KNN

Algorithms drive the machine learning world.

They're often praised for their predictive capabilities and spoken of as hard workers that consume huge amounts of data to produce instant results.

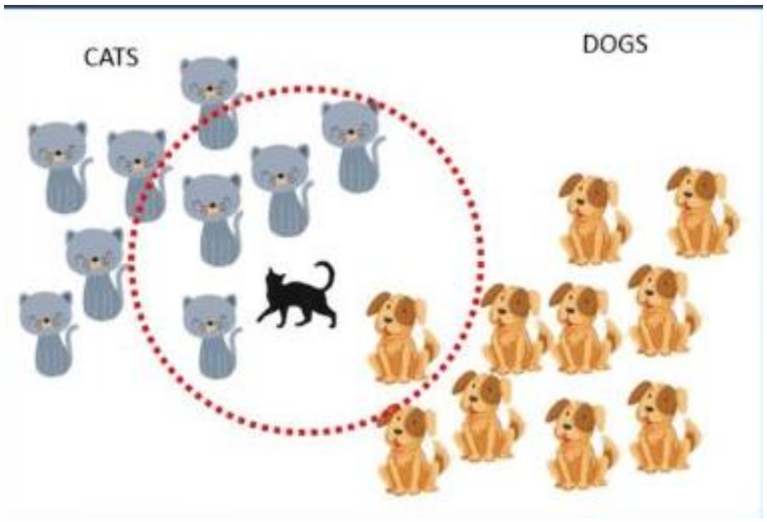
Among them, there's an algorithm often labeled as lazy. But it's quite a performer when it comes to classifying data points. It's called the k-nearest neighbors algorithm and is often quoted as one of the most important **machine learning** algorithms.

Consider the diagram below; it is straightforward and easy for humans to identify it as a “Cat” based on its closest allies. This operation, however, cannot be performed directly by the algorithm.

KNN calculates the distance from all points in the proximity of the unknown data and filters out the ones with the shortest distances to it. As a result, it’s often referred to as a distance-based algorithm.

In order to correctly classify the results, we must first determine the value of K (Number of Nearest Neighbours).

In the following diagram, the value of K is 5. Since there are four cats and just one dog in the proximity of the five closest neighbours, the algorithm would predict that it is a cat based on the proximity of the five closest neighbors in the red circle’s boundaries.



What is the k-nearest neighbors algorithm?

The **k-nearest neighbors (KNN) algorithm** is a data classification method for estimating the likelihood that a data point will become a member of one group or another based on what group the data points nearest to it belong to.

The k-nearest neighbor algorithm is a type of **supervised machine learning** algorithm used to solve classification and regression problems. However, it's mainly used for classification problems.

KNN is a *lazy learning* and *non-parametric* algorithm.

It's called a lazy learning algorithm or lazy learner because it doesn't perform any training when you supply the training data. Instead, it just stores the data during the training time and doesn't perform any calculations. It doesn't build a model until a query is performed on the dataset. This makes KNN ideal for **data mining**.

Did you know? The "K" in KNN is a parameter that determines the number of nearest neighbors to include in the voting process.

It's considered a non-parametric method because it doesn’t make any assumptions about the underlying data distribution. Simply put, KNN tries to determine what group a data point belongs to by looking at the data points around it.

Consider there are two groups, A and B.

To determine whether a data point is in group A or group B, the algorithm looks at the states of the data points near it. If the majority of data points are in group A, it's very likely that the data point in question is in group A and vice versa.

In short, KNN involves classifying a data point by looking at the nearest annotated data point, also known as the *nearest neighbor*.

Don't confuse K-NN classification with K-means clustering. KNN is a supervised classification algorithm that classifies new data points based on the nearest data points. On the other hand, K-means clustering is an **unsupervised** clustering algorithm that groups data into a K number of clusters.

How does KNN work?

As mentioned above, the KNN algorithm is predominantly used as a classifier. Let's take a look at how KNN works to classify unseen input data points.

Unlike classification using artificial neural networks, k-nearest neighbors classification is easy to understand and simple to implement. It's ideal in situations where the data points are well defined or non-linear.

In essence, KNN performs a voting mechanism to determine the class of an unseen observation. This means that the class with the majority vote will become the class of the data point in question.

If the value of K is equal to one, then we'll use only the nearest neighbor to determine the class of a data point. If the value of K is equal to ten, then we'll use the ten nearest neighbors, and so on.

Tip: Automate tasks and make data-driven decisions using **machine learning software**.

To put that into perspective, consider an unclassified data point X. There are several data points with known categories, A and B, in a scatter plot.

Suppose the data point X is placed near group A.

As you know, we classify a data point by looking at the nearest annotated points. If the value of K is equal to one, then we'll use only one nearest neighbor to determine the group of the data point.

In this case, the data point X belongs to group A as its nearest neighbor is in the same group. If group A has more than ten data points and the value of K is equal to 10, then the data point X will still belong to group A as all its nearest neighbors are in the same group.

Suppose another unclassified data point Y is placed between group A and group B. If K is equal to 10, we pick the group that gets the most votes, meaning that we classify Y to the group in which it has the most number of neighbors. For example, if Y has seven neighbors in group B and three neighbors in group A, it belongs to group B.

The fact that the classifier assigns the category with the highest number of votes is true regardless of the number of categories present.

You might be wondering how the distance metric is calculated to determine whether a data point is a neighbor or not.

There are four ways to calculate the distance measure between the data point and its nearest neighbor: **Euclidean distance**, **Manhattan distance**, **Hamming distance**, and **Minkowski distance**. Out of the three, Euclidean distance is the most commonly used distance function or metric.

K-nearest neighbor algorithm pseudocode

Programming languages like Python and R are used to implement the KNN algorithm. The following is the pseudocode for KNN:

```
1. Load the data
2. Choose K value
3. For each data point in the data:
       o Find the Euclidean distance to all training data samples
       o Store the distances on an ordered list and sort it
       o Choose the top K entries from the sorted list
       o Label the test point based on the majority of classes present in the
         selected points
4. End
```

To validate the accuracy of the KNN classification, a **confusion matrix** is used. Other statistical methods such as the likelihood-ratio test are also used for validation.

In the case of KNN regression, the majority of steps are the same. Instead of assigning the class with the highest votes, the average of the neighbors' values is calculated and assigned to the unknown data point.

Why use the KNN algorithm?

Classification is a critical problem in data science and machine learning. The KNN is one of the oldest yet accurate algorithms used for pattern classification and regression models.

Here are some of the areas where the k-nearest neighbor algorithm can be used:

- **Credit rating:** The KNN algorithm helps determine an individual's credit rating by comparing them with the ones with similar characteristics.
- **Loan approval:** Similar to credit rating, the k-nearest neighbor algorithm is beneficial in identifying individuals who are more likely to default on loans by comparing their traits with similar individuals.
- **Data preprocessing:** Datasets can have many missing values. The KNN algorithm is used for a process called **missing data imputation** that estimates the missing values.
- **Pattern recognition:** The ability of the KNN algorithm to identify patterns creates a wide range of applications. For example, it helps detect patterns in credit card usage and spot unusual patterns. Pattern detection is also useful in identifying patterns in customer purchase behavior.
- **Stock price prediction:** Since the KNN algorithm has a flair for predicting the values of unknown entities, it's useful in predicting the future value of stocks based on historical data.
- **Recommendation systems:** Since KNN can help find users of similar characteristics, it can be used in recommendation systems. For example, it can be used in an online video streaming platform to

suggest content a user is more likely to watch by analyzing what similar users watch.

- **Computer vision:** The KNN algorithm is used for image classification. Since it's capable of grouping similar data points, for example, grouping cats together and dogs in a different class, it's useful in several **computer vision** applications.

How to choose the optimal value of K

There isn't a specific way to determine the best K value – in other words – the number of neighbors in KNN. This means that you might have to experiment with a few values before deciding which one to go forward with.

One way to do this is by considering (or pretending) that a part of the training samples is "unknown". Then, you can categorize the unknown data in the test set by using the k-nearest neighbors algorithm and analyze how good the new categorization is by comparing it with the information you already have in the training data.

When dealing with a two-class problem, it's better to choose an odd value for K. Otherwise, a scenario can arise where the number of neighbors in each class is the same. Also, the value of K must not be a multiple of the number of classes present.

Another way to choose the optimal value of K is by calculating the sqrt(N), where N denotes the number of samples in the training data set.

However, K with lower values, such as K=1 or K=2, can be noisy and subjected to the effects of outliers. The chance for overfitting is also high in such cases.

On the other hand, K with larger values, in most cases, will give rise to smoother decision boundaries, but it shouldn't be too large. Otherwise, groups with a fewer number of data points will always be outvoted by other groups. Plus, a larger K will be computationally expensive.

Advantages and disadvantages of KNN

One of the most significant advantages of using the KNN algorithm is that there's no need to build a model or tune several parameters. Since it's a lazy learning algorithm and not an eager learner, there's no need to train the model; instead, all data points are used at the time of prediction.

Of course, that's computationally expensive and time-consuming. But if you've got the needed computational resources, you can use KNN for solving regression and classification problems. Albeit, there are several faster algorithms out there that can produce accurate predictions.

Here are some of the **advantages** of using the k-nearest neighbors algorithm:

- It's easy to understand and simple to implement
- It can be used for both classification and regression problems
- It's ideal for non-linear data since there's no assumption about underlying data
- It can naturally handle multi-class cases
- It can perform well with enough representative data

Of course, KNN isn't a perfect machine learning algorithm. Since the KNN predictor calculates everything from the ground up, it might not be ideal for large data sets.

Here are some of the **disadvantages** of using the k-nearest neighbors algorithm:

- Associated computation cost is high as it stores all the training data
- Requires high memory storage
- Need to determine the value of K
- Prediction is slow if the value of N is high
- Sensitive to irrelevant features

KNN and the curse of dimensionality

When you have massive amounts of data at hand, it can be quite challenging to extract quick and straightforward information from it. For that, we can use dimensionality reduction algorithms that, in essence, make the data "get directly to the point".

The term "curse of dimensionality" might give off the impression that it's straight out from a sci-fi movie. But what it means is that the data has too many features.

If data has too many features, then there's a high risk of overfitting the model, leading to inaccurate models. Too many dimensions also make it harder to group data as every data sample in the dataset will appear equidistant from each other.

The k-nearest neighbors algorithm is highly susceptible to overfitting due to the curse of dimensionality. However, this problem can be resolved with the **brute force implementation** of the KNN algorithm. But it isn't practical for large datasets.

KNN doesn't work well if there are too many features. Hence, dimensionality reduction techniques like **principal component analysis (PCA)** and **feature selection** must be performed during the data preparation phase.

KNN: the lazy algorithm that won hearts

Despite being the laziest among algorithms, KNN has built an impressive reputation and is a go-to algorithm for several classification and regression problems. Of course, due to its laziness, it might not be the best choice for cases involving large data sets. But it's one of the oldest, simplest, and accurate algorithms out there.

Training and validating an algorithm with a limited amount of data can be a Herculean task. But there's a way to do it efficiently. It's called cross-validation and involves reserving a part of the training data as the test data set.

Here, 'K' is the hyperparameter for KNN. For proper classification/prediction, the value of K must be fine-tuned.

But, How do we select the right value of K?

We don't have a particular method for determining the correct value of K. Here, we'll try to test the model's accuracy for different K values. The value of K that delivers the best accuracy for both training and testing data is selected.

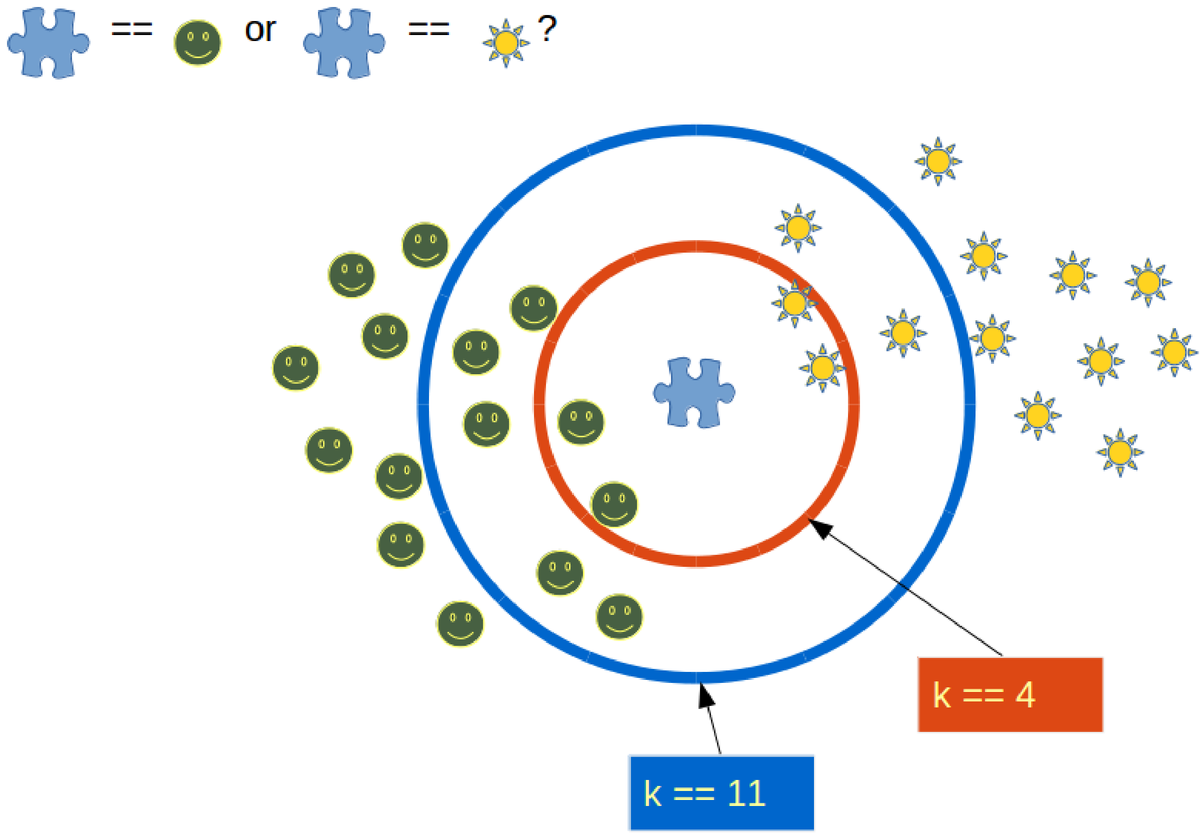
Note!!

It is recommended to always select an odd value of K ~

When the value of K is set to even, a situation may arise in which the elements from both groups are equal. In the diagram below, elements from both groups are equal in the internal “Red” circle (k == 4).

In this condition, the model would be unable to do the correct classification for you. Here the model will randomly assign any of the two classes to this new unknown data.

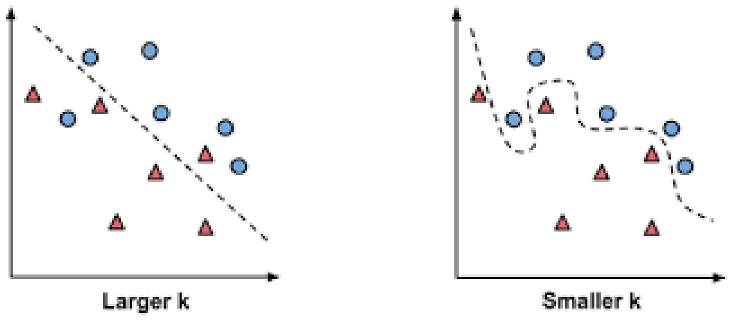
Choosing an odd value for K is preferred because such a state of equality between the two classes would never occur here. Due to the fact that one of the two groups would still be in the majority, the value of K is selected as odd.



Src: https://images.app.goo.gl/Q8ZKxQ8mhP68yxqn7

The impact of selecting a smaller or larger K value on the model

- **Larger K value:** The case of underfitting occurs when the value of k is increased. In this case, the model would be unable to correctly learn on the training data.
- **Smaller k value:** The condition of overfitting occurs when the value of k is smaller. The model will capture all of the training data, including noise. The model will perform poorly for the test data in this scenario.



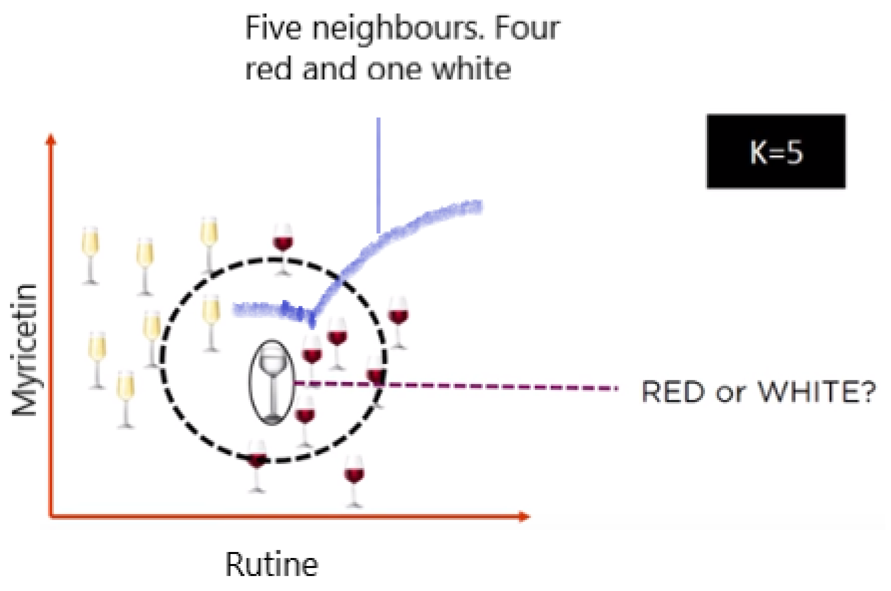
Src: https://images.app.goo.gl/vXStNS4NeEqUCDXn8

How does KNN work for ‘Classification’ and ‘Regression’ problem statements?

- **Classification~**

When the problem statement is of ‘classification’ type, KNN tends to use the concept of “Majority Voting”. Within the given range of K values, the class with the most votes is chosen.

Consider the following diagram, in which a circle is drawn within the radius of the five closest neighbours. Four of the five neighbours in this neighbourhood voted for ‘RED,’ while one voted for ‘WHITE.’ It will be classified as a ‘RED’ wine based on the majority votes.



Src: https://images.app.goo.gl/Ud42nZn8Q8FpDVcs5

Real-world example:

Several parties compete in an election in a democratic country like India. Parties compete for voter support during election campaigns. The public votes for the candidate with whom they feel more connected.

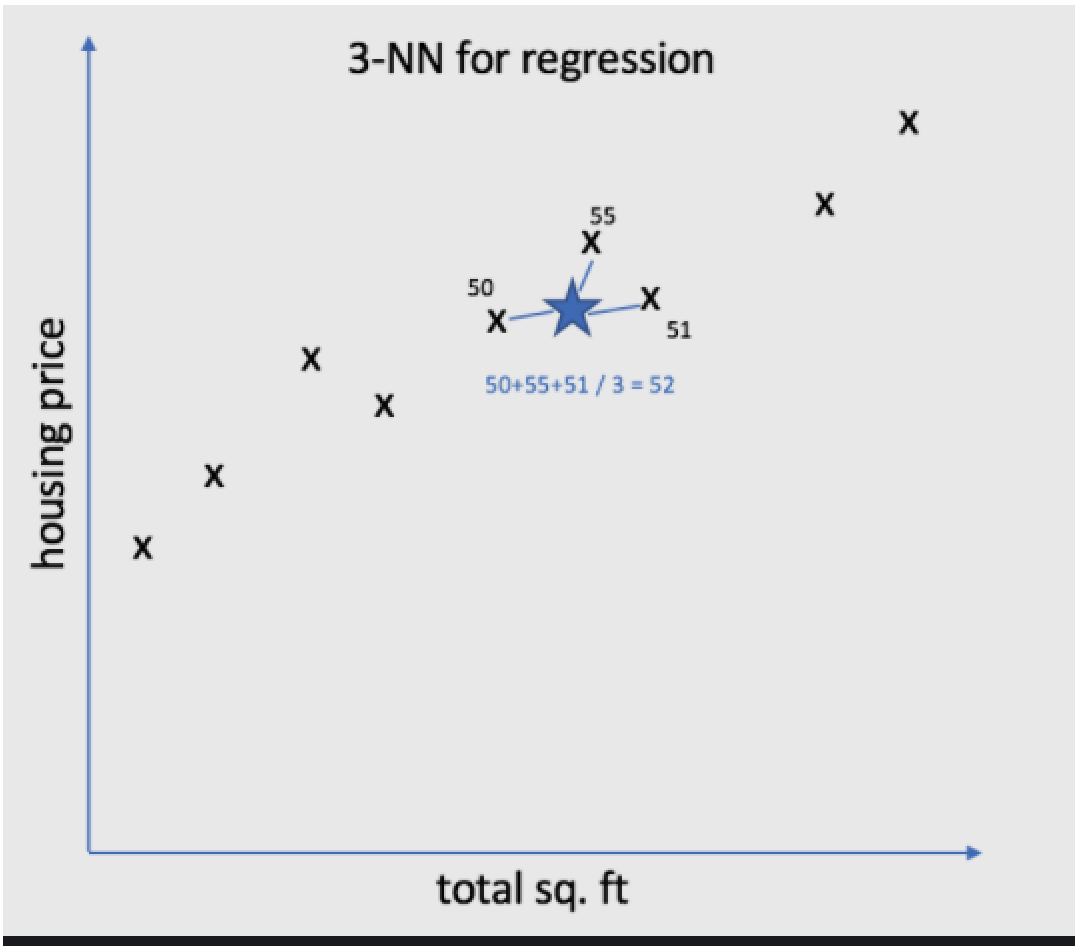
When the votes for all of the candidates have been recorded, the candidate with the most votes is declared as the election's winner.

- **Regression~**

KNN employs a mean/average method for predicting the value of new data. Based on the value of K, it would consider all of the nearest neighbours.

The algorithm attempts to calculate the mean for all the nearest neighbours' values until it has identified all the nearest neighbours within a certain range of the K value.

Consider the diagram below, where the value of k is set to 3. It will now calculate the mean (52) based on the values of these neighbours (50, 55, and 51) and allocate this value to the unknown data.



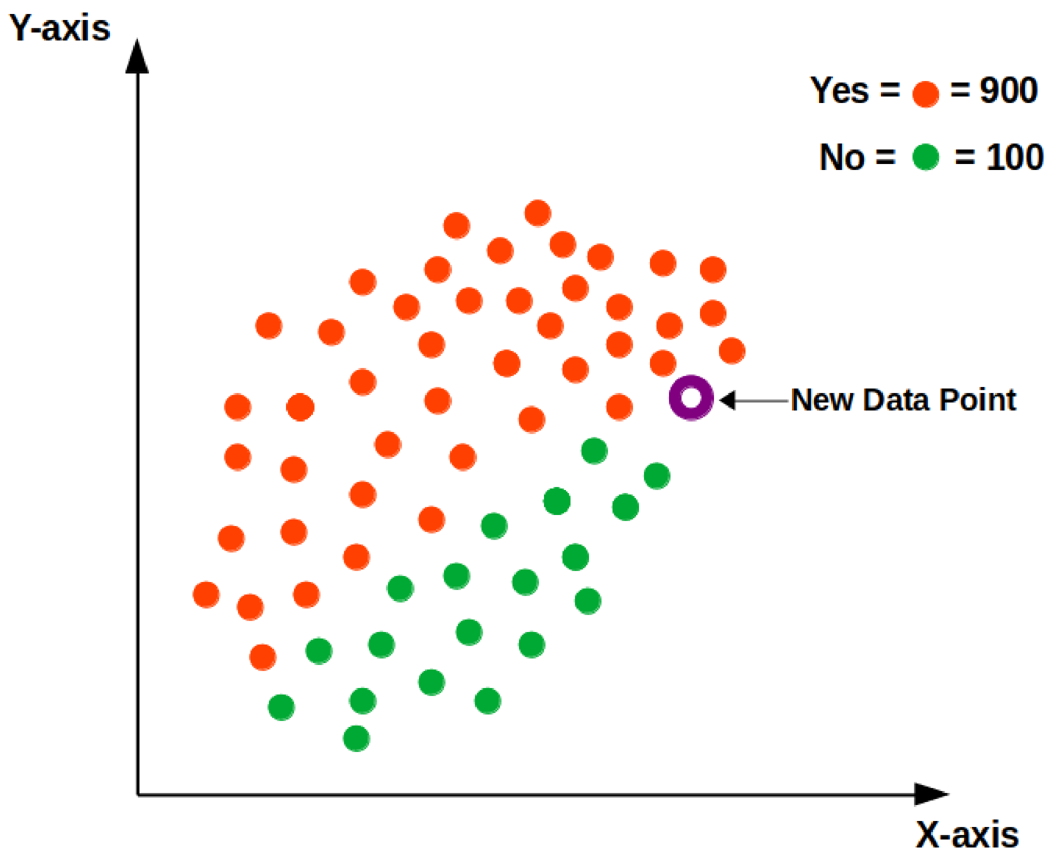
Src: https://images.app.goo.gl/pzW97weL6vHJByni8

Impact of Imbalanced dataset and Outliers on KNN

- **Imbalanced dataset~**

When dealing with an imbalanced data set, the model will become biased. Consider the example shown in the diagram below, where the “Yes” class is more prominent.

As a consequence, the bulk of the closest neighbours to this new point will be from the dominant class. Because of this, we must balance our data set using either an “Upscaling” or “Downscaling” strategy.



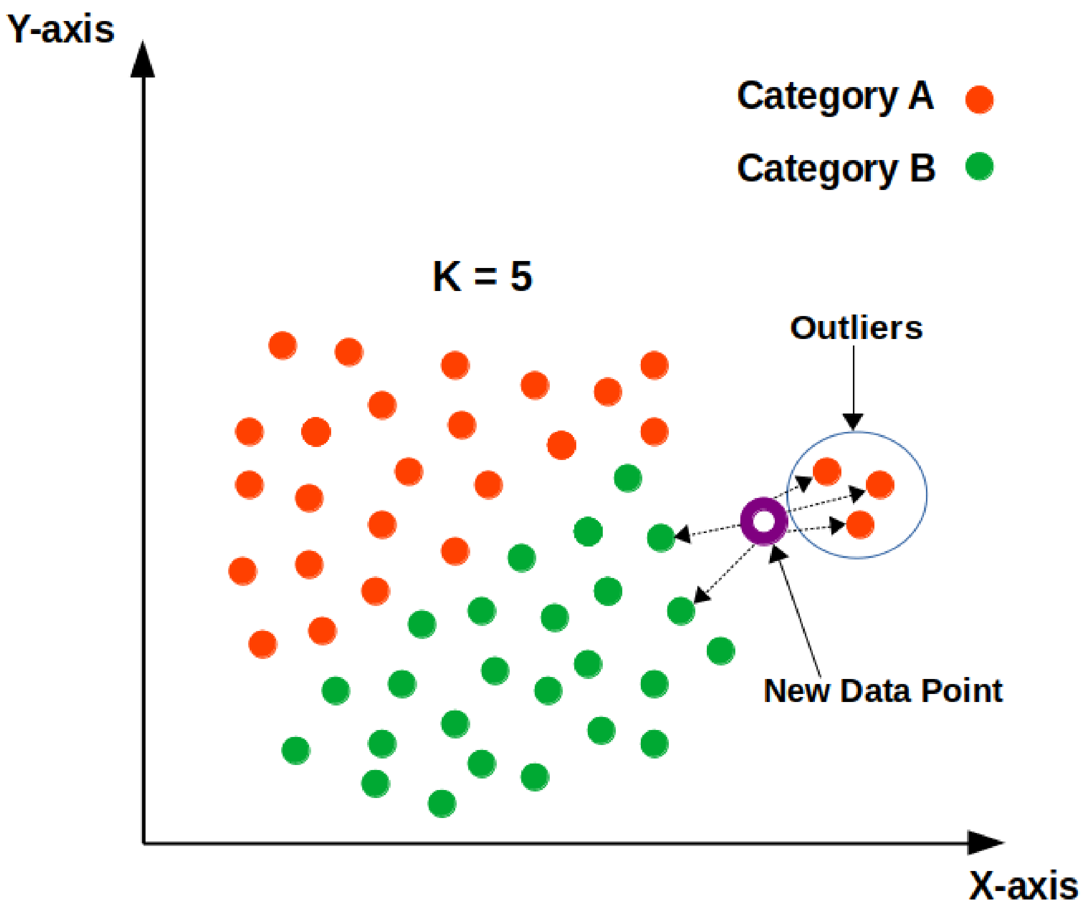
Src: https://images.app.goo.gl/1XkGHtn16nXDkrTL7

- **Outliers~**

Outliers are the points that differ significantly from the rest of the data points.

The outliers will impact the classification/prediction of the model. The appropriate class for the new data point, according to the following diagram, should be "Category B" in green.

The model, however, would be unable to have the appropriate classification due to the existence of outliers. As a result, removing outliers before using KNN is recommended.



Src: https://images.app.goo.gl/K35WtKYCTnGBDLW36