Supervised Machine Learning

\*SVM

The SVM (Support Vector Machine) is a supervised learning

algorithm mainly used for binary classification. It learns from labeled data (xᵢ, yᵢ) where xᵢ represents an n-dimensional feature vector (e.g., word count, link count) of an example, and yᵢ is its class

label (+1 or -1, such as spam or not-spam).

The goal is to find a hyperplane (w.x + b = 0) that maximizes the margin between classes, ensuring values ≥ 1 for positive class examples and ≤ -1 for negative class examples.

How does it find this hyperplane?

By finding values of w\*(weight/normal) and b\*(intercept) which help us to define this hyperplane.

The optimal values of w\* and b\* can be find out by minimising the cost function.

f(x)=sign(w\*⋅x+b\*)—————-> SVM model

pasted-image.tiff

**\*Bias and Variance:**

Machine learning algorithms typically require two datasets: a training dataset and a testing dataset. Let's consider a dataset that compares the weight and height of mice. Generally, smaller mice tend to have lower weights, while larger mice have higher weights. However, this relationship breaks down past a certain weight threshold, often indicating obesity. When we don't have a precise understanding of the height-weight relationship, we can use two methods to explore it:

Linear Regression

Non-linear (e.g., polynomial regression or kernel regression) models represented by squiggly lines.

Bias refers to the algorithm's inability to capture the true relationship. In the case of mouse weight versus height, a linear regression model typically exhibits higher bias compared to a non-linear model represented by a squiggly line.

Ideally, we aim for models with low bias and low variance. However, in practice, there's often a trade-off. For instance, a polynomial regression model with lower bias may exhibit higher variability, while a simple linear regression model has lower variability but higher bias. This trade-off can be likened to a high-risk, high-reward scenario for non-linear models, potentially providing very accurate or highly inaccurate predictions, and a lower-risk, lower-reward scenario for linear models, offering consistently good but not exceptional predictions.

Overfitting occurs when a model fits the training data well but performs poorly on the testing dataset. In contrast, underfitting refers to a model that is too simplistic and fails to capture the underlying patterns in the data.

Support Vector Machines (SVMs) often encounter issues related to bias and variance. A model with high variance is prone to overfitting, while a model with high bias tends to underfit the data.

We employ several techniques to strike a balance between bias and variance in machine learning models:

**Regularization**: This involves adjusting models to minimize loss and prevent overfitting or underfitting. It includes two main types:

**Ridge (L2) Regularization**: Adds a penalty equivalent to the sum of squares of coefficient magnitudes.

**Lasso (L1) Regularization**: Adds a penalty equivalent to the sum of absolute values of coefficients.

**Boosting**: This technique combines weak learners with high bias to collectively reduce overall bias. It sequentially trains models where each subsequent model improves upon the errors of previous ones.

**Bagging**: Bagging combines weak learners with high variance to collectively reduce overall variance. It involves using bootstrapping to create data subsets and aggregating predictions.

**Ensemble Methods:** Ensemble methods combine multiple models to create a more robust predictive model by leveraging the strengths of each individual model.

**Margin Classifiers:** Margin classifiers, used in tasks like classifying mice as obese or not, involve setting a threshold parameter and adjusting it based on distances between extreme category ends. The Soft Margin approach allows for misclassifications to handle outliers effectively.

**Cross-validation:** This technique evaluates model performance by splitting data into training and validation sets. There are various methods of cross-validation, including hold-out cross-validation, k-fold cross-validation, and stratified k-fold cross-validation. These methods help assess how well the model generalizes to unseen data and mitigate the risk of overfitting or underfitting.

K-means Clustering

\*clustering

Clustering is a widely used exploratory data analysis technique aimed at revealing the data's underlying structure. It involves identifying similar subgroups (clusters) within the data, with data points in the same cluster being very similar and those in different clusters being dissimilar. This process relies on similarity measures like Euclidean or correlation-based distances, chosen based on the specific application. Clustering can focus on features, grouping samples based on their attributes, and is utilized in market segmentation, image compression, document clustering, and other areas

The most widely used clustering algorithm are-

1.K-mean Clustering

2. Hierarchical Clustering

\*K-mean clustering

K-means is an iterative algorithm that partitions a dataset into K distinct, non-overlapping clusters. It aims to maximize similarity within clusters while minimizing the sum of squared distances between data points and cluster centroids.

Steps for k-mean clustering-

1.Choose the number of clusters (K).

2.Initialize centroids by randomly selecting K data points from the shuffled dataset.

3.Iterate until centroids no longer change:

-Compute the sum of squared distances between data points and centroids.

-Assign each data point to the nearest centroid (cluster).

-Compute new centroids by averaging data points in each cluster.

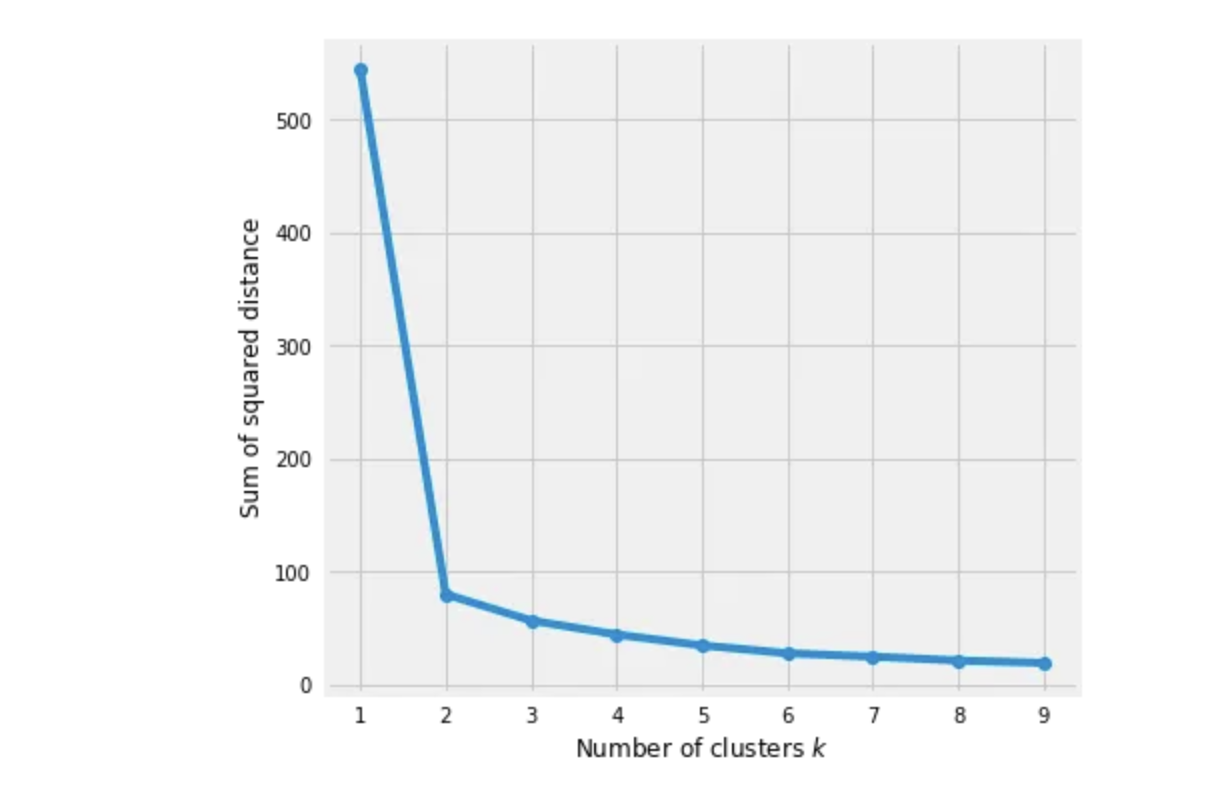
\*Evaluation

Evaluating clustering algorithms is tricky because unlike supervised learning, we don't have a clear way to measure their accuracy. K-means requires us to guess the number of clusters (K), and there's no one-size-fits-all answer.

To get information on k we use following methods-

1.Elbow method

2.Silhouette analysis

1.Elbow method

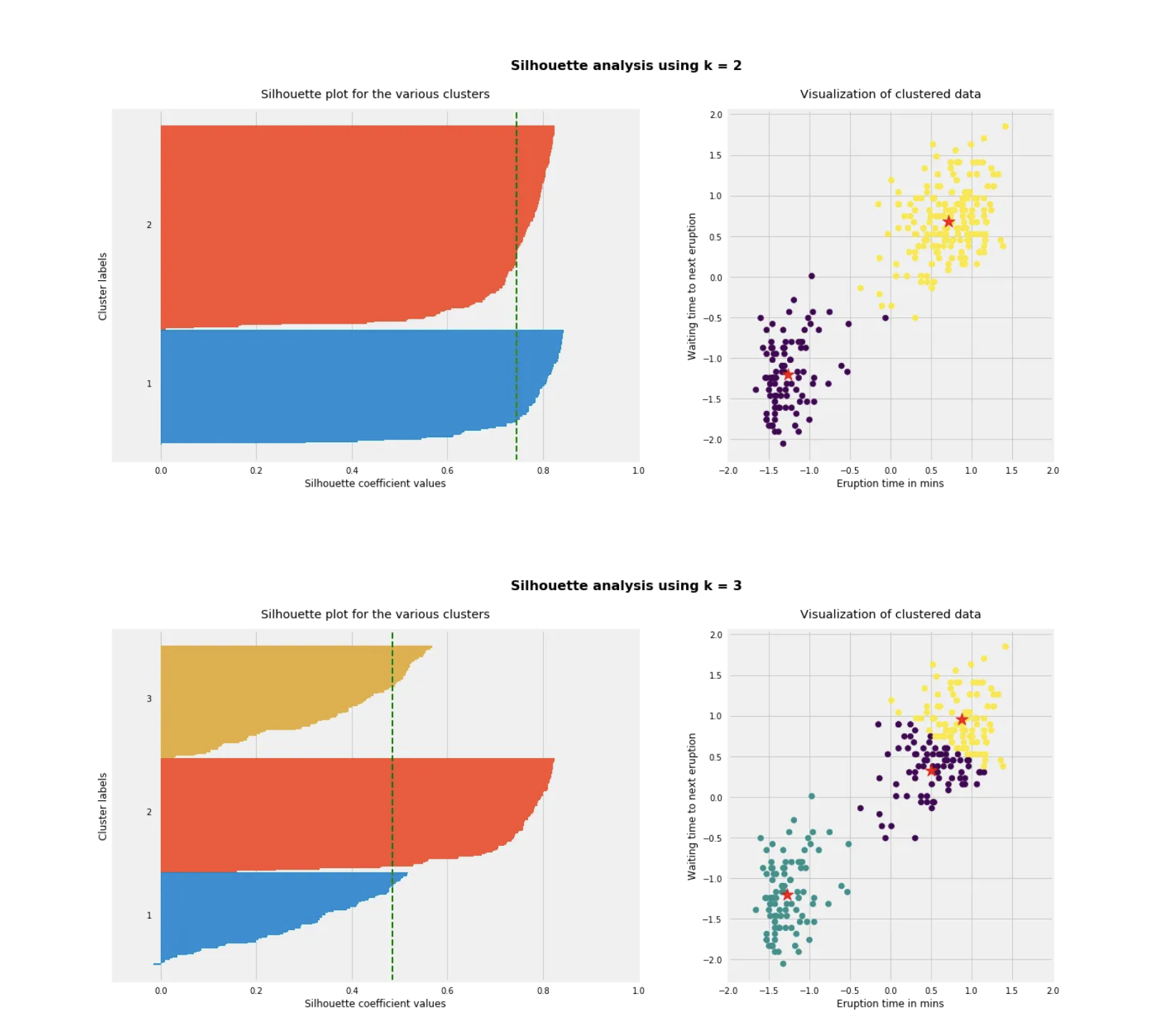
Here we would run K-mean clustering on a range of K values and plot the “percentage of variance explained” on the Y-axis and “K” on X-axis.

In the above graph you would notice that as we add more clusters after 3 it doesn't give much better modeling on the data. The first cluster adds much information, but at some point, the marginal gain will start dropping.

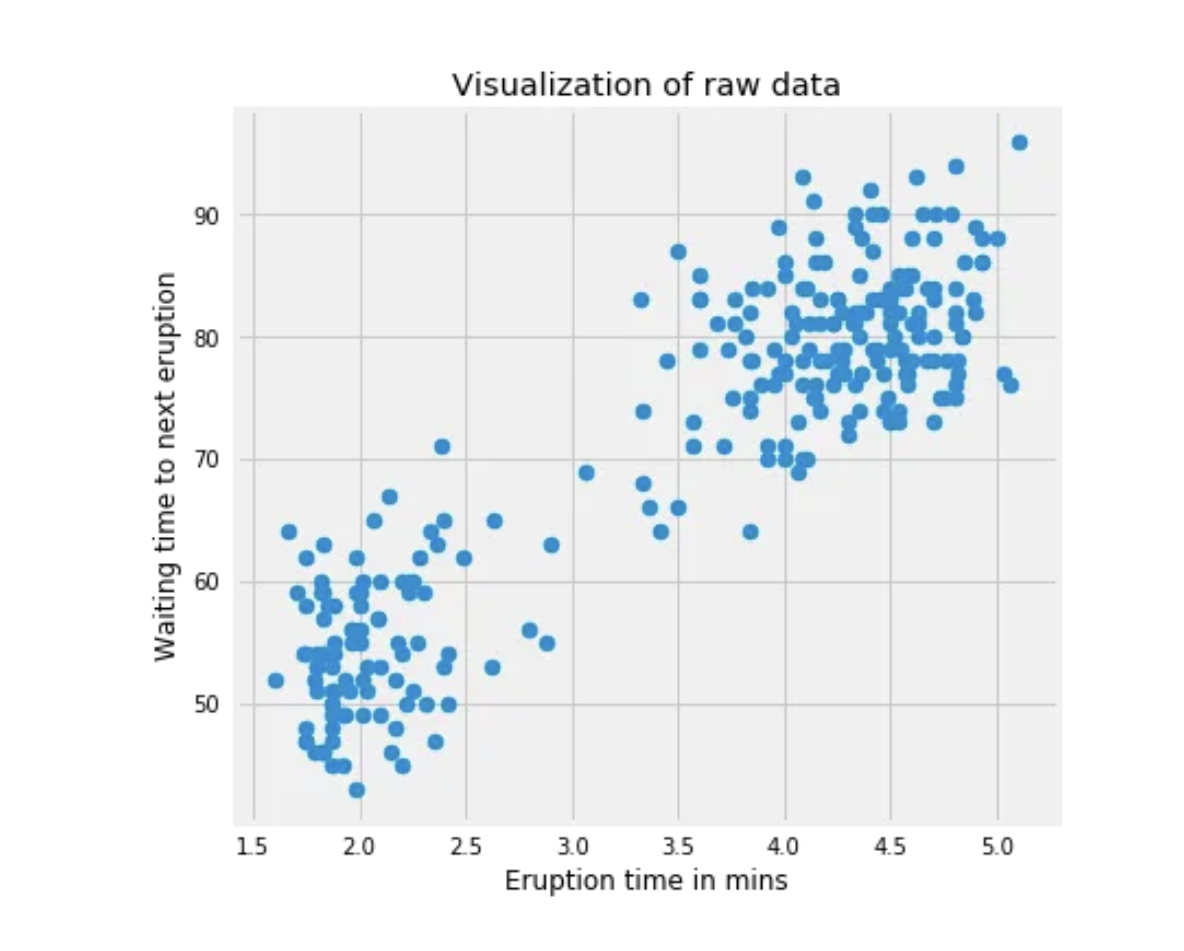
2.Silhouette Analysis

It is used to find degree of separation between the two clusters .

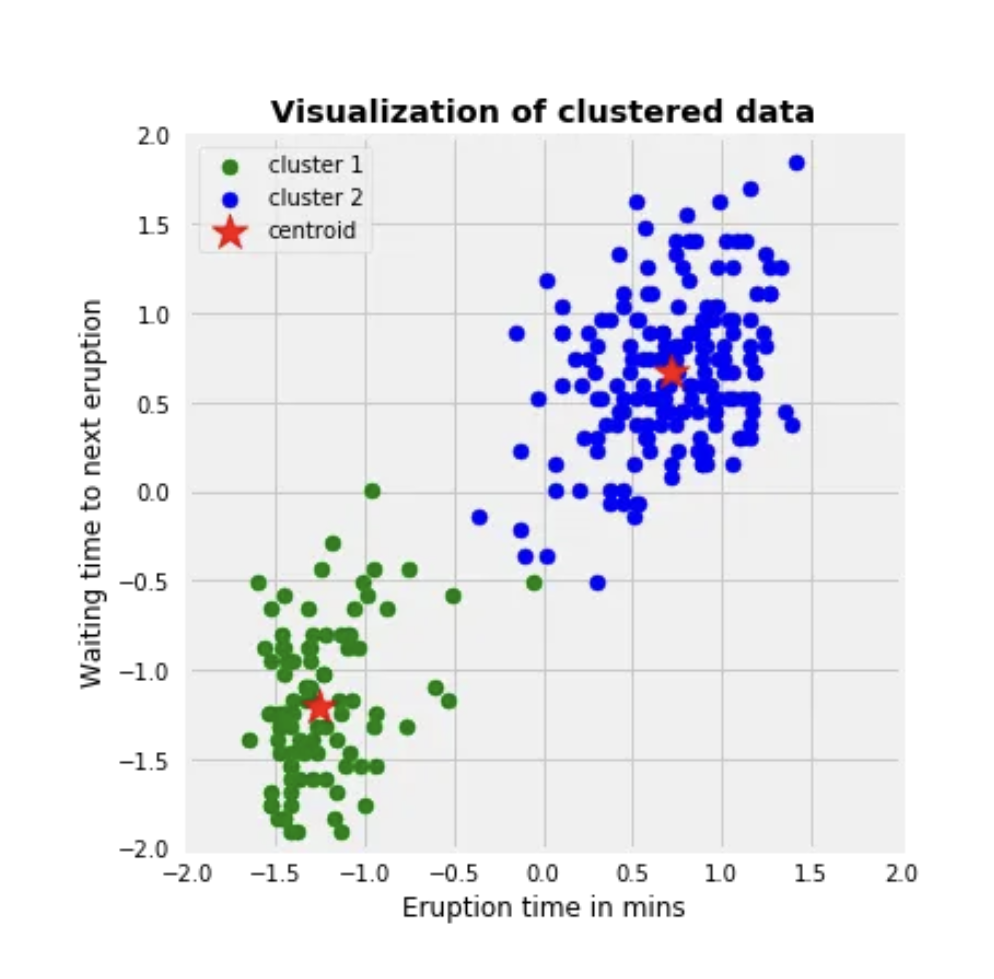
* Compute the average distance from all data points in the same cluster (ai).
* Compute the average distance from all data points in the closest cluster (bi).
* Compute the coefficient:pasted-image.tiff
* If it is 0 –> the sample is very close to the neighboring clusters.
* It it is 1 –> the sample is far away from the neighboring clusters.
* It it is -1 –> the sample is assigned to the wrong
* clusters.



\*Kmeans on Geyser’s Eruptions Segmentation

We'll start by applying the K-means algorithm to a 2D dataset, specifically the data from Old Faithful geyser in Yellowstone National Park. This dataset includes 272 observations with two features: eruption time in minutes and waiting time to the next eruption.

The goal is to find subgroups (clusters) within the data based on these features. Since it's a 2D dataset, we can easily visualize and identify the clusters. In this case, it's clear that we have 2 clusters. After standardizing the data, we run K-means with K=2.

The scatter plot below shows the data points colored by the clusters they belong to, with '\*' indicating the centroid of each cluster. These clusters represent different behaviors of the geyser under different conditions.

It's important to note that

different initializations of centroids can lead to different

results. To demonstrate this, we'll use 9 different random states to initialize centroids and plot the results. Each plot's title will

display the sum of squared

distances for each initialization.

While this example converges in less than 10 iterations due to its simplicity, real-world datasets are often more complex. Random initialization can

significantly impact convergence, as shown by the different clusterings based on initializations. We typically choose the initialization resulting in the lowest sum of squared distances.

