

Mean Squared Error (MSE)

- Fundamental concept in statistics and machine learning playing a crucial role in **assessing the accuracy** of the predictive models.
- It is a parameter to calculate the accuracy of the model.
- It measures the **average squared difference between predicted values and the actual values** in the dataset.
- **Mean squared error (MSE)** is a metric used to measure the average squared difference between the predicted values and the actual values in the dataset.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Loss Functions

- n is the number of observations in the dataset.
- y_i is the actual value of the observation.
- \hat{y}_i is the predicted value of the i^{th} observation.

Interpretation of Mean Squared Error

The Interpreting MSE involves understanding the magnitude of the error and its implications for model's performance.

- A lower MSE indicates that the model's predictions are closer to the actual values signifying better accuracy.
- Conversely, a higher MSE suggests that the model's predictions deviate further from true values indicating the poorer performance.

Significance of Mean Squared Error

The Mean Squared Error is widely used in the various fields including the statistics, machine learning and econometrics due to its several important properties:

- It provides the quantitative measure of the accuracy of the predictive models.
- It penalizes large errors more heavily than small errors making it sensitive to the outliers.
- It is mathematically convenient and easy to the interpret making it a preferred choice for the evaluating model performance.

Applications of Mean Squared Error

The Mean Squared Error is extensively used in the various applications including:

- **Regression analysis:** Assessing the goodness of fit of the regression models.
- **Model evaluation:** Comparing the performance of the different machine learning algorithms.
- **Optimization:** Minimizing MSE during the model training to the improve predictive accuracy.

Advantages and Limitations of Mean Squared Error

The advantages and limitations of mean squared error is mentioned below:

Advantages

- Provides the comprehensive measure of the model accuracy.
- Sensitive to the both large and small errors.
- Easy to the calculate and interpret.

Limitations

- It can be heavily influenced by the outliers.
- It penalizes large errors disproportionately which may not always be desirable.

Loss Functions

Month	Actual	Predicted
January	42	46
February	51	48
March	53	55
April	68	73
May	74	77
June	81	83
July	88	87
August	85	85
September	79	75
October	67	70
November	58	55
December	43	41

$$\text{MSE} = \frac{\sum (y_i - p_i)^2}{n} = \frac{106}{12} = 8.83.$$

Loss Functions

y_i	y^*
49	47.48
63	62.6
58	53.15
60	62.6
58	64.49
61	60.71
60	60.71
63	55.04
60	55.04
52	49.37
62	64.49
30	39.92
59	64.49
49	49.37
68	62.6

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Loss Functions

y_i	y^*	$y_i - y^*$	Square of ($y_i - y^*$)
49	47.48	1.52	2.31
63	62.6	0.4	0.16
58	53.15	4.2	17.64
60	62.6	-2.6	6.76
58	64.49	-6.49	42.12
61	60.71	0.29	0.08
60	60.71	-0.71	0.50
63	55.04	7.96	63.36
60	55.04	4.96	24.60
52	49.37	2.63	6.92
62	64.49	-2.49	6.2
30	39.92	-9.92	98.41
59	64.49	-5.49	30.14
49	49.37	-0.37	0.14
68	62.6	5.4	29.16
		SSE = 328.51 MSE = 328.51/15 = 21.90	

Loss Functions

What is Mean Absolute Error (MAE)?

Mean Absolute Error calculates the average difference between the calculated values and actual values. It is also known as scale-dependent accuracy as it calculates error in observations taken on the same scale used to predict the accuracy of the machine learning model

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Loss Functions

Mean Absolute Error (MAE)

Normalized MAE = MAE / (max value – min value) = 3.69/(9.92-0.29) = 0.38

Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the **RMSE should be more useful when large errors are particularly undesirable**. Both ranges from 0 to infinity.

y_i	y^*	$ y_i - y^* $
49	47.48	1.52
63	62.6	0.4
58	53.15	4.2
60	62.6	2.6
58	64.49	6.49
61	60.71	0.29
60	60.71	0.71
63	55.04	7.96
60	55.04	4.96
52	49.37	2.63
62	64.49	2.49
30	39.92	9.92
59	64.49	5.49
49	49.37	0.37
68	62.6	5.4
		AE = 55.43 MAE = 55.43/15 = 3.69

Huber's Loss Functions

Huber Loss is a loss function used in regression problems. It is robust to outliers and combines the best properties of both Mean Squared Error (MSE) and Mean Absolute Error (MAE). The loss function is defined as:

Key Features:

- Quadratic for small errors ($|a| \leq \delta$): Acts like MSE, giving a smooth gradient and is sensitive to small errors.
- Linear for large errors ($|a| > \delta$): Acts like MAE, reducing the impact of outliers by not letting the error grow quadratically.

$$L_{\delta}(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$

Loss Functions

Categorical cross entropy:

```
import tensorflow as tf
import numpy as np

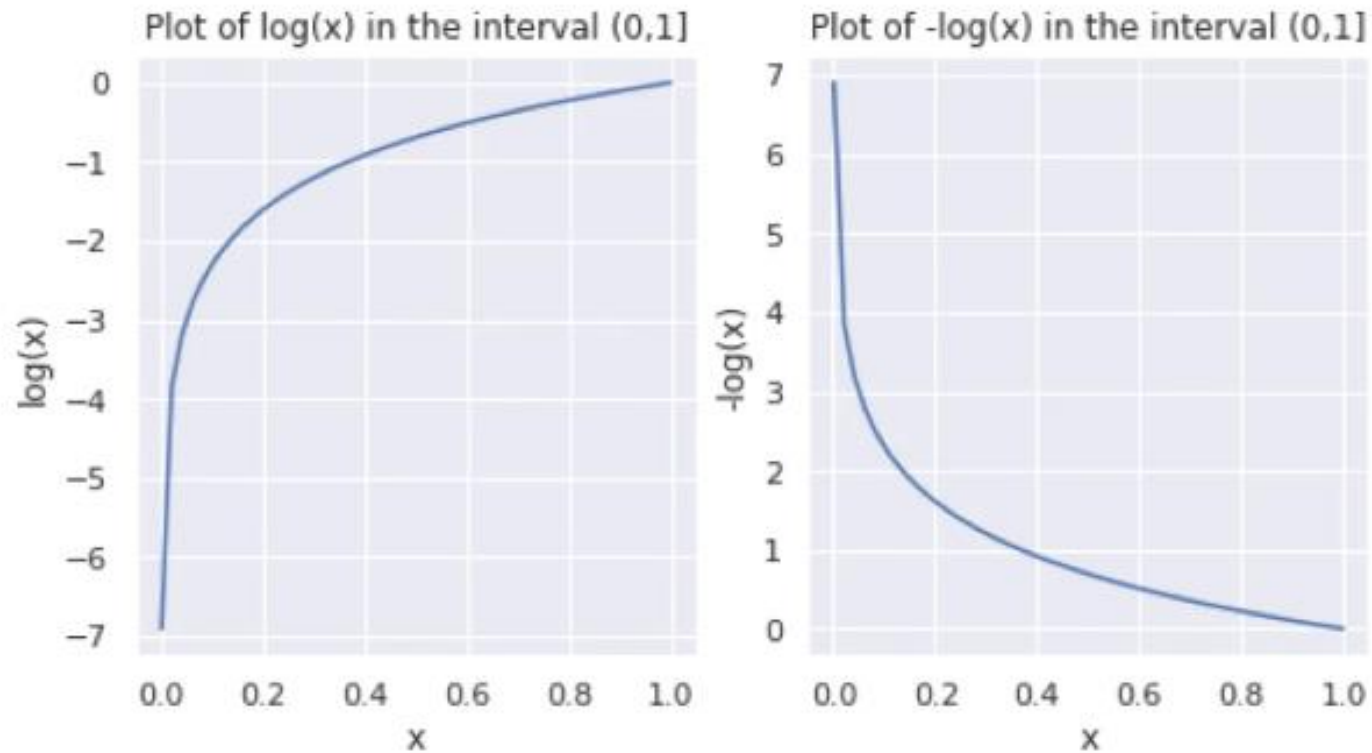
# True labels (one-hot encoded)
y_true = np.array([[0, 1, 0], [1, 0, 0], [0, 0, 1]])

# Predicted probabilities
y_pred = np.array([[0.1, 0.8, 0.1], [0.7, 0.2, 0.1], [0.2, 0.3, 0.5]])

# Categorical Cross-Entropy loss calculation
loss = tf.keras.losses.categorical_crossentropy(y_true, y_pred)

print("Loss:", loss.numpy())
```

Loss Functions



Why One Hot Encoding

Explicit Representation of Class Membership:

- **One-hot encoding** clearly shows which class a data point belongs to. Each class has a unique position in the vector, making the membership explicit.
- This representation is particularly useful in situations where the model or algorithm requires all classes to be represented equally in the input.

Compatibility with Some Algorithms:

- Some machine learning algorithms, like certain neural network architectures, expect input in a particular format, often as vectors. One-hot encoding ensures that the target labels meet these input requirements.
- For example, when computing dot products or performing matrix operations, one-hot encoded vectors work seamlessly.

Why One Hot Encoding

Interpretability:

- One-hot encoded vectors make it easier to interpret which class the model is predicting or working with because each dimension corresponds to a specific class.
- This can be beneficial in debugging or understanding the model's predictions.

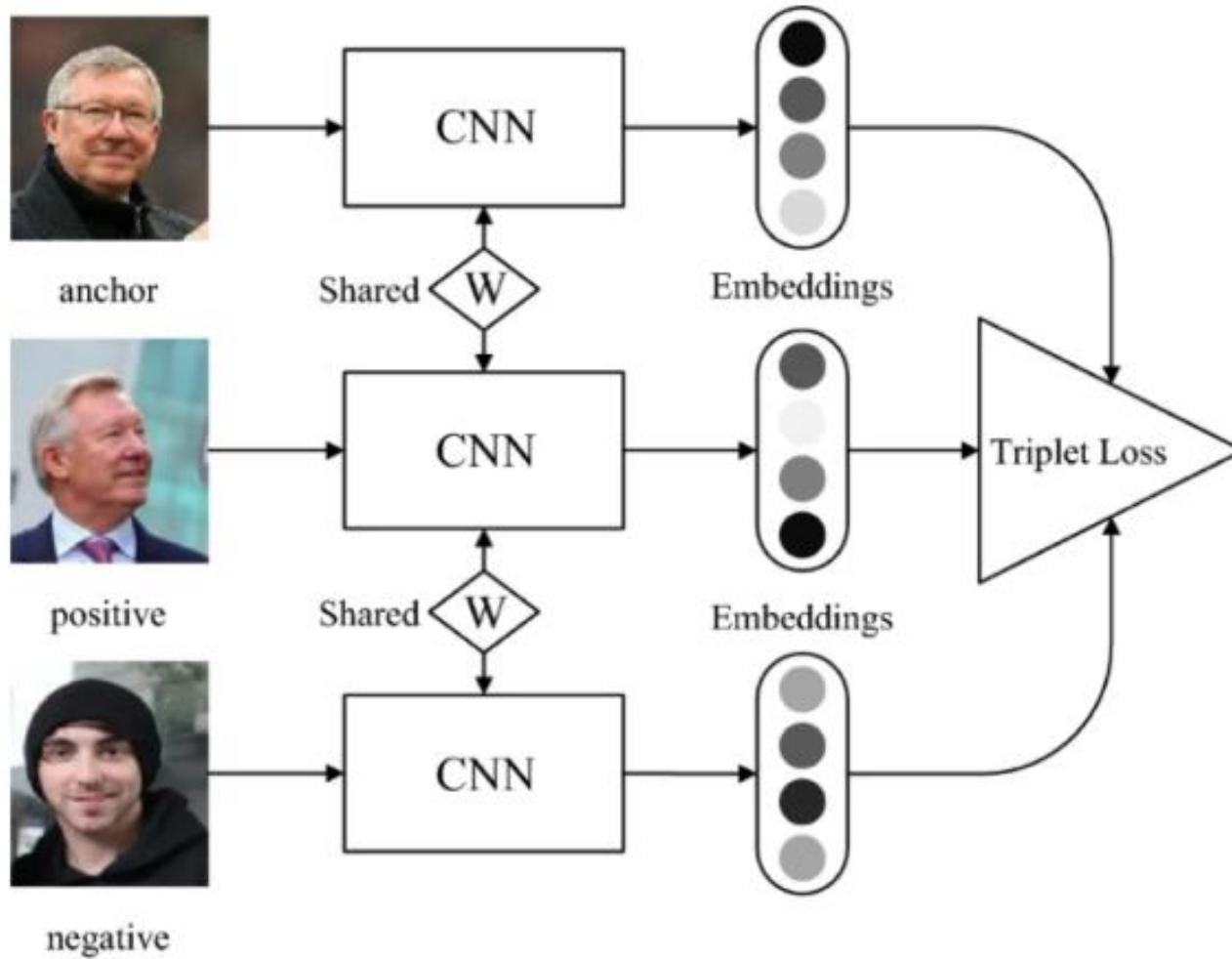
Flexibility in Loss Functions

- While **Sparse Categorical Cross Entropy** simplifies the process by using integer labels, **Categorical Cross Entropy** with one-hot encoded labels provides flexibility for different scenarios, like when modifying or creating custom loss functions.

Triplet Loss

- Triplet loss is a way to teach a machine-learning model how to recognize the similarity or differences between items. It uses groups of three items, called triplets, which consist of an anchor item, a similar item (positive), and a dissimilar item (negative).
- The goal is to make the model understand that the anchor is closer to the positive than the negative item. This helps the model distinguish between similar and dissimilar items more effectively.
- In face recognition, for example, the model compares two unfamiliar faces and determines if they belong to the same person.

Triplet Loss



Triplet Loss

This scenario uses triplet loss to learn embeddings for every face. Faces from the same individual should be close together again and form well-separated clusters in the embedding space.

The objective of triplet loss is to build a representation space where the gap between similar samples is smaller than between different examples. By enforcing the order of distances, triplet loss models are embedded so that samples with identical labels appear nearer than those with other labels.

Hence, the triplet loss architecture helps us learn distributed embedding through the concept of similarity and dissimilarity. The mathematical depiction is shown below:

$$\sum_i^N [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]$$

Triplet Loss

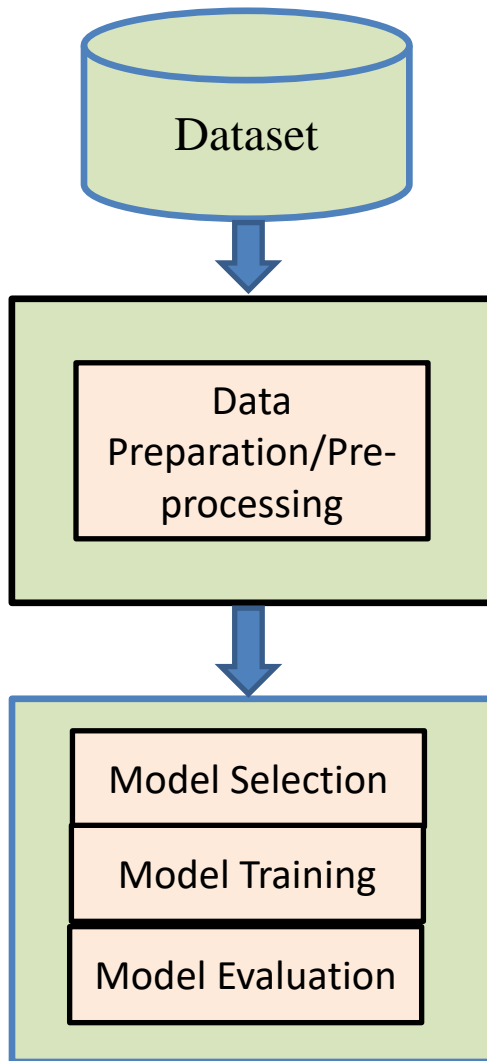
Where

- $f(x)$ accepts an input of x and generates a 128-dimensional vector w
- i represents the i 'th input
- The subscript a denotes an anchor image, p is a positive image, and n is a negative image
- α refers to the bias

The goal is to minimize the above equation by minimizing the first term and maximizing the second term, and bias acts as a threshold.

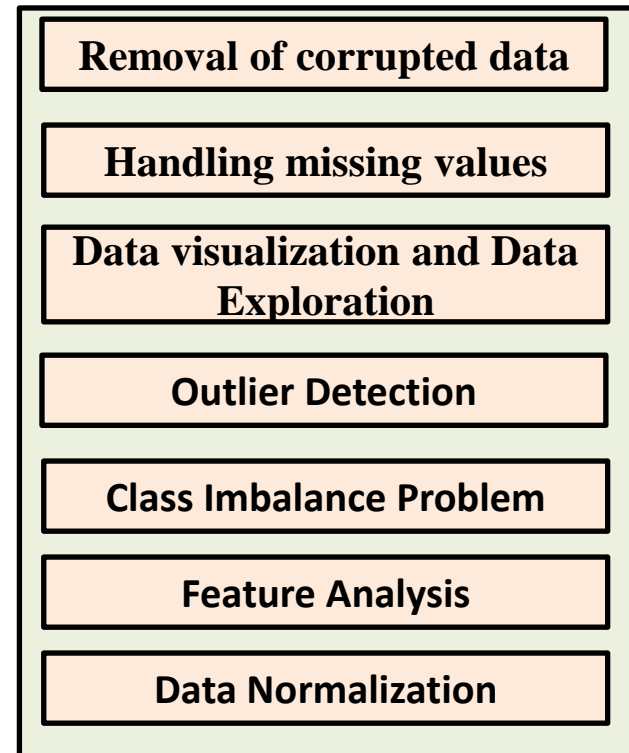
An anchor (with fixed identity) negative is an image that doesn't share the class with the anchor—so, with a greater distance. In contrast, a **positive** is a point closer to the anchor, displaying a similar image. The model attempts to diminish the difference between similar classes while increasing the difference between different classes.

ML Model



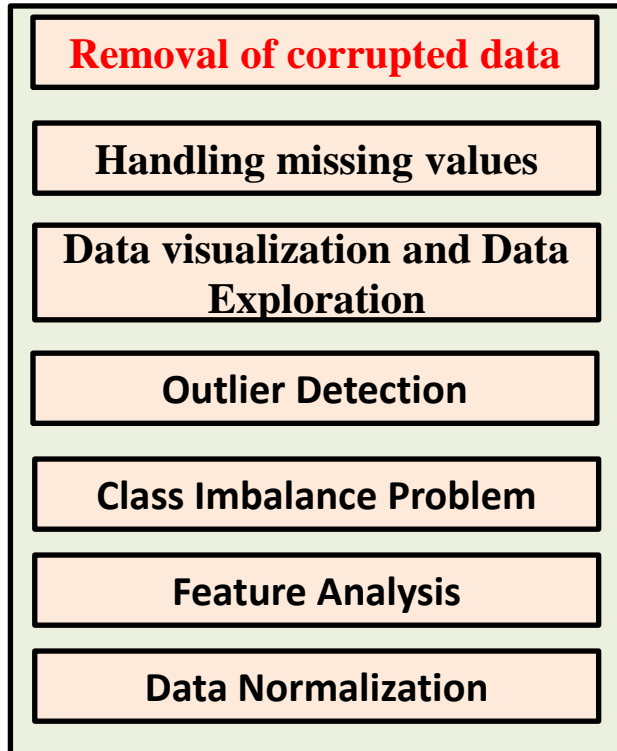
Modelling & Evaluation

Pre-processing/ Data Preparation



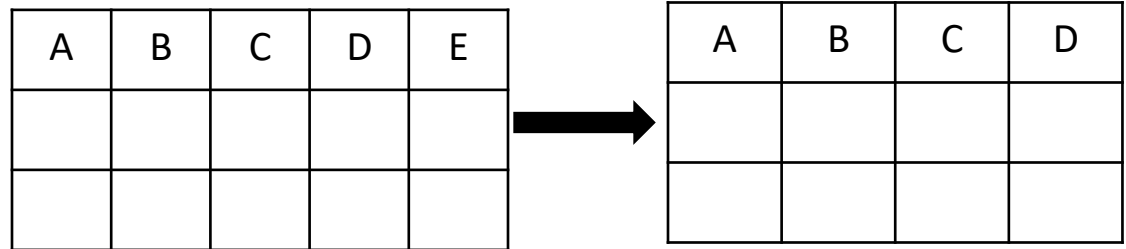
ML Model

Pre-processing/ Data Preparation



Improves quality of the training and reliability

Removal of erroneous data: not in format,
not in range, outliers, missing values

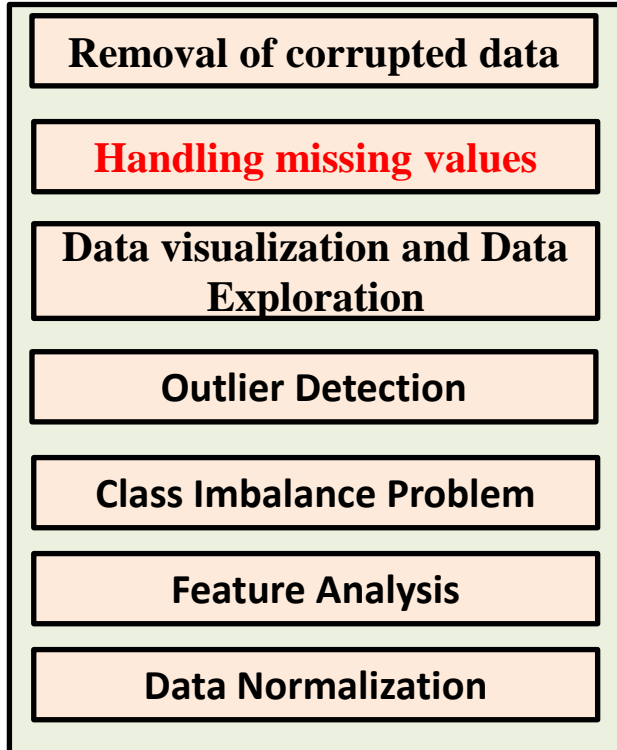


Age	weight	Height	press ure	Heart Beat	class
25	54	5.6	110	300	Y
30	64	5.7	130	400	N

Age	weight	Height	press ure	class
25	54	5.6	110	Y
30	64	5.7	130	N

ML Model

Pre-processing/ Data Preparation



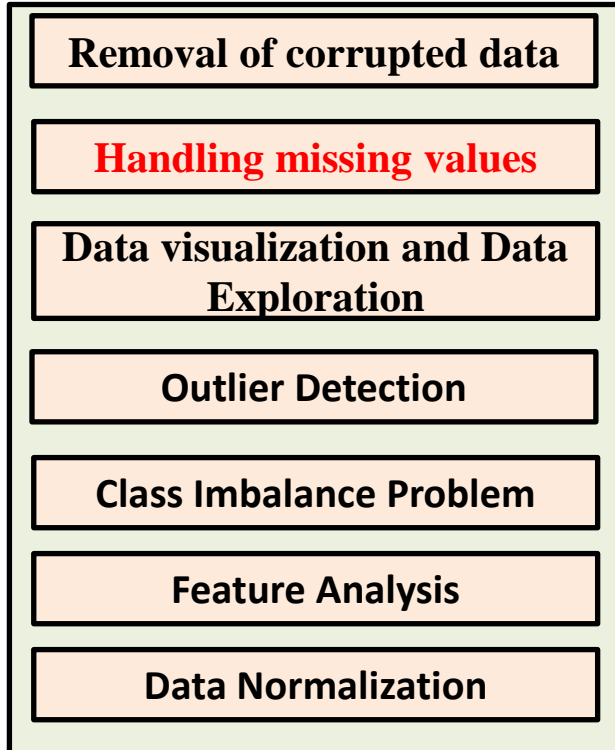
What is a Missing Value?

How is it created ?

BP	Heart Beat	Weight	Class
100	70	50	Y
101		60	Y
102	59		N
120		70	N
120	66	85	N
124	75	82	N
154	76		Y
124	56	81	Y
115	70	73	Y

ML Model

Pre-processing/ Data Preparation



Type of Missing values:

- MCAR: Missing Completely At random
- MAR: Missing At Random
- MNAR: Missing Not At Random

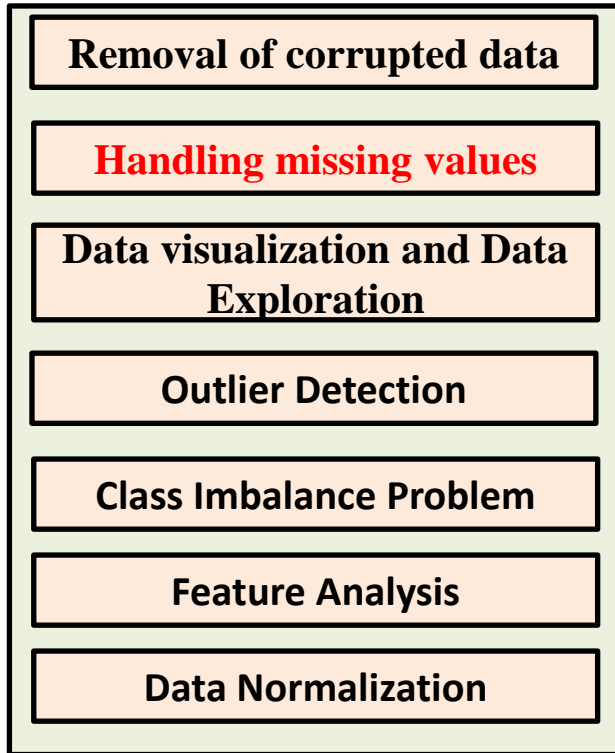
MCAR: Missing Completely At random

- If there is no relationship among the missing data and any other variable of the dataset
- Probability of missing is not related with any other variable.

Roll. No	Due book
120	05
101	
102	03
112	09
105	02

ML Model

Pre-processing/ Data Preparation



Type of Missing values:

- MCAR: Missing Completely At random
- MAR: Missing At Random
- MNAR: Missing Not At Random

MAR: Missing At Random

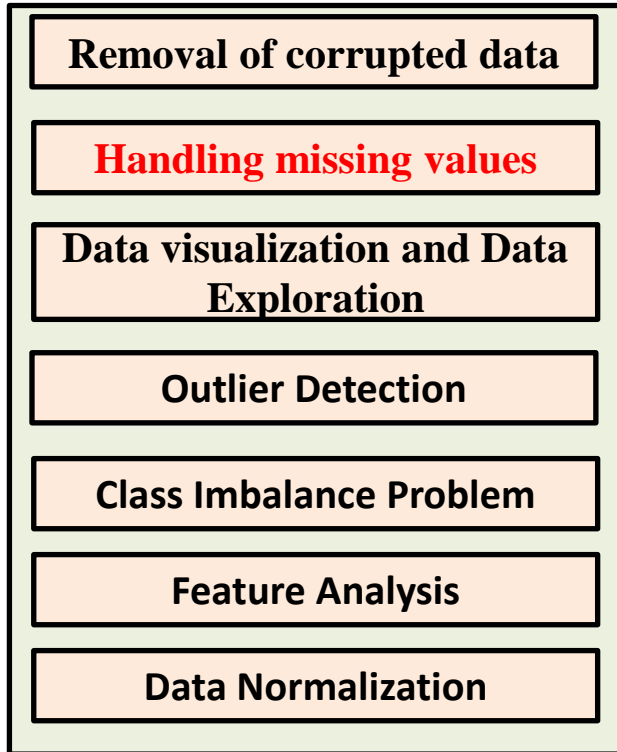
If there is a relationship among the missing data and any other variable of the dataset. Therefore need to analyze the relationship between the missing data and the variable on which it depends upon.

If the probability of being missing is the same only within groups defined by the *observed* data, then the data are missing at random (MAR). MAR is a much broader class than MCAR

Roll. No	Due book	Sex	Roll. No	Due book	Sex
120	05	M	100		M
101		F	103		M
102	03	F	115	03	F
112	09	M	111	09	F

ML Model

Pre-processing/ Data Preparation



Type of Missing values:

- **MCAR: Missing Completely At random**
- **MAR: Missing At Random**
- **MNAR: Missing Not At Random**

MNAR: Missing Not At Random

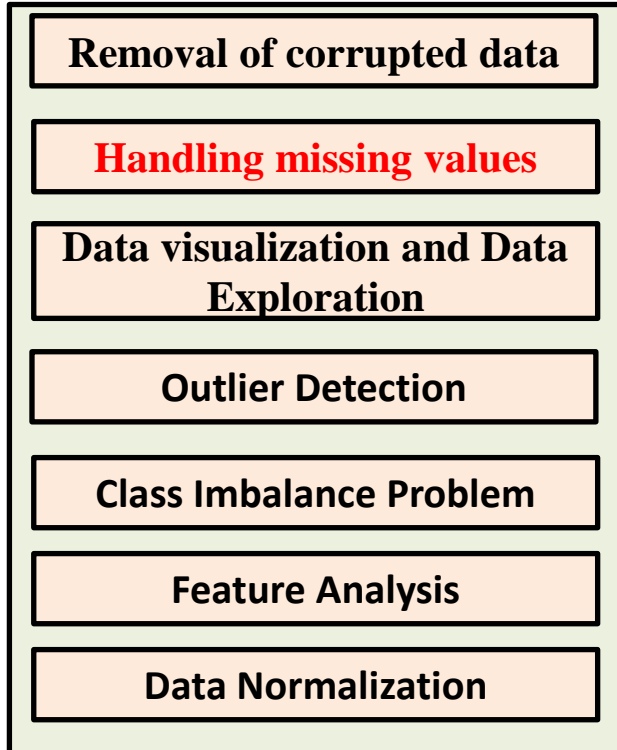
There is a relationship between the missing data and the variable itself in which the data is missing.

Required to proper understanding about the variable before making any imputation.

Year	No. Population	Year	No. Population
2005	1000	2010	1800
2006	1100	2011	2000
2007	1300	2012	2300
2008		2013	
2009	1600	2014	2800

ML Model

Pre-processing/ Data Preparation



Improves quality of the training

Handling Missing Values:

1. Deletion
2. Data Imputation

BP	Heart Beat	Weight	Class
100	70	50	Y
101		60	Y
102	59		N
120		70	N
120	66	85	N
124	75	82	N
154	76		Y
124	56	81	Y
115	70	73	Y

Handling Missing Values

BP	Heart Beat	Weight	Class
100	70	50	Y
101	65	60	Y
102	59	52	N
120		70	N
120	66	85	N
124	75	82	N
154	76		Y
124	56	81	Y
115	70	73	Y

Deletion:

Deletion methods are used when missing is occurred due to “missing completely at random” and “missing at random”

1. Deleting Rows
2. Deleting Columns
3. Pairwise

Handling Missing Values

Deletion:

1. Deleting Rows
2. Deleting Columns
3. Pairwise

BP	Heart Beat	Weight	Class
100	70	50	Y
101	65	60	Y
102	59	52	N
120		70	N
120	66	85	N
124	75	82	N
154	76	52	Y
124	56	81	Y
115	70	73	Y



BP	Heart Beat	Weight	Class
100	70	50	Y
101	65	60	Y
102	59	52	N
120	66	85	N
124	75	82	N
154	76	52	Y
124	56	81	Y
115	70	73	Y

Handling Missing Values

BP	Heart Beat	Weight	Class
100	70	50	Y
101		60	Y
102	59	52	N
120		70	N
120	66	85	N
124	75	82	N
154	76		Y
124		81	Y
115	70	73	Y



BP	Weight	Class
100	50	Y
101	60	Y
102	52	N
120	70	N
120	85	N
124	82	N
154	52	Y
124	81	Y
115	73	Y

Deletion:

1. Deleting Rows
2. Deleting Columns
3. Pairwise

Handling Missing Values

Deletion:

1. Deleting Rows
2. Deleting Columns
3. Pairwise

BP	Heart Beat	Weight	Class
100	70	50	Y
101		60	Y
102	59	52	N
120		70	N
120	66	85	N
124	75	82	N
154	76		Y
124		81	Y
115	70	73	Y



BP	Weight	Class
100	50	Y
101	60	Y
102	52	N
120	70	N
120	85	N
124	82	N
124	81	Y
115	73	Y

Pairwise correlation between predictor and target is found to help in deletion

BP--Class-----High

Heart Beat—Class-----Low

Weight-Class-----High

Handling Missing Values

BP	Heart Beat	Weight	Class
100	70	50	Y
101		60	Y
102	59	52	N
120		70	N
120	66	85	N
124	75	82	N
154	76		Y
124		81	Y
115	70	73	Y



BP	Heart Beat	Class
100	70	Y
102	59	N
120	66	N
124	75	N
154	76	Y
115	70	Y

Deletion:

1. Deleting Rows
2. Deleting Columns
3. Pairwise

Pairwise correlation between predictor and target is found to help in deletion

BP--Class-----High

Heart Beat—Class-----High

Weight-Class-----Low

Handling Missing Values

Deletion:

Pros: Trained model becomes robust as all the missing values are deleted.

Cons: 1. loss of information and 2. trained model works poorly if deletion is excessive

Handling Missing Values

Imputation:

1. **Mean**
 2. **Median**
 3. **Mode**
 4. **Linear Interpolation**
 5. **Linear Regression**
 6. **K-NN**
- Different ML techniques**

BP	Heart Beat	Weight	Class
100	70	50	Y
101	65	60	Y
102	59	52	N
120	67	70	N
120	66	85	N
124	75	82	N
154	76	52	Y
124	56	81	Y
115	70	73	Y

$$\begin{aligned}\text{Mean} &= (70+65+59+66+75+76+56+70)/8 \\ &= 67.125 \\ &= 67\end{aligned}$$

Handling Missing Values

BP	Heart Beat	Weight	Class
100	Y	50	Y
101	Y	60	Y
102	N	52	N
120	Y	70	N
120	Y	85	N
124	Y	82	N
154	Y	52	Y
124	Y	81	Y
115	N	73	Y

Imputation:

1. Mean
2. Median
3. Mode
4. Linear Interpolation
5. Linear Regression (ML)
6. K-NN (ML)
7. SoftImpute (ML)
8. Mice (ML) (good)
9. MatrixFactorization (ML)
10. miss- Forest (ML)
11. Deductive Imputation (LR)
12. Factor Analysis of Mixed Data (FAMD) (ML)

Categorical Data

K-NN is used value of $k = 4$

Handling Missing Values

Last Observation Carried Forward (LOCF)

If data is time-series data, one of the most widely used imputation methods is the last observation carried forward. Whenever a value is missing, it is replaced with the last observed value.

Mobile ID	Date	Download Speed	Data Limit Usage
1	1-Jan	157	80%
2	2-Jan	99	81%
3	3-Jan	167	83%
4	4-Jan	90	84%
5	5-Jan	N/A	86%
6	6-Jan	155	87%
7	7-Jan	N/A	89%
8	8-Jan	N/A	90%
9	9-Jan	180	92%



Mobile ID	Date	Download Speed	Data Limit Usage
1	1-Jan	157	80%
2	2-Jan	99	81%
3	3-Jan	167	83%
4	4-Jan	90	84%
5	5-Jan	90	86%
6	6-Jan	155	87%
7	7-Jan	155	89%
8	8-Jan	155	90%
9	9-Jan	180	92%

Handling Missing Values

Next Observation Carried Backward (NOCB)

It is a similar approach like LOCF which works oppositely by taking the first observation after the missing value and carrying it backward ("next observation carried backwards", or NOCB).

Mobile ID	Date	Download Speed	Data Limit Usage
1	1-Jan	157	80%
2	2-Jan	99	81%
3	3-Jan	167	83%
4	4-Jan	90	84%
5	5-Jan	N/A	86%
6	6-Jan	155	87%
7	7-Jan	N/A	89%
8	8-Jan	N/A	90%
9	9-Jan	180	92%



Mobile ID	Date	Download Speed	Data Limit Usage
1	1-Jan	157	80%
2	2-Jan	99	81%
3	3-Jan	167	83%
4	4-Jan	90	84%
5	5-Jan	155	86%
6	6-Jan	155	87%
7	7-Jan	180	89%
8	8-Jan	180	90%
9	9-Jan	180	92%

Handling Missing Values

Linear Interpolation

It is a mathematical method that adjusts a function to data and uses this function to extrapolate the missing data. **The simplest type of interpolation is linear interpolation, where the values before the missing data and after the same is used.**

Mobile ID	Date	Download Speed	Data Limit Usage
1	1-Jan	157	80%
2	2-Jan	99	81%
3	3-Jan	167	83%
4	4-Jan	90	84%
5	5-Jan	N/A	86%
6	6-Jan	150	87%
7	7-Jan	160	89%
8	8-Jan	N/A	90%
9	9-Jan	180	92%



Mobile ID	Date	Download Speed	Data Limit Usage
1	1-Jan	157	80%
2	2-Jan	99	81%
3	3-Jan	167	83%
4	4-Jan	90	84%
5	5-Jan	120	86%
6	6-Jan	150	87%
7	7-Jan	160	89%
8	8-Jan	170	90%
9	9-Jan	180	92%

$$(90+150)/2 = 120$$

$$(160+180)/2 = 170$$

Handling Missing Values

Adding a category to capture NA

This is perhaps the most widely used method of missing data imputation for categorical variables.

This method consists of treating missing data as an additional label or category of the variable.

Mobile ID	Mobile Package	Download Speed	Data Limit Usage
1	Fast+	157	80%
2	N/A	99	70%
3	Fast+	167	10%
4	Fast+	90	80%
5	Lite	76	70%
6	N/A	155	10%
7	Fast+	200	95%
8	Lite	76	77%
9	N/A	180	95%



Mobile ID	Mobile Package	Download Speed	Data Limit Usage
1	Fast+	157	80%
2	Missing	99	70%
3	Fast+	167	10%
4	Fast+	90	80%
5	Lite	76	70%
6	Missing	155	10%
7	Fast+	200	95%
8	Lite	76	77%
9	Missing	180	95%

Handling Missing Values



Imagine you have a categorical variable Education with the following categories:

- High School
- Bachelor's
- Master's

Some values are missing (NaN). To handle this:

1. Replace the missing values with a new category, e.g., "Unknown".
2. Encode with level encoding or One-hot encoding

Handling Missing Values

Adding a category to capture NA



```
import pandas as pd
```

```
# Example dataset
```

```
data = {'Education': ['High School', 'Bachelor\'s', None,  
                      'Master\'s', None]}
```

```
df = pd.DataFrame(data)
```

```
# Replace missing values with a new category
```

```
df['Education'] = df['Education'].fillna('Unknown')
```

```
print(df)
```


Handling Missing Values

Adding a category to capture NA



```
from sklearn.preprocessing import LabelEncoder

# Label Encoding
label_encoder = LabelEncoder()
df['Education_Label'] =
label_encoder.fit_transform(df['Education'])
print(df)
```

Handling Missing Values

Frequent category imputation

Replacement of missing values by the most frequent category is the equivalent of mean/median imputation. It consists of replacing all occurrences of missing values within a variable with the variable's most frequent label or category.

Mobile ID	Mobile Package	Download Speed	Data Limit Usage
1	Fast+	157	80%
2	N/A	99	70%
3	Fast+	167	10%
4	Fast+	90	80%
5	Lite	76	70%
6	N/A	155	10%
7	Fast+	200	95%
8	Lite	76	77%
9	N/A	180	95%



Mobile ID	Mobile Package	Download Speed	Data Limit Usage
1	Fast+	157	80%
2	Fast+	99	70%
3	Fast+	167	10%
4	Fast+	90	80%
5	Lite	76	70%
6	Fast+	155	10%
7	Fast+	200	95%
8	Lite	76	77%
9	Fast+	180	95%

Handling Missing Values

Missing Value Treatment using most recent data imputation techniques



MICE (Multiple Imputation by Chained Equation)

Handling Missing Values

Multiple Imputation

Multiple Imputation (MI) is a statistical technique for handling missing data.

The key concept of MI is to use the distribution of the observed data to estimate a set of plausible values for the missing data.

Estimates are combined to obtain a set of parameter estimates.

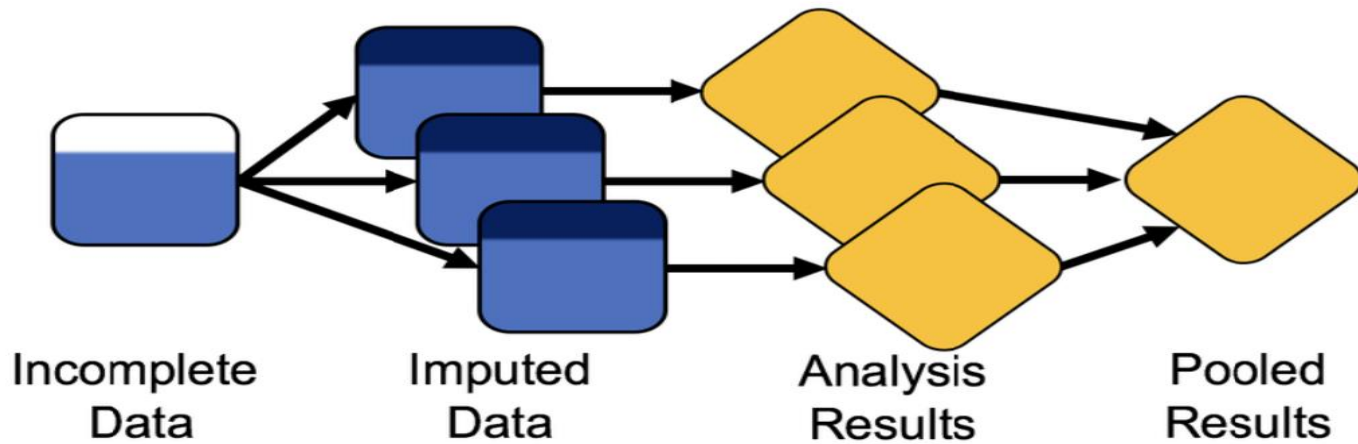
Multiple datasets are created and then analysed individually but identically to obtain a set of parameter estimates.

Multiple Imputation by Chained Equations (MICE) approach is a flexible way of handling more than one missing variable,

The benefit of the multiple imputations is to restore the natural variability of the missing values.

Handling Missing Values

Multiple Imputation



Multiple Imputation

First step would be to remove the "Personal Loan" column as it is the target column, we will not need this column for imputation.

age	experience	salary(K)	Personal loan
25		50	1
27	3		1
29	5	80	0
31	7	90	0
33	9	100	1
	11	130	0



age	experience	salary(K)
25		50
27	3	
29	5	80
31	7	90
33	9	100
	11	130

Multiple Imputation

Second step would be a simple imputation, such as imputing the mean, which is performed for every missing value in the dataset that leads to the formation of zeroth dataset.

age	experience	salary(K)
25		50
27	3	
29	5	80
31	7	90
33	9	100
	11	130



age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
29	11	130

zeroth dataset

Multiple Imputation

Third step would be to remove the "age" imputed values and keep the imputed values in other columns as shown here. Now, we will be imputing the columns from left to right.

age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
29	11	130



age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
	11	130

Multiple Imputation

In the fourth step, the remaining features and rows(top 5 rows of experience and salary) become the feature matrix (purple cells), "age" becomes the target variable(yellow cells).

We will run the linear regression model on the fully filled rows with $X = \text{experience and salary}$ and $Y = \text{age}$. To estimate the missing age, we will use the missing value row (white cells) as the test data. So, top 5 rows will be training data and the last row that has missing age will be test data. We will use (experience = 11 and salary = 130) to predict corresponding "age" value. When I did this, I found that my model predicted the age as 34.99.

age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
	11	130

Multiple Imputation

In the fifth step, we update the predicted age value in the missing cell in "age" column.

Now, remove "experience" imputed value. The remaining features and rows becomes the feature matrix(purple cells) and "experience" becomes the target variable(yellow cells). We will run the linear regression model on the fully filled rows with $X = \text{age and salary}$ and $Y = \text{experience}$. To estimate the missing experience, we will use the missing value row (white cells) as the test data. The predicted value for experience is 0.98.

age	experience	salary(K)
25		50
27	3	90
29	5	80
31	7	90
33	9	100
34.99	11	130


Multiple Imputation

In the sixth step, we update the predicted experience value in the missing cell in "experience" column and remove "salary" imputed value.

The remaining features and rows becomes the feature matrix(purple cells) and "salary" becomes the target variable(yellow cells). We will run the linear regression model on the fully filled rows with $X = \text{age and experience}$ and $Y = \text{salary}$. To estimate the missing salary, we will use the missing value row (white cells) as the test data. The predicted value for Salary is 70.

age	experience	salary(K)
25	0.98	50
27	3	
29	5	80
31	7	90
33	9	100
34.99	11	130

Multiple Imputation




Now we **impute the missing values in the original dataset** and the predicted values after 1st iteration is shown here.

Let's name this as **"First" dataset**.

age	experience	salary(K)
25	0.98	50
27	3	70
29	5	80
31	7	90
33	9	100
34.99	11	130

Multiple Imputation

In the seventh step, We will subtract the two datasets (zeroth and first). The resultant dataset is as below:

age	experience	salary(K)	minus	age	experience	salary(K)		age	experience	salary(K)
25	7	50		25	0.98	50		0	6.02	0
27	3	90		27	3	70		0	0	20
29	5	80		29	5	80		0	0	0
31	7	90		31	7	90		0	0	0
33	9	100		33	9	100		0	0	0
29	11	130		34.99	11	130		-5.99	0	0

If we observe, the absolute difference between 2 datasets are higher in few imputed values. Our aim is to reduce these differences close to 0. To achieve this we have to do many iterations. So, now we repeat the steps 2-6 with the new dataset (first), until we get a stable model. i.e. until the difference between the 2 latest imputed datasets becomes very small, close to 0. Technically, we stop the iterations when a pre-defined threshold is reached or a pre defined maximum number of iterations gets completed.

Multiple Imputation

Now we will use the "first" dataset as our base dataset to do imputations, and discard the "Zeroth" dataset which had the mean imputations. With "first" dataset as base, let's perform all the previous steps and again predict the imputed values for the initial 3 missing values.

age	experience	salary(K)
25	0.98	50
27	3	70
29	5	80
31	7	90
33	9	100
34.99	11	130

Multiple Imputation

Here's is the iteration 2 values and the new dataset values are subtracted from the first dataset and got the difference matrix as below:

Iteration 2

age	experience	salary(K)
25	0.98	50
27	3	70
29	5	80
31	7	90
33	9	100
34.99	11	130

First Dataset

After all
imputations



age	experience	salary(K)
25	0.975	50
27	3	70
29	5	80
31	7	90
33	9	100
34.95	11	130

Second Dataset

After
Second - First



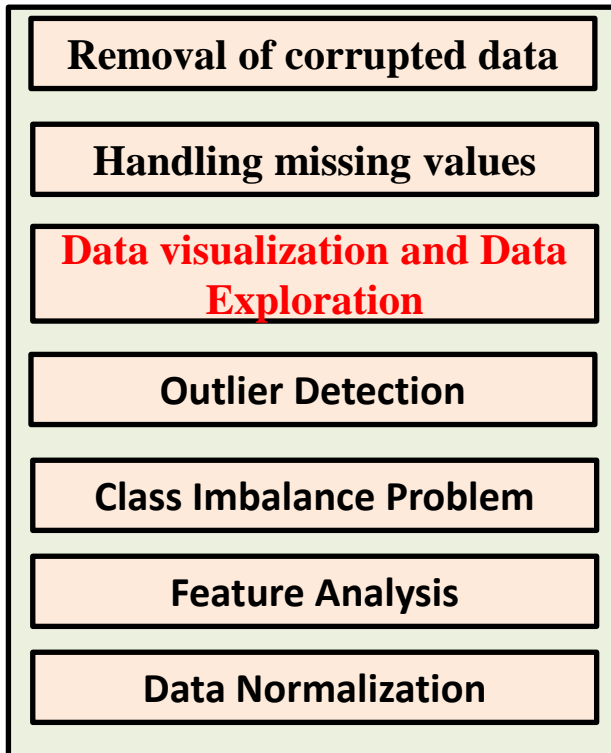
age	experience	salary(K)
0	0.005	0
0	0	0
0	0	0
0	0	0
0	0	0
0.004	0	0

Difference Matrix

Now, after second iteration, we can see that the difference is very negligible. We can either stop here as we almost got the same numbers, or proceed with next iteration until we get 0 difference.

Machine Learning Model

Pre-processing/ Data Preparation



Helps to understand underlying behaviour of data which helps to take right steps in data preparation and modelling.

Data Visualization and Data Exploration

Data Exploration:

- **Mean (central tendency)**
- **Median (central tendency)**
- **Variance (Data Spread)**

Set of observation=21 89 34 67 96

Mean= $(21+89+34+67+96)/5 = 61.4$

21 34 67 89 96; Median = 67

Set of observation= 21, 20, 23,24, 25 84 67, 55 96

Mean= $(21+20+23+24+25+84+67+55+96)/9 = 46.11$

20, 21, 23, 24, 25, 55, 67, 84, 96; Median= 25

Set of observation= 21 89 34 67 200

Mean= $(21+89+34+67+200)/5 = 82.2$

21 34 67 89 200; Median = 67

A1: 44, 46, 48, 45, 47

A2: 34, 46, 59, 39, 52

For both mean and median 46

To measure data dispersion or data spread, variance is measured

variance(A1)= 2; variance(A2)= 79.6

A1 values are concentrated around mean

A2 values are extremely spread out

Data Visualization and Data Exploration

Data Visualization:

Box Plot: An effective mechanism to get a one-shot view and understand the nature of data.

It gives a standard visualization of five statistical summary: **minimum**, **first quartile(Q1)**, **median(Q2)**, **third quartile(Q3)** and **maximum**.

Box spans from Q1 to Q3 = **Inter-Quartile Range (IQR)**

Lower Range (LR) extends up to = **(Q1 - 1.5 times of IQR)**

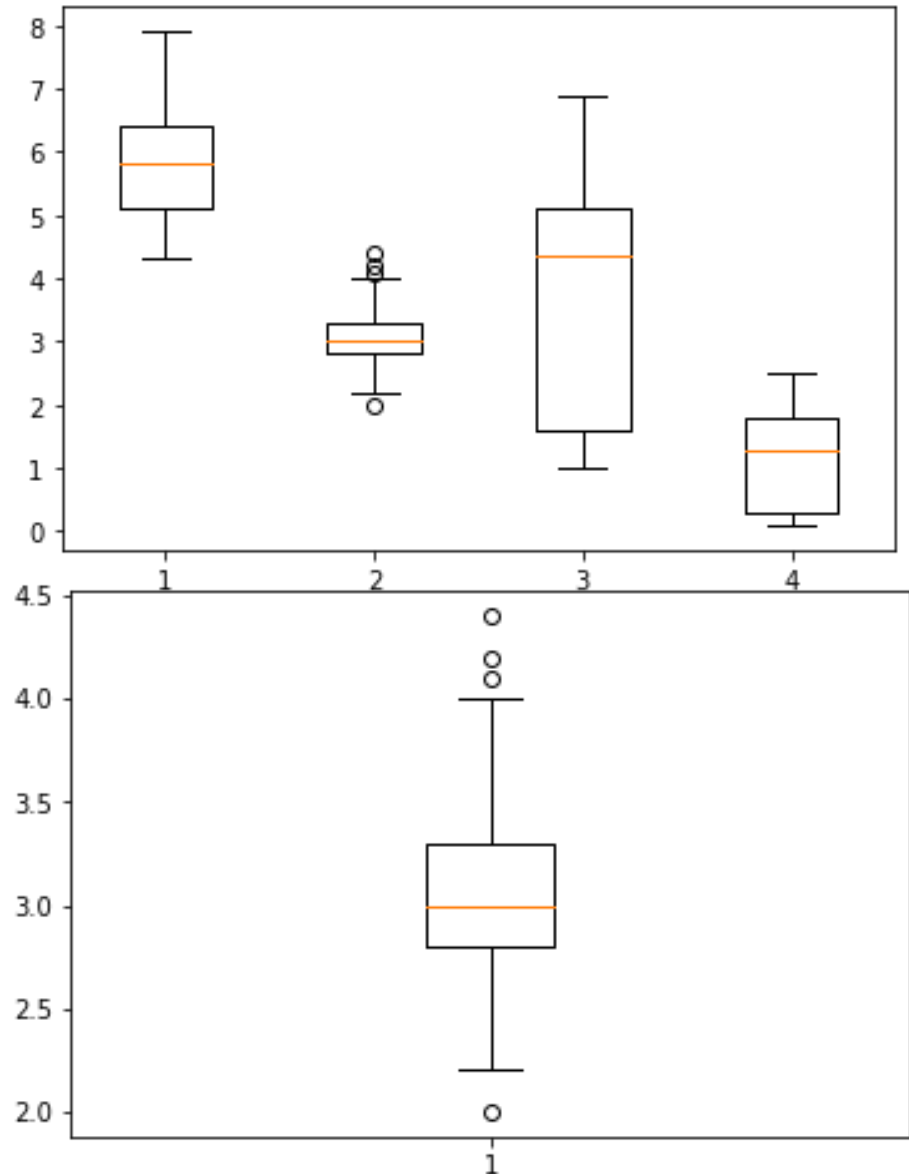
For some x: Q1=73, Q2=76 and Q3=79

IQR=(Q3-Q1)=(79-73)= **6**

LR=(Q1-1.5*IQR)=(73-1.5*6)=(73-9)=**64**

Say some lower data values of x: 70, 63, 60

Minimum= 70 which is larger than 64



Data Visualization and Data Exploration

Data Visualization:

Upper Range (UR) extends up to = (Q3 + 1.5 times of IQR)

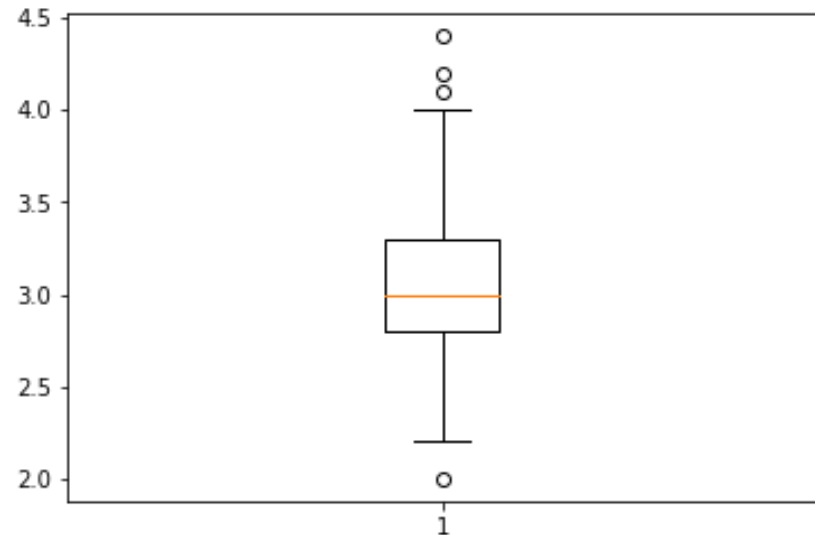
For some x: Q1=73, Q2=76 and Q3=79
IQR=(Q3-Q1)=(79-73)= 6

UR=(Q3+1.5*IQR)=(79+1.5*6)=(79+9)= 88

Say some upper values x: 82, 84, 89

Maximum= 84 which is highest value lower than 88.

x	Frequency	C Frequency
3	4	4
4	204	208 (=4+204)
5	3	211
6	84	295
7	0	295
8	103	398



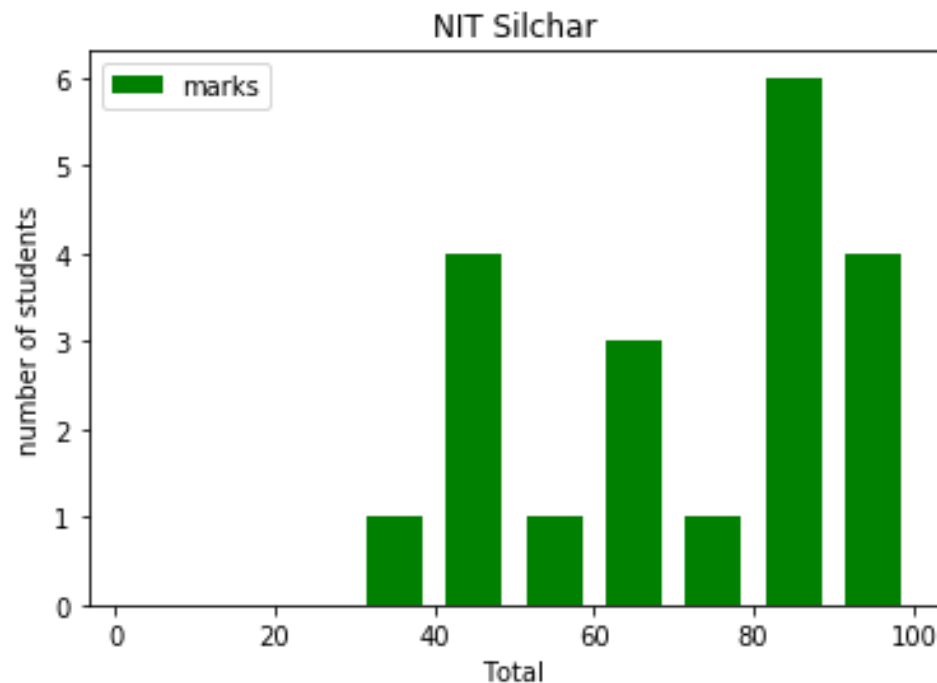
	Fequency/observation	x
Q1	Avg of 99 th and 100 th	4
Q2	199	4
Q3	Avg of 298 th and 299 th	8
IQR	(Q3-Q1)	4
LR	Q1-1.5*4=4-6	-2
Min		3
UR	Q3+1.5*4=8+6	14
Max		8

It can finds outliers.

Data Visualization and Data Exploration

Data Visualization:

- **Histogram (ranges of data values):** helps in understanding the distribution of a numeric data into series of intervals. Gives us quick understanding of the data.

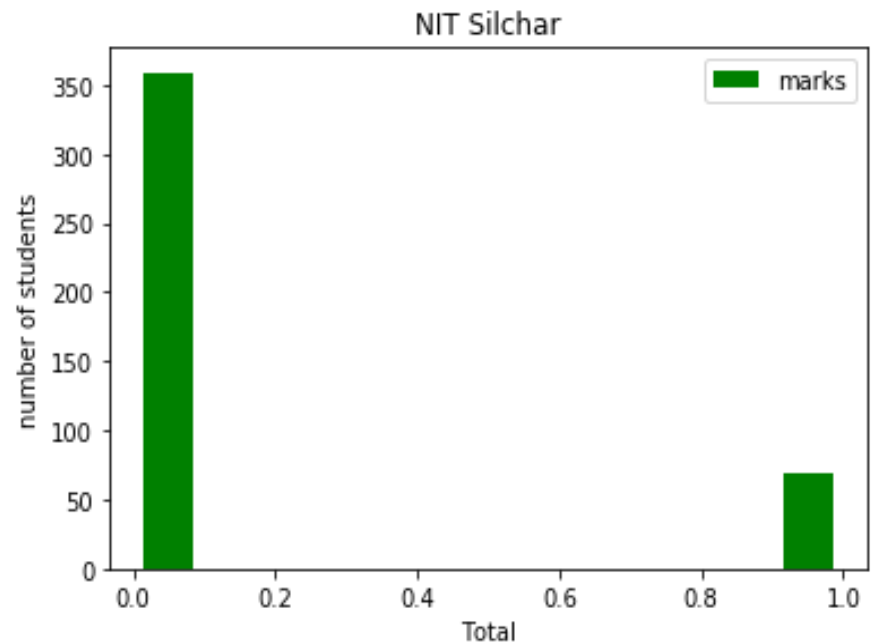
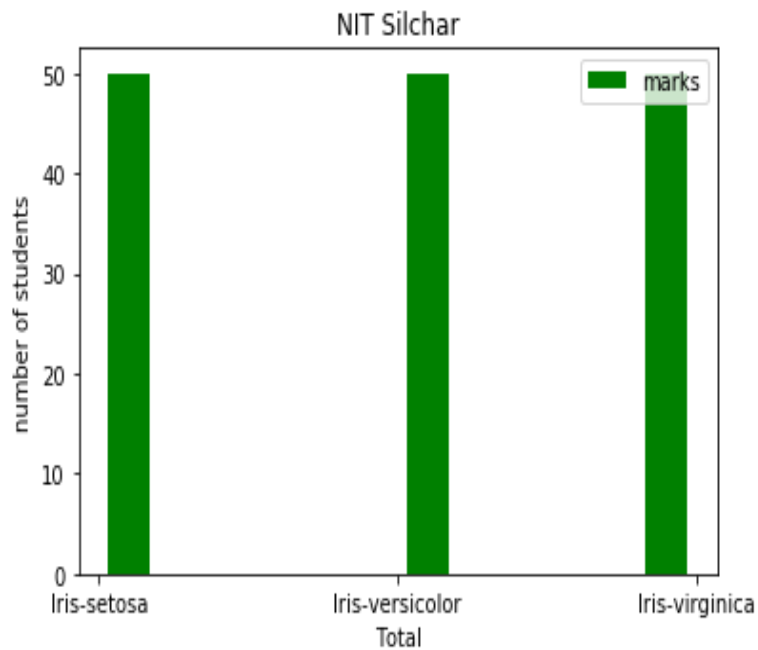


Total
52
45
65
45
68
98
46
88
36
45
86
87
68
86
94
74
93
89
90
86

Data Visualization and Data Exploration

Data Visualization:

- **Histogram:** Gives us quick understanding of the data.

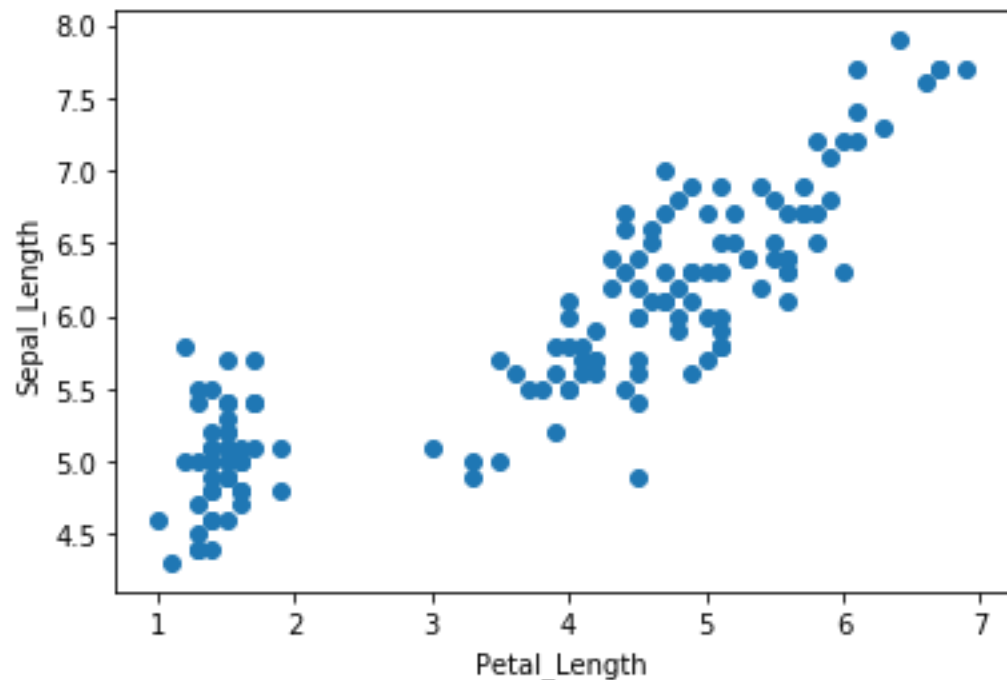


Data Visualization and Data Exploration

Data Visualization:

- **Scattered plot: shows relationship between two variables**

IRIS DATASET: ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width', 'Species'],



Python Libraries for Machine Learning

1. NumPy (Numerical Python):

- It is array-processing package
- It is used to process large multi-dimensional arrays and matrices
- It is used for handling linear algebra, Fourier transforms, and random numbers.
- Other libraries like TensorFlow uses NumPy at the backend for manipulating tensors.,
- With NumPy, we can define arbitrary data types and easily integrate with most databases.

Some important type of function under NumPy:

i. NumPy **Array Manipulation functions**

```
import numpy
```

```
arr1 = numpy.arange(4)  
print('Elements of an array1:\n',arr1)
```

```
res1 = arr1.reshape(2,2)  
print('Reshaped array with 2x2 dimensions:\n',res1)
```

Python Libraries for Machine Learning

Some important type of function unders NumPy:

i. NumPy Array Manipulation functions

```
import numpy
```

```
concat = numpy.concatenate((arr1,arr2),axis=1)  
print(concat)
```

ii. NumPy String functions

`numpy.char.add()` function: Concatenates data values of two arrays, merges them and represents a new array as a result.

`numpy.char.capitalize()` function: It capitalizes the first character of the entire word/string.

Python Libraries for Machine Learning

ii. NumPy String functions

numpy.char.lower() function: Converts the case of the string characters to lower string.

numpy.char.upper() function: Converts the case of the string characters to upper string.

numpy.char.replace() function: Replaces a string or a portion of string with another string value.

iii. NumPy Arithmetic functions

numpy.add() function : It adds two arrays and returns the result.

numpy.subtract() function : Subtracts elements of array2 from array1 and returns the result.

numpy.multiply() function : Multiplies the elements of both the arrays and returns the product.

numpy.divide() function : Divides the array1 by array2 and returns the quotient of array values.

numpy.mod() function: Performs modulus operation and returns the remainder array.

numpy.power() function: Returns the exponential value of array1 ^ array2.

Python Libraries for Machine Learning

iv. NumPy Statistical functions

`numpy.median()` : Calculates the median value of the passed array.

`numpy.mean()` : Returns the mean of the data values of the array.

`numpy.average()` : It returns the average of all the data values of the passed array.

`numpy.std()` : Calculates and returns the standard deviation of the data values of the array.

Python Libraries for Machine Learning

2. Pandas: Pandas are turning up to be the most popular Python library that is used for **data analysis**.

- **The two main types of data structures used by pandas are :** Series (1-dimensional)
- DataFrame (2-dimensional)
- These two put together can handle a vast majority of data: sectors like science, statistics, social, finance, and of course, analytics and other areas of engineering.
- Tabular data with columns of heterogeneous data.
- Ordered and unordered time series data.
- Arbitrary matrix data with the homogeneous or heterogeneous type of data in the rows and columns
- Any other form of statistical or observational data sets.

Some important type of function unders Pandas:

1. read_csv(): read_csv() function helps read a comma-separated values (csv) file into a Pandas DataFrame. It can also read files separated by delimiters other than comma, like | or tab

2. head(): head(n) is used to return the first n rows of a dataset. By default, df.head() will return the first 5 rows of the DataFrame

Python Libraries for Machine Learning

Some important type of function unders Pandas:

3. describe(): **describe() is used to generate descriptive statistics of the data in a Pandas DataFrame or Series.** It summarizes central tendency and dispersion of the dataset. describe() helps in getting a quick overview of the dataset.

4. memory_usage(): memory_usage() returns a Pandas Series having the memory usage of each column (in bytes) in a Pandas DataFrame.

```
data_1.memory_usage(deep=True)
```

5. astype(): astype() is used to cast a Python object to a particular data type.

6. loc[:]: loc[:] helps to access a group of rows and columns in a dataset, a slice of the dataset, as per our requirement.

7. to_datetime(): to_datetime() converts a Python object to datetime format.

8. value_counts(): value_counts() returns a Pandas Series containing the counts of unique values.

9. drop_duplicates(): drop_duplicates() returns a Pandas DataFrame with duplicate rows removed.

```
data_1.drop_duplicates(inplace=True)
```

10. groupby(): groupby() is used to group a Pandas DataFrame by 1 or more columns, and perform some mathematical operation on it.

Python Libraries for Machine Learning

Some important type of function unders Pandas:

```
data_1.groupby(by='State').Salary.mean()
```

11. merge(): merge() is used to merge 2 Pandas DataFrame objects or a DataFrame and a Series object on a common column (field)

12. sort_values(): sort_values() is used to sort column in a Pandas DataFrame (or a Pandas Series) by values in ascending or descending order.

```
data_1.sort_values(by='Name', inplace=True)
```

13. fillna(): fillna() helps to replace all NaN values in a DataFrame or Series by imputing these missing values with more appropriate values.

```
data_1['City temp'].fillna(38.5, inplace=True)
```

14. Shape: property will return a tuple of the shape of the data frame.

```
f1.shape
```

15. f1.columns: will give you the column values

16. f1.tail():

17. DataFrame.info(): Pandas **dataframe.info()** function is used to get a concise summary of the dataframe.

Python Libraries for Machine Learning

Some important type of function unders Pandas:

18. dtypes: (f1.dtypes) dtypes shows the data type of each column. (f1.dtypes)

19. Size: (f1.size) Size, as the name suggests, returns the size of a dataframe which is the number of rows multiplied by the number of columns.

20. Sample: (f1.sample(n=8)) Sample method allows you to select values randomly from a **Series** or **DataFrame**.

21. isnull:(f1.isnull()) To handle missing values

22. isna() : (f1.isna().any()) Isna function returns a dataframe filled with boolean values with true indicating missing values.

23. f1.isnull().sum() : We can calculate the number of missing values in each column

24. nunique(): (f1. nunique()) Nunique counts the number of unique entries over columns or rows. It is very useful in categorical features especially in cases where we do not know the number of categories beforehand

25. index() (f1.index) searches for a given element from the start of the list and returns the lowest index where the element appears.

26. nsmallest() (f1. nsmallest(5,'Sepal_Width')) finds the 5 observations with the smallest value

27. nlargest() (f1. nlargest(5,'Sepal_Width')) finds the 5 observations with the Largest value.

Python Libraries for Machine Learning

Some important type of function unders Pandas:

28. Loc and iloc

Loc and iloc are used to select rows and columns.

loc: select by labels

iloc: select by positions

```
f1.loc[:5,['Sepal_Length', 'Sepal_Width']]  
f1.iloc[:5,:6]
```

29. **Slicing:** Slicing Rows and Columns using labels.

```
f1[0:4]
```

30. **dropna ()** function is used to remove a row or a column from a dataframe which has a NaN or no values in it

31. **query():** We sometimes need to filter a dataframe based on a condition or apply a mask to get certain values.

```
f1.query('3000<median_value<1000')[ :4]
```

32. **insert()** : offers the option to add the new column in any position using **insert** function

```
f1.insert(5, 'new_name', new_col)
```

Python Libraries for Machine Learning

3. Matplotlib:

- Matplotlib is a data visualization library
- It is used for 2D plotting to produce publication-quality image plots and figures in a variety of formats.
- The library helps to generate histograms, plots, error charts, scatter plots, bar charts with just a few lines of code.

4. SciPy (Scientific Python): This is a python library for **machine learning**, especially for scientific and analytical computing.

- SciPy uses **multi-dimensional array** provided by the NumPy module.
- SciPy depends on NumPy for the array manipulation subroutines.
- The SciPy library offers **modules for linear algebra, image optimization, integration interpolation, special functions, Fast Fourier transform, signal and image processing, Ordinary Differential Equation (ODE) solving, and other computational tasks in science and analytics.**

5. Scikit-learn: It has become the most popular Python machine learning library for developing machine learning algorithms.

Python Libraries for Machine Learning

6. Scikit-learn:

- The library can be used for **data-mining and data analysis**.
- The main machine learning functions that the Scikit-learn library can handle are **classification, regression, clustering, dimensionality reduction, model selection, and preprocessing**.

7. Theano:

- Theano is a **python machine learning library** that can act as an optimizing compiler for evaluating and manipulating mathematical expressions and matrix calculations.
- It is built on NumPy.
- **Theano can work on Graphics Processing Unit (GPU) and CPU.**
- Theano has built-in tools for unit-testing and validation, thereby avoiding bugs and problems.

8. TensorFlow: TensorFlow is a popular computational framework for creating **machine learning models**.

- TensorFlow has a flexible architecture with which it can run on a variety of **computational platforms CPUs, GPUs, and TPUs**.
- TPU stands for Tensor processing unit, a hardware chip built around TensorFlow for machine learning and artificial intelligence.

Python Libraries for Machine Learning

9.Keras: Keras is an open-source library used for **neural networks and machine learning**. Keras can run on top of TensorFlow, Theano etc.

- Keras works with neural-network building blocks like layers, objectives, activation functions, and optimizers.
- Keras also have a bunch of features **to work on images and text images that comes handy when writing Deep Neural Network code**.
- **Keras supports convolutional and recurrent neural networks.**

10. PyTorch: PyTorch has a range of tools and libraries that support **computer vision, machine learning, and natural language processing**

- PyTorch can smoothly integrate with the python data science stack, including NumPy.
- We will hardly make out a difference between NumPy and PyTorch.
- PyTorch include multi GPU support, simplified preprocessors, and custom data loaders.

11. Neurolab: **It is a simple and powerful Neural Network Library for Python**. It is a library for basic neural networks algorithms with flexible network configurations and learning algorithms for Python.

Python Module and Packages

Modules: Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*

A module is a file containing Python definitions and statements. **The file name is the module name with the suffix .py appended.**

Packages:

Packages are a way of structuring Python's module namespace by using "dotted module names". For example, **the module name A.B designates a submodule named B in a package named A.**

```
import sound.effects.echo
```

This loads the submodule **echo** from package **sound.effects**. It must be referenced with its full name.

An alternative way of importing the submodule is:

```
from sound.effects import echo
```

This also loads the submodule **echo**, and makes it available without its package prefix, so it can be used as follows:

```
echo.echofilter(input, output, delay=0.7, atten=4)
```

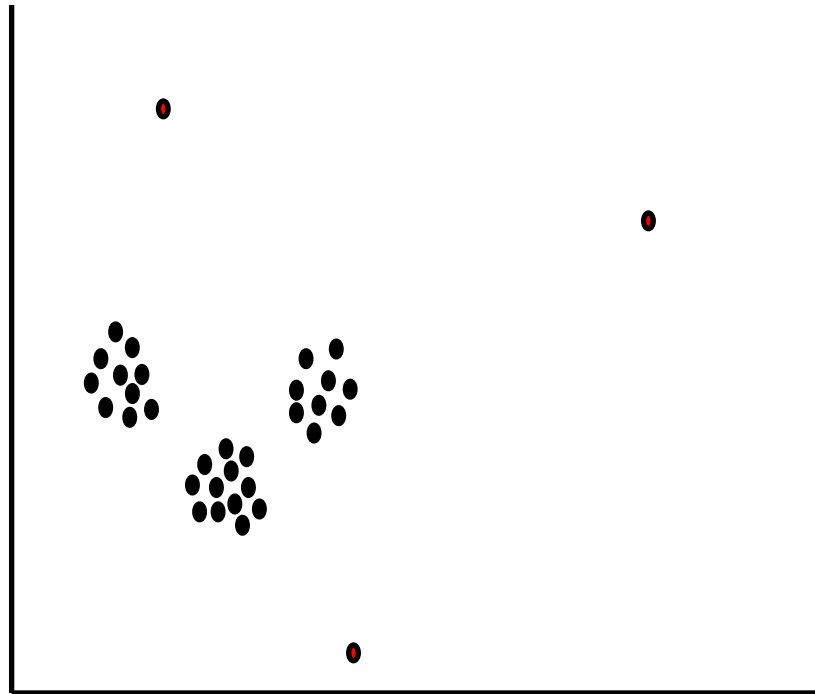
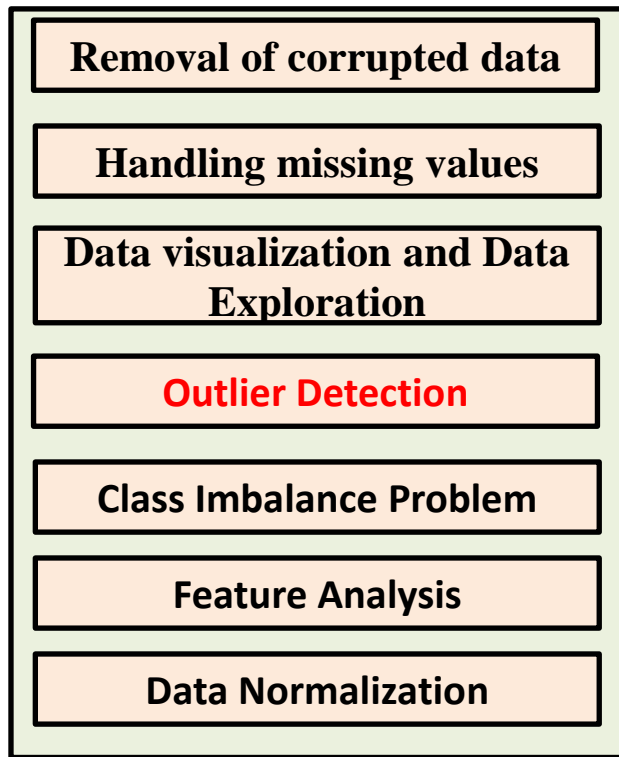
```
from sound.effects.echo import echofilter
```

Again, this loads the submodule **echo**, but this makes its function **echofilter()** directly available:

```
echofilter(input, output, delay=0.7, atten=4)
```

Machine Learning Model

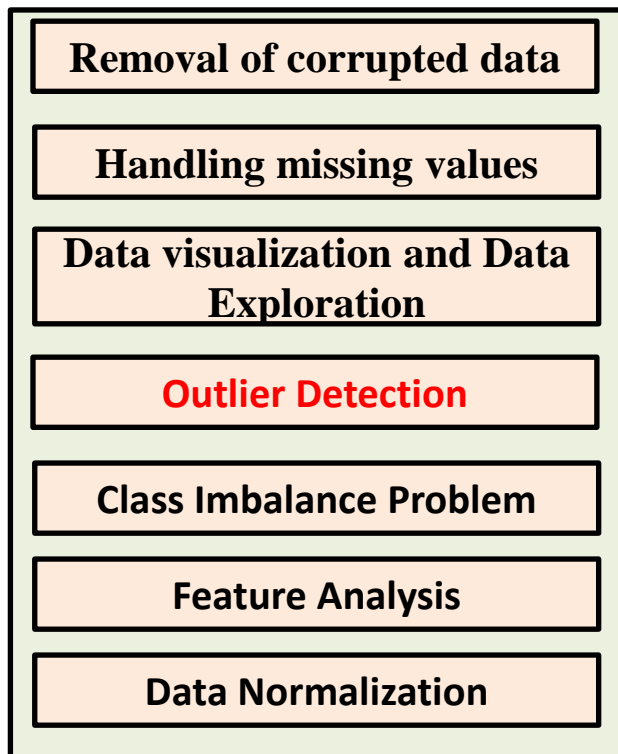
Pre-processing/ Data Preparation



Far from the rest of the observations or the center of mass of observations.

Outlier Detection

Pre-processing/ Data Preparation



Can result in a poor fit and lower predictive modelling performance.

Can fit the data properly and enhance the performance of the model

Histogram or scatter plot can be used for one or two dimensional data.

For high dimensional data, simple statistical methods for identifying outliers can break down.

Outlier Detection

Many techniques are found to detect outlier. Based on learning style, these techniques can be categorized into three classes:

- supervised outlier detection techniques,
- unsupervised outlier detection techniques.
- Semi-supervised outlier detection techniques.

Based on learning measures, these techniques can be categorized into four approaches

- Statistical based
- Distance based
- Density based
- Tree-based

LIMITATIONS OF STATISTICAL BASED APPROACH

- Works well for a single attribute
- In many cases, data distribution may not be known. However statistical based approach needs to know data distribution.
- For multi-dimensional data, it may be difficult to estimate the true distribution

Outlier Detection

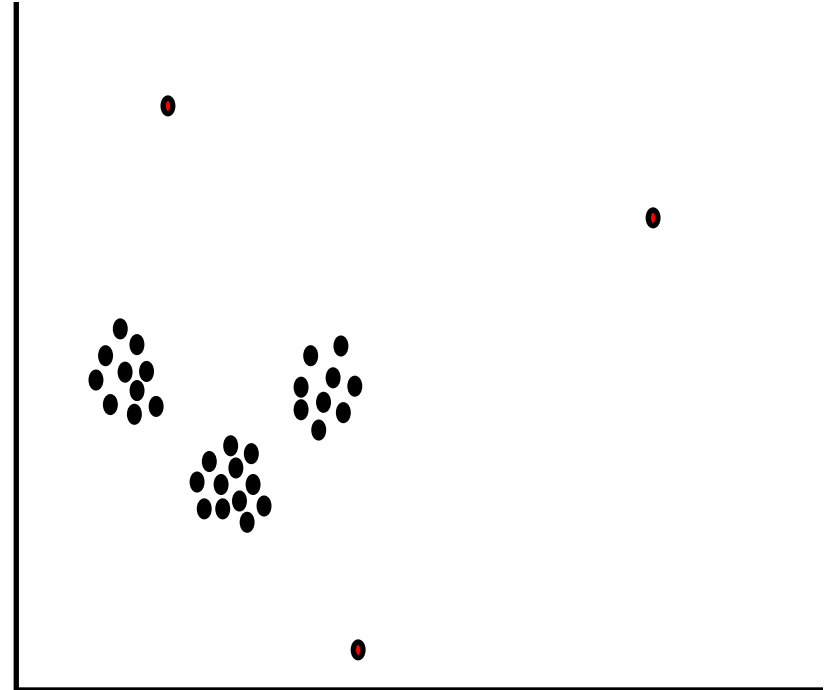
Local Outlier Factor: Each example is assigned a scoring of how isolated or how likely it is to be outliers based on the size of its local neighborhood. Examples with the largest score are more likely to be outliers. (density based unsupervised algorithm)

Isolation Forest: is a tree-based anomaly detection algorithm:

Outliers have attribute-values that are very different from those of normal instances. (Tree based unsupervised algorithm)

One Class SVM: can be used to discover outliers in input data for both regression and classification datasets. This is specially used in imbalanced dataset. (density based unsupervised algorithm)

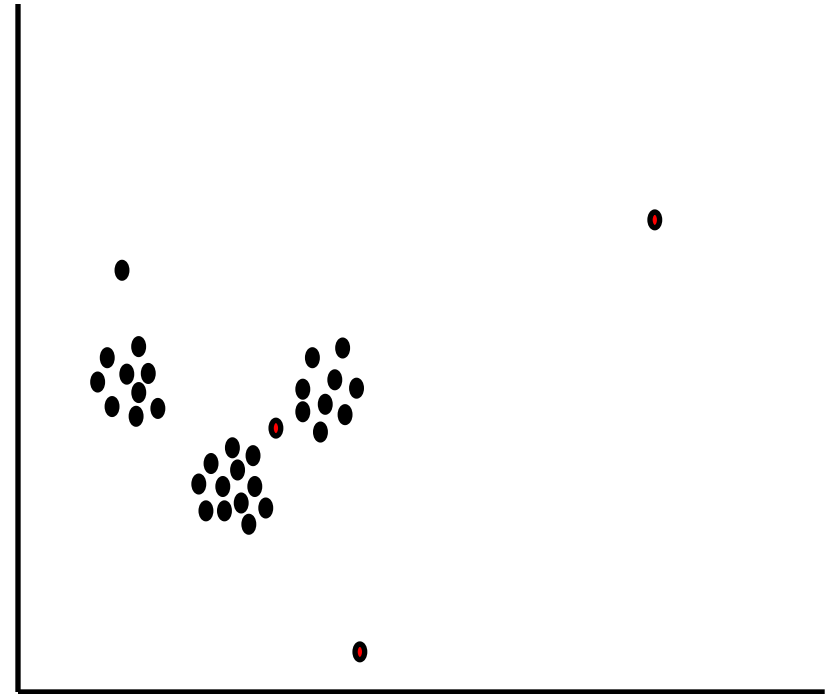
Minimum Covariance Determinant: If the input variables have a Gaussian distribution, then this simple statistical methods can be used to detect outliers. (Statistical based unsupervised algorithm)



Local and Global Outlier

Outliers can be local outlier and global outlier.

Global outlier can be found using distance measure; however local outlier can be found using density based measure



Local Outlier Factors (LOF)

The concept of LOF is based on the statistics of K-Nearest Neighbours (K-NN).

Need to calculate reachability distance.

Need to find LOF score for all the instances.

Kth distance of A [dist-k(A)]= Distance between A and its k-nearest neighbour = DA (k=3)

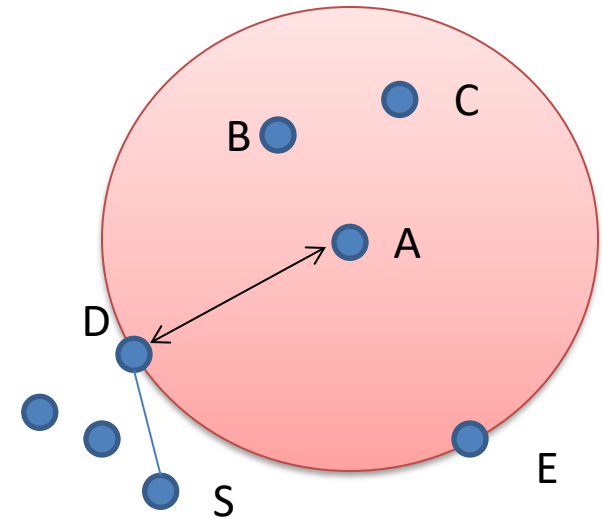
K distance neighbour of A=

$$K\text{-dn}(A) = \{A' | A' \in D, \text{dist}(A, A') \leq \text{dist} - k(A)\} \\ = \{B, C, D, E\}$$

$$\text{Reach-Dist}(A, D) = \max\{\text{Kth-distance}(D), d(A, D)\}, \\ = \max(DS, AD) = AD$$

where Kth-distance(D) is the distance of instance D from its Kth nearest neighbor=DS

d(A,D) is the actual distance between A and D=AD



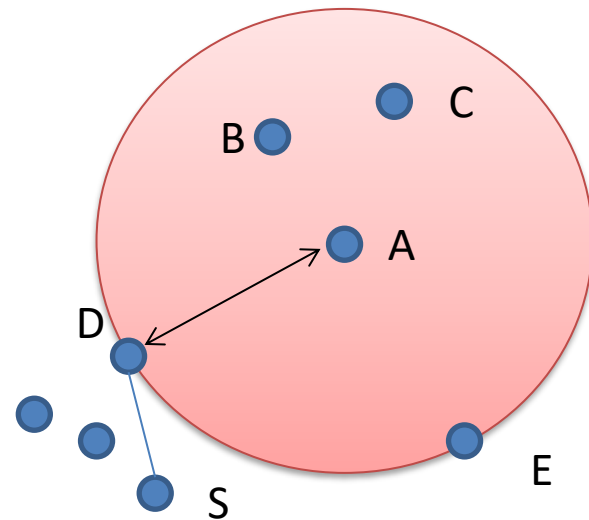
Local Outlier Factors (LOF)

Average RD_A =

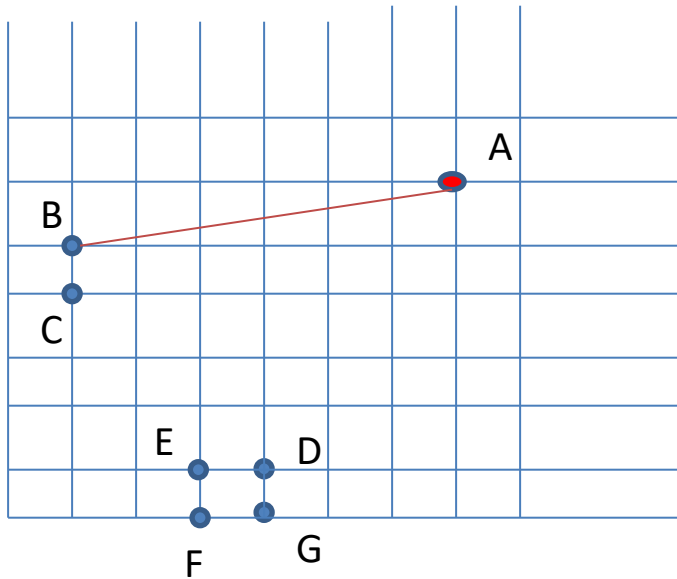
$$\frac{1}{k-dn} \sum_{k-dn} \max[(k^{th} \text{ distance of } A's \text{ neighbor}, \text{distance}(A, \text{the neighbor}))]$$

=

$$\frac{1}{4} \sum_4 \max[(3^{th} \text{ distance of } A's \text{ neighbor}, \text{distance}(A, \text{the neighbor}))]$$



Local Outlier Factors (LOF)



	A1	A2
A	7	6
B	1	5
C	1	4
D	4	1
E	3	1
F	3	0
G	4	0

$$\begin{aligned}
 AB^2 &= 6^2 + 1^2 \\
 &= 36 + 1 \\
 &= 37 \\
 AB &= 6.08
 \end{aligned}$$

Kth distance of A
[dist-k(A)] = AC = 6.32

K distance neighbour
of A = {B, C, D}

Distances	
AB	6.08
AC	6.32
AD	5.8
AE	6.4
AF	7.2
BA	6.08
BC	1
BD	5
BE	4.4
BF	5.3
CA	6.32
CB	1
CD	4.2
CE	3.6
CF	4.4
DA	5.8
DB	5
DC	4.2
DE	1
DF	1.4
DG	1

Local Outlier Factors (LOF)

Average $RD_A = \frac{1}{3} \sum_3 \max[(3^{th} \text{ distance of } A's \text{ neighbor}, \text{distance}(A, \text{the neighbor}))]$

$$= \frac{1}{3} [\max(3^{th} \text{ dist } B, \text{dist}(A, B)) + \max(3^{th} \text{ dist } C, \text{dist}(A, C)) + \max(3^{th} \text{ dist } D, \text{dist}(A, D))]$$

$$= \frac{1}{3} [\max(5, 6.08) + \max(4.2, 6.32) + \max(1.4, 5.8)]$$

$$= \frac{1}{3} [6.08 + 6.32 + 5.8] = 6.06$$

Density is reverse of distance therefore **Local Reachability score LRD**

$$LRD_A = \frac{1}{RD_A}$$

$$= 1/6.06 = 0.165$$

Distances	
AB	6.08
AC	6.32
AD	5.8
AE	6.4
AF	7.2
BA	6.08
BC	1
BD	5
BE	4.4
BF	5.3
CA	6.32
CB	1
CD	4.2
CE	3.6
CF	4.4
DA	5.8
DB	5
DC	4.2
DE	1
DF	1.4
DG	1

Local Outlier Factors (LOF)

Similarly calculated,

Average $RD_B = 5.17$ and $LRD_B = 0.193$

Average $RD_C = 5.17$ and $LRD_C = 0.193$

Average $RD_D = 5.17$ and $LRD_D = 0.193$

$$LOF_A = \frac{\frac{1}{3}(LRD_B + LRD_C + LRD_D)}{LRD_A} = \frac{\frac{1}{3}(0.193 + 0.193 + 0.193)}{0.165} = \frac{0.193}{0.165} = 1.17$$

Generally, if **LOF > 1**, it is considered as an outlier, but that is not always true.

Distances	
AB	6.08
AC	6.32
AD	5.8
AE	6.4
AF	7.2
BA	6.08
BC	1
BD	5
BE	4.4
BF	5.3
CA	6.32
CB	1
CD	4.2
CE	3.6
CF	4.4
DA	5.8
DB	5
DC	4.2
DE	1
DF	1.4
DG	1

Isolation Forest (IF)

- It's an unsupervised learning algorithm that identifies anomaly by isolating outliers in the data.
- Isolation forests are an effective method for detecting outliers or novelties in data.
- It is a relatively novel method based on binary decision trees.
- Isolation forest's basic principle is that outliers are few and far from the rest of the observations.
- It can work for large datasets with one or multi dimensional feature space.

Isolation Forest (IF)

Isolation Forest isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that selected feature. This split depends on how long it takes to separate the points.

Random partitioning produces noticeably shorter paths for anomalies. When a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies.

- An outlier score can be computed for each observation:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

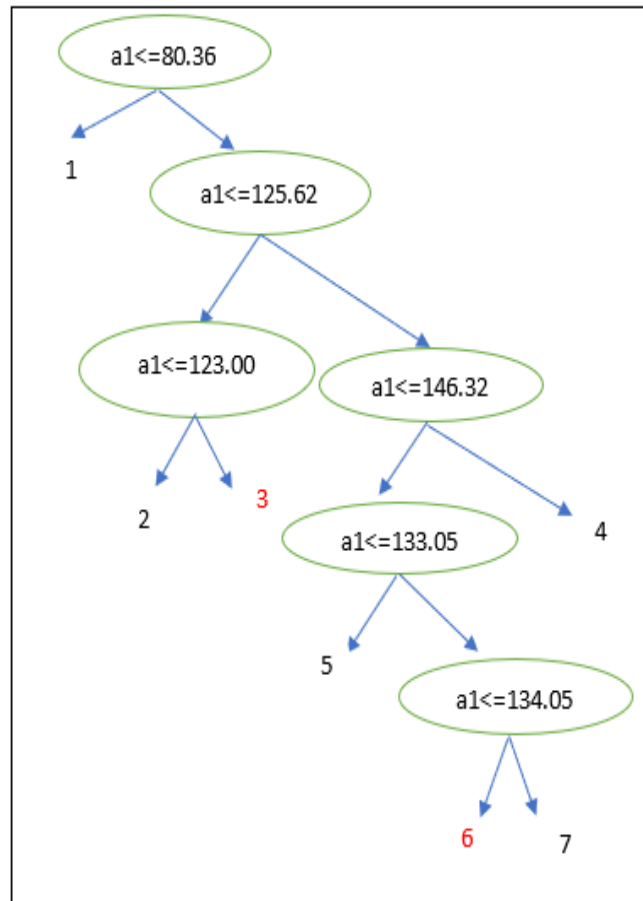
outlier score

- Where $h(x)$ is the path length of the sample x , and $c(n)$ is the ‘unsuccessful length search’ of a binary tree (the maximum path length of a binary tree from root to external node) n is the number of external nodes. After giving each observation a score ranging from 0 to 1; 1 meaning more outlyingness and 0 meaning more normality. A threshold can be specified (ie. 0.55 or 0.60)

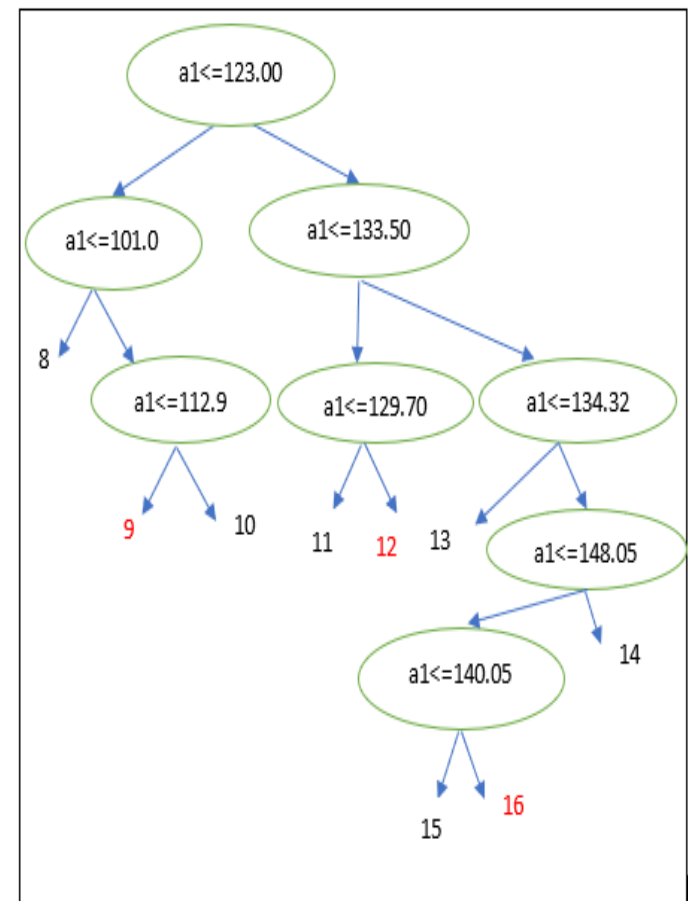
Isolation Forest (IF)

Sl no.	a1	Leaf node
1	150	4, 14
2	122	2, 10
3	135	7, 15
4	132	5, 13
5	4	1, 8
6	138	7, 15
7	121	2, 10
8	127	5, 11

Sample
set



Decision
tree 1



Decision
tree 2

ISOLATION FOREST

Node no. 3,6,9,12,16 are unsuccessful path
 The average unsuccessful path length is given by node 6 and 16
 which is 5. Since 2 trees are considered therefore $h(x)$ is the average path of the datapoint for the 2 trees.

$$s_1 = 2^{-\left(\frac{3+4}{2}\right)} = 2^{-(0.7)} = 0.615$$

$$s_2 = 2^{-\left(\frac{3+3}{2}\right)} = 2^{-(0.6)} = 0.65$$

$$s_3 = 2^{-\left(\frac{5+5}{2}\right)} = 2^{-(1)} = 0.5$$

$$s_4 = 2^{-\left(\frac{4+3}{2}\right)} = 2^{-(0.7)} = 0.615$$

$$s_5 = 2^{-\left(\frac{1+2}{2}\right)} = 2^{-(0.3)} = 0.812$$

$$s_6 = 2^{-\left(\frac{5+5}{2}\right)} = 2^{-(1)} = 0.5$$

$$s_7 = 2^{-\left(\frac{3+3}{2}\right)} = 2^{-(0.6)} = 0.65$$

$$s_8 = 2^{-\left(\frac{4+3}{2}\right)} = 2^{-(0.7)} = 0.615$$

$$\text{Score} = s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Sl no.	weight	scores
1	150	0.615
2	122	0.65
3	135	0.50
4	132	0.615
5	4	0.812
6	138	0.50
7	121	0.65
8	127	0.615

one anomaly is detected

a1	scores
4	0.812

ISOLATION FOREST

Isolation Forest pros:

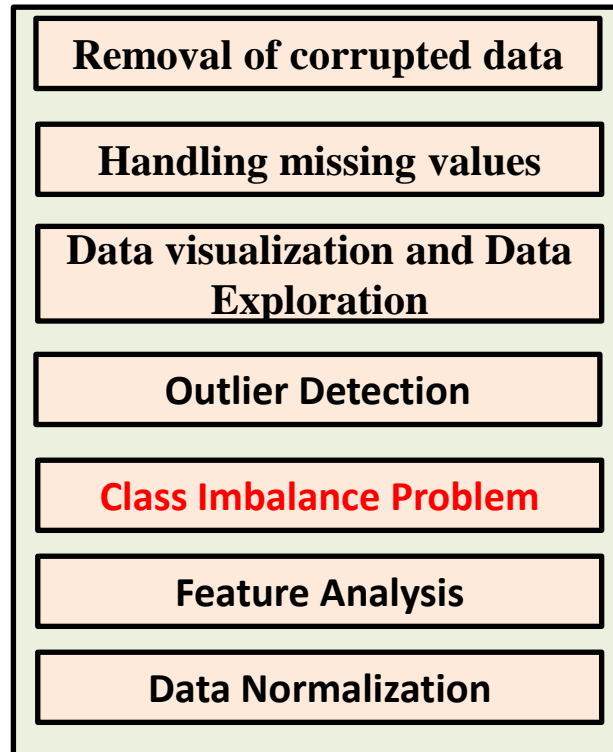
- There is no need of scaling the values in the feature space.
- It is an effective method when value distributions can not be assumed.
- It has few parameters, this makes this method fairly robust and easy to optimize.

Isolation Forest cons:

- Visualizing results is complicated.
- Sometimes, training time can be very long and computationally expensive.

Machine Learning Model

Pre-processing/ Data Preparation

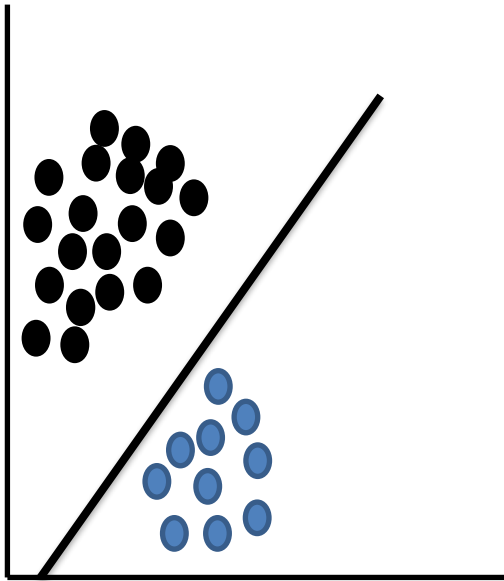


Class imbalance Problem

Class Imbalance: Biases towards majority class(es)

Approaches: Sampling, Algorithmic, Ensembling

Sampling: Undersampling & oversampling



Undersampling: reduces majority class instances: examples RUS, Tomek link based undersampling, Condensed Nearest Neighbor (CNN) undersampling etc.

Oversampling: increases minority class instances: examples ROS, SMOTE, Border-Line SMOTE, Safe level SMOTE, ADASYN etc.

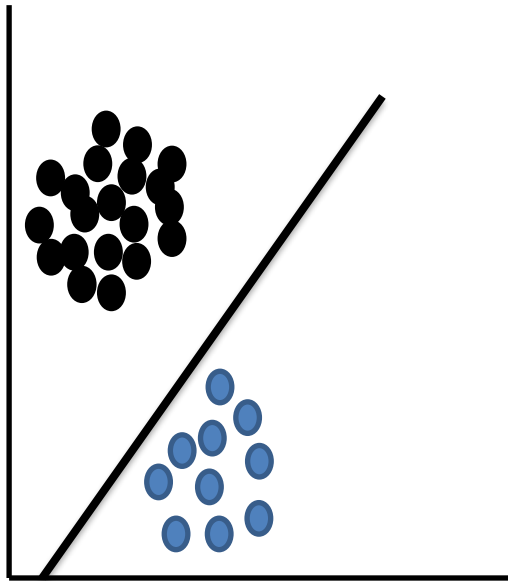
[\[Gaussian SMOTE\]](#) (2017)

[\[kmeans SMOTE\]](#) (2018)

Algorithmic: changes the cost function, higher misclassification cost for minority class cost-sensitive neural network etc.

Ensembling: SMOTEBoost, RUSBoost etc

Class imbalance Problem



RUS: Random Under Sampling technique is random undersampling of the majority class.

This can potentially lead to loss of information about the majority class.

However, in cases where each example of the majority class is near other examples of the same class, this method might yield good results.

$$R = S/L = 0.5$$

$$S = 500$$

$$L = 1500$$

$$L = 1000 \text{ (after undersampling)}$$

Distance Measure

Let $\mathbf{p} = (p_1, p_2)$ and $\mathbf{q} = (q_1, q_2)$ be two points:

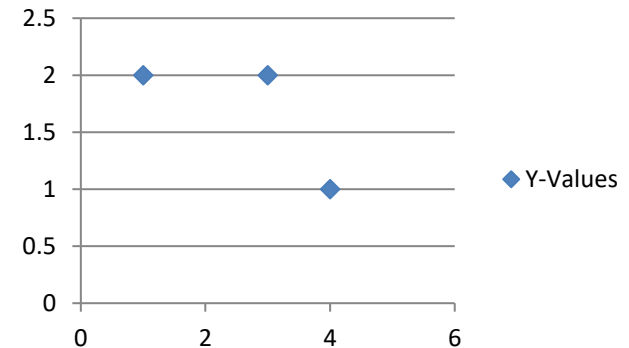
- ✓ City block distance $d(\mathbf{p}, \mathbf{q}) = |p_1 - q_1| + |p_2 - q_2|$
- ✓ Euclidean distance $d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$
- ✓ Minkowski distance $d(\mathbf{p}, \mathbf{q}) = (\sum_{i=1}^M |p_i^n - q_i^n|^r)^{\frac{1}{r}}$

For $r=1$, Minkowski distance = City block distance

For $r=2$, Minkowski distance = Euclidean distance

	M=1	M=2
1st	1	2
2nd	3	2
3rd	4	1

Y-Values



1st = (1, 2)

2nd = (3, 2)

3rd = (4, 1)

$$\text{Dis}(1^{\text{st}}, 2^{\text{nd}}) = |1-3| + |2-2| = 2$$

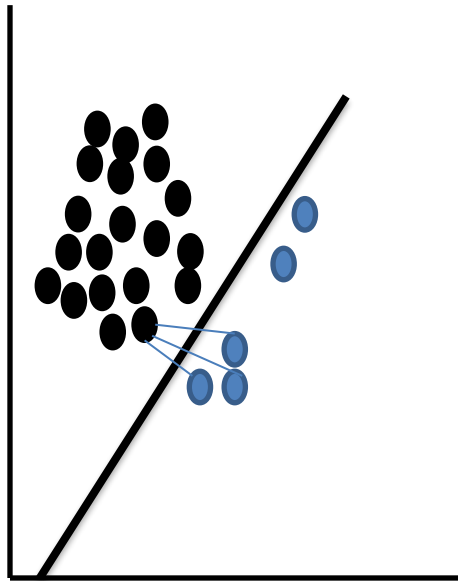
Undersampling

NearMiss-1: Those points from L are retained whose mean distance to the k nearest points in S is lowest.

where k is a tunable hyperparameter

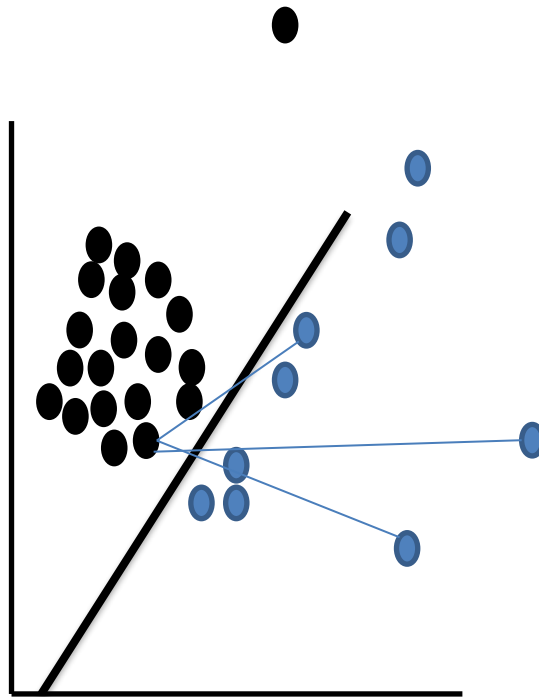
$R = S/L = 0.5$ (As we wish)

$$s1 = \frac{l1 + l2 + l3}{3} = x1$$



8, 6, 4, 5, 1, 2, 3, 7, 9, 10, 12, 16, 17, 18, 20, 19, 15, 14, 13, 11
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

Undersampling



NearMiss-2: keeps those points from L whose mean distance to the k farthest points in S is lowest.

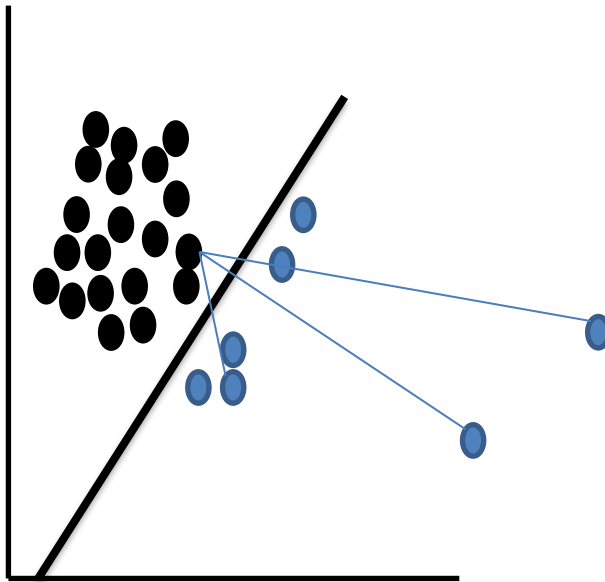
where k is a tunable hyperparameter

$R = S/L = 0.5$ (As we wish)

$$s1 = \frac{f1 + f2 + f3}{3} = x1$$

8, 6, 4, 5, 1, 2, 3, 7, 9, 10, 12, 16, 17, 18, 20, 19, 15, 14, 13, 11
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

Undersampling



NearMiss-2: keeps those points from L whose mean distance to the k farthest points in S is lowest.

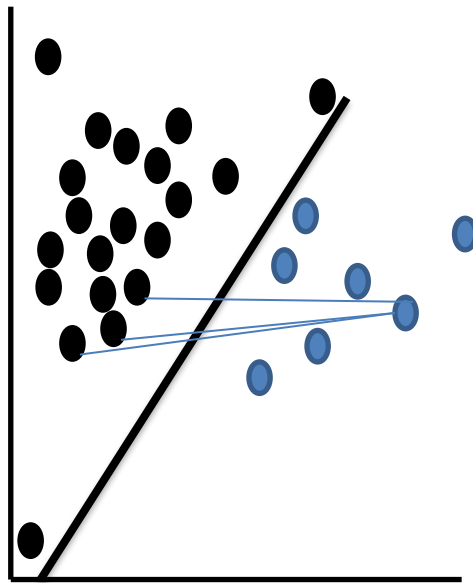
where k is a tunable hyperparameter

$R = S/L = 0.5$ (As we wish)

$$s1 = \frac{f1 + f2 + f3}{3} = x1$$

8, 6, 4, 5, 1, 2, 3, 7, 9, 10, 12, 16, 17, 18, 20, 19, 15, 14, 13, 11
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

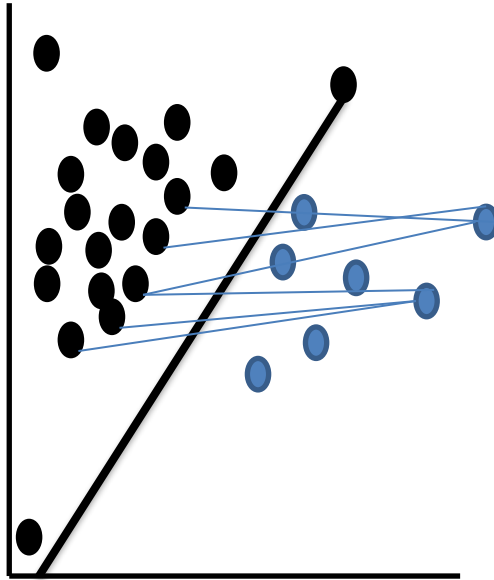
Undersampling



NearMiss-3: selects k nearest neighbors in L for every point in S . In this case, the undersampling ratio is directly controlled by k and is not separately tuned.

where k is a tunable hyperparameter

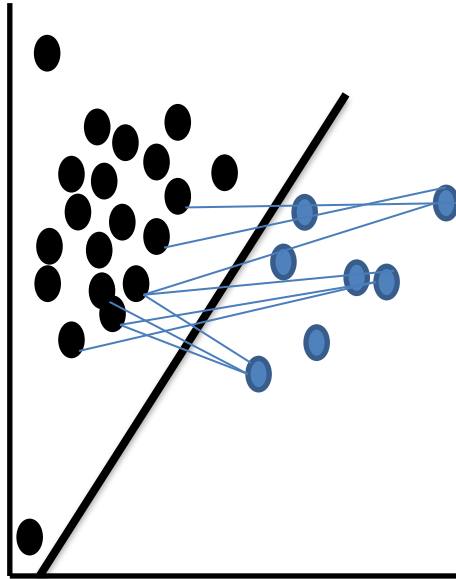
Undersampling



NearMiss-3: selects k nearest neighbors in L for every point in S . In this case, the undersampling ratio is directly controlled by k and is not separately tuned.

where k is a tunable hyperparameter

Undersampling



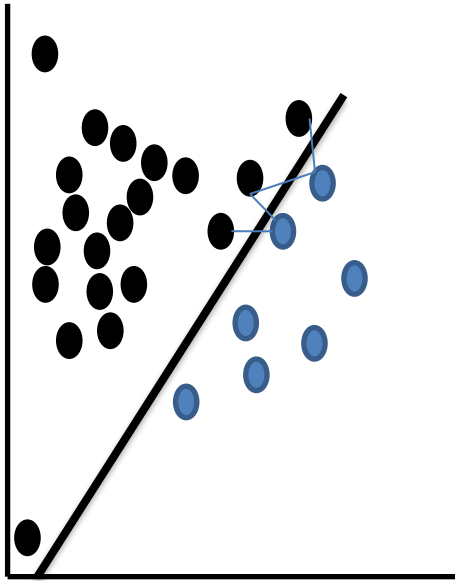
NearMiss-3: selects k nearest neighbors in L for every point in S . In this case, the undersampling ratio is directly controlled by k and is not separately tuned.

where k is a tunable hyperparameter

Undersampling

Condensed Nearest Neighbor (CNN): the goal is to choose a subset U of the training set T such that for every point in T its nearest neighbor in U is of the different class. U can be grown iteratively as follows:

1. Select a random point from T and set $U = \{p\}$.
2. Scan $T - U$ and add to U the first point found whose nearest neighbor in U is of a different class
3. Repeat step 2 until U is maxima

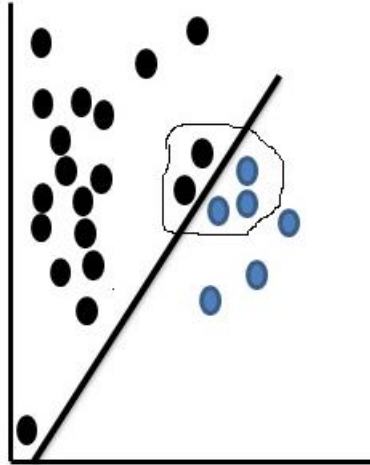


Undersampling via CNN can be slower compared to other methods since it requires many passes over the training data.

Further, because of the randomness involved in the selection of points at each iteration, the subset selected can vary significantly.

A variant of CNN is to only undersample L i.e. retain all points from S but retain only those points in L that belong to U .

Undersampling

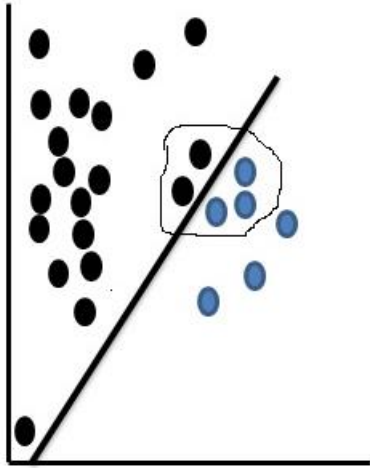


Edited Nearest Neighbor (ENN):

undersampling of the majority class is done by removing points whose class label differs from a majority of its k nearest neighbors.

Say $k=4$

Undersampling



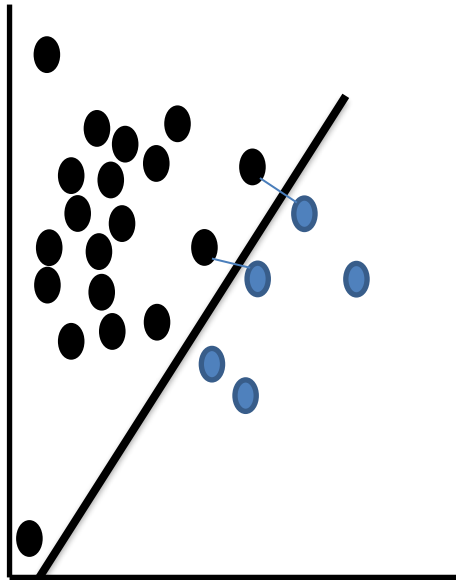
Repeated Edited Nearest Neighbor:

The ENN algorithm is applied successively until ENN can remove no further points.

Say $k=4$

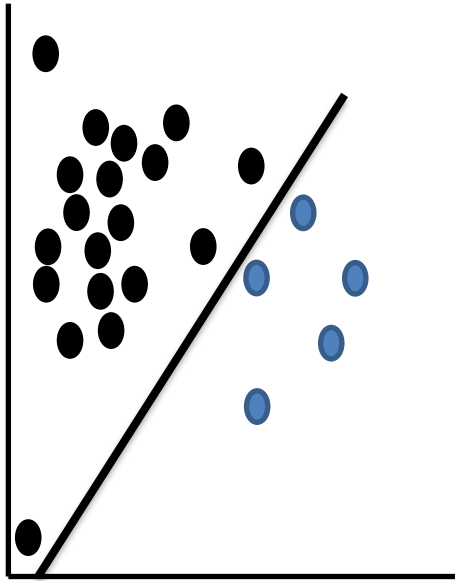
Undersampling

Tomek Link Removal:



- A pair of examples is called a Tomek link if they belong to different classes and are each other's nearest neighbors.
- Undersampling can be done by removing all tomek links from the dataset.
- An alternate method is to only remove the majority class samples that are part of a Tomek link.

Oversampling



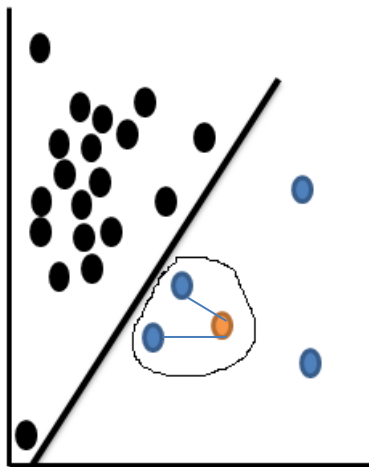
Random oversampling of minority class:

- Points from the minority class may be oversampled randomly.
- This method is prone to overfitting.
- We consider the result of oversampling of S to achieve $r = 0.5$.

Oversampling

SMOTE:

A more sophisticated means for oversampling is Synthetic Minority Oversampling Technique (SMOTE)



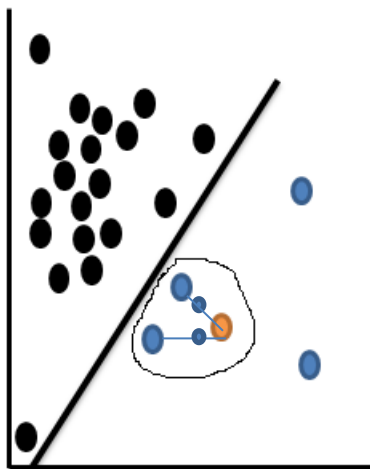
For each point p in S :

1. Compute its k nearest neighbors in S .
 2. Randomly choose $r \leq k$ of the neighbors (with replacement).
 3. Choose a random point along the lines joining p and each of the r selected neighbors.
 4. Add these synthetic points to the dataset with class S .
- We may consider the result of oversampling of S to achieve $r = 0.5$.

Oversampling

SMOTE:

A more sophisticated means for oversampling is Synthetic Minority Oversampling Technique (SMOTE)



For each point p in S :

1. Compute its k nearest neighbors in S .
 2. Randomly choose $r \leq k$ of the neighbors (with replacement).
 3. Choose a random point along the lines joining p and each of the r selected neighbors.
 4. Add these synthetic points to the dataset with class S .
- We may consider the result of oversampling of S to achieve $r = 0.5$.

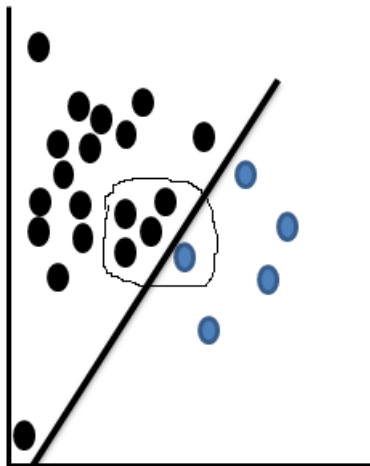
Oversampling

BORDERLINE SMOTE-1:

There are two enhancements of SMOTE, termed borderline SMOTE, which may yield better performance than SMOTE.

For each point p in S :

1. Compute its m nearest neighbors in T . Call this set M_p and let $m_0 = |M_p \cap L|$.
2. If $m_0 = m$, p is a noisy example. Ignore p and continue to the next point.
3. If $0 \leq m_0 \leq m/2$, p is safe. Ignore p and continue to the next point.
4. If $m/2 < m_0 < m$, add p to the set DANGER.
6. For each point d in DANGER, apply the SMOTE algorithm to generate synthetic examples.



- We may consider the result of oversampling of S to achieve $r = 0.5$.

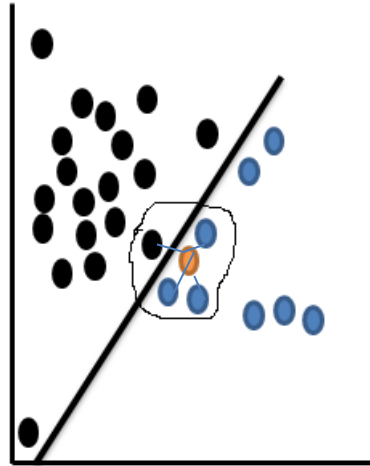
Oversampling

Borderline-SMOTE1:

There are two enhancements of SMOTE, termed borderline SMOTE, which may yield better performance than SMOTE.

For each point p in S :

1. Compute its m nearest neighbors in T .
Call this set M_p and let $m_0 = |M_p \cap L|$.
2. If $m_0 = m$, p is a noisy example. Ignore p and continue to the next point.
3. If $0 \leq m_0 \leq m/2$, p is safe. Ignore p and continue to the next point.
4. If $m/2 < m_0 < m$, add p to the set DANGER.
6. For each point d in DANGER, apply the SMOTE algorithm to generate synthetic examples.



- We may consider the result of oversampling of S to achieve $r = 0.5$.

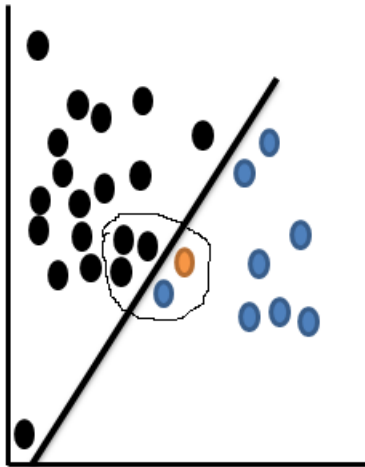
Oversampling

Borderline-SMOTE1:

There are two enhancements of SMOTE, termed borderline SMOTE, which may yield better performance than SMOTE.

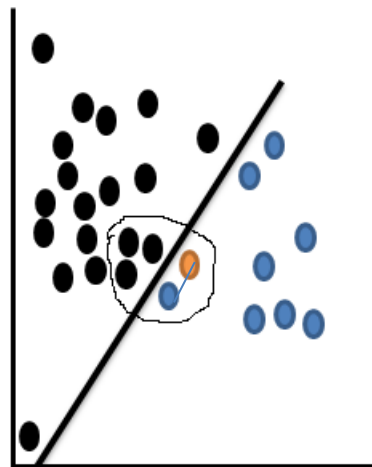
For each point p in S :

1. Compute its m nearest neighbors in T .
Call this set M_p and let $m_0 = |M_p \cap L|$.
2. If $m_0 = m$, p is a noisy example. Ignore p and continue to the next point.
3. If $0 \leq m_0 \leq m/2$, p is safe. Ignore p and continue to the next point.
4. If $m/2 < m_0 < m$, add p to the set DANGER.
6. For each point d in DANGER, apply the SMOTE algorithm to generate synthetic examples.



- We may consider the result of oversampling of S to achieve $r = 0.5$.

Oversampling



Borderline-SMOTE2:

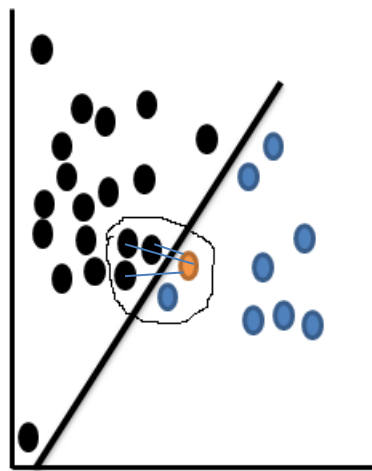
Borderline-SMOTE2 is similar to Borderline-SMOTE1 except in the last step, new synthetic examples are created along the line joining points in DANGER to either their nearest neighbors in S or their nearest neighbors in L.

- We may consider the result of oversampling of S to achieve $r = 0.5$.

Oversampling

Borderline-SMOTE2:

Borderline-SMOTE2 is similar to Borderline-SMOTE1 except in the last step, new synthetic examples are created along the line joining points in DANGER to either their nearest neighbors in S or their nearest neighbors in L.



- We may consider the result of oversampling of S to achieve $r = 0.5$.

Oversampling + Undersampling

Combination methods

Performing a combination of oversampling and undersampling can often yield better results than either in isolation.

SMOTE + Tomek Link Removal

	L	S
Before resampling	6320	680
After resampling	6050	3160

SMOTE + ENN

	L	S
Before resampling	6320	680
After resampling	4894	3160

Treatment of Class imbalance Problem

Algorithmic level solution

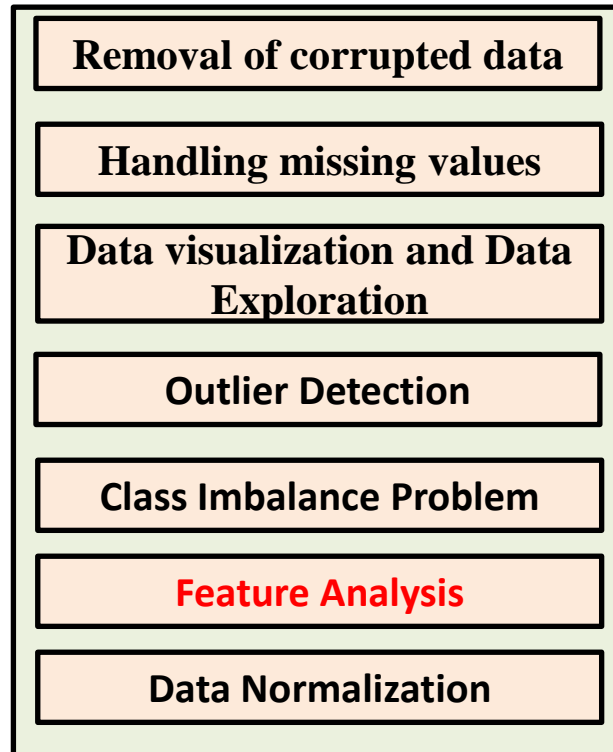
- Cost Sensitive Neural Network

Ensemble methods

- EasyEnsemble
- BalanceCascade
- SMOTEboost
- RUSboost

Machine Learning Model

Pre-processing/ Data Preparation



What is a Feature?

Feature is a quantifiable/measurable characteristic of a phenomenon being observed.

BP	Heart Beat	Weight	Diabetes
120	70	50	Y
125	65	60	Y
130	59	52	N

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
6.7	3.3	5.7	2.5	Virginica
4.9	3	1.4	0.2	Setosa
5.5	2.6	4.4	1.2	Versicolor

Types of feature?

Discrete Feature: can only take certain values. Can be numeric or categorical: the results of rolling 2 dice.

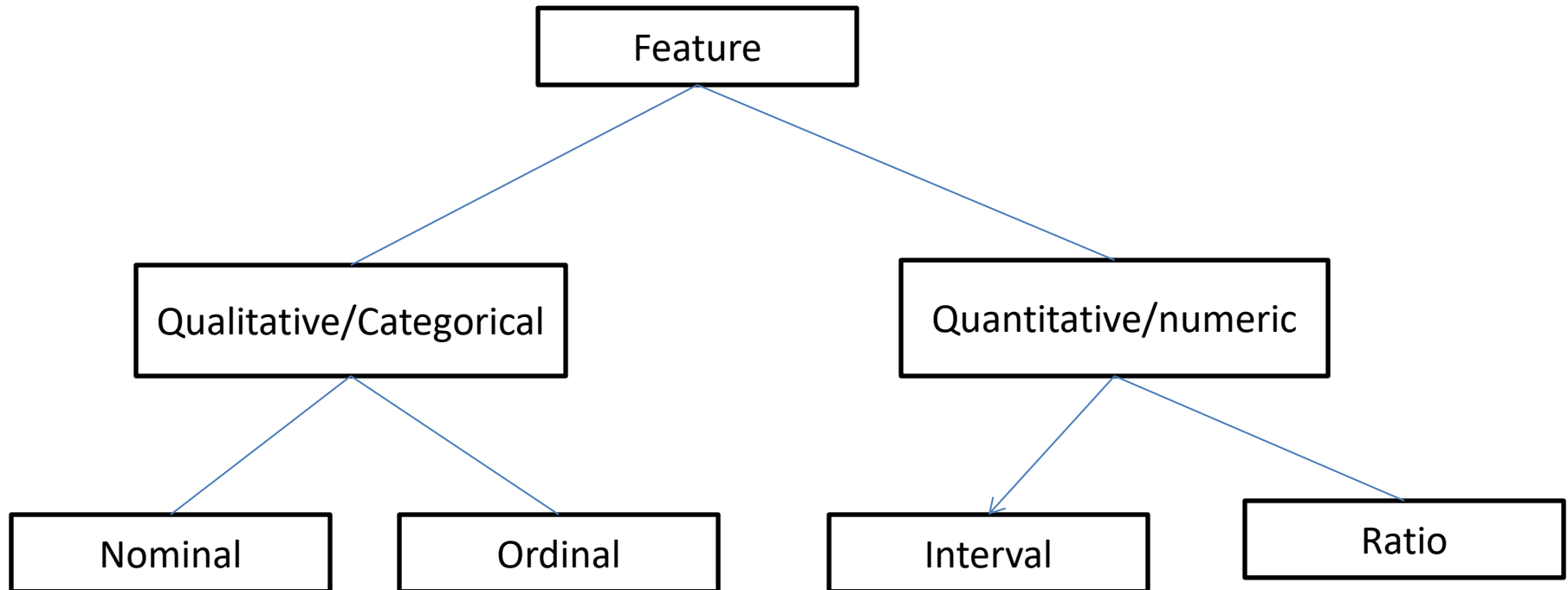
Sepal Length	Sepal Width	Petal Length	Petal Width	Species
big	medium	medium	small	Virginica
big	medium	small	small	Setosa
big	small	medium	small	Versicolor

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
3	2	2	1	Virginica
3	2	1	1	Setosa
3	1	2	1	Versicolor

Continuous Feature: can take any value (within a range)

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
6.7	3.3	5.7	2.5	Virginica
4.9	3	1.4	0.2	Setosa
5.5	2.6	4.4	1.2	Versicolor

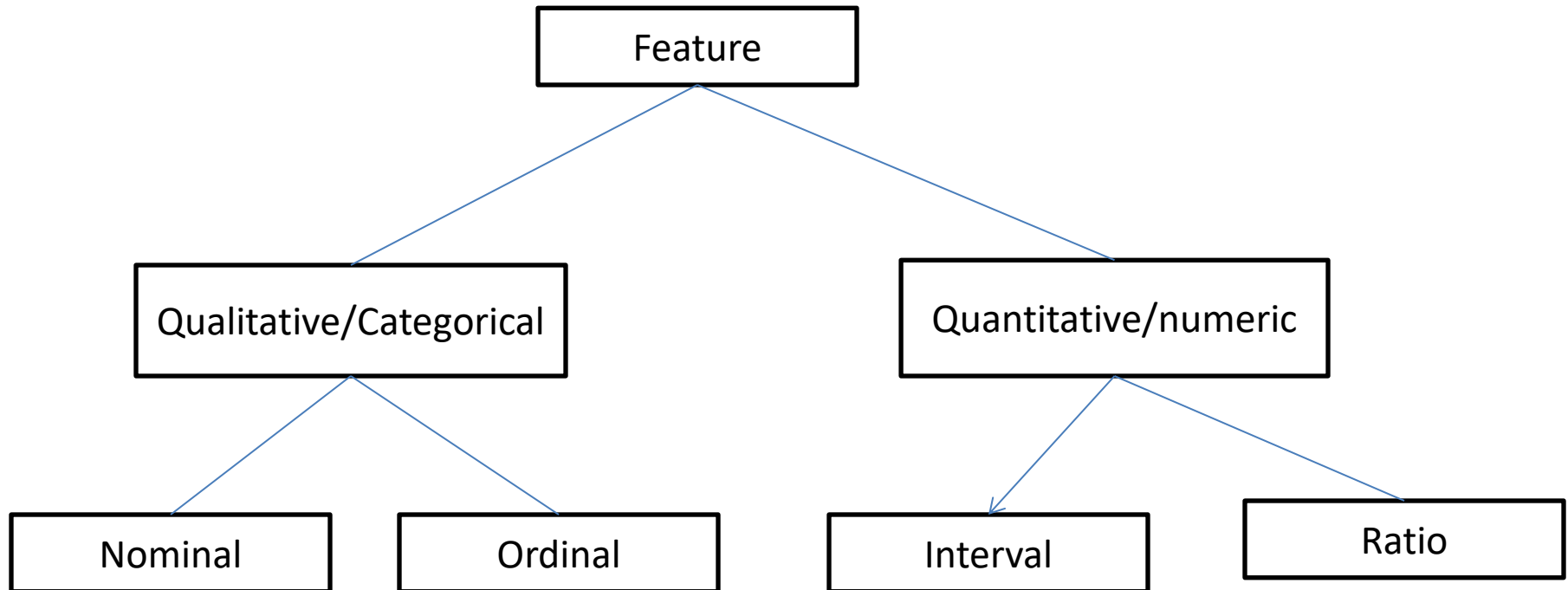
Types of feature?



Nominal Data: the range of values is not ordered in any sense, but simply **named** (hence the nom). Blood groups, gender, etc.

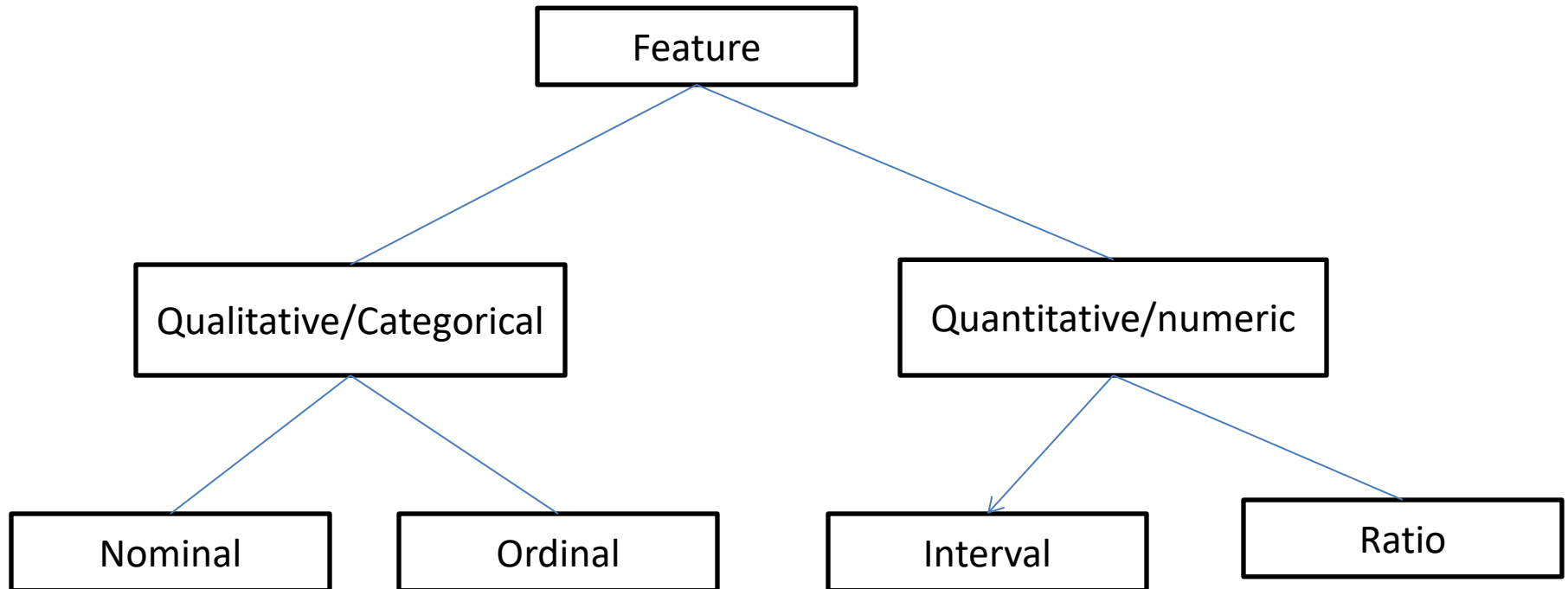
Ordinal Data: the range of values is ordered: **socio economic status** (“low income”, “middle income”, “high income”), education level (“high school”, “BS”, “MS”, “PhD”), income level (“less than 50K”, “50K-100K”, “over 100K”), satisfaction rating (“extremely dislike”, “dislike”, “neutral”, “like”, “extremely like”)

Types of feature?



Interval Data: Interval data is a type of data which is measured along a scale, in which each point is placed at an equal distance (interval) from one another. **The difference between 100 degrees Fahrenheit and 90 degrees Fahrenheit is the same as 60 degrees Fahrenheit and 70 degrees Fahrenheit.** Time of each day, Age, Dates, Voltage.

Types of feature?

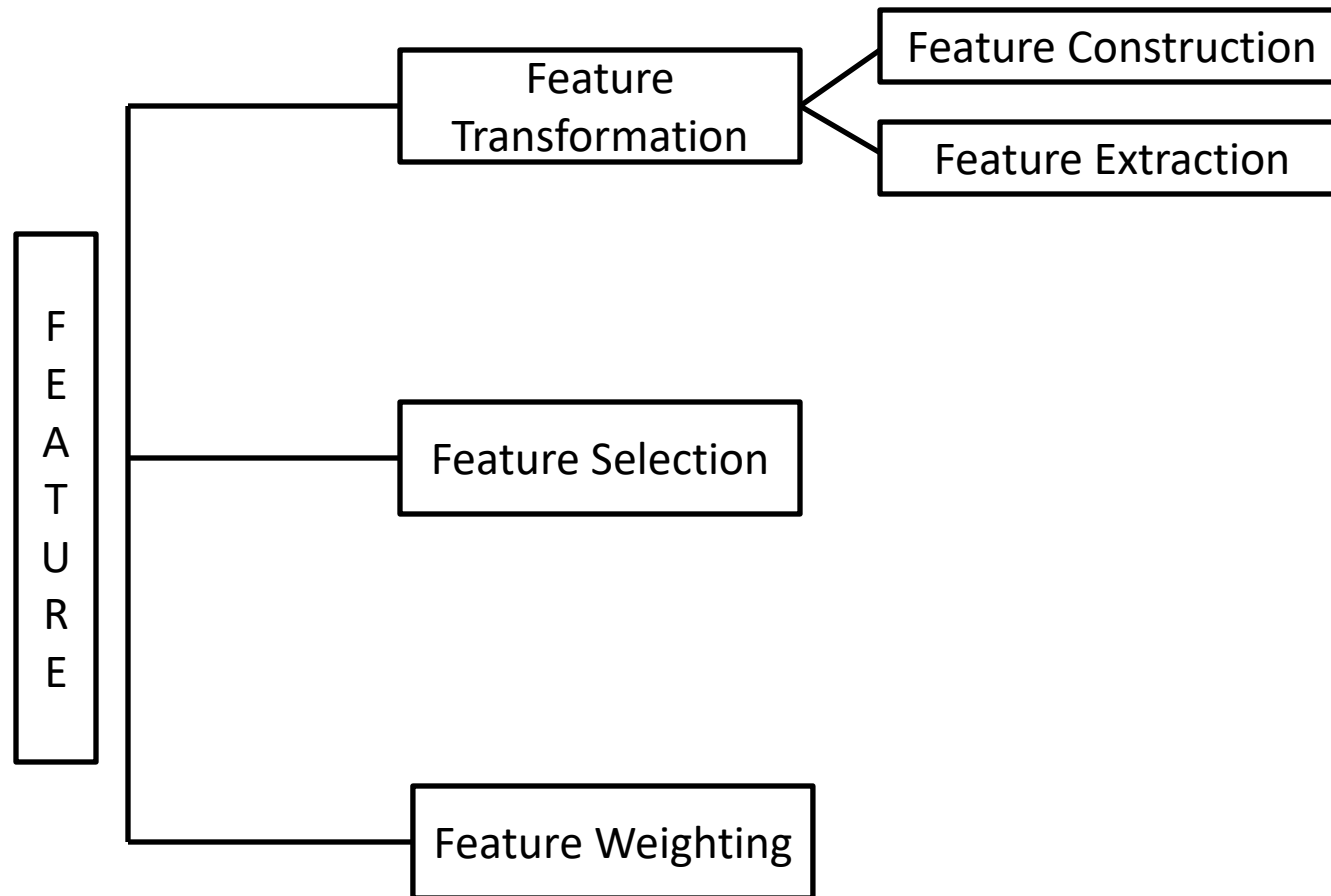


Ratio Data: Unlike interval data, ratio data has a true zero. This basically means that zero is an absolute, below which there are no meaningful values. Speed, age, or weight are all excellent examples since none can have a negative value (you cannot be -10 years old or weigh -160 pounds!)

THE FOUR LEVELS OF MEASUREMENT:

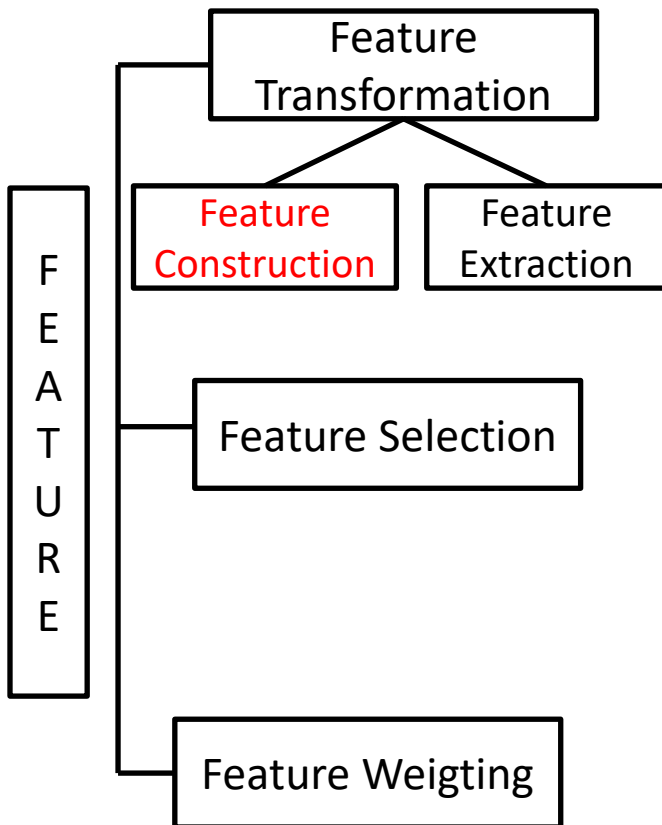
	Nominal	Ordinal	Interval	Ratio
Categorizes and labels variables	✓	✓	✓	✓
Ranks categories in order		✓	✓	✓
Has known, equal intervals			✓	✓
Has a true or meaningful zero				✓

Feature Analysis



Feature Construction

Feature Construction: Generates a new set of more powerful feature from a given set of input feature. Say, there are n features, after feature construction m more feature are added. Now total features= $(n+m)$



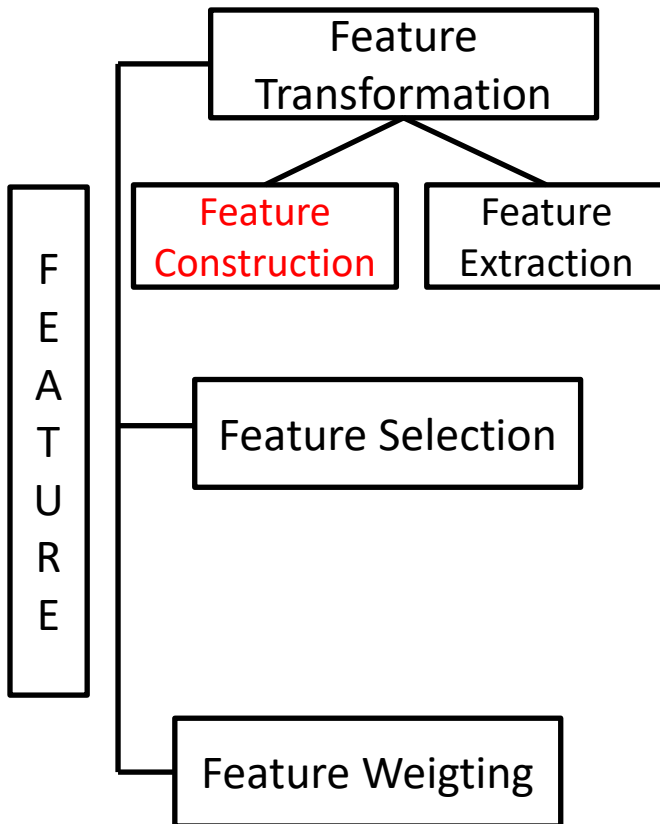
A Length	A Breadth	A price in RS
80	59	2360000
54	45	1215000
78	56	2184000

A Length	A Breadth	A Area	A price in RS
80	59	4720	2360000
54	45	2430	1215000
78	56	4368	2184000

Feature Construction

Need of Feature Construction:

- When features have categorical value and machine learning needs numeric value inputs.
- When features having continuous value and need to be converted to discrete/ordinal values.
- When text-specific feature construction needs to be done.
- Sometimes improves system performance



Feature Construction

Encoding categorical (nominal) variables:

Age	City of origin	Parents athlete	Chance of win
18	City A	Yes	Y
20	City B	No	Y
23	City B	Yes	Y
19	City A	No	N
18	City C	Yes	N

Age	City A	City B	City C	Parents athlete_Y	Win_chance_1
18	1	0	0	1	1
20	0	1	0	0	1
23	0	1	0	1	1
19	1	0	0	0	0
18	0	0	1	1	0

Feature Construction

Encoding categorical (ordinal) variables:

Science	maths	Grade
78	75	B
56	62	C
87	90	A
91	95	A
45	42	D

Science	maths	Grade
78	75	2
56	62	3
87	90	1
91	95	1
45	42	4

Feature Construction

Numeric features (continuous) to categorical features:

A_Area	A_Price
4720	2300000
2430	1200000
4368	2100000
3969	1900000
6142	3000000

A_Area	A_Grade
4720	Medium
2430	Low
4368	Medium
3969	Low
6142	High

A_Area	A_Grade
4720	2
2430	1
4368	2
3969	1
6142	3

Feature Construction

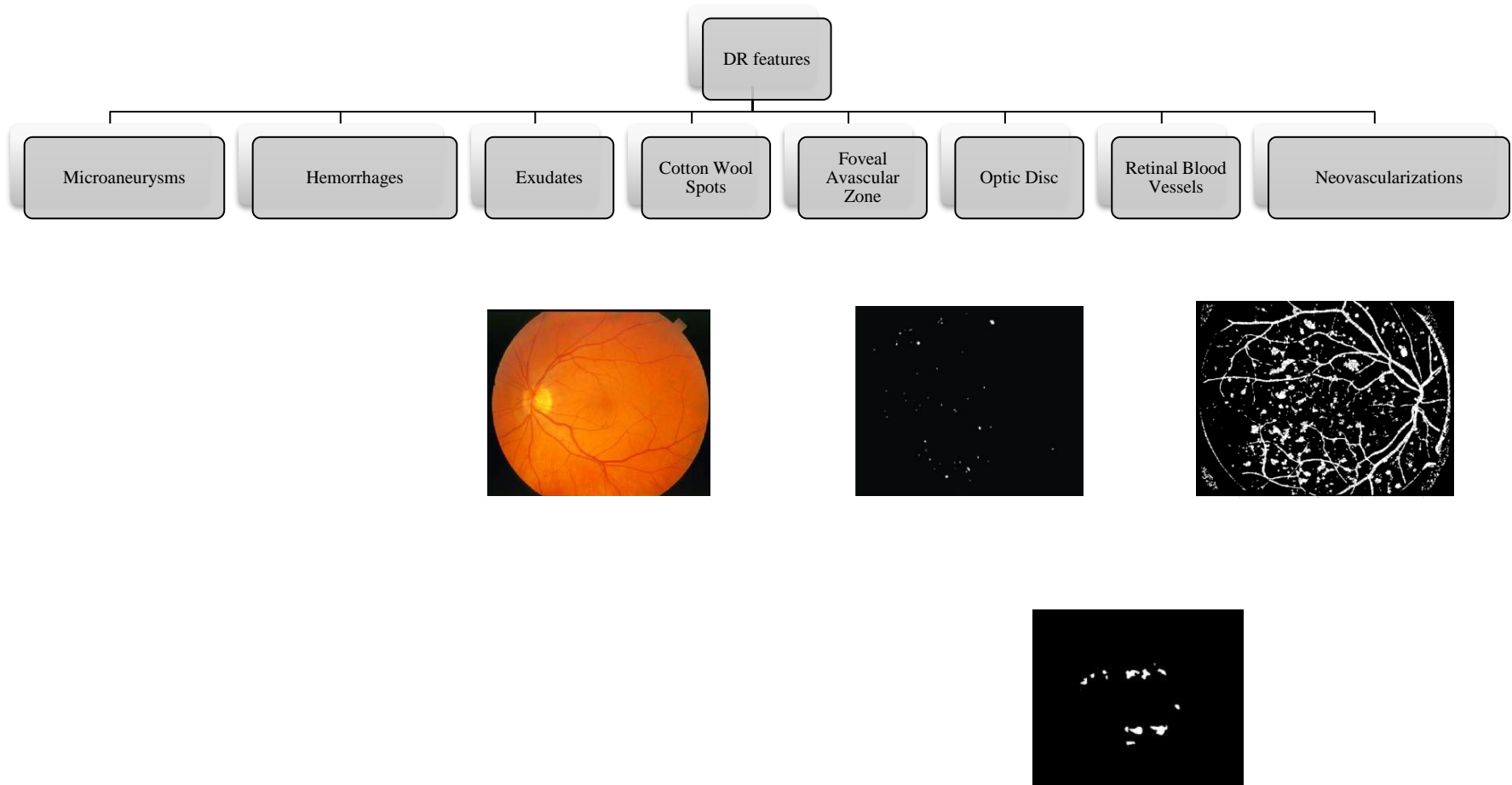
Text Specific feature construction and representation: Vector Space Model (VSM) and Graph Based Model (GBM)

Thrilling	movie	good	attractive	looser	lonely	expectation	bore	class
1	1	1	1	0	0	1	0	1
1	0	1	0	0	0	1	0	1
0	1	1	1	0	0	1	0	1
0	0	0	0	1	1	1	1	0
1	1	0	0	1	1	0	1	0
0	1	0	0	1	1	0	1	0

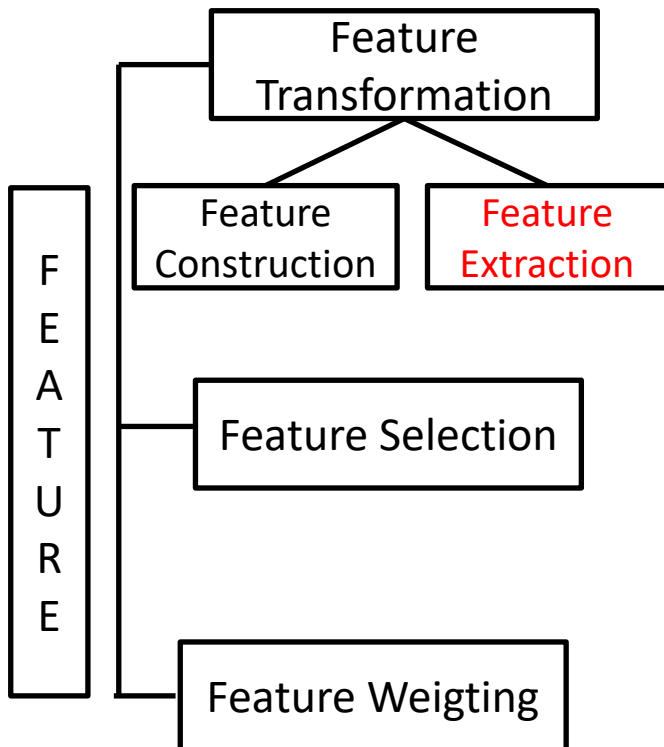
1. The **movie** is **good**, **thrilling**, **attractive**. It is up to my **expectation**-----1 (good)
3. The **movie** is **good** and **attractive** and is up to **expectation** -----1 (good)
5. The **movie** was **looser**, **lonely**. It was **bore**. It was **thrilling**. ----- 0 (bad)

Feature Construction

Image Feature Construction for diabetic retinopathy



Feature Extraction



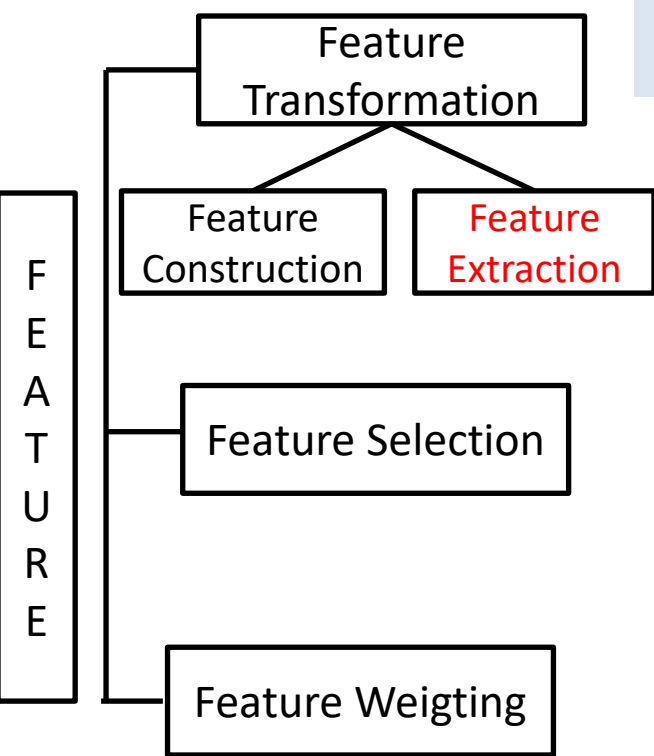
Feature Extraction: New features are created from a combination of original features: some operators:
Boolean Features: Conjunction, Disjunction, Negation etc.

Nominal Features: Cartesian product, M of N etc.

Numerical Features: Min, Max, Addition, Subtraction, Multiplication, Division, Average, Equivalence, Inequality etc.

$$F' = f(F); \quad F1' = K1F1 + k2F2; \quad m < n$$

Feature Extraction



Feat_A	Feat_B	Feat_C	Feat_D
34	34.5	23	233
44	45.56	11	3.44
78	22.59	21	4.5
22	65.22	11	322.3
22	33.8	355	45.2

Feat_1	Feat_2
41.25	185.80
54.20	53.12
43.73	35.79
65.30	264.10
37.02	238.42

$$\text{Feat_1} = 0.3 * \text{Feat_A} + 0.9 * \text{Feat_B}$$

$$\text{Feat_2} = \text{Feat_B} + 0.5 * \text{Feat_C} + 0.6 * \text{Feat_D}$$

Feature Extraction

Some of the popular feature extraction algorithms used in ML are given below:

1. Singular Value Decomposition (SVD)
2. Linear Discriminant Analysis (LDA)
3. Principle Component Analysis (PCA)
4. Fisher's Linear Discriminant (FLD)

Principal Component Analysis

PCA is a useful statistical ML technique that has found application in fields such as face recognition and image compression, and **is a common technique for finding patterns in data of high dimension.**

The other main advantage of PCA is that once we have found these patterns in the data, then we can compress the data, ie. by reducing the number of dimensions, without much loss of information.

Method:

Step 1: Get some data

Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Principal Component Analysis

Step 2: Subtract the mean

		$2.5 - 1.81 = .69$		$2.4 - 1.91 = .49$	
X	Y	(X-X')		(Y-Y')	
2.5	2.4	.69		.49	
0.5	0.7	-1.31		-1.21	
2.2	2.9	.39		.99	
1.9	2.2	.09		.29	
3.1	3.0	1.29		1.09	
2.3	2.7	.49		.79	
2	1.6	.19		-.31	
1	1.1	-.81		-.81	
1.5	1.6	-.31		-.31	
1.1	0.9	-.71		-1.0	
Mean					
X'=1.81	Y'=1.91				

DataAdjust =

Principal Component Analysis

Step 3: Calculate the covariance matrix of x and y (two dimensional data)

$$\begin{pmatrix} cov(x, x) & cov(x, y) \\ cov(y, x) & cov(y, y) \end{pmatrix}$$

If you calculate the covariance between one dimension and itself, you get the variance.

The formula for variance could also be written like this:

$$var(X) = \frac{\sum_1^n (X_i - X') (X_i - X')}{(n - 1)}$$

$$cov(X, Y) = \frac{\sum_1^n (X_i - X') (Y_i - Y')}{(n - 1)}$$

Principal Component Analysis

$$\begin{pmatrix} cov(x, x) & cov(x, y) \\ cov(y, x) & cov(y, y) \end{pmatrix}$$

$$var(X) = \frac{\sum_1^n (X_i - X') (X_i - X')}{(n - 1)}$$

$$cov(X, Y) = \frac{\sum_1^n (X_i - X') (Y_i - Y')}{(n - 1)}$$

$$\begin{pmatrix} 0.616555556 & 0.615444444 \\ 0.615444444 & 0.714322222 \end{pmatrix}$$

$$\begin{aligned} cov(X, Y) &= (.69 * .49) + (-1.31 * - \\ &1.21) + (.39 * .99) + (.09 * .29) + (1.29 * 1.09) + (.49 * .79) + (.19 * - \\ &.31) + (-.81 * -.81) + (-.31 * -.31) + (-.71 * -1.0) = 5.539 \end{aligned}$$

(X-X')	(Y-Y')
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.0
<i>cov(X, Y)</i>	
5.539/9=0.615444444	
Var(X)= 5.549/9=0.616555556	
Var(Y)= 6.4289/9=0.714322222	

Principal Component Analysis

Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

$$\begin{pmatrix} 0.616555556 & 0.615444444 \\ 0.615444444 & 0.714322222 \end{pmatrix}$$

$$eigenvalues = \begin{pmatrix} 0.0490833989 & \\ & 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

Step 5: Choosing components and forming a feature vector

It turns out that the eigenvector with the highest eigenvalue is the principle component of the data set.

The eigenvector with the largest eigenvalue is the one that points the middle of the data.

It is the most significant relationship between the data dimensions.

Principal Component Analysis

Step 5: Choosing components and forming a feature vector

$$eigenvalues = \begin{pmatrix} 0.0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

$$\text{FeatureVector} = eig_1 eig_2 eig_3 eig_4$$

$$= \begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

Principal Component Analysis

Step 6: Deriving the new data set

we simply take the transpose of the vector and multiply it on transpose of *DataAdjust*

$$= \begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

$$FinalData = RowFeatureVector \times RowDataAdjust$$

$$\begin{pmatrix} -.68 & -.74 \end{pmatrix} \times \begin{pmatrix} .69 & -1.31 & .39 & .09 & 1.29 & .49 & .19 & -.81 & -.31 & -.71 \\ .49 & -1.21 & .99 & .29 & 1.09 & .79 & -.31 & -.81 & -.31 & -1.0 \end{pmatrix}$$

$$= (-.68*.69) + (-.74*.49)$$

$$=(-0.8318, 1.7862, -0.9978, -0.2758, -1.6838, -0.9178, 0.1002, 1.1502, 0.4402, 1.2228)$$

Feature Selection

Feature Subset Selection: is the most critical pre-processing in ML and selects a subset which keeps meaningful contribution in a ML task.

Roll. No	Age	Height	Weight
12	12	1.1	23
14	11	1.05	21.6
19	13	1.2	24.7
32	11	1.07	21.3

Age	Height	Weight
12	1.1	23
11	1.05	21.6
13	1.2	24.7
11	1.07	21.3

Feature set= {F1, F2, F3, F4,,FN}

Feature Subset= {F1, F2,.,FM}

Where $M \leq N$

- High-dimensional data: DNA analysis, GIS, Social Networking etc.
- DNA microarray data can have up to 450000 variables(gene)
- Text data is extremely high dimensional data

Feature Selection

Feature Subset Selection: is the most critical pre-processing in ML and selects a subset which keeps meaningful contribution in a ML task.

Roll. No	Age	Height	Weight
12	12	1.1	23
14	11	1.05	21.6
19	13	1.2	24.7
32	11	1.07	21.3

Age	Height	Weight
12	1.1	23
11	1.05	21.6
13	1.2	24.7
11	1.07	21.3

Advantages of Feature Selection:

1. Improve accuracy
2. Improve efficiency by reducing time complexity
3. It helps in the simplification of the model so that it can be easily interpreted by the researchers.

Feature Selection

Jobs of Feature Subset Selection—1. Feature relevance 2. Feature redundancy

2. Feature Relevance: i. Irrelevant feature ii. Relevant feature

Roll. No	Age	Height	Weight
12	12	1.1	23
14	11	1.05	21.6
19	13	1.2	24.7
32	11	1.07	21.3

2. Feature Redundancy:

Age	Height	Weight
12	1.1	23
11	1.05	21.6
13	1.2	24.7
14	1.25	25.3

Measures of feature relevance:

- **Information Gain**
- **Mutual information**
- **Fisher score**
- **Analysis of Variance (ANOVA)**
- **Chi-Square**
- **Dispersion ratio**
- **Relief**

Feature Selection

Measures of feature relevance using **Information Gain**:

NAME of training pattern	Attributes			Class
	HABITS	EAT	FOOTWEAR	
T1	Gabby	Baked	Clogs	Students
T2	Gabby	Roasted	Sandals	Professor
T3	Gabby	Baked	Sandals	Students
T4	Quiet	Fried	Sandals	Professor
T5	Gabby	Fried	Clogs	Students
T6	Quiet	Baked	Sandals	Students
T7	Gabby	Fried	Sandals	Professor
T8	Quiet	Fried	Clogs	Students

Information Gain

- Let C_1, C_2, \dots, C_m be the number of classes where $m > 1$.
- V_0 = Database (set of data)
- $Y(k, 0)$ = number of training patterns of class C_k in the set V_0
- $Z(0)$ = number of patterns in the set V_0

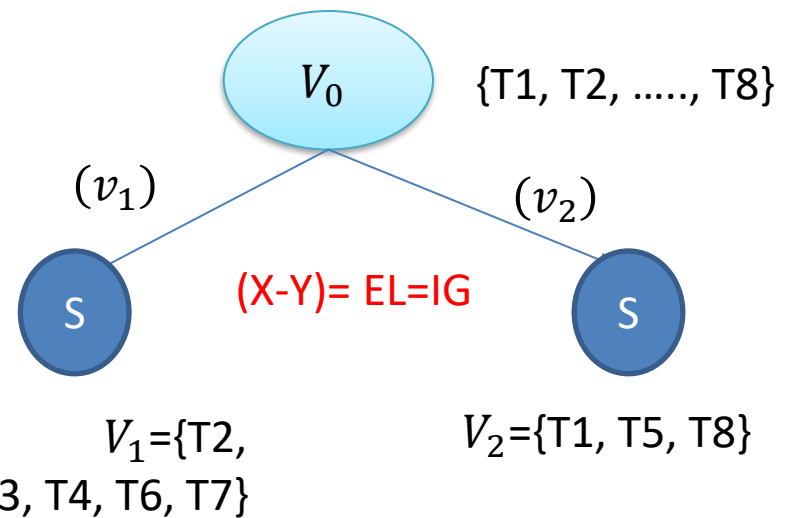
$$Z(0) = \sum_{k=1}^m Y(k, 0)$$

- The probability that a pattern in V_0 belongs to class C_k is

$$\frac{Y(k, 0)}{Z(0)}$$

- The information required to classify a pattern of V_0 into the class C_k , for $1 \leq k \leq m$ is expressed as

$$-\log \frac{Y(k, 0)}{Z(0)}$$

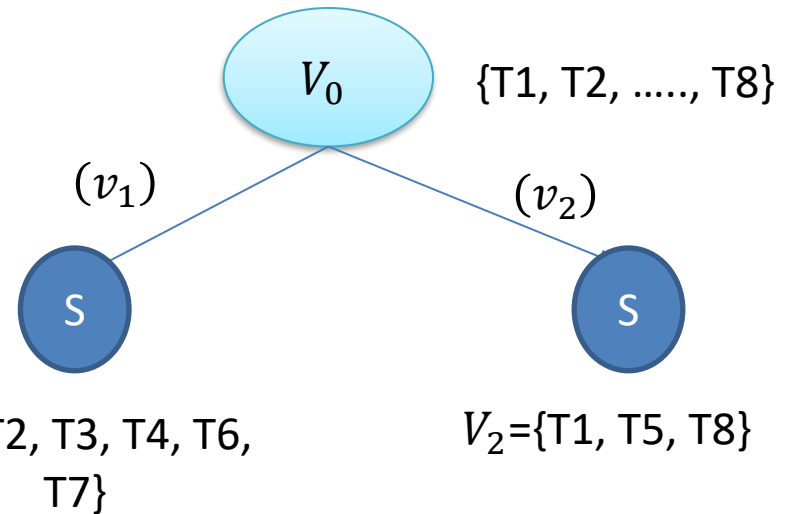


NAME of training pattern	Attributes			Class
	HABITS	EAT	FOOTWEAR	
T1	Gabby	Baked	Clogs	S
T2	Gabby	Roasted	Sandals	P
T3	Gabby	Baked	Sandals	S
T4	Quiet	Fried	Sandals	P
T5	Gabby	Fried	Clogs	S
T6	Quiet	Baked	Sandals	S
T7	Gabby	Fried	Sandals	P
T8	Quiet	Fried	Clogs	S

Information Gain

- The weighted average information required to classify a pattern of set V_0 into one of the m classes is expressed as

$$I(V_0) = \sum_{k=1}^m \frac{Y(k, 0)}{Z(0)} \left(-\log \frac{Y(k, 0)}{Z(0)} \right)$$



$I(V_0)$ is called the entropy of the set V_0 . $V_1 = \{T2, T3, T4, T6, T7\}$ $V_2 = \{T1, T5, T8\}$

- Similarly for the set V_j ,

$$I(V_j) = \sum_{k=1}^m \frac{Y(k, j)}{Z(j)} \left(-\log \frac{Y(k, j)}{Z(j)} \right)$$

- The weighted average information required to classify a pattern into one of class k in set V_0 after it has been split by the attribute A into sets V_1 to V_n is given by

$$I_A(V_0) = \sum_{j=1}^n \frac{Z(j)}{Z(0)} I(V_j)$$

$$I_A(V_0) = \sum_{j=1}^n \frac{Z(j)}{Z(0)} \sum_{k=1}^m \frac{Y(k,j)}{Z(j)} \left(-\log \frac{Y(k,j)}{Z(j)} \right)$$

$I_A(V_0)$ is called the entropy of the attribute A for the set V_0

- The gain in information caused by attribute A splitting set V_0 into sets V_1 to V_n is

$$g_A(V_0) = I(V_0) - I_A(V_0)$$

NAME of training pattern	Attributes			Class
	HABITS	EAT	FOOTWEAR	
T1	Gabby	Baked	Clogs	S
T2	Gabby	Roasted	Sandals	P
T3	Gabby	Baked	Sandals	S
T4	Quiet	Fried	Sandals	P
T5	Gabby	Fried	Clogs	S
T6	Quiet	Baked	Sandals	S
T7	Gabby	Fried	Sandals	P
T8	Quiet	Fried	Clogs	S

Example

Evaluate the entropy $I(V_0)$ of the Professor-student training set.

NAME of training pattern	Attributes			Class
	HABITS	EAT	FOOTWEAR	
T1	Gabby	Baked	Clogs	S
T2	Gabby	Roasted	Sandals	P
T3	Gabby	Baked	Sandals	S
T4	Quiet	Fried	Sandals	P
T5	Gabby	Fried	Clogs	S
T6	Quiet	Baked	Sandals	S
T7	Gabby	Fried	Sandals	P
T8	Quiet	Fried	Clogs	S

- S and P are the two classes, hence $m=2$.
- The training set $V_0 = \{T1, T2, T3, \dots, T8\}$.
- $Y(1,0)=3$ (number of patterns in V_0 of class P)
- $Y(2,0)=5$ (number of patterns in V_0 of class S)

Therefore,

$$\begin{aligned}
 I(V_0) &= - \sum_{k=1}^m \frac{Y(k,0)}{Z(0)} \log \frac{Y(k,0)}{Z(0)} \\
 &= - \sum_{k=1}^2 \frac{Y(k,0)}{Z(0)} \log \frac{Y(k,0)}{Z(0)} \\
 &= - \frac{3}{8} \log \frac{3}{8} - \frac{5}{8} \log \frac{5}{8} \\
 &= 0.9544
 \end{aligned}$$

Example

Information gain of HABIT

- S and P are the two classes, hence $m=2$.
- The training set $V_0 = \{T1, T2, T3, \dots, T8\}$.
- $Y(1,0)=3$ (number of patterns in V_0 of class P)
- $Y(2,0)=5$ (number of patterns in V_0 of class S)
- $Z(0)=8$ (number of patterns in V_0)
- $I(V_0)=0.9544$

$$\begin{aligned}
 I_{Habit}(V_0) &= - \sum_{j=1}^n \frac{Z(j)}{Z(0)} \sum_{k=1}^m \frac{Y(k,j)}{Z(j)} \log \frac{Y(k,j)}{Z(j)} \\
 &= [5/8 \{ (2/5)(-\log 2/5) + (3/5)(-\log 3/5) \} \\
 &\quad + 3/8 \{ (1/3)(-\log 1/3) + (2/3)(-\log 2/3) \}] \\
 &= 0.9499
 \end{aligned}$$

- $V_1 = \{T1, T2, T3, T5, T7\}$ at node y_1 , where HABIT=gabby.
- $Y(1,1)=2$ (number of patterns in V_1 of class P)
- $Y(2,1)=3$ (number of patterns in V_1 of class S)
- $Z(1)=5$ (number of patterns in V_1)

$$\begin{aligned}
 G_{Habit}(V_0) &= \frac{I(V_0) - I_{Habit}(V_0)}{1} \\
 &= (0.9544 - 0.9499) \\
 &= 0.0100
 \end{aligned}$$

- $V_2 = \{T4, T6, T8\}$ at node y_2 , where HABIT=quiet.
- $Y(1,2)=1$ (number of patterns in V_2 of class P)
- $Y(2,2)=2$ (number of patterns in V_2 of class S)
- $Z(2)=3$ (number of patterns in V_2)

Example

Information gain of EATS

- S and P are the two classes, hence $m=2$.
- The training set $V_0 = \{T1, T2, T3, \dots, T8\}$.
- $Y(1,0)=3$ (number of patterns in V_0 of class P)
- $Y(2,0)=5$ (number of patterns in V_0 of class S)
- $Z(0)=8$ (number of patterns in V_0)
- $I(V_0)=0.9544$
- $V_1 = \{T1, T3, T6\}$ at node y_1 , where EATS=baked.
- $Y(1,1)=0$ (number of patterns in V_1 of class P)
- $Y(2,1)=3$ (number of patterns in V_1 of class S)
- $Z(1)=3$ (number of patterns in V_1)
- $V_2 = \{T4, T5, T7, T8\}$ at node y_2 , where EATS=fried.
- $Y(1,2)=2$ (number of patterns in V_2 of class P)
- $Y(2,2)=2$ (number of patterns in V_2 of class S)
- $Z(2)=4$ (number of patterns in V_2)
- $V_3 = \{T2\}$ at node y_3 , where EATS=roasted.
- $Y(1,3)=1$ (number of patterns in V_3 of class P)
- $Y(2,3)=0$ (number of patterns in V_3 of class S)
- $Z(3)=1$ (number of patterns in V_3)

$$I_{Eats}(V_0) = - \sum_{j=1}^n \frac{Z(j)}{Z(0)} \sum_{k=1}^m \frac{Y(k,j)}{Z(j)} \log \frac{Y(k,j)}{Z(j)}$$

$=0.5$

$$G_{Eats}(V_0) = \frac{I(V_0) - I_{Eats}(V_0)}{= (0.9544 - 0.5)}$$

$= 0.4544$

Example

information gain of FOOTWEAR

- S and P are the two classes, hence $m=2$.
- The training set $V_0 = \{T1, T2, T3, \dots, T8\}$.
- $Y(1,0)=3$ (number of patterns in V_0 of class P)
- $Y(2,0)=5$ (number of patterns in V_0 of class S)
- $Z(0)=8$ (number of patterns in V_0)
- $I(V_0)=0.9544$
- $V_1 = \{T1, T5, T8\}$ at node y_1 , where FOOTWEAR=clogs.
- $Y(1,1)=0$ (number of patterns in V_1 of class P)
- $Y(2,1)=3$ (number of patterns in V_1 of class S)
- $Z(1)=3$ (number of patterns in V_1)
- $V_2 = \{T2, T3, T4, T6, T7\}$ at node y_2 , where FOOTWEAR=sandals.
- $Y(1,2)=3$ (number of patterns in V_2 of class P)
- $Y(2,2)=2$ (number of patterns in V_2 of class S)
- $Z(2)=5$ (number of patterns in V_2)

Therefore,

$$I_{Footwear}(V_0) = - \sum_{j=1}^n \frac{Z(j)}{Z(0)} \sum_{k=1}^m \frac{Y(k,j)}{Z(j)} \log \frac{Y(k,j)}{Z(j)}$$

$= 0.6066$

$$G_{Footwear}(V_0) = \frac{I(V_0) - I_{Footwear}(V_0)}{1} = (0.9544 - 0.6066) = 0.3478$$

NAME of training pattern	Attributes			Class
	HABIT S	EAT	FOOT WEAR	
T1	Gabby	Bake d	Clogs	S
T2	Gabby	Roas ted	Sandal s	P
T3	Gabby	Bake d	Sandal s	S
T4	Quiet	Fried	Sandal s	P
T5	Gabby	Fried	Clogs	S
T6	Quiet	Bake d	Sandal s	S
T7	Gabby	Fried	Sandal s	P
T8	Quiet	Fried	Clogs	S

Example

$$G_{HAbit}(V_0)= 0.0100$$

$$G_{Eats}(V_0)= 0.4544$$

$$G_{Footwear}(V_0)= 0.3478$$

Measures of feature redundancy:

1. Similarity-based measure

- i. Pearson Correlation Coefficient
- ii. Spearman's Correlation
- iii. Kendall's Tau
- iv. Jaccard Index/Coefficient
- v. Simple Matching Coefficient (SMC)
- vi. Cosine Similarity

2. Distance-based measure

- i. Euclidean distance
- ii. Manhattan distance

Correlation-based measure: Measures linear dependency between two random variables.

Feature Redundancy

Pearson Correlation Coefficient: measures linear dependency between two random variables

$$\alpha = \frac{\text{cov}(F1, F2)}{\sqrt{\text{var}(F1) \cdot \text{var}(F2)}}$$

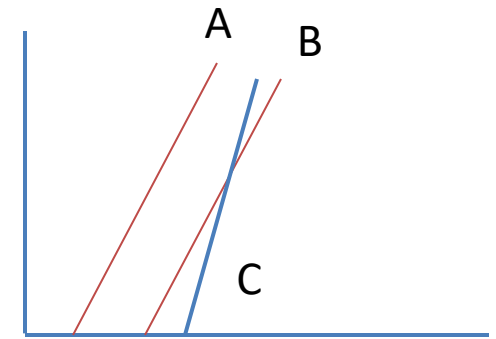
$$\text{cov}(F1, F2) = \sum \{(F1_i - F1') \cdot (F2_i - F2')\} / (n - 1)$$

$$\text{var}(F1) = \sum (F1_i - F1')^2 / (n - 1), \text{ where } F1' = \frac{1}{n} \sum F1_i$$

$$\text{var}(F2) = \sum (F2_i - F2')^2 / (n - 1), \text{ where } F2' = \frac{1}{n} \sum F2_i$$

It ranges between +1 and -1

Covariance is a measure of the relationship between two random variables. The metric evaluates how much – to what extent – the variables change together. In other words, it is essentially a measure of the variance between two variables



Feature Redundancy

Spearman's Correlation Coefficient: measures linear dependency between two random variables. It uses rank of each value. Data are represented as

$$\mathbf{x} = \mathbf{x}^r$$

$$\mathbf{y} = \mathbf{y}^r$$

If the raw data are [0, -5, 4, 7], the ranked values will be [2, 1, 3, 4].

$$S(F1, F2) = \frac{cov(F1, F2)}{\sqrt{var(F1) \cdot var(F2)}}$$

$$cov(F1, F2) = \sum \{(F1_i^r - F1'^r) \cdot (F2_i^r - F2'^r)\} / (n-1)$$

$$var(F1) = \sum (F1_i^r - F1'^r)^2 / (n-1) \quad \text{where } F1'^r = \frac{1}{n} \sum F1_i^r$$

$$var(F2) = \sum (F2_i^r - F2'^r)^2 / (n-1), \quad \text{where } F2'^r = \frac{1}{n} \sum F2_i^r$$

It ranges between +1 and -1

If $S > P$ it means that we have a monotonic relationship, not a linear relationship.

Feature Redundancy

Kendall's Tau: measures linear dependency between two random variables. It uses rank of each value. Kendall's Tau has smaller variability when using larger sample sizes. However, Spearman's measure is more computationally efficient, as Kendall's Tau is $O(n^2)$ and Spearman's correlation is $O(n\log(n))$.

Data are represented as

$$\mathbf{x} = \mathbf{x}^r$$
$$\mathbf{y} = \mathbf{y}^r$$

If the raw data are $[0, -5, 4, 7]$, the ranked values will be $[2, 1, 3, 4]$.

It ranges between +1 and -1

Feature Redundancy

Kendall's Tau = (C – D) / (C + D)

Feature1: 1 2 3 4 5 6 7 8 9 10 11 12

Feature 2: 1 2 4 3 6 5 8 7 10 9 12 11

Step1: Make a table of rankings. The rankings for Feature 1 should be in ascending order

Feature1: 1 2 3 4 **12** 6 7 8 9 10 11 **5**

Feature 2: 1 2 4 3 6 5 8 7 10 9 12 11

Feature1: 1 2 3 4 **5** 6 7 8 9 10 11 **12**

Feature 2: 1 2 4 3 **11** 5 8 7 10 9 12 **6**

Feature1	Feature 2
1	1
2	2
3	4
4	3
5	6
6	5
7	8
8	7
9	10
10	9
11	12
12	11

Feature Redundancy

Step 2: Count the number of concordant pairs, using the second column. Concordant pairs are how many larger ranks are below a certain rank. For example, the first rank in the second Feature's column is a "1", so all 11 ranks below it are larger.

Feature 1	Feature 2	Concordant	Discordant
1	1	11	
2	2	10	
3	4	8	
4	3	8	
5	6	6	
6	5	6	
7	8	4	
8	7	4	
9	10	2	
10	9	2	
11	12	0	
12	11		

Feature Redundancy

Step 3: Count the number of discordant pairs and insert them into the next column. The number of discordant pairs is similar to Step 2, only you're looking for smaller ranks, not larger ones.

Feature 1	Feature 2	Concordant	Discordant
1	1	11	0
2	2	10	0
3	4	8	1
4	3	8	0
5	6	6	1
6	5	6	0
7	8	4	1
8	7	4	0
9	10	2	1
10	9	2	0
11	12	0	1
12	11		

Feature Redundancy

Step 4: **Sum the values in the two columns:**

Step 5: **Insert the totals into the formula:**
Kendall's Tau = $(C - D / C + D) = (61 - 5) / (61 + 5)$
= 56 / 66 = .85.

The Tau coefficient is .85, suggesting a strong relationship between the rankings.

Feature 1	Feature 2	Concordant	Discordant
1	1	11	0
2	2	10	0
3	4	8	1
4	3	8	0
5	6	6	1
6	5	6	0
7	8	4	1
8	7	4	0
9	10	2	1
10	9	2	0
11	12	0	1
12	11		
Total		61	5

Feature Redundancy

Perfect Correlation

$\text{Tau} = (66 - 0) / (66 + 0) = 1$,
which is (as we expect) perfect
agreement.

Feature 1	Feature 2	Concor dant	Discord ant
1	1	11	0
2	2	10	0
3	3	9	0
4	4	8	0
5	5	7	0
6	6	6	0
7	7	5	0
8	8	4	0
9	9	3	0
10	10	2	0
11	11	1	0
12	12		
Total		66	0