Classification using k-Nearest Neighbourhood(k-NN) with applications and comparison with some other methods

Adrija Bhar, Srijani Das, Yenisi Das

April 4, 2024

Abstract

The k-Nearest Neighbors (kNN) algorithm is a fundamental method in the field of machine learning and pattern recognition, widely used for classification tasks. It is a non-parametric and instance-based learning algorithm that makes predictions based on the majority class of its k nearest neighbors in the feature space. Despite its simplicity, kNN exhibits powerful performance, especially in scenarios where the decision boundaries are non-linear or the data distribution is complex. This project aims to explore the theoretical foundations of the kNN algorithm, including its underlying principles, key concepts, and mathematical formulations. Additionally, it investigates various strategies for parameter tuning, distance metric selection, and handling imbalanced datasets to enhance the classification accuracy of kNN. Through theoretical analysis and experimental validation, this project aims to provide insights into the strengths, limitations, and best practices of the kNN algorithm for classification tasks.

Contents

I Classification using k-NN **4**

1 What is Classification? **4**
1.1 Solving Classification Problem: . . . 4
1.2 Understanding Different Types of Classification Problems . . . 5
1.3 Examples of Classification Problems . . . 5

2 Introduction to k-NN **5**

3 Key Characteristics of k-NN: **5**
3.1 Supervised Learning : . . . 5
3.2 Lazy Learning : . . . 6
3.3 Instance-based Learning . . . 6
3.4 Non-Parametric : . . . 6

4 Different Distance Metrics: **7**
4.1 Euclidean distance: . . . 7
4.2 Manhattan distance: . . . 7
4.3 Minkowski distance: . . . 7

5 k-Nearest Neighbor Learning **8**

6 The k-Nearest Neighbour Algorithm : **8**

7 Weighted k-NN Algorithm : **9**
7.1 Distance Weighted k-NN Algorithm : . . . 9
7.2 Kernel-weighted k-NN Algorithm : . . . 10

8 Understanding Decision Boundary in k-NN: **11**
8.1 Introduction: . . . 11
8.2 Mathematical Explanation of Decision Boundary in k-NN: . . . 11
8.3 Types of Decision Boundaries in k-NN: . . . 12
8.4 Factors that Influence Decision Boundary in k-NN: . . . 12

9 Visualization of the Decision Boundary in k-NN: **12**
9.1 Visualization method - 1 . . . 12
9.2 Visualization method - 2: Voronoi Diagram . . . 13

10 How to choose the optimum k (Number of Neighbors for k-NN)? **14**

11 Discriminant Analysis **14**
11.1 Linear Discriminant Analysis (LDA) . . . 14
11.2 Quadratic Discriminant Analysis (QDA) . . . 15

12 Data Description **15**

13 Implementing k-NN to drug consumption data **16**
13.1 Applying k-NN algorithm in data corresponding to the Heroin drug . . . 16
13.2 Findings after applying k-NN algorithm in data corresponding to the Heroin drug . 17
13.3 Applying weighted k-NN algorithm in data corresponding to the Heroin drug 17
13.4 Findings after applying weighted k-NN algorithm in data corresponding to the Heroin drug . . . 19
13.5 Exploratory analysis of the data corresponding to the Heroin drug . . . 19
13.5.1 Findings from the exploratory analysis of the data corresponding to the Heroin drug . . . 19

13.5.2 Scatterplot and Decision Boundaries - 1 20
13.5.3 Findings from Scatterplot and Decision Boundaries - 1 20
13.5.4 Scatterplot and Decision Boundaries - 2 20
13.5.5 Findings from Scatterplot and Decision Boundaries - 2 21
13.6 Exploratory analysis of the data corresponding to the Cannabis drug 22
13.6.1 Scatterplot and Decision Boundaries - 1 22
13.6.2 Findings from Scatterplot and Decision Boundaries - 1 23
13.6.3 Scatterplot and Decision Boundaries - 2 23
13.6.4 Scatterplot and Decision Boundaries - 3 23
13.6.5 Scatterplot and Decision Boundaries - 4 24
13.6.6 Scatterplot and Decision Boundaries - 5 24
13.7 Applying weighted k-nn to Cannabis data 24
13.8 Findings after applying weighted k-nn to Cannabis data 26

14 Comparison of kernel-weighted k-NN with QDA **27**

15 Demerits **28**
15.1 The curse of dimensionality 28

16 Common Uses of k-NN : **28**

17 References **29**

18 Acknowledgement **29**

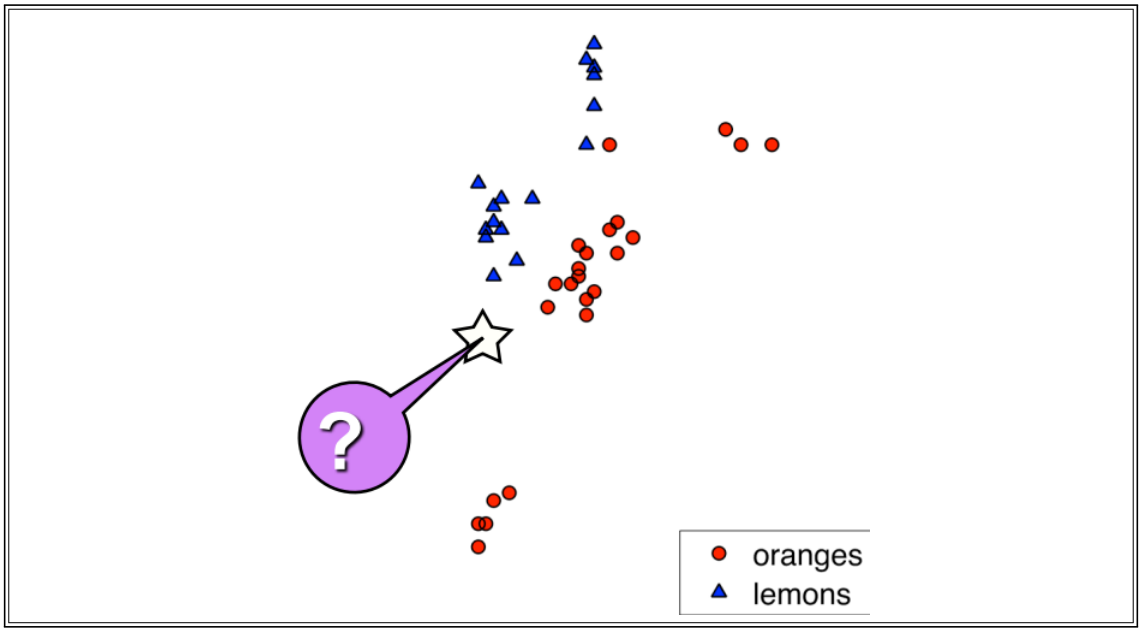
Part I
Classification using k-NN

1 What is Classification?

Classification problems are tasks where we need to sort things into different groups based on how similar they are to certain characteristics. These groups are called classes, and they consist of things that are alike in some way. We decide which class something belongs to by comparing its features, like color, shape, or size, to those of each class. Each class has its own label that helps us identify it. So, classification is about figuring out which group something fits into based on its features.

For example, let's think about sorting fruits. Imagine you have two baskets labeled A, B. Basket A has lemons(blue colour dots in the graph) and basket B has oranges(red colour dots). Now, if we get a new fruit and need to decide where to put it, you're facing a classification problem. We'll look at the fruit's type (whether it's a lemon or orange) and then place it in the basket that already has fruits of the same type. So, if the new fruit is a lemon, we'll add it to basket A with the other lemons. This is similar to how we classify things based on their similarities.

We are visualizing this classification problem through the graph below.Star reoresents the new fruit.



1.1 Solving Classification Problem:

We tackle classification problems by training specialized models. These models learn from examples we provide, where each example includes an object and its corresponding label. During training, the models learn to recognize patterns and similarities among objects belonging to the same class.

Once trained, we test the model using separate data it hasn't seen before. During testing, we only give the model the object itself, without telling it which class it belongs to. Then, the model predicts which class the object most likely belongs to.

We measure how well the model performs by checking how many times it correctly predicts the label of the objects in the testing data. This measure of accuracy helps us understand how good our classification model is at recognizing different objects.

1.2 Understanding Different Types of Classification Problems

- **Binary Classification:** These are classification problems where there are only two classes to choose from.
- **Multi-Class Classification:** In these problems, there are more than two classes to classify objects into.
- **Multi-Label Classification:** Here, objects can belong to more than one class simultaneously.
- **Imbalanced Classification:** These problems occur when there's an unequal distribution of objects among the classes, making some classes have many more objects than others.

1.3 Examples of Classification Problems

- **Spam Detection:** Deciding if an email is spam or not by looking at its content and features.
- **Image Classification:** Figuring out what's in a picture, like recognizing numbers in handwriting or identifying animals in photos.
- **Medical Diagnosis:** Deciding if a person's medical condition is normal or abnormal using their health data and test results.
- **Customer Churn Prediction:** Guessing if a customer might stop using a service by studying how they've used it before.
- **Sentiment Analysis:** Understanding if a piece of writing is expressing a positive, negative, or neutral feeling.

Among many algorithms for classification problems, we will discuss **k-Nearest Neighbourhood Method (k-NN).**

2 Introduction to k-NN

In the realm of supervised machine learning, the k-Nearest Neighbors (k-NN) algorithm stands as a versatile and intuitive approach to classification tasks. Rooted in the principle of similarity, k-NN offers a straightforward yet powerful method for pattern recognition and decision making. By leveraging the notion that similar instances tend to belong to the same class, KNN assigns labels to unclassified data points based on the consensus of their nearest neighbors. Despite the emergence of more complex algorithms, k-NN maintains its relevance due to its simplicity, interpretability, and effectiveness across various domains. This introductory guide explores the fundamentals of the k-NN classifier, its underlying principles, practical applications, and considerations for optimal performance.

3 Key Characteristics of k-NN:

3.1 Supervised Learning :

"Supervised" means that the algorithm learns from input-output pairs, where each input data point is associated with a corresponding output label or class. k-NN is considered a supervised learning algorithm because it requires labeled training data during its training phase. Then, when presented with new, unlabeled data, it uses the labeled data points to classify or predict the label of the new data point based on the labels of its nearest neighbors. Thus, the supervision provided by the labeled training data guides the algorithm in making accurate predictions for unseen instances.

3.2 Lazy Learning :

In k-NN we find the k training points $x_{(r)}, r = 1, 2, ..., k$ closest in distance to x_0, given a query point x_0, and then we classify it using majority vote among the k neighbors. For this reason, kNN is also called a lazy learning algorithm. Now, What it means to be a lazy learning algorithm?

Lazy learning, as its name suggests, defers processing of the training examples until prediction time. Essentially, the training phase involves nothing more than storing the training data itself. In any lazy learning algorithm, including k-Nearest Neighbors (k-NN), no actual learning occurs until a test example is provided. When a new observation is to be classified using k-NN, the algorithm identifies its k-nearest neighbors from the training dataset. These neighbors then participate in a "majority vote", where the class that appears most frequently among them is assigned to the new observation. This decision is made based on measuring the distance between the new data point and its neighbors.

3.3 Instance-based Learning

In contrast to constructing a single universal model, k-Nearest Neighbors (k-NN) generates a distinct local approximation for each data point, relying on both the data point itself and the training set. This approach, known as instance-based or "memory-based" learning.

In machine learning, an instance is a single observation of data. Instance-Based Learning (IBL) is a broader category of learning algorithms to which k-NN belongs. Instance-based learning methods such as nearest neighbor consists of simply storing the presented training data. When a new query instance is encountered, a set of similar related instances is retrieved from memory and used to classify the new query instance. This approach is also referred to as memory-based or lazy learning because, unlike eager learning algorithms that construct a general, abstract model of the data during training, IBL algorithms store instances of the training data and delay the generalization process until a new query is made to the system.

3.4 Non-Parametric :

Lastly, because we do not make any assumption about the functional form of the kNN algorithm, a kNN model is also considered a nonparametric model. However, classifying kNN as either discriminative or generative is less straightforward compared to other algorithms. While under certain assumptions, it's possible to estimate conditional and marginal probabilities, kNN primarily focuses on modeling posterior probabilities directly, typically represented as $p(x|f(x))$, where x is a data point and $f(x)$ represents its class label. For more detailed insights on kNN from a Bayesian perspective, refer to the corresponding section later in this document.

Hence, in short the key characteristics of k-NN is given by,

1. Memory-Based: This characteristic allows these algorithms to adapt quickly if new data is added to the dataset.

2. No Explicit Generalization: IBL algorithms use the entire dataset to make predictions, which involves searching through the stored instances to find the most similar examples.

3. Lazy Learning: Because generalization happens at the time of prediction, IBL algorithms are considered "lazy."

4. Similarity Measurement: The core mechanism of IBL is to measure the similarity (or distance) between instances. The specific method of calculating similarity can vary, but common measures include Euclidean distance, Manhattan distance, and Cosine similarity.

5. Non-parametric: k-Nearest Neighbors (k-NN) is considered a nonparametric model due to its lack of assumptions about the underlying data distribution or functional form.

4 Different Distance Metrics:

k-NN is compatible with any pairwise distance metric. However, the choice of the distance metric affects the runtime performance of the algorithm. For instance, computing the Mahalanobis distance is much more expensive than calculating the more straightforward Euclidean distance. There are a few conditions that the distance metric must satisfy:

- **Non-negativity:** $d(x, y) \geq 0$
- **Identity:** $d(x, y) = 0$ if and only if $x == y$
- **Symmetry:** $d(x, y) = d(y, x)$
- **Triangle Inequality:** $d(x, y) + d(y, z) \geq d(x, z)$

Some distance metrics are given below:

4.1 Euclidean distance:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

The Euclidean distance is suitable for data that has continuous and numerical features with similar scales and ranges. It can also handle outliers and noise well, as it gives more weight to larger differences. However, it can be affected by the curse of dimensionality, which means that as the number of features increases, the distance between any two points becomes less meaningful and more similar.

4.2 Manhattan distance:

$$d_{\mathrm{T}}(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_{\mathrm{T}} = \sum_{i=1}^{n} |p_i - q_i|$$

The Manhattan distance is suitable for data that has discrete and categorical features, as it does not penalize small differences as much as the Euclidean distance. It can also handle high-dimensional data better, as it is less sensitive to the curse of dimensionality. However, it can be influenced by the orientation and scale of the features, as it assumes that all directions are equally important and all units are comparable.

- **Mahalanobis distance:** Given a probability distribution Q on $\mathbb{R}^N$ and two points $\boldsymbol{p}$ and $\boldsymbol{q}$ $\in \mathbb{R}^N$, the Mahalanobis distance between them with respect to Q is

  $$d(\boldsymbol{p}, \boldsymbol{q}; Q) = \sqrt{(\boldsymbol{p} - \boldsymbol{q})' S^{-1} (\boldsymbol{p} - \boldsymbol{q})}.$$

  where S is the covariance matrix.

4.3 Minkowski distance:

$$d\left(\boldsymbol{p}, \boldsymbol{q}\right) = \left(\sum_{i=1}^{n} |p_i - q_i|^r \right)^{\frac{1}{r}}.$$

The Minkowski distance is suitable for data that has mixed types of features, as it allows you to adjust the parameter p to balance the importance of different features and distances. However, it can be computationally expensive and difficult to interpret, as the parameter p can have different effects on different data sets and problems.

To choose the best distance metric we can try the following ways:

- Experiment with Different Metrics: Try out various distance metrics and see how they perform on your data. Compare their results and use techniques like cross-validation to measure accuracy.
- Consider Evaluation: Think about the computational complexity and interpretability of each metric. Some may be more complex or easier to understand.
- Use Domain Knowledge: Use knowledge about the problem domain when selecting a metric.

5 k-Nearest Neighbor Learning

The most basic instance-based method is the k-Nearest Neighbor algorithm. This algorithm assumes all instances correspond to points in the n-dimensional space $\mathbb{R}^n$. The nearest neighbors of an instance are defined in terms of the standard Euclidean distance. A feature vector is a vector containing multiple elements about an object. More precisely, let an arbitrary instance $\boldsymbol{x}$ be described by the feature vector $(a_1(\boldsymbol{x}), a_2(\boldsymbol{x}), ..., a_n(\boldsymbol{x}))$ where $a_r(\boldsymbol{x})$denotes the value of the rth attribute of instance $\boldsymbol{x}$. The "closeness" or similarity between instances is quantified using distance metrics, with the standard Euclidean distance being the most common measure given by ,

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{r=1}^{n} (a_r(\boldsymbol{x}_i) - a_r(\boldsymbol{x}_j))^2}$$

Thus, in a classification task, k-NN uses the distances to identify the k-nearest neighbors to a query instance, and then assigns the most common class among these neighbors to the query. A target function represents the underlying rule or pattern that the algorithm tries to learn from the training data in order to make predictions on new, unseen data. In nearest-neighbor learning the target function may be either discrete-valued or real-valued. **Let us first consider learning discrete-valued target functions of the form** $f : \mathbb{R}^n \rightarrow V$**, where** V **is the finite set** $v_l, ...v_s$.

6 The k-Nearest Neighbour Algorithm :

The k-nearest neighbour algorithm estimates a discrete valued target function at the query instance x_q .The value $\hat{f}(x_q)$ returned by this algorithm as the estimate of $f(x_q)$, is just the most common value of f among the k training examples nearest to x_q. For a clear understanding of the concept, first we shall go through the nearest neighbour algorithm which is the simplest version of k-NN(with k=1).

If we choose k = 1, then the **1-NEAREST NEIGHBOUR** algorithm assigns $\hat{f}(x_q)$ to the value $f(x_i)$ where x_i is the training instance nearest to x_q.

Example: Let's say we have a dataset of images of handwritten digits (0-9). Each image is represented as a matrix of pixel values. We want to classify a new handwritten digit image based on its similarity to the images in our dataset using the 1-nearest neighbor algorithm.

- Each image is labeled with the corresponding digit it represents, ie, $V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- Suppose we have a new handwritten digit image(x_q) that we want to classify. This image is also represented as a matrix of pixel values.
- We calculate the distance$[d(\boldsymbol{x}_i, \boldsymbol{x}_j)]$ between the input image and each image in our dataset. This distance can be calculated using various methods, such as Euclidean distance or Manhattan distance, considering the pixel values as features.
- We find the image(suppose x_i) in our dataset that is closest (most similar) to the input image based on the calculated distances. This image is our nearest neighbour.

- We assign the label of the nearest neighbour($f(x_i)$) to the input image($\hat{f}(x_q)$). In other words, if the nearest neighbour image represents the digit 6, we classify the input image as digit 6.
- Hence we have classified the input image based on the 1-nearest neighbour algorithm.

This process demonstrates how the 1-nearest neighbour algorithm can be used for image classification tasks. It finds the most similar image in the dataset to the input image and assigns the same label to the input image.

The k-NEAREST NEIGHBOR algorithm for approximating a discrete-valued target function:

- **Training algorithm**: For each training example $(x, f(x))$, add the example to the list of training examples.
- **Classification algorithm**: Given a query instance x_q to be classified, Let $x_1, ..., x_k$ denote the k instances from training examples that are nearest to x_q.

Decide the class of x_q by the majority voting:

$$\hat{f}(x_q) = \underset{v \in V}{argmax} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where the Kronecker delta $\delta(a, b) = 1$ if $a = b$ and $\delta(a, b) = 0$ otherwise. **NOTE:** The **k-NEAREST NEIGHBOR** algorithm never forms an explicit general hypothesis regarding the target function $f(.)$. It simply computes the classification of each new query instance as needed.

7 Weighted k-NN Algorithm :

In case of k-NN, only the k nearest neighbors influence the prediction; however, this influence is the same for each of these neighbors, although the individual similarity to (x_q, x_i) might be widely different. To reach this aim, the distances, on which the search for the nearest neighbors is based , have to be transformed into similarity measures, which can be used as weights.

7.1 Distance Weighted k-NN Algorithm :

One refinement to the k-NEAREST NEIGHBOUR algorithm is to weight the contribution of each of the k neighbors according to their distance to the query point x_q giving greater weight to closer neighbors. For example, we might weight the vote of each neighbour according to the inverse square of its distance from x_q.

This can be accomplished by approximating the target function at the query point as,

$$\hat{f}(x_q) = \underset{v \in V}{argmax} \sum_{i=1}^{k} w\delta(v, f(x_i))$$

wherem

$$w = \frac{1}{d(x_i, x_q)^2}$$

To accommodate the case where the query point x_q, exactly matches one of the training instances x_i , the denominator $d(x_i, x_q)^2$ turns out to be zero, hence, we assign $\hat{f}(x_q)$ to be $f(x_i)$ in this case. If there are several such training examples, we assign the majority classification among them. The above variants of the k-NEAREST NEIGHBOUR algorithm consider only the k nearest neighbors to classify the query point. Once we add distance weighting, there is really no harm in allowing all training examples to have an influence on the classification of x_q, because very distant examples will have very little effect on $\hat{f}(x_q)$. The only disadvantage of considering all examples is

that our classifier will run more slowly. If all training examples are considered when classifying a new query instance, we call the algorithm a **global method**. If only the nearest training examples are considered, we call it a **local method**. The distance-weighted k-NEAREST NEIGHBOR algorithm is a highly effective inductive inference method for many practical problems. It is robust to noisy training data and quite effective when it is provided a sufficiently large set of training data. By taking the weighted average of the k neighbors nearest to the query point, it can smooth out the impact of isolated noisy training examples. The **INDUCTIVE BIAS of K-NN** corresponds to an assumption that the classification of an instance x, will be most similar to the classification of other instances that are nearby in Euclidean distance or the distance metric chosen.

7.2 Kernel-weighted k-NN Algorithm :

Kernels are functions K(.) of the distances d(corresponding to the distance metric chosen) with maximum in d = 0 and values that get smaller with growing absolute value of d. Thus the following properties must hold:

- K(d) ≥ 0 for all $d \epsilon \mathbb{R}$
- K(d) gets its maximum for d = 0
- K(d) descents monotonously for d □ ±∞

Typical examples for this kind of function are the following:

- rectangular kernel : $\frac{1}{2}$ I($|d| \leq 1$)
- triangular kernel :$(1 - |d|) \cdot$ I($|d| \leq 1$)
- Epanechnikov kernel : $\frac{3}{4}$ $(1 - d^2) \cdot$ I($|d| \leq 1$)
- quartic or biweight kernel : $\frac{15}{16}$ $(1 - d^2)^2 \cdot$ I($|d| \leq 1$)
- triweight kernel : $\frac{35}{32}$ $(1 - d^2)^3 \cdot$ I($|d| \leq 1$)
- cosine kernel : $\frac{\pi}{4}\cos(\frac{\pi}{2}d) \cdot$ I($|d| \leq 1$)
- Gauss kernel : $\frac{1}{\sqrt{2\pi}} exp(-\frac{d^2}{2})$ In the case of distances, which are defined as strictly positive values, of course only the positive domain of K has to be used.Every kernel function needs either a window width, if the values become zero in a certain distance from the maximum value, or a dispersion parameter, if the values are larger than zero for all $d \epsilon \mathbb{R}$. In weighted k-NN both are selected automatically according to the distance of the last neighbor, let that be x_{k+1}, that is not taken into consideration any more. This is done implicitly by standardization of all other distances with the distance of the (k + 1)th neighbor: $D(x_q, x_i) = \frac{d(x_q, x_i)}{d(x_q, x_{k+1})}$ i = 1,2,...,k . These standardized distances always take values within the interval [0, 1]. During implementation we add a small constant ϵ> 0 to $d(x_q, x_{k+1})$ in order to avoid weights of 0 for some of the nearest neighbors. This could happen if one or more of these neighbors show exactly the same distance as the (k + 1)th, as most of the kernels become 0 at the window boundary D = 1. This band width of 1 is an adequate choice, as all observations with a larger distance from x than the kth neighbor have no influence on the prediction. So the choice of the band width is adaptively based on the data. The main target of this extended method is to gain a technique, that up to a certain degree is independent of a bad choice for k resulting in a high misclassification error.Now this number of nearest neighbors is implicitly hidden in the weights: If k is too large k is adjusted to a lower value automatically.In this case a small number of neighbors with large weights dominates the other neighbors, whose classes have no influence on the prediction because of their low weights. The target function is approximated by,

$$\hat{f}(x_q) = argmax_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

$$where, w_i = K(D(x_i, x_q))$$

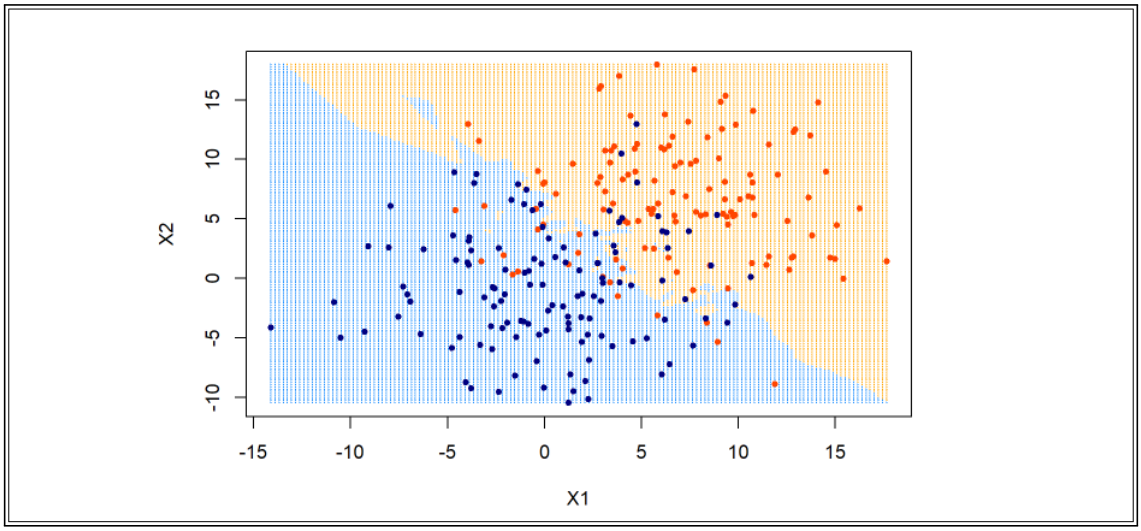
8 Understanding Decision Boundary in k-NN:

8.1 Introduction:

The decision boundary in k-NN is super important because it tells us how different classes are separated in the feature space. It's like drawing lines or shapes that help us decide which class a data point belongs to based on its neighbors.

Let's break it down into two main parts:

- **Decision Boundary for Binary Classification:** In binary classification, we're dealing with two classes, like positive and negative. The decision boundary is what splits the feature space into two parts, one for each class. When we have a new data point, we check which side of the boundary it falls on to classify it. This boundary is defined by points where there's an equal number of nearest neighbors from each class. These are the points where the classifier might be unsure and needs more information to make a decision. As we move away from this boundary, the majority class becomes clearer, and the classifier gets more confident in its predictions. The shape and position of this boundary depend on things like the distance between points, how many neighbors we consider (k), and how the data is spread out.
- **Decision Boundary for Multiclass Classification:** In multiclass classification, where we have more than two classes, things get a bit more complex. We still have boundaries, but now they divide the feature space into multiple regions, each corresponding to a different class. These boundaries work the same way as in binary classification, but now they can have different shapes and forms because there are more classes to consider.



8.2 Mathematical Explanation of Decision Boundary in k-NN:

Let's consider a binary classification scenario with two classes: Class $\mathcal{A}$ and Class $\mathcal{B}$. The decision boundary is where the number of nearest neighbors from Class $\mathcal{A}$ equals the number from Class $\mathcal{B}$.

Let's say we have a new data point, X, that needs classification. We calculate the distance between X and all data points in the training set, then select the k-nearest neighbors. Denoting NA as the count of neighbors from Class A and NB as the count from Class B among the k-nearest neighbors, the decision boundary is given by:

$$\mathrm{NA} = \mathrm{NB}$$

This equation signifies that the decision boundary comprises points where the count of Class $\mathcal{A}$ neighbors equals the count of Class $\mathcal{B}$ neighbors.

For multiclass classification, each class has its own decision boundary. For Class $\mathcal{A}$ the boundary is where the count of nearest neighbors from Class $\mathcal{A}$ is the highest among all classes.

Mathematically, in multiclass classification, the decision boundary can be expressed as:

$$max\{\text{NA}, \text{NB}, \text{NC}, \ldots\} = \text{Ni}$$

Here, NA, NB, NC, etc., represent the counts of neighbors from Class $\mathcal{A}$, Class $\mathcal{B}$, Class $\mathcal{C}$, and so on, among the k-nearest neighbors. Ni represents the maximum count among these for each class.

8.3 Types of Decision Boundaries in k-NN:

1. Li**near Decision Boundary:** This boundary separates classes with a straight line (or plane, or hyperplane in higher dimensions). It's based on how the data is spread out and the distance between points.
2. **Non-Linear Decision Boundary:** When classes aren't neatly separated with straight lines, we get a non-linear boundary. It can have curves, spirals, or other shapes to capture the complex relationships in the data.
3. **Varying Decision Boundary:** The shape of the boundary might change across different parts of the data. In sparse regions, it might be smoother and more linear, while in crowded areas, it might be more complex.
4. **Smooth vs. Discrete Decision Boundary:** A smooth boundary changes gradually between classes, while a discrete one has sharp edges.
5. **Adaptive Decision Boundary:** The boundary adjusts to the local data distribution. In areas with one dominant class, the boundary leans towards that class.
6. **Boundary with Outliers:** Outliers or noisy data can mess with the boundary, leading to misclassifications or making it less stable.

8.4 Factors that Influence Decision Boundary in k-NN:

1. **Choice of Distance Metric**: The type of distance measurement used, like Euclidean or Manhattan distance, can change the shape of the decision boundary. Different metrics focus on different aspects of the data, so they can lead to different boundaries.
2. **Value of k:** The number of neighbors considered for classification matters too. If k is small, the boundary might be more jagged and sensitive to noise. But if k is large, the boundary might be smoother, which could simplify things but might also miss important details.
3. **Distribution of the Data:** How the data points are spread out across the feature space is important. If they're well-separated and evenly balanced between classes, the boundary might be clear and straight. But if there's overlap between classes or one class has a lot more points than the others, the boundary could be more complicated.
4. **Dimensionality of the Feature Space:** If there are a lot of dimensions (features) in the data, it can affect the boundary. High-dimensional spaces can be sparse, which means the boundary might not be as clear or stable.
5. **Noise and Outliers:** Noise and outliers—those pesky data points that don't fit the pattern—can mess with the boundary too. They can make it less certain or even move its position.

9 Visualization of the Decision Boundary in k-NN:

9.1 Visualization method - 1

Seeing the decision boundary in k-NN can really help us understand how the algorithm works. But it's tough when we have lots of features. In two or three dimensions, we can plot the boundary

directly. For more dimensions, we might use tricks like Principal Component Analysis to simplify things.

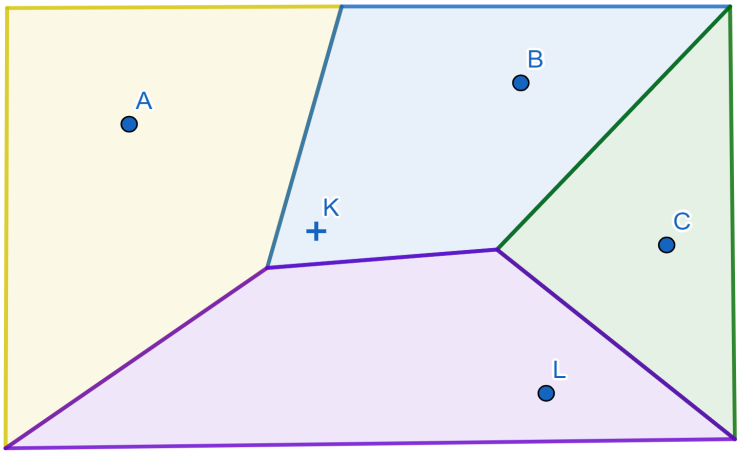
Here's how to visualize the decision boundary:

1. **Picking a Subset of Features:** Choose just a couple of features to focus on, like two or three dimensions. This makes it easier to see the boundary.
2. **Making a Grid:** Create a grid of points in this subset of features. We want to cover the whole area we're interested in.
3. **Classifying Points:** For each point in the grid, use k-NN to decide which class it belongs to. Then, color the point based on its predicted class. When we plot all these points, we see the decision boundary where the colors change.

9.2 Visualization method - 2: Voronoi Diagram

Using Euclidean distance, when we have **two training examples**, the decision boundary between them is just a **straight line**. If a new point falls exactly on this boundary, it means it's the same distance away from both training examples. But when we consider the **entire training set** (> 2), the decision boundary of the Nearest Neighbor model it becomes a bunch of connected, curved shapes called **polyhedra**. Inside each shape, all points are closest to one specific training example. Outside the shape, points are closer to a different training example. This helps the model decide which group a point belongs to based on where it falls relative to these shapes.

This partitioning of regions on a plane in 2D is also called "Voronoi diagram" or Voronoi tessellation.In a Voronoi diagram, each straight line is at the same distance from two different training examples. But at a point called a vertex or node, this distance is the same from three training examples. So, when we're drawing the decision boundary of a nearest neighbor classifier in two dimensions, we combine the straight lines between pairs of training examples from the same class. This helps us define the boundary between different classes.



'**Note: What is special about Voronoi diagram** → Imagine we have a plane with several points scattered across it. Now, we want to divide this plane into sections so that any point we pick from a section is closest to the point that defines that section, rather than any other point on the plane. These sections end up having shapes like polygons.

For example, let's say we have a Voronoi diagram for 4 points (A, B, C, and D) on the plane. Now, if we pick a random point, let's call it K, we want to know which of the original points (A, B, C, or D) is closest to K.

One way to find out is by calculating the distances between K and each of the points (KA, KB, KC, and KD).

Another way is to see which polygon K falls into. In the Voronoi diagram, if K lies within the polygon associated with point B, then B is the closest neighbor of K.

10 How to choose the optimum k (Number of Neighbors for k-NN)?

When using the k-NN (K-Nearest Neighbors) algorithm, the "k" value is crucial. It tells us how many nearby points to consider when making predictions for a new data point. Picking the right "k" is important because it affects how accurate our model is and how well it works with new data.

Methods to find the best "k":

- **Cross-Validation:** Use methods like cross-validation to try out different "k" values and see which one gives the best results. This helps ensure that the chosen "k" works well with new data too.
- **Elbow Method:** Plot the error rate or accuracy against different "k" values and look for the point where the improvement starts to level off. This point is often called the "elbow," and it helps us find a good "k" value.
- **Domain Knowledge:** Think about the specific problem you're working on and what you know about the data. Sometimes, certain "k" values make more sense based on what you understand about the situation. For instance, if you're trying to predict the price of a house based on its features, you might know from experience that looking at the prices of the ten nearest houses might give a good estimate.
- **Square Root of N rule:** If you don't have much other information to go on, you can start with a simple rule of thumb. Set "k" to the square root of the total number of data points in your dataset. For example, if you have 100 data points, you might start with "k" as 10.

Finding the best "k" involves finding a balance between making accurate predictions and not being too sensitive to small changes in the data. It's like deciding how many neighbors to ask for advice when trying to choose a restaurant in a new neighborhood. You don't want too few opinions (biased), but you also don't want so many that it's overwhelming (variance). Experimenting with different "k" values and understanding your data well can help you find the right balance for your KNN model.

11 Discriminant Analysis

Discriminant analysis is a statistical technique used to classify or predict a categorical dependent variable based on one or more continuous or binary independent variables.

Assume the prior probability or the marginal pmf for class k is denoted as π_k .

π_k is usually estimated simply by empirical frequencies of the training set:

$$\hat{\pi_k} = \frac{\text{Number of samples in kth class}}{\text{Total number of samples}}$$

Assume that the samples of the k^{th}class has density $f_k(x)$.

11.1 Linear Discriminant Analysis (LDA)

This type of discriminant analysis is used when all the predictor variables are continuous and normally distributed, and the groups have equal covariance matrices. LDA seeks to find a linear combination of the predictors that separates the groups as much as possible.

Suppose there are m populations(classes) and we assume that the k^{th} polpulation has a multivariate normal distribution $N_p(\boldsymbol{\mu_k}, \sum)$for all k=1,2,...,m.Let $C(j|i)$ be the cost of missclassifying an instance **x** from the i^{th} population into the j^{th} population where i,j=1,2,...,m.

Note that $C(i|i)$ = 0 for all i=1,2,...,m.

We classify **x(query object)** to the l^{th} population if

$$d_l(\boldsymbol{x}) = log(\pi_l) + \boldsymbol{\mu_l} \sum^{-1} \boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu_l^T} \sum^{-1} \boldsymbol{\mu_l}$$

is the largest among the m populations.(Classification Rule)

Since we don't know the values of the actual parameters we use their estimates to work out the classification rule.

11.2 Quadratic Discriminant Analysis (QDA)

QDA is similar to LDA, but it does not assume that the groups have equal covariance matrices. This means that it can model more complex group boundaries, but it also requires estimating more parameters than LDA and can be more prone to overfitting.

Suppose there are m populations(classes) and we assume that the k^{th} polpulation has a multivariate normal distribution $N_p(\boldsymbol{\mu_k}, \sum_k)$for all k=1,2,...,m. We classify **x(query object)** to the l^{th} population if

$$d_l^Q(\boldsymbol{x}) = -\frac{1}{2}log|\sum_l| + log(\pi_l) - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_l})^T \sum_l^{-1} (\boldsymbol{x} - \boldsymbol{\mu_l})$$

,is the largest among the m populations.(Classification Rule)

Since we don't know the values of the actual parameters we use their estimates to work out the classification rule.

12 Data Description

The dataset used for the KNN classification analysis is sourced from the UCI Machine Learning Repository. It comprises responses collected through an anonymous online survey methodology conducted by Elaine Fehrman, encompassing 2051 participants. The dataset is publicly available online.

Each respondent in the dataset is associated with twelve attributes:

1. **Personality Measurements**: This includes scores on the Big Five personality traits - Neuroticism (N), Extraversion (E), Openness to Experience (O), Agreeableness (A), and Conscientiousness (C) obtained from the NEO-FFI-R inventory.

- **Openness to experience:** is a broad appreciation for art, emotion, adventure, and curiosity. Those high in openness tend to be intellectually curious, emotionally aware, and willing to try new things. They often possess a rich vocabulary, vivid imagination, and enjoy reflecting on ideas. Conversely, those low in openness may be pragmatic and less interested in abstract concepts.
- **Conscientiousness:** reflects self-discipline, attention to detail, and a preference for planned behavior. High conscientiousness individuals are organized and reliable, while low conscientiousness individuals may appear less disciplined and more spontaneous.
- **Extraversion:** describes engagement with the external world, with extraverts enjoying social interactions and being perceived as energetic and talkative. Introverts, on the other hand, have lower social engagement and may appear quieter and more independent.
- **Agreeableness:** is characterized by concern for social harmony and consideration for others. Agreeable individuals are generally kind, generous, and empathetic, while those low in agreeableness may be more skeptical and competitive.

- **Neuroticism:** refers to the tendency to experience negative emotions such as anxiety and depression. High neuroticism individuals are emotionally reactive and prone to stress, while those low in neuroticism are more emotionally stable and less easily upset.

2. **Impulsivity**: Assessed using the Barratt Impulsiveness Scale (BIS-11).
3. **Sensation Seeking (SS)**: Derived from the Impulsiveness Sensation Seeking scale (ImpSS).
4. **Level of Education (Edu.)**: Indicates the educational attainment of the respondent.
5. **Age:** The age of the respondent.
6. **Gender**: Categorization of respondents based on gender.
7. **Country of Residence**: The country in which the respondent resides.
8. **Ethnicity**: Ethnic background of the respondent.

The dataset also includes information on the consumption of 18 central nervous system psychoactive drugs, along with one fictitious drug, Semeron. These drugs are:

1. Alcohol 2. Amphetamines 3. Amyl nitrite 4. Benzodiazepines 5. Cannabis 6. Chocolate 7. Cocaine 8. Caffeine 9. Crack 10. Ecstasy 11. Heroin 12. Ketamine 13. Legal highs 14. LSD 15. Methadone 16. Magic mushrooms 17. Nicotine 18. Volatile Substance Abuse (VSA).

Participants indicated their usage patterns for each drug, categorized into different timeframes ranging from never used, used over a decade ago, to usage in the last decade, year, month, week, or day.

The drugs were classified into three main categories based on their pharmacological effects:

1. **Depressants:** Including alcohol, amyl nitrite, benzodiazepines, tranquilizers, gamma-hydroxybutyrate solvents and inhalants, and opiates such as heroin and methadone/prescribed opiates.
2. **Stimulants:** Consisting of amphetamines, nicotine, cocaine powder, crack cocaine, caffeine, and chocolate.
3. **Hallucinogens:** Comprising cannabis, ecstasy, ketamine, LSD, and magic mushrooms.

Legal highs, such as mephedrone and salvia, along with various legal smoking mixtures, were also included in the dataset.

The primary objective of the study was to analyze the relationship between personality traits, impulsivity, sensation-seeking, demographic data, and drug consumption patterns. The study aimed to identify associations between personality profiles and drug consumption, as well as predict individual drug consumption risk based on personality profiles.

We use the data after clubbing the last 3 different timeframes, so now we have data ranging from never used, used over a decade ago, to usage in the last decade, year, less than a month. The data has been quantified using numerical values for various categories of each attribute. It can be found here.

13 Implementing k-NN to drug consumption data

13.1 Applying k-NN algorithm in data corresponding to the Heroin drug

We have 18 drugs and 2051 participants. We shall implement k-NN technique over this data. We shall consider a single drug , Heroine, and subset the data.The 5 levels of Heroine intake are our 5 classes. We split the data into training set and test set in 7:3 ratio. We fit the k-NN based on the training data and use the test data to check how well k-NN is working as a classifier.

We use the knn() function in R to learn the data based on the training set. We used 10-fold C.V to choose the optimal value of k. Next we classify the observations in the test set,(we considered them as query instances) based on k-NN and the training data learnt by it .Next we calculated the proportion of data rightly classified into each class and drew the confusion matrix. We also

computed the accuracy of the k-NN method which is proportion of data rightly classified to the total number of observations.

```
k-Nearest Neighbors

1322 samples
  10 predictor
   5 classes: 'CL0', 'CL1', 'CL2', 'CL3', 'CL4'

No pre-processing
Resampling: Cross-Validated (9 fold)
Summary of sample sizes: 1174, 1175, 1175, 1176, 1174, 1176, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.7292623  0.042723896
   3  0.8146713  0.071978636
   5  0.8350952  0.041902296
   7  0.8457085  0.017576920
   9  0.8479658  0.008812641
  11  0.8510048  0.012753431
  13  0.8502437  0.003975353
  15  0.8502437  0.000000000
  17  0.8502437  0.000000000
  19  0.8502437  0.000000000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 11.

predictions CL0 CL1 CL2 CL3 CL4
        CL0 480  20  28  19  15
        CL1   0   0   0   0   0
        CL2   0   0   0   0   0
        CL3   0   0   0   0   0
        CL4   1   0   0   0   0
[1] "Accuracy: 0.852575488454707"
```

13.2 Findings after applying k-NN algorithm in data corresponding to the Heroin drug

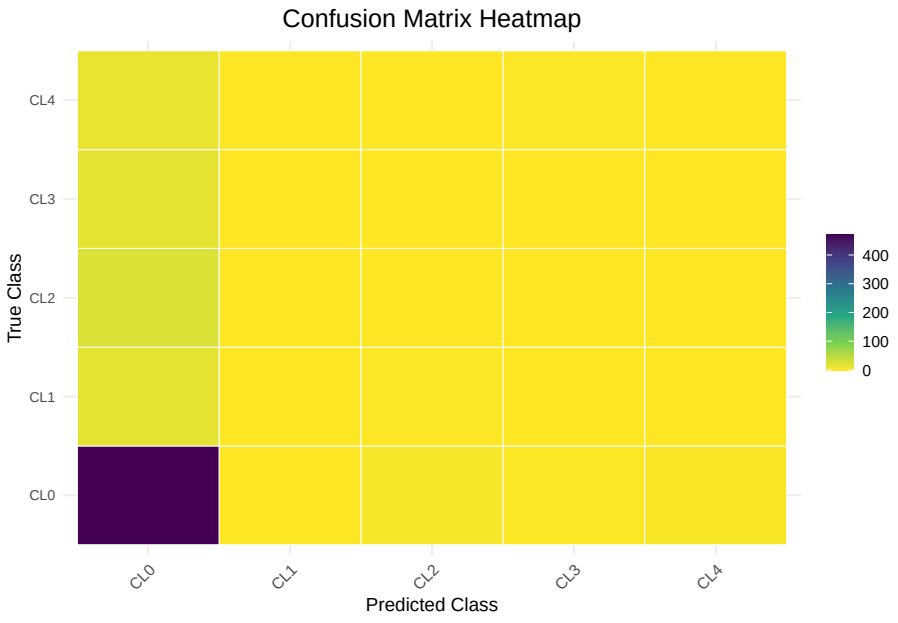
- The accuracy turns out to be 0.852, which is pretty high so we can conclude that k-NN is working very good.
- The confusion matrix obtained turns out to be $\begin{bmatrix} Prediction & CL0 & CL1 & CL2 & CL3 & CL4 \\ CL0 & 480 & 20 & 28 & 19 & 15 \\ CL1 & 0 & 0 & 0 & 0 & 0 \\ CL2 & 0 & 0 & 0 & 0 & 0 \\ CL3 & 0 & 0 & 0 & 0 & 0 \\ CL4 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$ which is a bit disturbing since it is classifying almost every observation to class CL0.

13.3 Applying weighted k-NN algorithm in data corresponding to the Heroin drug

Now we perform **kernel -weighted k-NN** on the data and the kernel chosen for this is optimal kernel. We again choose the optimal k by 10-fold cross validation.

```
k-Nearest Neighbors

1322 samples
  10 predictor
   5 classes: 'CL0', 'CL1', 'CL2', 'CL3', 'CL4'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1189, 1191, 1189, 1191, 1190, 1190, ...
Resampling results across tuning parameters:

  kmax  distance  Accuracy   Kappa
   1    1         0.7201200  0.03662267
   1    2         0.7337512  0.06492139
```

```
   1    3         0.7307724  0.06588494
   2    1         0.7201200  0.03662267
   2    2         0.7337512  0.06492139
   2    3         0.7307724  0.06588494
   3    1         0.7201200  0.03662267
   3    2         0.7337512  0.06492139
   3    3         0.7307724  0.06588494
   4    1         0.7201200  0.03662267
   4    2         0.7337512  0.06492139
   4    3         0.7307724  0.06588494
   5    1         0.7935396  0.06912893
   5    2         0.7958410  0.08222155
   5    3         0.7890052  0.07948669
   6    1         0.8124685  0.06868073
   6    2         0.8094320  0.06992163
   6    3         0.8086457  0.06490012
   7    1         0.8185236  0.05581190
   7    2         0.8245499  0.06221474
   7    3         0.8207503  0.06208540
   8    1         0.8275747  0.05820904
   8    2         0.8290898  0.06120297
   8    3         0.8306164  0.06198249
   9    1         0.8336184  0.05145910
   9    2         0.8351508  0.05697992
   9    3         0.8351450  0.04938512
  10    1         0.8366546  0.03270867
  10    2         0.8389330  0.03989589
  10    3         0.8396849  0.02826465

Tuning parameter 'kernel' was held constant at a value of optimal
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were kmax = 10, distance = 3 and kernel
 = optimal.
[1] "Accuracy: 0.838365896980462"

prediction CL0 CL1 CL2 CL3 CL4
       CL0 472  19  28  19  14
       CL1   0   0   0   0   0
       CL2   5   1   0   0   0
       CL3   1   0   0   0   1
       CL4   3   0   0   0   0
```



13.4 Findings after applying weighted k-NN algorithm in data corresponding to the Heroin drug

- The accuracy turns out to be 0.838.
- The confusion matrix obtained turns out to be $\begin{bmatrix} Prediction & CL0 & CL1 & CL2 & CL3 & CL4 \\ CL0 & 472 & 19 & 28 & 19 & 14 \\ CL1 & 0 & 0 & 0 & 0 & 0 \\ CL2 & 5 & 1 & 0 & 0 & 0 \\ CL3 & 1 & 0 & 0 & 0 & 1 \\ CL4 & 3 & 0 & 0 & 0 & 0 \end{bmatrix}$ according to which the weighted k-NN fails to classify correctly any data point coming from a class other than CLo .

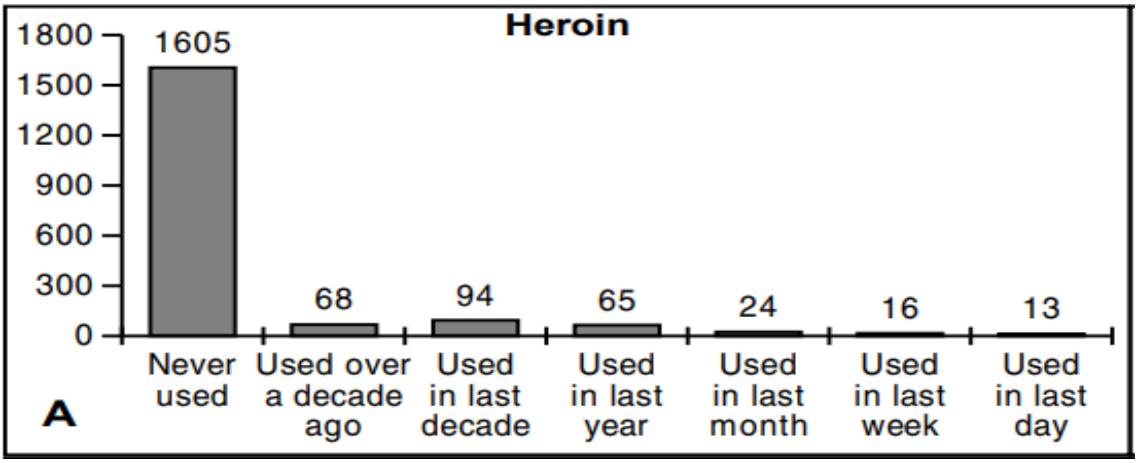
A plausible explanation of such occurrences may be that a very large proportion of data(both train and test) falls under CLo, therefore, the k-NN is assigning almost all new query to CLo and the accuracy is being overestimated.

The weighted k-NN, although shows similar results has atleast reduced the accuracy so that the we can get a glimpse of the actual performance of k-NN method corresponding to this dataset.

13.5 Exploratory analysis of the data corresponding to the Heroin drug

We explore the drug data a bit to see why such misleading results are occurring.

13.5.1 Findings from the exploratory analysis of the data corresponding to the Heroin drug



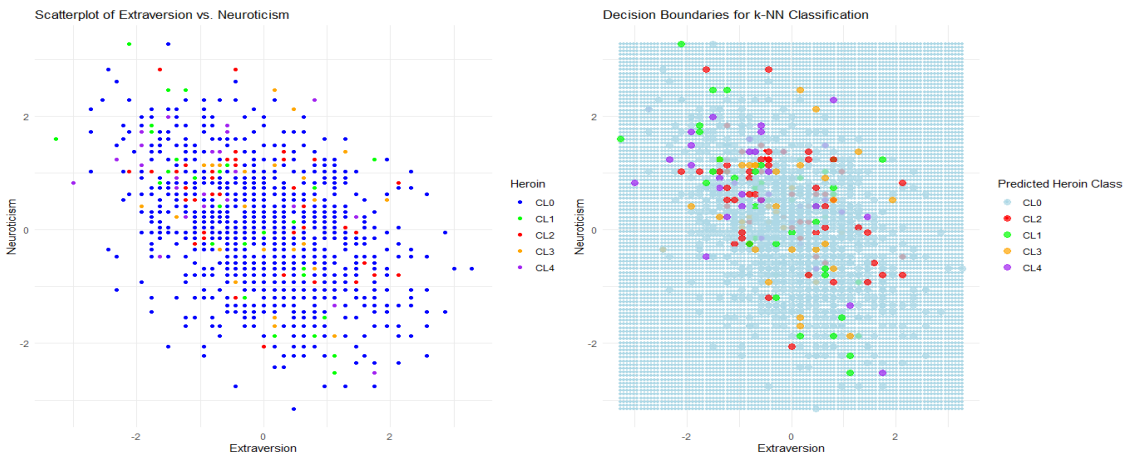
- This is a column diagram of observations where the bar represents the number of data points belonging to each class before CL4,CL5 and CL6 are clubbed into a single class.From the diagram we see that CLo has an extremely large frequency compared to the other classes, hence whenever a new query instance is occurring , k-NN is assigning it to the class CLo.

13.5.2 Scatterplot and Decision Boundaries - 1

To visualize this phenomenon more clearly we have graphed a scatterplot of the drug data with "neuroticism" on the y axis and "extraversion" on the x axis, the colour of each point corresponds to the class in which it belongs(after clubbing CL4, CL5, CL6 into a single class).We have also graphed the decision boundaries if a k-NN is performed over the data by considering "agreeableness", and "conscientiousness" as the only attributes.

13.5.3 Findings from Scatterplot and Decision Boundaries - 1

- Scatterplot shows that a large proportion of data falls under the class CLo as we had seen in the column diagram.
- A very interesting observation came up in the plot of the decision boundary, the entire region has been colored blue which shows that any new data point will be assigned to CLo if k-NN is based on only the two above mentioned attributes.



13.5.4 Scatterplot and Decision Boundaries - 2

Now we graph a scatterplot of the drug data with "agreeableness" on the y axis and "openness_to_experience" on the x axis, the colour of each point corresponds to the class in which it belongs(after clubbing

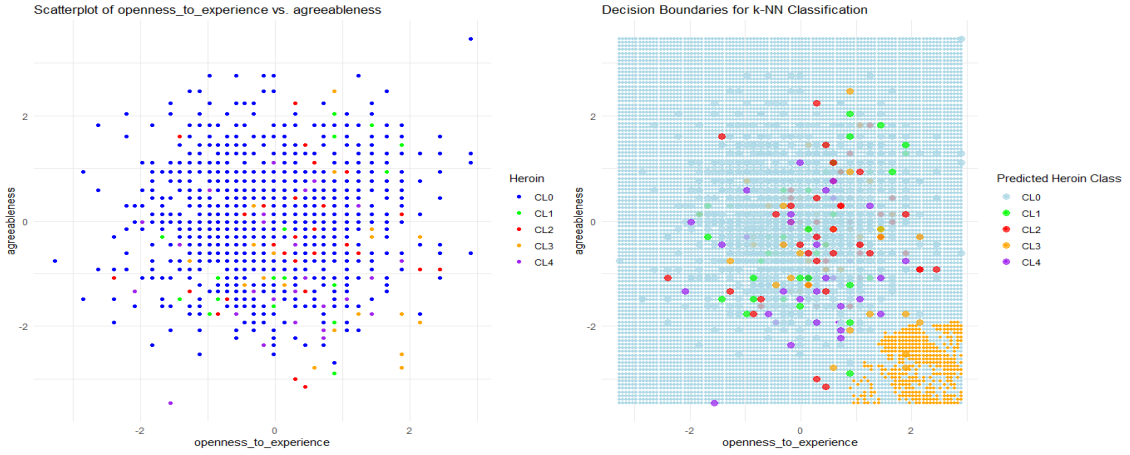
CL4, CL5, CL6 into a single class).We have also graphed the decision boundaries if a k-NN is performed over the data by considering , “agreeableness” and “openness_to_experience” as the only attributes.

13.5.5 Findings from Scatterplot and Decision Boundaries - 2

- Scatterplot shows that a large proportion of data falls under the class CL0 as we had seen in the column diagram.
- A very large portion of the entire region has been colored blue which shows that most of the new data points will be assigned to CL0 if k-NN is based on only the two above mentioned attributes.

Since for both cases such observations have come up, we can fairly conclude that due to a large proportion of data falling under a particular class the accuracy of k-NN is being overestimated.

Therefore, we shall try to work with some other drug in which the the observations are more or less evenly distributed(atleast not like Heroin), so that we can fairly judge the performance of k-NN.




```
Shapiro-Wilk normality test

data:  Z
W = 0.4759, p-value < 2.2e-16

Box's M-test for Homogeneity of Covariance Matrices

data:  Y
Chi-Sq (approx.) = 706.68, df = 220, p-value < 2.2e-16
Call:
qda(cannabis ~ ., data = train_data)

Prior probabilities of groups:
      CL0       CL1       CL2       CL3       CL4
0.2193646 0.1096823 0.1414523 0.1119516 0.4175492

Group means:
           age      gender  education neuroticism extraversion
CL0  0.5088298  0.19298400 -0.3570060 -0.19946731  0.060117897
CL1  0.7577026  0.07652814 -0.3120994 -0.20517110 -0.005246069
CL2  0.1543481  0.11610000 -0.3064888  0.09729134 -0.068733155
CL3 -0.2112033 -0.03911838 -0.3053145  0.28074196 -0.158029324
CL4 -0.3779846 -0.16256804 -0.2839512  0.09191772 -0.035375797
    openness_to_experience agreeableness conscientiousness impulsivity
CL0            -0.54938386     0.2674941        0.46888659 -0.50569579
CL1            -0.39345538     0.1367559        0.29323366 -0.28663455
CL2            -0.14037086    -0.1441344        0.05085551 -0.01836914
CL3             0.06672047    -0.1693428       -0.26181514  0.09106588
CL4             0.44862458    -0.1453314       -0.30664868  0.32766172
    sensation_seeking
CL0        -0.6900145
CL1        -0.3792782
CL2        -0.1316187
```

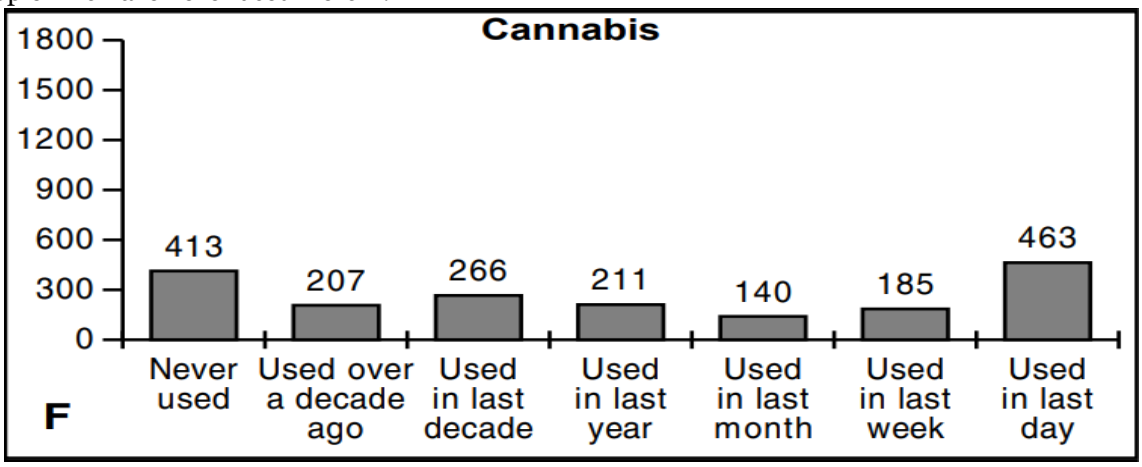
```
CL3         0.1639857
CL4         0.4442319
[1] 0.4849023
```

We observed from the column diagram that the data for Heroin actually shows that 1605 people have never used it! Less than 100 people have used Heroin both in the last decade and in the last year. This is why the decision boundary plot mostly seems to be a blue rectangle. There are not enough training data so that we can mark a region to assign to some class other than the class of people who have never used Heroin.

To find out a more even distribution over the timeframes, we plotted the column diagrams for other drugs and found the data for Cannabis to be more or less evenly spread. We proceed in the next subsection about our analysis on Cannabis.

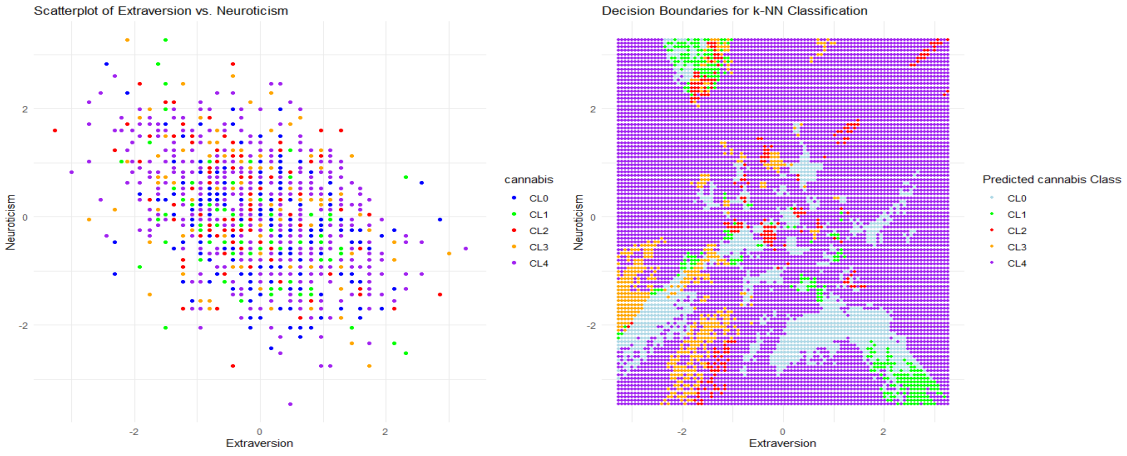
13.6 Exploratory analysis of the data corresponding to the Cannabis drug

We observed from the column diagram that the data for Heroin actually shows that 1605 people have never used it! Less than 100 people have used Heroin both in the last decade and in the last year. This is why the decision boundary plot mostly seems to be a blue rectangle. There are not enough training data so that we can mark a region to assign to some class other than the class of people who have never used Heroin.



To find out a more even distribution over the timeframes, we plotted the column diagrams for other drugs and found the data for Cannabis to be more or less evenly spread. We proceed in the next subsection about our analysis on Cannabis.

13.6.1 Scatterplot and Decision Boundaries - 1



13.6.2 Findings from Scatterplot and Decision Boundaries - 1

We plot the decision boundaries using 2 of the 5 personality trait attributes to observe how the KNN-Classification works.

On the left, we have a scatter plot that maps individual data points according to their scores in Extraversion and Neuroticism. Each point represents an individual, and the color of the point indicates the class of cannabis use, with different classes from CL0 to CL4.

On the right, there is a plot illustrating the decision boundaries for a k-Nearest Neighbors (k-NN) classification model. This plot is a visual representation of how the k-NN algorithm has classified the entire space of possible Extraversion and Neuroticism scores into categories of cannabis use (again from CL0 to CL4). Each small square represents a point in the feature space which is of the form (extraversion, neuroticism), and the color of the square indicates the predicted class of cannabis use for that point.

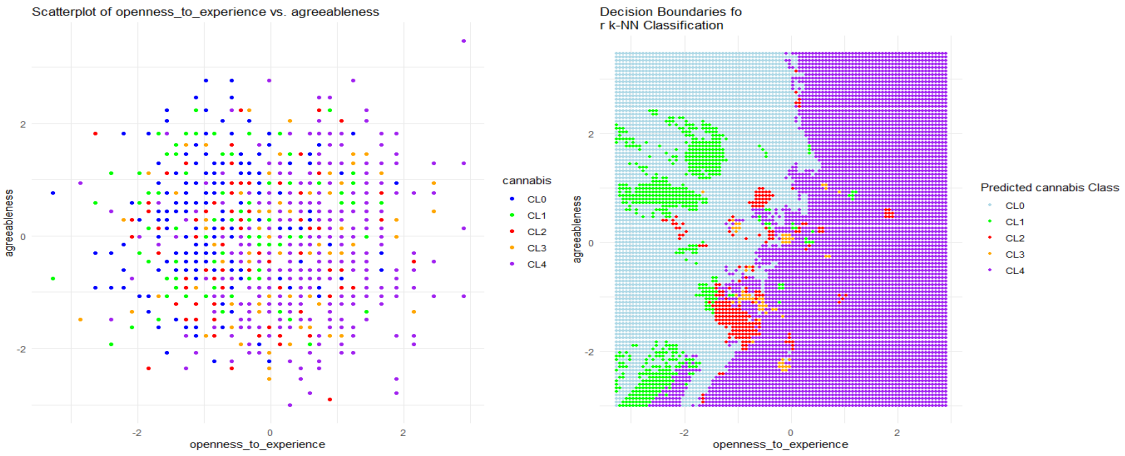
The decision boundary itself is where the color changes, signifying a shift from one predicted class to another. These boundaries show how any new individual, based on their Extraversion and Neuroticism scores, would be classified by the model in terms of their predicted cannabis use.

From the plot, we can see that the decision boundaries are not linear, which means that the relationship between the traits and cannabis use classes is not simple or direct; instead, it varies in a more complex way across the trait spectrum. It's also evident that some classes have very mixed boundaries with each other, suggesting overlap in the traits' scores among different classes of cannabis use.

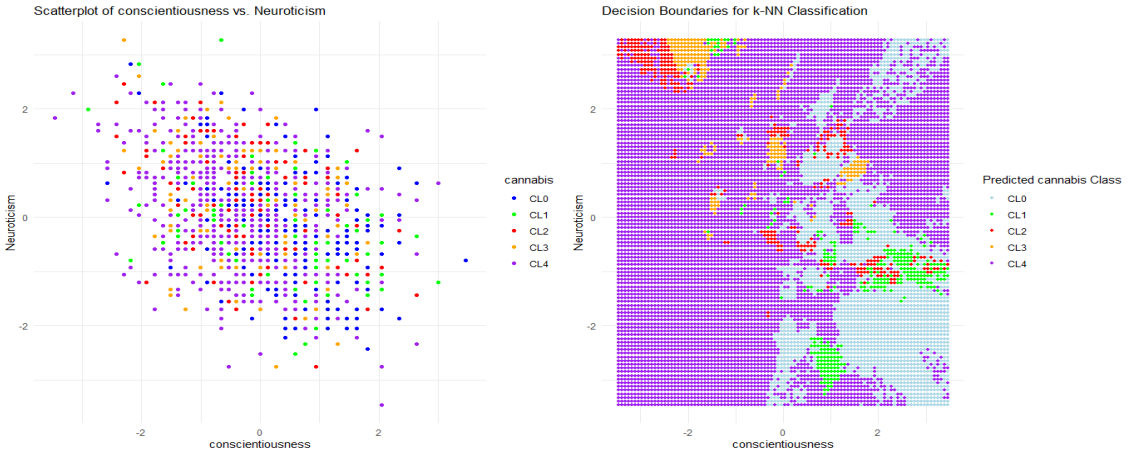
Overall, the decision boundary plot is a useful way to understand the model's predictions and to identify how different regions of the feature space are being associated with the various cannabis use classes. It also provides insight into the model's complexity and how well it may handle new data.

We do this for a couple of more combinations from the personality traits and the decision boundary plots are given below.

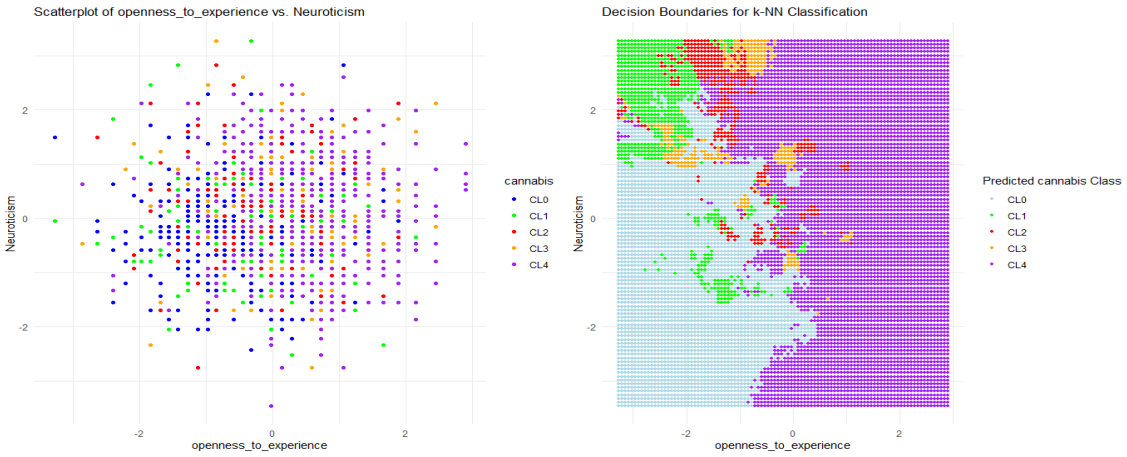
13.6.3 Scatterplot and Decision Boundaries - 2



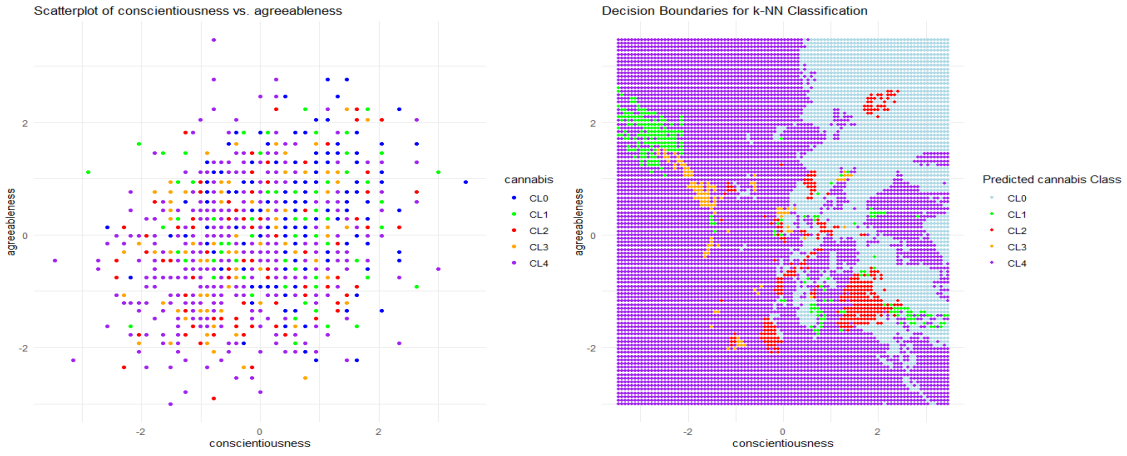
13.6.4 Scatterplot and Decision Boundaries - 3



13.6.5 Scatterplot and Decision Boundaries - 4



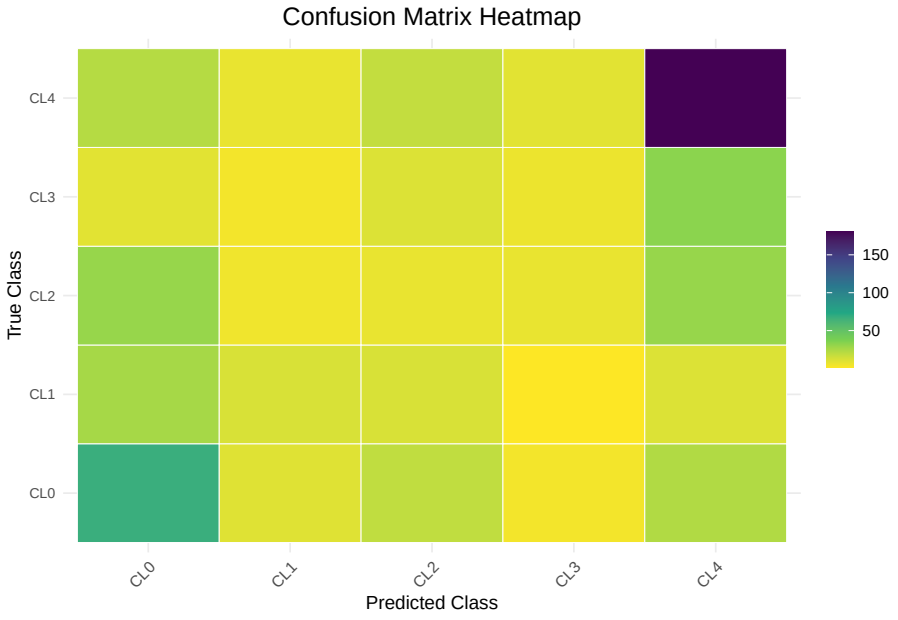
13.6.6 Scatterplot and Decision Boundaries - 5

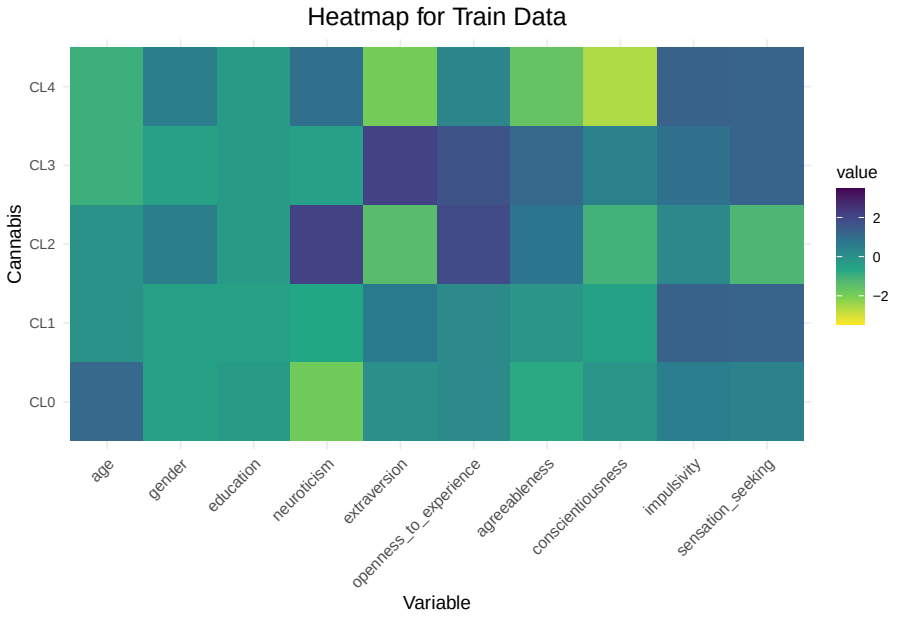


13.7 Applying weighted k-nn to Cannabis data

We perform kernel -weighted k-NN on the data and the kernel chosen for this is Optimal. We again choose the optimal k by 10-fold cross validation.

```
k-Nearest Neighbors

1322 samples
  10 predictor
   5 classes: 'CL0', 'CL1', 'CL2', 'CL3', 'CL4'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1190, 1191, 1189, 1190, 1191, 1189, ...
Resampling results across tuning parameters:

  kmax  distance  Accuracy   Kappa
   1    1         0.4154324  0.1976143
   1    2         0.4176598  0.2026132
   1    3         0.4063300  0.1903283
   2    1         0.4154324  0.1976143
   2    2         0.4176598  0.2026132
   2    3         0.4063300  0.1903283
   3    1         0.4154324  0.1976143
   3    2         0.4176598  0.2026132
   3    3         0.4063300  0.1903283
   4    1         0.4154324  0.1976143
   4    2         0.4176598  0.2026132
   4    3         0.4063300  0.1903283
   5    1         0.4464660  0.2297835
   5    2         0.4517754  0.2380575
   5    3         0.4434703  0.2286591
   6    1         0.4562350  0.2375608
   6    2         0.4691436  0.2559399
   6    3         0.4646203  0.2510591
   7    1         0.4720938  0.2538109
   7    2         0.4880324  0.2762588
   7    3         0.4759274  0.2629658
   8    1         0.4872176  0.2709078
   8    2         0.4933069  0.2807030
   8    3         0.4850356  0.2699347
   9    1         0.4962576  0.2817848
   9    2         0.4993453  0.2875882
   9    3         0.4918084  0.2773857
  10    1         0.5045627  0.2906189
  10    2         0.5046255  0.2912954
  10    3         0.4948047  0.2784229

Tuning parameter 'kernel' was held constant at a value of optimal
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were kmax = 10, distance = 2 and kernel
 = optimal.
[1] "Accuracy: 0.483126110124334"

predictions CL0 CL1 CL2 CL3 CL4
        CL0  67  26  30   9  22
        CL1  10  12   5   4   7
        CL2  19  12   7  11  18
        CL3   4   1   7   6   9
        CL4  23  11  30  33 180
```





13.8 Findings after applying weighted k-nn to Cannabis data

• The accuracy turns out to be 0.483.

• The confusion matrix obtained turns out to be

$$\begin{bmatrix} Prediction & CL0 & CL1 & CL2 & CL3 & CL4 \\ CL0 & 67 & 26 & 30 & 9 & 22 \\ CL1 & 10 & 12 & 5 & 4 & 7 \\ CL2 & 19 & 12 & 7 & 11 & 18 \\ CL3 & 4 & 1 & 7 & 6 & 9 \\ CL4 & 23 & 11 & 30 & 33 & 180 \end{bmatrix}$$

according to which the weighted k-NN correctly classifies a lot of observations in every class as compared to the confusion matrix from obtained using Heroin which evident from the diagonal of the confusion matrix.

14 Comparison of kernel-weighted k-NN with QDA

We wish to compare k-NN with some other method of classification. We shall perform discriminant analysis[LDAor QDA] on drug data. Since k-NN is a non-parametric method we did not have to assume anything about the underlying distribution of the datapoints coming from various classes but in case of LDA we need assumptions of normality and homoscedasticity of the classes and in case of QDA we need the assumption of normality for the data.

We subset the train data into groups, each group consists of data which belongs to a particular class. We get a total of 5 such groups corresponding to the 5 classes.We perform the multivariate shapiro wilk's test on each of the groups to check for multivariate normality for each of the groups, using the mshapiro.test() function in R.

```
$CL0

Shapiro-Wilk normality test

data:  Z
W = 0.97768, p-value = 0.0001693


$CL1

Shapiro-Wilk normality test

data:  Z
W = 0.33681, p-value < 2.2e-16


$CL2

Shapiro-Wilk normality test

data:  Z
W = 0.51105, p-value < 2.2e-16


$CL3

Shapiro-Wilk normality test

data:  Z
W = 0.5099, p-value < 2.2e-16


$CL4

Shapiro-Wilk normality test

data:  Z
W = 0.44153, p-value < 2.2e-16
```

We see that for each group the p-value is less than 0.0002 so we reject (for each group) the null hypothesis that the underlying distribution of the class is multivariate normal at 0.05 level of significance.

We perform BoxM test to check for homoscedasticity .

The p value turns out to be 0.484 hence we reject the null hypoyhesis that the classes are homoscedastic under the null hypothesis

Therefore, we perform QDA , though the data is not normal , for practical purposes QDA works fine.

```
Call:
qda(cannabis ~ ., data = train_data)

Prior probabilities of groups:
      CL0       CL1       CL2       CL3       CL4
0.2193646 0.1096823 0.1414523 0.1119516 0.4175492

Group means:
           age      gender  education neuroticism extraversion
CL0  0.5088298  0.19298400 -0.3570060 -0.19946731  0.060117897
CL1  0.7577026  0.07652814 -0.3120994 -0.20517110 -0.005246069
CL2  0.1543481  0.11610000 -0.3064888  0.09729134 -0.068733155
CL3 -0.2112033 -0.03911838 -0.3053145  0.28074196 -0.158029324
```

```
CL4 -0.3779846 -0.16256804 -0.2839512  0.09191772 -0.035375797
    openness_to_experience agreeableness conscientiousness impulsivity
CL0            -0.54938386     0.2674941        0.46888659 -0.50569579
CL1            -0.39345538     0.1367559        0.29323366 -0.28663455
CL2            -0.14037086    -0.1441344        0.05085551 -0.01836914
CL3             0.06672047    -0.1693428       -0.26181514  0.09106588
CL4             0.44862458    -0.1453314       -0.30664868  0.32766172
    sensation_seeking
CL0        -0.6900145
CL1        -0.3792782
CL2        -0.1316187
CL3         0.1639857
CL4         0.4442319
[1] 0.4849023
```

We perform QDA using the qda() function in R and the accuracy turns out to be 0.84 while accuracy of kernel-weighted k-NN turns out to be 0.483 , hence we conclude that for this dataset QDA is working much better than kernel weighted k-NN.

15 Demerits

15.1 The curse of dimensionality

It refers to a phenomenon where the performance of certain algorithms, such as k-Nearest Neighbor (k-NN), deteriorates as the dimensionality of the feature space increases. In the context of k-NN, this issue arises because distance-based similarity measures become less reliable in high-dimensional spaces, leading to misleading results.

When applying k-NN to a problem with a high-dimensional feature space, such as one where instances are described by a large number of attributes, only a subset of these attributes may be relevant for determining the classification. In such cases, instances with identical values for the relevant attributes may still be distant from each other in the high-dimensional space due to the presence of irrelevant attributes.

Consequently, the similarity metric used by k-NN, which relies on all attributes in the instance space, may be skewed by the large number of irrelevant attributes. As a result, the distance between neighbors may be dominated by these irrelevant attributes, leading to suboptimal classification performance. This issue is particularly pronounced in k-NN approaches, which rely solely on local information and are highly sensitive to variations in distance metrics. Other machine learning algorithms, such as rule-based and decision tree learning systems, have mechanisms for feature selection or dimensionality reduction, which can mitigate the effects of irrelevant attributes on classification performance. Overall, the curse of dimensionality poses a significant challenge in high-dimensional data spaces, and it underscores the importance of feature selection, dimensionality reduction, and careful consideration of algorithmic choices when working with such data

16 Common Uses of k-NN :

Although neural networks are increasingly favored in computer vision and pattern recognition, k-nearest neighbors (KNN) models maintain prominence, particularly in the intersection of these fields with biometrics. Notwithstanding its straightforwardness, KNN has demonstrated success across a wide array of classification tasks, such as identifying handwritten digits, categorizing satellite image scenes, and analyzing EKG patterns.

17 References

- E. Fehrman, A.K. Muhammad, E.M. Mirkes, V. Egan, A.N. Gorban, The Five Factor Model of personality and evaluation of drug consumption risk. Accessed from: https://arxiv.org/abs/1506.06297
- Hechenbichler, Schliep: Weighted k-Nearest-Neighbor Techniques and Ordinal Classification Sonderforschungsbereich 386, Paper 399 (2004). Online unter: http://epub.ub.uni-muenchen.de/
- Mitchell, Machine Learning

18 Acknowledgement

We would like to thank Professor Swagata Nandi for her guidance and support in this endeavour.