PROSIT 1 - PARSING APARTMENTS

0 - Topic

Topic

1 - Clarification

1.1 - Keywords

- K-means algorithm
- AI
- Census tracts
- Hierarchical clustering
- Classification
- Dendrogram
- Intelligent analysis
- Data set
- Supervised learning

1.2 - Context

A real estate agency wants to minimize the number of useless visits for apartment renting by optimizing apartment suggestions according to the client. This problem being too difficult, they

want to be helped by artificial intelligence.

1.3 - Constraints (Where? How? Why? When?)

- Given dataset
- Using K-means algorithm (with $k = 2$)
- Missing & Non numerical values present in the dataset
- Dataset restricting to California in 1990.

1.4 - Problematic

How to make the 1990 dataset relevant to today's economy?

Can the dendrogram categorize part of the dataset?

How to use k-mean algorithms? → separate data in 2 parts

Why such representation? (colors)

How to manage missing & non numerical values in the most optimal way? (in the text they removed the whole attribute column where there were a missing value which is absolutely irrelevant). Is it a good idea to replace those values or to just not use them?

What are the consequence of using an other clustering number than 2? ($k \neq 2$)

Does k-means uses average distances between datas?

What is the best way to compare results?

How to effectively pre-process the data?

How to identify if there is data mistakes and how to manage them?

1.5 - Generalization

- AI
- Classification
- Dataset
- Data processing

2 - Preparation

2.1 - Deliverables

- Steps of data processing
- K-means model

- Result visualization and analysis

2.2 - Solution ideas

- Use the K-means algorithm
- Remove rows from missing values or replace the missing values by average OR medium OR random in bounds...
- Use elbow method to find optimal k value
- Calculate correlation between variables
- Compare k-means with scatter plot
- Transform numerical values with algorithm such as One Hot Encoding.
- Add a new column OR reduce number of columns
- Pre process the data by transforming the data between 0 & 1 with given ranges.
- Check data mistakes and manage them.

2.3 - Action plan

Studies

- Data processing
- Classification
- K-means algorithm
- Supervised learning
- Elbow method

Outcomes

- Cleaned dataset
- Processed dataset
- Application of the K-mean algorithm
- Comparison of results

3 - Studies

BasicsStatisticsForDataScience.pdf

3.1 - Data Preprocessing

3.1.1 - Basic Definitions

Universe = A Population (Universe) is the Whole Collection of Things Under Consideration

Sample = A Sample is a Portion of the Population Selected for Analysis

Parameter = A Parameter is a Summary Measure Computed to Describe a Characteristic of the Population

Statistic = A Statistic is a Summary Measure Computed to Describe a Characteristic of the Sample

Descriptive Statistics = Collecting, presenting, and characterizing data

Inferential Statistics = Drawing conclusions and/or making decisions concerning a population based only on sample data

Nominal Scale = distinct categories in which **no ordering** is implied

Ordinal Scale = distinct categories in which **ordering** is implied

Interval Scale = an ordered scale in which the difference between the measurements **does not involve a true zero point**

Ratio Scale = an ordered scale in which the difference between the measurements **involves a true zero point**

3.1.2 - Basic Calculus

Mean: $\bar{X} = \frac{x_1+x_2+\cdots+x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n}$

Weighted Mean: $\frac{x_1+x_2+\cdots+x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n}$

Geometric Mean: $\sqrt[n]{x_1 \cdot x_2 \cdot \cdots \cdot x_n}$

Geometric Mean Rate of Return: $\left(\prod_{i=1}^{n}(1+R_i)\right)^{\frac{1}{n}} - 1$

Median: $\frac{\text{Middle value 1}+\text{Middle value 2}}{2}$

Mode: The **mode** is the value or values that occur **most frequently** in a dataset. It is a measure of central tendency that identifies the most common data point(s). $mode = \max(f(x_1), f(x_2), \ldots, f(x_n))$ with f the frequency and $\max$ the maximum frequency.

Sample Variance: $S^2 = \frac{\sum_{i=1}^{n}(X_i-\bar{X})^2}{n-1}$

Population Variance: $\sigma^2 = \frac{\sum_{i=1}^{N}(X_i-\mu)^2}{N}$

Coefficient of Variation: $CV = \left(\frac{S}{\bar{X}}\right) \times 100$ (is used to compare two or more sets of data measured in different units)

Coefficient of Correlation: $r = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}}$

3.1.3 - Normalization

Population Standard Deviation: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$

Sample Standard Deviation: $s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2}$

Standardisation (Z-score Normalization): $X_{stand} = \frac{X-\bar{X}}{\sigma \text{ or } s}$

Max-Min Normalization: $X_{norm} = \frac{X - \min(X)}{\max(X) - \min(X)}$

3.1.4 Overview of Sampling Methods in Research

What is a Stratified Sample?

A **stratified sample** is a sampling method where the total population is divided into **subgroups** or **strata** based on shared characteristics (e.g., age, gender, region). Then, a sample is taken from each stratum, either proportionally to its size in the population or based on specific criteria.

Why Use Stratified Sampling?

- **Increased Representativeness:** Ensures that every significant subgroup in the population is well-represented in the sample.
- **Improved Precision:** Reduces variance in estimates compared to simple random sampling.
- **Targeted Applications:** Useful when there is high variability between strata but homogeneity within strata.

1. Simple Random Sampling

Definition:

Simple random sampling is a method where every individual in the population has an **equal chance** of being selected. This is usually done through random draws or random number generators.

Why Use Simple Random Sampling?

- **Fairness:** All individuals have the same probability of being selected, reducing bias.
- **Ease of Implementation:** Straightforward to execute when the population is small and well-defined.
- **Representative (if population is homogeneous):** Works well when there are no significant subgroups in the population.

2. Systematic Sampling

Definition:

Systematic sampling involves selecting individuals at regular intervals from a list or dataset, starting from a randomly chosen point.

Why Use Systematic Sampling?

- **Efficiency:** Faster than simple random sampling for large datasets.
- **Uniform Coverage:** Ensures that samples are spread across the entire population.
- **Ease of Implementation:** Only the interval needs to be defined after random selection of the first point.

3. Cluster Sampling

Definition:

Cluster sampling divides the population into **naturally occurring groups** (clusters) and then randomly selects a few clusters. All individuals within these selected clusters are included in the sample.

Why Use Cluster Sampling?

- **Cost-Effective:** Reduces the need to sample individuals from a wide geographic area.
- **Feasible for Large Populations:** Useful when the population is spread out and difficult to access.
- **Preserves Group Characteristics:** Captures naturally occurring group structures.

4. Quota Sampling

Definition:

Quota sampling involves setting **specific quotas** for each subgroup (e.g., age, gender) to ensure the sample represents those subgroups. However, the selection of individuals within each subgroup is **non-random.**

Why Use Quota Sampling?

- **Practicality:** Easier to implement than stratified sampling.
- **Ensures Representation:** Guarantees representation of subgroups in the final sample.
- **Useful for Limited Resources:** Allows for targeted data collection when random sampling isn't feasible.

5. Snowball Sampling

Definition:

Snowball sampling is a method where existing participants help recruit other participants who share similar characteristics. It’s often used for hard-to-reach or specialized populations.

Why Use Snowball Sampling?

- **Access to Hidden Populations:** Useful for communities that are difficult to identify (e.g., undocumented workers, people with rare medical conditions).
- **Efficient Recruitment:** Builds on referrals, reducing the effort needed to find participants.
- **Exploratory Research:** Helps in gaining insights into less-studied groups.

3.2 - Automatic Classification Algorithms (Unsupervised)

https://www.seldon.io/supervised-vs-unsupervised-learning-explained

3.2.1 - Supervised / Unsupervised Learning

Supervised: Supervised machine learning requires labelled input and output data during the training phase of the machine learning model lifecycle. The reason it is called supervised machine learning is because at least part of this approach requires human oversight. The vast majority of available data is unlabelled, raw data. Human interaction is generally required to accurately label data ready for supervised learning.

Supervised machine learning is used to classify unseen data into established categories and forecast trends and future change as a predictive model.

A model developed through supervised machine learning will learn to recognize objects and the features that classify them. Predictive models are also often trained with supervised machine learning techniques. By learning patterns between input and output data, supervised machine learning models can predict outcomes from new and unseen data.

> **Supervised** machine learning is often used for:
>
> - **Classifying** different file types such as images, documents, or written words.
> - **Forecasting** future trends and outcomes through learning patterns in training data.

Unsupervised: Unsupervised machine learning is the training of models on raw and unlabelled training data. It is often used to identify patterns and trends in raw datasets, or to cluster similar data into a specific number of groups. It's also often an approach used in the early exploratory phase to better understand the datasets. As the name suggests, unsupervised machine learning is more of a hands-off approach compared to supervised machine learning.

A human will set model hyperparameters such as the number of cluster points, but the model will process huge arrays of data effectively and without human oversight.

Unsupervised machine learning is therefore suited to answer questions about unseen trends and relationships within data itself.

The vast majority of available data is unlabelled, raw data. By grouping data along similar features or analysing datasets for underlying patterns, unsupervised learning is a powerful tool used to gain insight from this data. In contrast, supervised machine learning can be resource intensive because of the need for labelled data.

> **Unsupervised** machine learning is mainly used to:
>
> - **Cluster** datasets on similarities between features or segment data

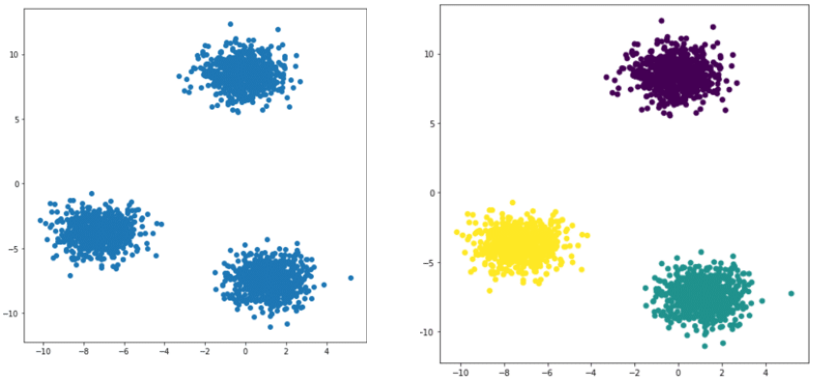
> - Understand **relationship** between different data point such as automated music recommendations
> - Perform initial **data analysis**

The main difference between supervised vs unsupervised learning is the need for labelled training data.

3.3 - K-means algorithm

Definition

Clustering is a specialized discipline within Machine Learning aimed at separating your data into homogeneous groups with common characteristics. It's a highly valued field, especially in marketing, where there is often a need to segment customer databases to identify specific behaviors. The K-means algorithm is a well-known **unsupervised** algorithm in the realm of Clustering.



How do we go about it?

The idea is quite simple and intuitive. The first step is to randomly define 3 centroids and associate them with 3 labels, for example, 0, 1, 2. Then, for each point, we calculate its distance to the 3 centroids and associate the point with the closest centroid and its corresponding label. This labels our data.

Finally, we recalculate 3 new centroids, which will be the centers of gravity of each labeled cluster of points. We repeat these steps until the new centroids no longer move from the previous ones. The final result is shown in the figure on the right.

Concept of distance and initialization

You've understood that in this algorithm, two key points are: What metric is used to evaluate the distance between points and centroids? What is the number of clusters to choose?

In the k-means algorithm, the **Euclidean** distance is generally used, where $p = (p_1, \ldots, p_n)$ and $q = (q_1, \ldots, q_n)$.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} = \sqrt{(p_1 - q_1)^2 + \ldots + (p_n - q_n)^2}$$

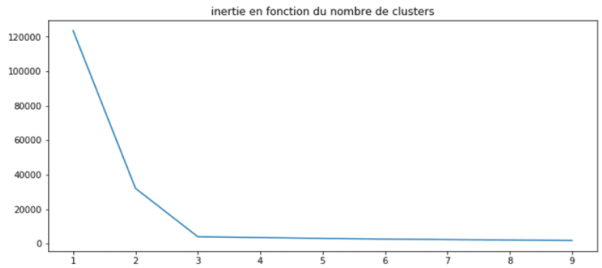
How to get the centroids?

1. **Initialization**:
   - **Random Initialization**: Centroids are initialized randomly by selecting K points from the dataset or by assigning random values within the data range.
   - **K-Means++ Initialization** (Improved version): Selects initial centroids in a way that ensures they are spread out, improving convergence.
2. **Assignment Step**:
   - For each data point in the dataset, calculate its distance (usually Euclidean) to each centroid.
   - Assign the point to the cluster represented by the nearest centroid.
3. **Update Step**:
   - After all points are assigned, recompute the centroid for each cluster by taking the **mean** of all points assigned to that cluster.
   - If a cluster has no points assigned (rare), reinitialize its centroid randomly or use other strategies to

handle empty clusters.

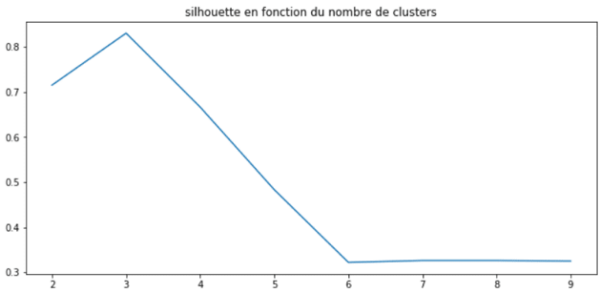
Ideal number of clusters?

There are multiple methods to determine the ideal number of clusters. The most well-known one is the elbow method. It relies on the concept of inertia, which is defined as follows: the sum of the **Euclidean distances** between each point and its associated centroid. Naturally, the higher the initial number of clusters we set, the lower the inertia becomes: points are more likely to be close to a centroid. Let's see how this works on our example:



This method is conclusive. It can be coupled with a more precise but computationally intensive approach: the silhouette coefficient. It is defined as follows:

$$s = \frac{b - a}{\max(a, b)}$$



- a The average distance to other observations in the same cluster (i.e., the intra-cluster average).
- b The average distance to the nearest cluster.

This coefficient can vary between -1 and +1. A coefficient close to +1 means that the observation is well inside its own cluster, while a coefficient close to 0 means it is close to a boundary; finally, a coefficient close to -1 means that the observation is associated with the wrong cluster

$$\text{Distance} = \frac{|(x_2 - x_1)(y_1 - y) - (x_1 - x)(y_2}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

This method also helps identify the ideal number of clusters, which in this case is 3.

In addition to these two methods, it is essential to conduct a detailed analysis of the created clusters. By that, I mean a precise and in-depth descriptive analysis to determine the common characteristics of each cluster. This will allow you to understand the typical profiles of each cluster.

While there isn't a strict mathematical formula for detecting the elbow point, there are methods to approximate or automate its identification:

Mathematical Approximation

A mathematical approach involves computing the **distance to the curve** from a straight line between the first and last points:

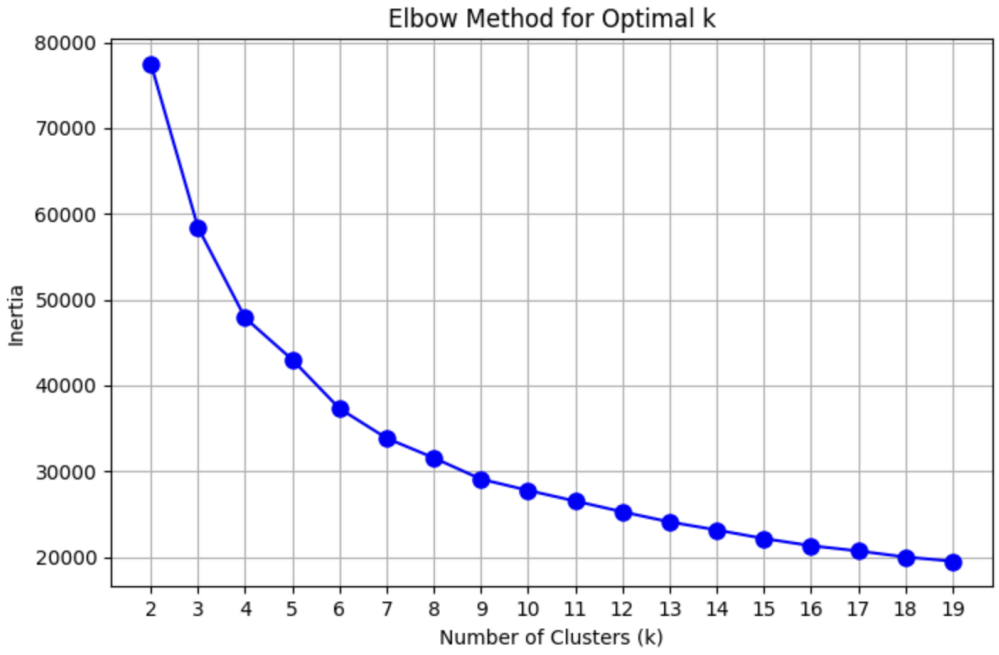
1. Consider the plot (x_i, y_i) as points .
2. Form a line connecting (x_{min}, y_{min}) and (x_{max}, y_{max}).
3. Compute the perpendicular distance of each intermediate point to this line:
4. The elbow point corresponds to the maximum distance.

4 - Outcomes

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

### STEP 1 - IMPORT DATAS
def import_data(filename):
    df = pd.read_csv(filename)
    return df

### STEP 2 - MANAGE MISSING VALUES
def drop_missing(df):
    df = df.dropna()
    return df
def mean_missing(df):
    mean = df.mean(numeric_only=True)
    df = df.fillna(mean)
    return df
def median_missing(df):
```

```
    median = df.median(numeric_only=True)
    df = df.fillna(median)
    return df
def fill_random(df):
    for i in range(0, df.columns.count):
        min_val = df.columns[i].min()  # Minimum value of the column
        max_val = df.columns[i].max()  # Maximum value of the column
        df.columns[i].apply(lambda x: np.random.uniform(min_val, max_val) if
    return df

### STEP 3 - MANAGE NON-NUMERIC VALUES
def all_numerics(df):
    df['ocean_proximity'].replace('<1H OCEAN', 0)
    df['ocean_proximity'].replace('NEAR BAY', 1)
    df['ocean_proximity'].replace('INLAND', 2)
    df['ocean_proximity'].replace('NEAR BAY', 3)
    return df

### STEP 3 - K MEANS ALGORITHM
    # STEP 3.1 - SELECTION OF RELEVANT FEATURES
df = import_data("housing.csv")
df = mean_missing(df)
df = all_numerics(df)
features = df[['housing_median_age', 'total_rooms', 'median_house_value', 'm

# STEP 3.1 - ELBOW METHOD TO FIND OPTIMAL K VALUES
def eblow():
    # Standardize the data (recommended for KMeans)
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(features)
    inertia = []
    k_values = range(2, 20)  # Test k values from 1 to 10
    for k in k_values:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(scaled_features)
        inertia.append(kmeans.inertia_)
    return k_values, inertia

# STEP 3.2 - APPLICATION OF THE K-MEANS AND DISPLAY THE RESULTS TO FIND THE
k_values, inertia = eblow()
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, 'bo-', markersize=8)
```

```
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')
plt.xticks(k_values)
plt.grid()
plt.show()

### STEP 4 - APPLY TO THE DATASET AND TEST DIFFERENT PARAMETERSj
```



TO FINISH...