Threshold',temp_thresh,'--',roc_auc_score(y_test, predicted))
    if roc_score<roc_auc_score(y_test, predicted): #store the threshold for best classification
        roc_score = roc_auc_score(y_test, predicted)
        thrsh_score = threshold_value
    threshold_value = threshold_value + step_factor
print('---Optimum Threshold ---',thrsh_score,'--ROC--',roc_score)
```

Output:

```
Threshold 0.2 -- 0.7778223115708524
Threshold 0.25 -- 0.7851940092858101
Threshold 0.3 -- 0.7857354318529113
Threshold 0.35 -- 0.7770714585514514
Threshold 0.3999999999999999 -- 0.7700307442691748
Threshold 0.4499999999999996 -- 0.7617379569415947
Threshold 0.4999999999999994 -- 0.75203898429419
Threshold 0.5499999999999999 -- 0.7412452995865506
Threshold 0.5 -- 0.7305482030821056
Threshold 0.55 -- 0.7183183540989817
Threshold 0.6 -- 0.7000000000000001 -- 0.6977172374709829
Threshold 0.7500000000000001 -- 0.6742221224415976
---Optimum Threshold --- 0.3 --ROC-- 0.7857354318529113
```