k-Nearest Neighbors method and Random Forests

Marcin Dudar

June 2024

1 Introduction to Nearest Neighbour algorithm

1: Good afternoon everyone! I hope your day is great Thank you all for being here today. I'm excited to dive into two fascinating machine learning algorithms which naming is on the screen: K-Nearest Neighbors, commonly known as KNN, and Random Forests. Whether you're new to these concepts or looking to deepen your understanding, I hope this workshop provides valuable insights and practical knowledge that you can apply somewhere in your projects.

2: Before we dive into the content, let's quickly go over today's agenda. Firsly we will look for our objectives for today workshop and then we will jump into basics of KNN, later I will show you my quick jupyter notebook file of my very simple implementation of KNN and we will do the same for Random forests. I understand your time is valuable, so I want to assure you that while this workshop is scheduled for two hours, I've structured it to cover all essential topics efficiently. My aim is to deliver the key concepts and practical examples much quicker, so we might finish earlier and you will be able to enjoy our friday.

3: Our main goal today is to provide you with a understandin those two algorithms and how to implement them in practical scenarios.

4: K-Nearest Neighbors (kNN) is a simple, but powerful machine learning algorithm that is widely used for **classification** and **regression** tasks. It is typically used as a classification algorithm, working off the assumption that similar points can be found near one another. Unlike many other algorithms, kNN is easy to understand and implement, making it a great starting point for those new to machine learning and statistics.

KNN is an instance-based (or memory-based) learning algorithm, which means it makes predictions based on the specific examples in the training data rather than creating a generalized model - in other words kNN is an algorithm for supervised learning that simply stores the labeled training examples during the training phase and when a new data point needs to be classified or predicted kNN looks at the most similar instances in the training data.
KNN is one of the lazy learning algorithms - which means that algorithm store the data while training and preprocessing it during the testing phase. Then, to make a prediction (class label or continuous target) the kNN algorithm find the k nearest neighbors of a query point and compute these class label (classification) or continuouns target (regression) based on the k nearest (most "similar") points.

5: On the presentation you can see the mechanic how does it work.

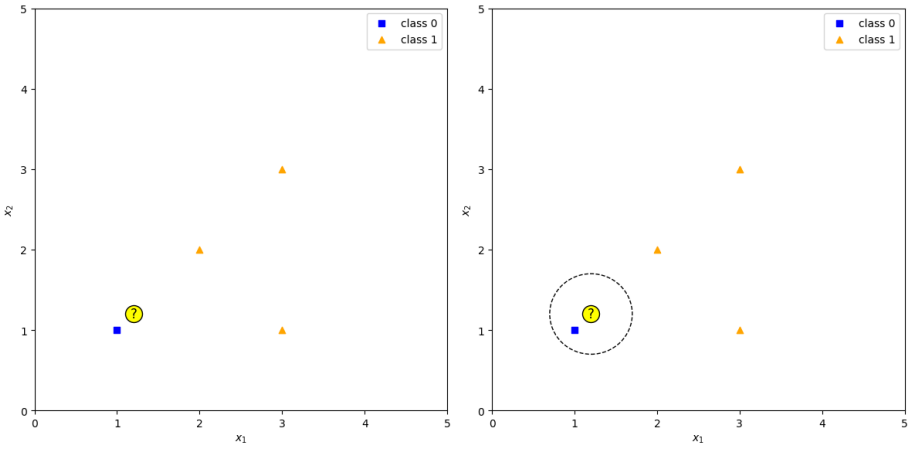

Figure 1: Illustrating a different aspect of the K-Nearest Neighbors (KNN) algorithm.

on the Left plot: We can see 3 types of data points, class 0, class 1 and point with a question mark, called query point "?". This is the new data point we want to classify or predict.

on the other hand on the Right plot: I added dashed circle around our new point which indicates the neighborhood around the query point. The radius of the circle represents the distance which the nearest neighbors are considered. KNN algorithm classifies a new data point by looking at the nearest neighbors. The classification of the query point depends on the class of the majority of the points within the circle.

So, simple interpretation would look like this, inside the Circle there is one point, which belongs to (class 0). so the classification would look on the majority of the nearest neighbors and see that it belongs to class 0, so the KNN algorithm would classify the query point as class 0.

7: but what is that dashed circle around the point? The dashed circle that you saw is **distance metric**, it determines which data points are closest to a given query point point. The distance between our query point and other data points will need to be calculated. So basically These distance metrics help us to form decision boundaries. There are many distance metrics or measures we can use to select k nearest neighbors. There is no "best" distance measure, and the choice is highly context or problem-dependent. Most commonly used are **Euclidean distance** and Manhatan distance. Lets look on the example on the right: Imagine you have two points on a piece of paper, each with an A and a B coordinate. The Euclidean distance is the length of the straight line that connects these two points. It's the shortest path between them.
On the other hand Manhattan Distance - is the distance between two points

measured along axes at right angles. Manhattan Distance metric is preferred over Euclidean Distance when there is a high dimensionality in the data.

8: And finally, we will answer question what is this "k" it our algorithm. The "K" in K-Nearest Neighbors (KNN) is a parameter that significantly impacts the performance and behavior of the algorithm. It represents the number of nearest neighbors considered when making a prediction about the class or value of a new data point.
So basically we are choosing the Value of K, later in code I will show you one method called "Elbow method" that will help us to choose proper K.

On the right we can see how the complexity of the model changes depending on how big the k is

- Small "k":
  - High variance: the model might overfit the training data, capturing the noise as if it were a part of the actual pattern. That means we will misallocate the query points.
  - If K is too small, the model becomes very sensitive to noise in the data. For example, with K equal to 1, a single noisy data point can drastically change the prediction.
- Large "k":
  - Low variance: If K is too large, the model can become too simplistic, missing important patterns and nuances in the data
  - Smoothing Effect: A larger K provides a smoother decision boundary and reduces the impact of noise by averaging over more points.

In practice, we try to find a good trade-off between high bias (the model is not complex enough to fit the data well) and high variance (the model fits the training data too closely), Some literature says that k=5 is optimal in most cases but we have a way to choose proper K.

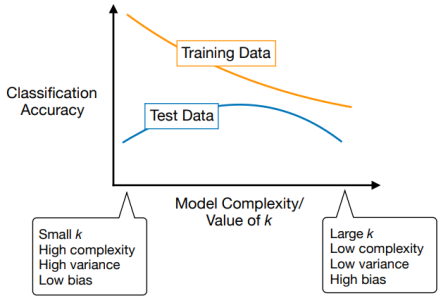

Figure 2: Model complexity.

I was thinking about examples connected to our company where we can use it. As we understood from previous slides KNN has it best use for classification.

Customer Segmentation and Marketing: Example: Segmenting customers for targeted marketing campaigns. Usage: KNN can help segment customers based on purchasing behavior, preferences, and demographics. By analyzing customer data, the KNN model can group customers with similar characteristics. This information can be used to tailor marketing campaigns and product recommendations, ensuring that promotions are relevant and more likely to convert into sales.

Demand Forecasting for New Products:

Example: Estimating the sales volume of a new tobacco product. Usage: When launching a new product, KNN can help predict its sales by finding similar existing products in the market. By analyzing features such as product type, target demographic, pricing, and launch marketing strategies, KNN can identify the nearest neighbors (similar products) and use their sales data to estimate the potential sales of the new product. This approach helps in setting realistic sales targets and planning production accordingly.

Step-by-Step explanation of how KNN works after preparation of the data:

1. Selecting the optimal value of K
2. Calculating distance (algorithm in Python will do it for us)
3. Take the K nearest neighbors as per the calculated distance.
4. Among these k neighbors, count the number of the data points in each category.
5. Assign the new data points to that category for which the number of the neighbor is maximum.
6. Our model is ready

For the practical part, I'll be demonstrating the examples using a Jupyter notebook. Instead of asking you to code along, I'll explain each line of code in detail so you can focus on understanding how everything works. This way, you can follow along and ask questions without the pressure of coding in real-time. Let's get started!"

The data we'll be using today comes from some of my old files that I used while learning. It includes four columns: 'Gender', 'Age', 'Salary', and 'Purchase'. The 'Gender', 'Age', and 'Salary' columns are our independent variables, meaning they are the input features we'll use to make predictions. The 'Purchase' column is our dependent variable, indicating whether a purchase was made or not, with 0 representing no purchase and 1 representing a purchase. This setup will allow us to explore how these features influence the purchasing decision.

Advantages:

- The effectiveness of KNN increase with large training data and so it supports sufficient data representation
- KNN can be used for both classification and regression
- Implementation of KNN is simple

Disadvantages:

- Measuring the distance between all data points in training samples is a computationally expensive task
- If the training data is excessively randomized, then assigning a class label based on distance measure is messy
- KNN is not a suitable option for query points at farther distance from the training data

1.1 Interview question for KNN that I met

1. What is the intuition behind the KNN algorithm?
2. How do you choose the value of K in the KNN algorithm?
3. What are the different distance metrics used in KNN?
4. What are the advantages and disadvantages of the KNN algorithm?

2 Random Forests

2.1 Introduction to Random Forests

12: Random forest is a supervised learning method that encompasses two forms (similarly to KNN): one for classification tasks and the other for regression tasks. Known for its flexibility and simplicity, it constructs decision trees based on provided data samples. Each tree generates predictions, and the final outcome is determined through a voting mechanism. Random forest algorithm combines multiple decision-trees, resulting in a forest of trees, hence the name Random Forest. Usually In the random forest classifier, the higher the number of trees in the forest results in higher accuracy. 13: Now you can see how does random forest work. It contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

14:

2.2 Practical implementation

Step-by-Step explanation of how Random Forest works (divided into two stages):

1. Select random K data points from the training set.
2. Build the decision trees associated with the selected data points (Subsets).
3. Choose the number N for decision trees that you want to build.
4. Repeat Step 1 and 2.
5. For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Of course Python code will do it for us, i know it takes more time to understand but dont worry, its not that hard.

15: Ok - But how does it work? Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision.

16: The advantages of Random forest algorithm are as follows:- Advantages and disadvantes: So lets start with advantages: It is considered as very accurate and robust model because it uses large number of decision-trees to make predictions. Random forests takes the average of all the predictions made by the decision-trees, which cancels out the biases. So, it does not suffer from the overfitting problem. Random forest classifier can handle the missing values. There

are two ways to handle the missing values. First is to use median values to replace continuous variables and second is to compute the proximity-weighted average of missing values.

Disadvantages: The biggest disadvantage of random forests is its computational complexity. Random forests is very slow in making predictions because large number of decision-trees are used to make predictions. All the trees in the forest have to make a prediction for the same input and then perform voting on it. So, it is a time-consuming process. The model is difficult to interpret as compared to a decision-tree, where we can easily make a prediction as compared to a decision-tree.

17: Now lets jump to jupyter notebook For the practical example we will use the same data input

KONIEC

Thank you all for attending today's workshop. I wanted to demonstrate how to integrate KNN and Random Forests into Power BI, but unfortunately, due to restrictions on the company laptop, I couldn't perform the live demonstration. However, I've successfully set it up on my personal computer and documented everything extensively. I will share all the materials, including my notes and sources, with you shortly after the workshop concludes. Please feel free to reach out if you have any questions or need further clarification. "