Data Preparation

Fátima Rodrigues
mfc@isep.ipp.pt
Departamento de Engenharia Informática (DEI/ISEP)

Motivation

In real applications the data tend to be inconsistent, incomplete and/or wrong

- What happens when the data is not correct?
- Can the knowledge extracted from the data be trusted?
- Obstacles to knowledge discovery: **poor data**

GIGO law: Garbage In, Garbage Out

Data Preparation

- Set of steps that may be necessary to carry out before any further analysis takes place on the available data
- It is estimated that data preparation takes 70-80% of all development effort of a data mining project
- Good data preparation is key to produce valid and reliable models

Data Preparation - Goals

- Understanding the nature of the data
- Solve problems inherent to the data
- Adapting the data according to the Data Mining algorithms
- Provide more meaningful data analysis and extract knowledge with meaning
- Know what useful information exists in a particular data set, so when random samples are formed from it, the information is preserved

Major Tasks in Data Preparation

Data Selection

- Creates the appropriate set of data to explore

Data Cleaning

- Handling missing values, smooth noisy data, identify or remove outliers and resolve inconsistencies

Data Transformation

- Data conversion, data scaling/normalisation

Data Reduction

- Obtains reduced representation in volume but produces the same or similar analytical results

Data Selection

Sampling

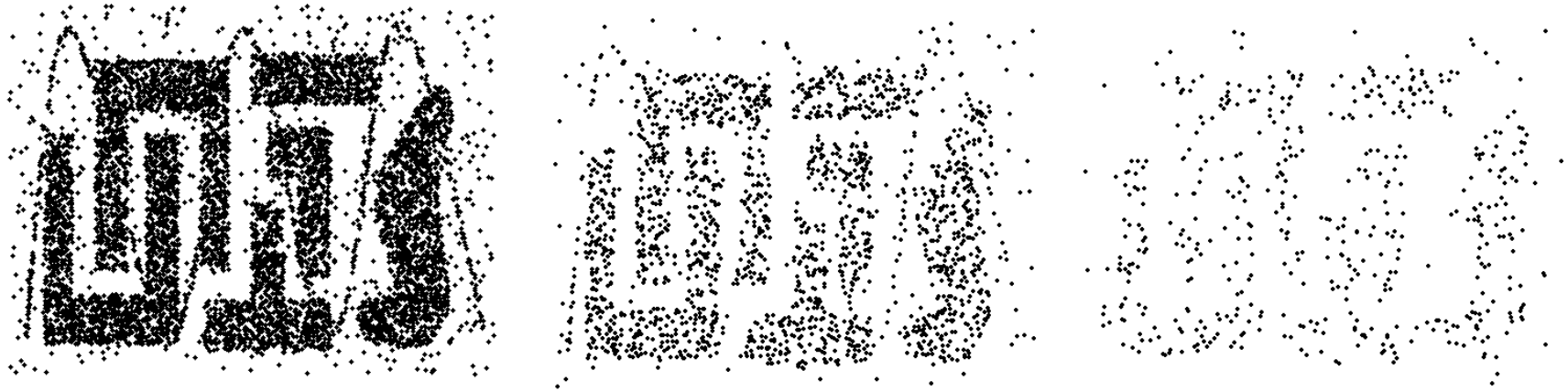
- Sampling is the main technique employed for data selection
  - It is often used for both the preliminary investigation of the data and the final data analysis

- Sampling is used in data mining because processing the entire set of data of interest may be too expensive or time-consuming

Sampling …

The key principle for effective sampling is the following:

- using a sample will work almost as well as using the entire data set, **if the sample is representative**
- a sample is representative if it has approximately the same properties (of interest) as the original set of data



8000 points

2000 Points

500 Points

Types of Sampling

- **Simple random sampling**
  There is an equal probability of selecting any particular item

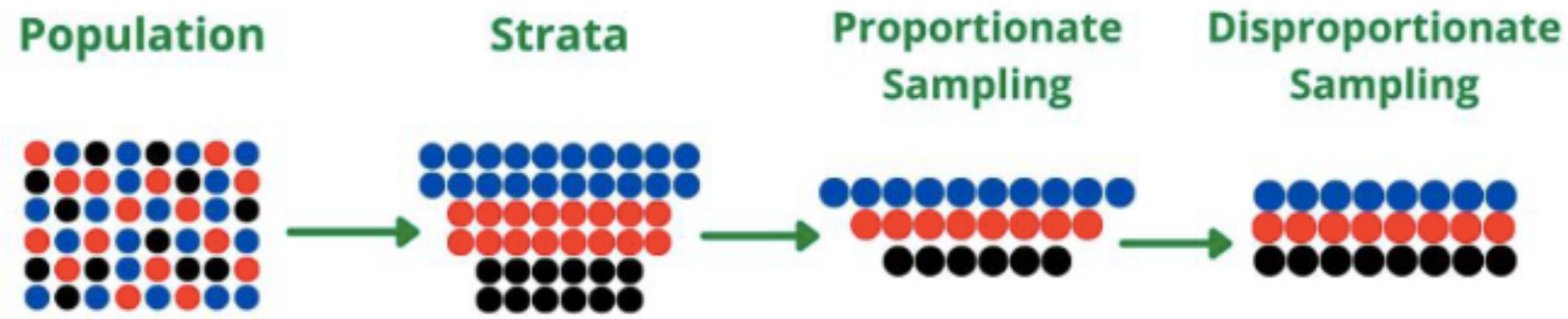
- **Sampling without replacement**
  As each item is selected, it is removed from the population

- **Sampling with replacement**
  Objects are not removed from the population as they are selected for the sample. The same object can be picked up more than once

- **Stratified sampling**
  Split the data into several partitions; then draw random samples from each partition

Stratified random sampling

The sample is created preserving the target variable's proportion from the original population



Data Cleaning

Missing Values

- Missing data can appear in several forms:
  - <empty field> “0” “.” “999” “NA” …
  - Standardize missing value code(s)

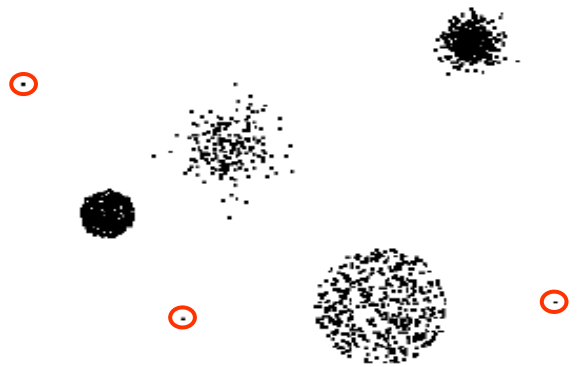
Dealing with Missing Values

if in a small number:

- ignore records with missing values
- manually fill in the missing values

- Fill in with common values:
  - mode for categorical data
  - median for ordered values
- Use the most likely value according to a model based on the values of other attributes:
  - Linear Regression
  - Instance-Based Learning
  - Mean of the k-nearest neighbours for numerical data

Outliers



- They have distinct characteristics from most of the other objects in the dataset, but **they are valid**

- **Isolated values** in some applications often indicate critical situations with high costs, requiring preventive or corrective actions. Example, fraud detection applications

- Other applications may view isolated values as **noise or outliers**, often treating them as errors and requiring their removal from the analysis

- Outliers don't affect decision trees and support vector machines, but they can have a significant impact on distance calculation-based models

Methods to Detect Outliers

Univariate methods: focus on examining each variable individually for outliers

- **Z-Score**
  - calculate the z-score for each data point and identify those with z-scores exceeding a certain threshold

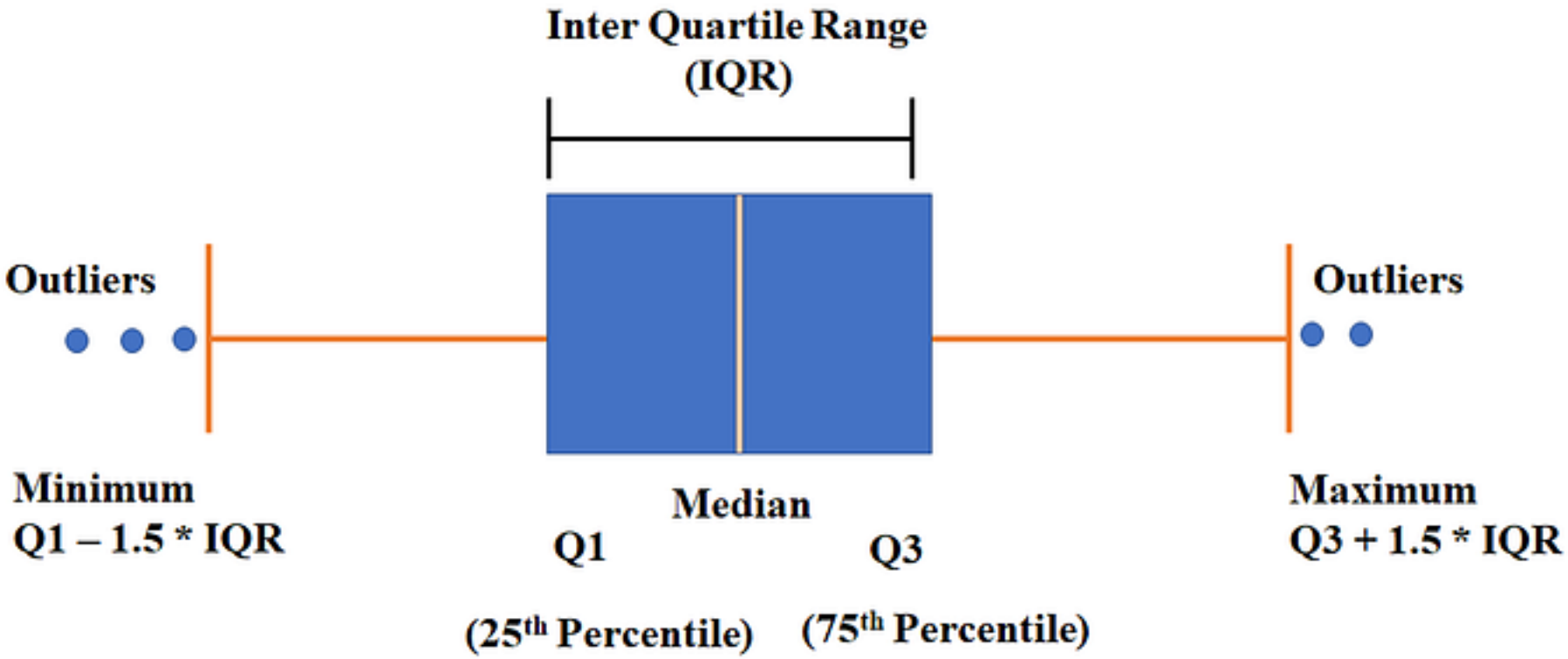
- **Tukey's Method (IQR)**
  - outliers are defined as data points that fall below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR, where Q1 and Q3 are the first and third quartiles, respectively, and IQR is the interquartile range

Outliers Identification

- **Graphical analysis with Boxplot**

  Outliers are extreme values any values out of range:

  [Q1-1.5 × IQR, ..., Q3 + 1.5 × IQR]



Methods to Detect Outliers

Model-Based methods: These methods involve building a predictive model and identifying data points with large residuals or prediction errors

- **Regression (Residuals)**

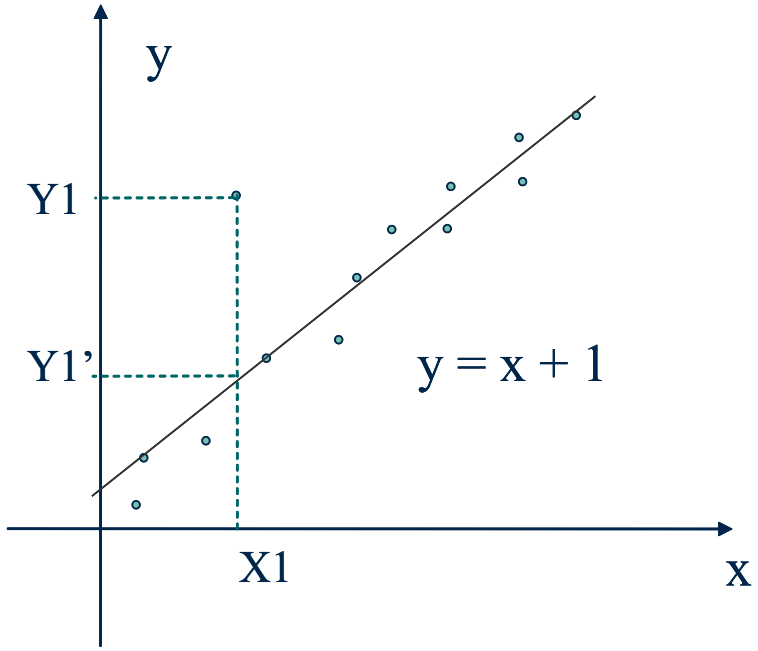
  For regression models, identify data points with large residuals, with large vertical distances between observed and predicted values

- **Cluster-Based Methods**

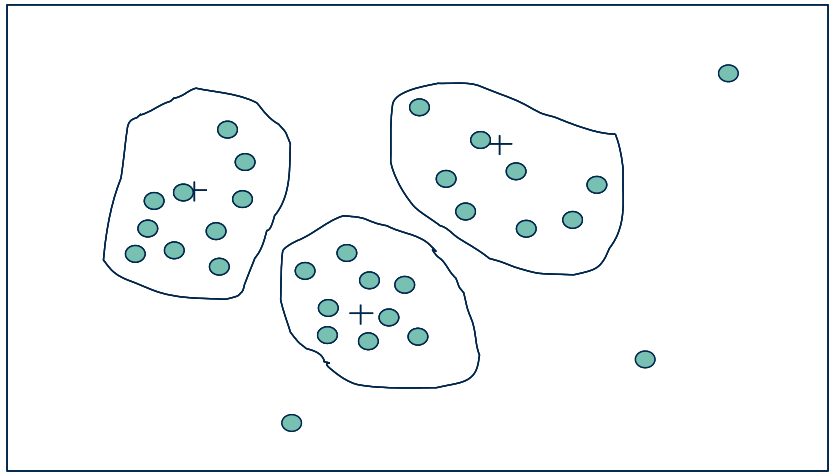
  Use clustering algorithms to group similar data points and identify outliers as points that do not belong to any cluster or belong to sparse clusters

Outliers Identification

- **Regression**



- **Clustering**



Data Transformation

Conversion: Nominal to Numeric

- Some methods can deal with nominal values; other methods (neural nets, regression, nearest neighbours) require only numeric inputs
- In such methods, it is necessary to convert nominal fields to a numeric value
- Converting a nominally ordered attribute (e.g., grade) to a number is necessary to preserve its *natural* order, to be able to use “>” and “<“ comparisons on these fields
- If no sequential order exists, it should be careful not to create one

Conversion: Nominal to Numeric

- **Binary attributes:** encoded 0/1

```
df[feature] = (df[feature].values == 'Yes').astype(int)

diag_map = {'M':1, 'B':0}
df['diagnosis']= df['diagnosis'].map(diag_map)
```

- Attributes with 3 or more values: **dummy encoding**

participant_id	race	asian	black	hispanic	height
1	Asian	1	0	0	67
2	Black	0	1	0	69
3	Hispanic	0	0	1	66
4	White	0	0	0	68

```
df= pd.get_dummies(df, drop_first=True)
```

Data Scaling

Data is scaled to fall within a small, specified range, typically [0, 1] or [-1, 1]

- min-max transformation
- transformation by decimal scaling
- …

Prevent attributes with large ranges outweigh ones with small ranges

- Example: income has range 3000-200000 and age has range 10-80

The scaling of the data affects only its scale and not its distribution

It simply forces the values into a certain range

Min-Max Transformation

Performs a linear transformation from the original dataset to a new specific dataset (typically 0-1):

- the old minimum (min_1) is mapped to a new minimum: min_2
- the old maximum (max_1) is mapped to a new maximum: max_2

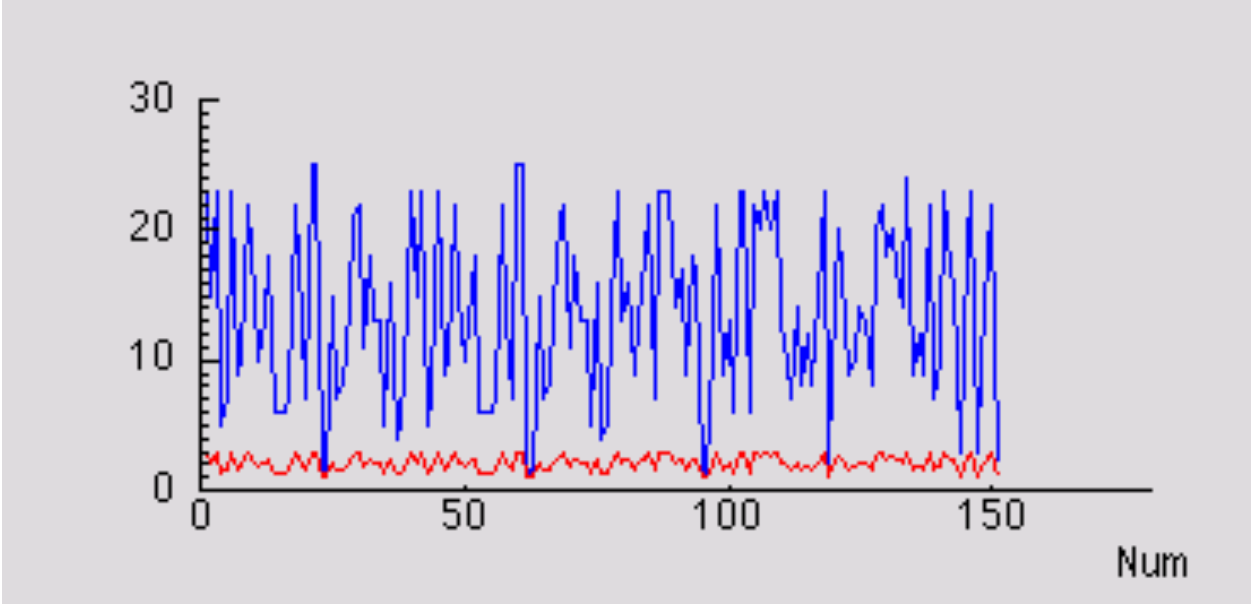
All points between these extremes are mapped to the new scale

The mathematical formula for the Min-Max normalization :

$$y' = \frac{y - \min_y}{\max_y - \min_y}$$

Advantages of the Min-Max transformation

- Preserves exactly all initial relationships of data values
- Doesn't introduce any changes in the data - the form of the histogram is maintained
- doesn't work well in samples with **isolated values**



Data Normalization

- The normalization of the data affects its distribution and its scale
- Normalizing the data forces them to have a Normal distribution or a Gaussian distribution
- The normalization of data is only useful when the models require the data to be normal. For example, linear discriminant analysis (LDA) or regression

Zscore Normalization

Also referred as medium-zero normalization or uni-variant normalization, transforms data so that:

- the average is zero
- the standard deviation is one

The formula applied is as follows:

$$x' = \frac{x - \mu}{\sigma}$$

The zscore normalization works well when:

- the sample has isolated values that dominate the normalization Min-max

Sigmoidal Normalization

Transforms the non-linear input data into the range [-1,1] using the sigmoid function

The formula applied by this type of normalization is as follows:

$$y' = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}} \qquad \alpha = \frac{1 - \mu}{\sigma}$$

The sigmoid normalization is appropriate to:

- Incorporate isolated points into the data set for analysis
- Prevent the most common values from being compressed without losing the ability to represent outliers

Data Reduction

Reducing the dimension of the data set

- Some data mining methods may be unable to handle very large data sets
- The computation time to obtain a certain model may be too large for the application
- We may want simpler models

Some strategies

- Reduce the number of instances
- Reduce the number of features

Instances Reduction

Discretization

The goal of discretization is to reduce the number of values for a continuous attribute assumed by grouping them into a small number of intervals (bins)

Any discretization process consists of two steps:

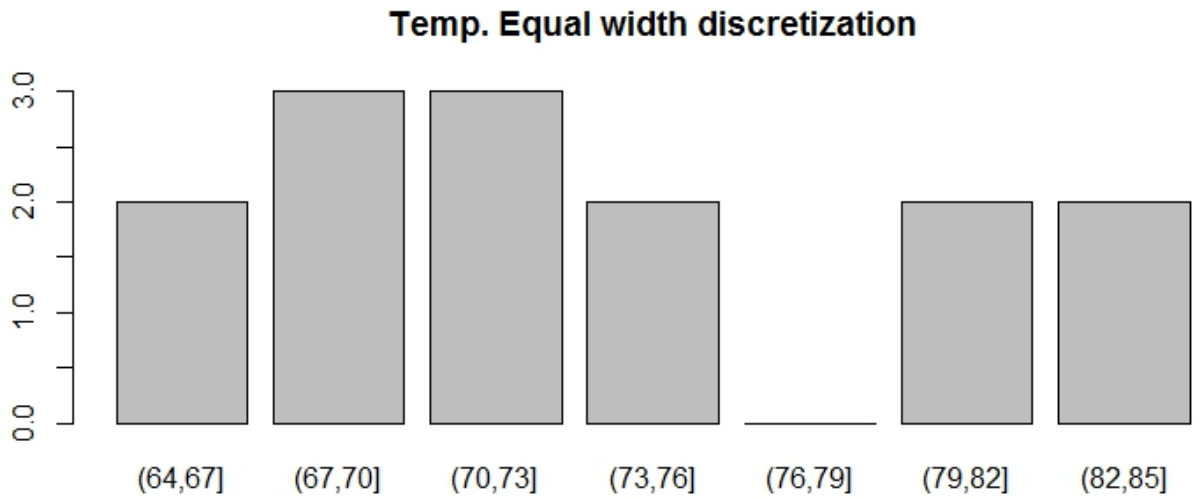
- 1st, the number of discrete intervals needs to be decided
- 2nd, the width (boundary) of each interval must be determined

Number of discrete intervals:

- large number: more of the original information is retained
- small number: looses information but is “easier” for the learning algorithms

Discretization: Equal-width

```
df.plot.box(by='Interval', column='tempdiscr')
```

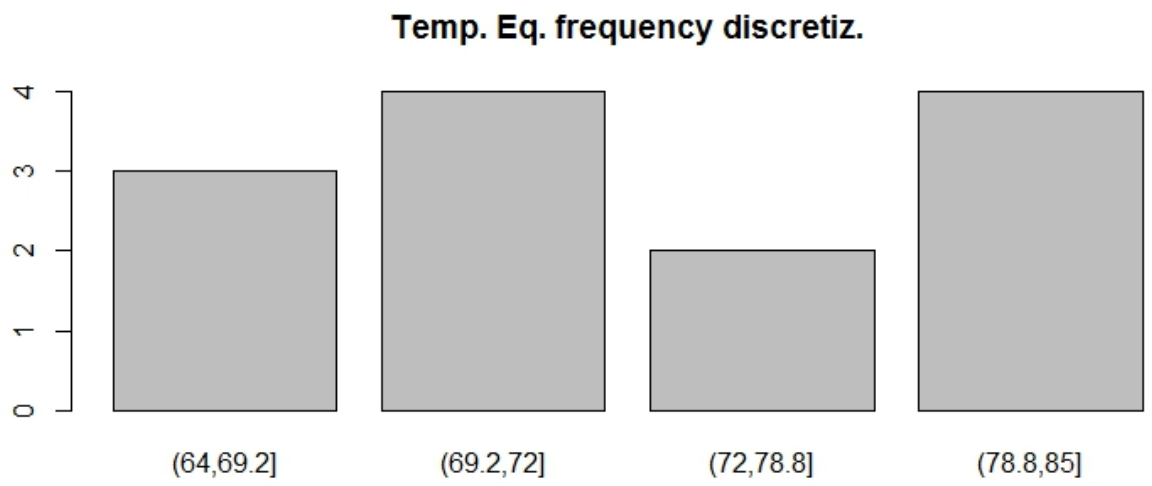


May produce clumping

Discretization: Equal-frequency

```
df[‘Tempdiscr’].value_counts()

df.plot.box(by=‘Interval', column='tempdiscr’,
                          tittle="Temp. Eq. frequency discretiz.")
```



Discretization considerations

- Equal Width is simplest, good for many classes
  - can fail miserably for unequal distributions
- Equal frequency gives better results
- Class-dependent can be better for classification
  - Note: decision trees build discretization on the fly
  - Naïve Bayes requires initial discretization
- Many other methods exist …

Features Reduction

Features to remove

- Features with no or little variability
  - *remove a feature where almost all values are the same*

- Key fields
  - *where all the values are distinct, as they haven't any semantic associated*

- False predictors or information "leakers"
  - *attributes similar to the goal attribute but with different values*

Features to remove

- Two highly correlated predictors represent the same underlying information and often add more complexity to the model than information
- The removal of one of them does not compromise the model's performance and may lead to a model that is easier to interpret
- It is possible to obtain relationships of collinearity among several predictors at the same time (multicollinearity) – represented by a **correlation matrix**
- The **Pearson correlation** between two attributes A and B measures the linear relationship between these two attributes

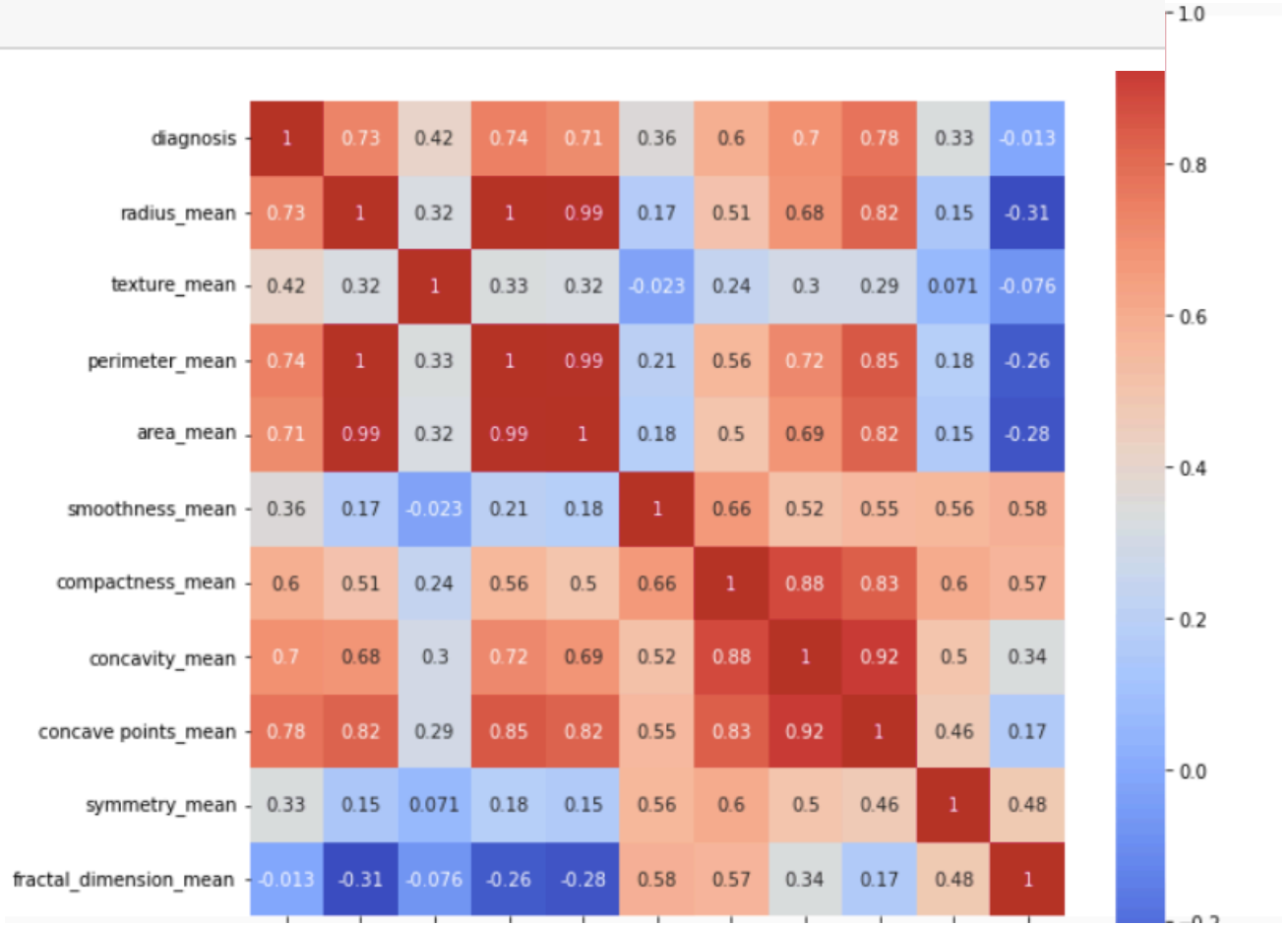
$$r_{A,B} = \frac{\sum(A-\bar{A})(B-\bar{B})}{(n-1)\sigma_A\sigma_B} \qquad \bar{A} = \frac{\sum A}{n} \qquad \sigma_A = \sqrt{\frac{\sum(A-\bar{A})^2}{n-1}}$$

Correlation Matrix

```
plt.figure(figsize=(12,6))

corr = df.select_dtypes(include='number').corr()
sns.heatmap(corr, annot=True, square=True, cmap='coolwarm')

plt.show()
```



Feature Selection

Feature Selection

- Feature selection is the process of selecting a subset of features from the total variables in a data set to train machine learning algorithms
- Feature selection techniques do not alter the original representation of the variables, but merely select a subset of them

Aims of Feature Selection:

- Improve model performance
- Create faster and cost-effective models
- Facilitate understanding of predictions

Feature selection methods

- **Filter methods**

  Select features based on the intrinsic characteristics of the data, ignoring their interaction with the ML model, they are **model agnostic**

- **Wrapper methods**

  They generate multiple feature subsets and then evaluate their performance based on the classification or regression model

- **Embedded methods**

  The selection of features is made in the training or induction of the predictive model by the ML algorithm that has a way to discriminate among the features

Filter methods

Filter methods involves:

- Ranking features based on some criteria
- Selecting high-ranking features

Features can be ranked based on:

- how well they separate the classes
- how well they correlate with the target

- This can be achieved by using statistical tests like:
  - ANOVA, Chi-square, or correlation, among others

Filter methods

Chi-square:

- is used to determine the association between categorical variables and categorical target
- features with large chi-square statistic or small p-values have strong associations with the target

ANOVA:

- is suitable for continuous variables and a categorical target
- selects features whose p-value is bigger than 0.05

Filter methods

Chi-square assumptions:

- the observations are independent
- the frequencies in the expected distribution must be greater than 5

ANOVA:

- Is less reliable when the target is imbalanced
- may not perform equally well in normal and skewed variables

Attention:

Statistical tests neglect feature interactions

Wrapper methods

Wrapper methods involves:

1. Create all possible feature subsets
2. Evaluate those subsets with the machine learning model
3. Select the best subset

For n features, the number of possible subsets is $\mathbf{2^n}$

For big datasets evaluating all possible feature combinations is impossible

- Exhaustive search
- Forward feature selection
- Backward feature elimination

Embedded methods

- The search for an optimal subset of features is built into the construction of the classifier or the regression model
- Embedded approaches train only one machine learning model to select features

Embedded methods:

- Lasso Regularization
- Feature importance from decision trees (Random Forest algorithm)
- Principal Component Analysis (PCA)

Lasso Regularization

Linear regression models aim to predict the outcome based on a linear combination of the predictor variables given by:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + e$$

With the **Lasso or l1-regularized regression** the coefficients are estimated by minimizing the following equation:

$$minimize\{\sum_{j=1}^{N}(y_j - \beta_0 - \sum_{i=1}^{n}\beta_i x_{ij})^2 + \lambda \sum_{i=1}^{n}|\beta_i|\}$$

The Lasso regularization has the ability to set some of the coefficients to zero, allowing for **feature selection**

Features whose **coefficients are zero** can be **safely removed**

Random Forest Feature Importance

- The feature importance is calculated based on the reduction in impurity that each feature contributes to the model
- Different metrics that can be used to determine the “best” feature:
  - In classification trees, the induction algorithm minimizes the Gini index or the entropy
  - In regression trees, the induction algorithm minimizes the mean squared error, the mean absolute error, or the Poisson deviance

- Decision tree algorithms assign feature importance during model induction, selecting the highest importance features
- Scikit-learn automatically selects features greater than the mean importance of all features in the data but can be adjusted to an arbitrary number

Principal Component Analysis (PCA)

General Idea

- Substitute the set of variables by a new (smaller) set where most of the "information" on the problem is still expressed

Goal

- Find a new set of axes onto which we will project the original data points

Data Reduction

- Use a smaller set of variables that contain the relevant information that is in the complete data
- The goal is to distinguish what is similar/different from the data using a smaller set of attributes

Principal Component Analysis (PCA)

- With PCA a new set of axes y_1, y_2, …, y_q are formed by linear combinations of the original variables $x_1,x_2,\ldots,x_n$ with q <n

$$y_1 = a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n$$
$$y_2 = a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n$$
$$\ldots$$
$$y_q = a_{q1}x_1+a_{q2}x_2+\ldots+a_{qn}x_n$$

- We search for the linear combinations that “explain” most of the variability that existed among the data points on the original axes
- If we are “lucky” with a few of these new axes (ideally two for easy data visualization), we are able to explain most of the variability on the original data
- Each original observation is then “projected” into these new axes

Principal Component Analysis (PCA)

Algorithm

- Find a first linear combination which better captures the variability in the data

- After finding this linear combination (the first direction), PCA looks for a second linear combination that is orthogonal to the first one and tries to capture the variability not explained by the first one, and so on..

- Continue until the set of new variables explains most of the variability (frequently **90% is considered enough**)

Unbalanced Classes

Class imbalance

Class imbalance occurs when one or more classes have very low proportions in the data compared to the other classes

Class imbalance frequently occurs in applications:

- Operator abandonment: 97% clients stay, 3% leave
- Medical diagnosis: 90% healthy, 10% unhealthy
- eCommerce: 99% don't buy, 1% buy
- Fraud detection: 99% of transactions are not fraudulent

In these cases, the classifier even shows a high accuracy rate (in the majority class), but with little or no usefulness

Management of Class imbalance

Two target classes

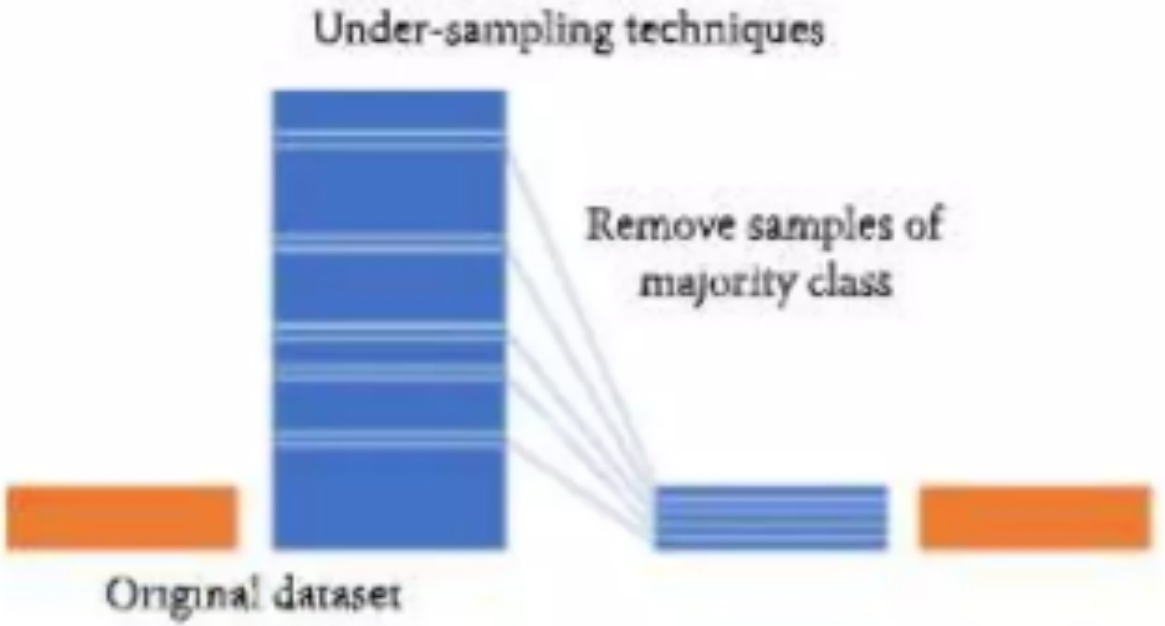
- From the initial data set, apply an appropriate sampling method
- Generate a balanced training set
- Build the models using the balanced data
- Evaluate the model on the test set with the initial data distribution - this way, an honest estimate of the model's future performance is presented

Multiple classes

Ensure that each class is approximately equally represented in the training set

Random Undersampling

Creates a new training set including all examples from the "positive" (minority) class and randomly select "negative" examples



Advantages:

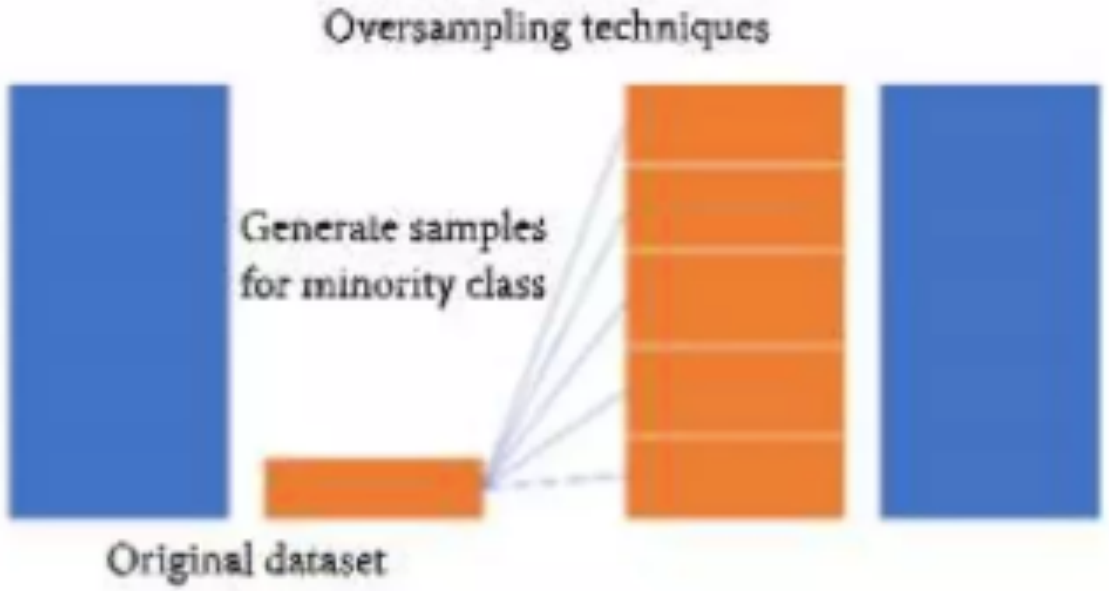
- easy to implement
- training is faster (smaller training set)
- for some domains, it can work very well

Disadvantages

- Loss of data/information

Random Oversampling

The process of sampling involves expanding the distribution of the minority class



Advantages:

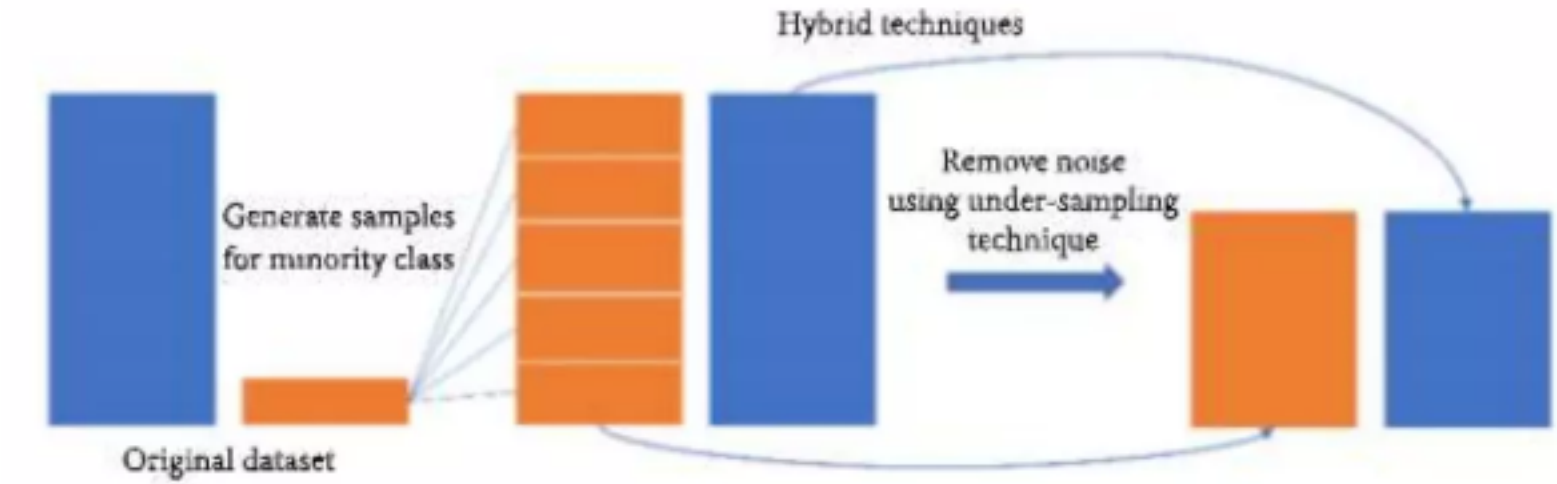
- Easy to implement
- Use all the training data
- It tends to perform better on a broader dataset than subsampling

Disadvantages

- Computationally more expensive to train the classifier
- Can lead to overfitting

Hybrid approach

Combines undersampling and oversampling (subsampling) to achieve data balancing



Hybrid approach

Can offer several advantages, but it also comes with certain challenges:

- Risk of Overfitting (from Oversampling)
- Loss of Important Information (from Undersampling)
- Model Instability
- Deciding the correct ratio for oversampling and undersampling is not straightforward
- The process of removing or generating samples can distort the natural distribution of the data

Algorithms for data balancing

SMOTE (Synthetic Minority Over-sampling Technique)

Generates synthetic samples for the minority class by interpolating between existing minority class instances

Advantages:

- Reduces the risk of overfitting compared to simple duplication of minority class examples
- Often improves the model's ability to generalize when trained on highly imbalanced datasets

Challenges:

- Can lead to overfitting in noisy datasets
- May create unrealistic synthetic data, leading to a distorted decision boundary

ADASYN (Adaptive Synthetic Sampling)

Is a variant of SMOTE that generates synthetic data adaptively by focusing more on minority class examples that are difficult to classify

Advantages:

- Focuses on the most difficult examples, which helps improve classification boundaries.
- Less risk of introducing redundant or irrelevant synthetic examples

Challenges:

- May still introduce noise if too much synthetic data is generated
- Increased computational cost compared to SMOTE

Tomek Links

A data cleaning method used after oversampling. Tomek Links remove noisy examples from the dataset, particularly those that are near the decision boundary between classes

Advantages:

- Removes ambiguity and noise from the data
- Reduces the risk of overfitting by removing problematic examples after oversampling

Challenges:

- It does not inherently balance the dataset but is effective when used in combination with oversampling

Best Practices

Combination of Techniques:
Often, combining techniques like SMOTE with Tomek Links or using oversampling with undersampling can provide better results

Cross-validation: Always validate the results using cross-validation on imbalanced datasets to avoid overfitting or losing critical information during data balancing

Choosing Algorithms Based on Dataset: The choice of algorithm depends on the specific characteristics of the dataset: the size of the majority class, the degree of imbalance, and the type of classification task

Best Practices

There is no single "best" algorithm for data balancing, but rather a combination of approaches may be the most effective

Experimentation with different methods is necessary to find the optimal solution for a specific problem

Careful cross-validation, experimentation with different algorithms, and ensuring that performance metrics (like AUC, precision-recall, etc.) are evaluated on **imbalanced test data** are crucial