Data Mining

J A K U B
M A R C I N E K

A N D R E J
Z I D E K

03/10/2023

PROJECT ASSIGNEMNT

Table of Contents

1. Introduction

In this chapter a small overview of the applied models is going to be presented along with their description and additional information such as their real-life applications.

1.1. K-Nearest Neighbors Algorithm (kNN)

kNN is a simple and intuitive supervised machine learning algorithm which can be used for both classification and regression problems. The main principle is instance learning – where the algorithm memorizes the provided training dataset and makes predictions based on the affinities between the new data points and the existing data points (Harrison, 2018).

How KNN works:

1. **Choosing the Value of K:** The first step in KNN is to choose the value of K, which represents the number of nearest neighbors to consider when making a prediction. It is up to the researcher to set this value.
2. **Calculating Distances:** For a given data point that is to be classified or predicted, KNN calculates the distance (usually Euclidean distance) between that point and all other points in the training dataset.
3. **Selecting Nearest Neighbors**: KNN then selects the K data points from the training dataset that are closest to the new data point based on the calculated distances. These are the so called "nearest neighbors".
4. **Majority Voting (Classification) or Averaging (Regression):** For classification tasks, kNN takes a majority vote among the K nearest neighbors to determine the class of the new data point. For regression tasks, it calculates the average of the K nearest neighbors' target values to make a prediction (IBM, 2022).

Some real-life uses of kNN include image recognition (by comparing the pixel values), text classification (finds similar text samples to classify new ones), medical diagnosis (categorizing patient conditions based on similarities to historical patient data), recommendation systems (finds users with similar preferences and recommends items they have liked) and anomaly detection (detect anomalies or outliers in datasets) (Harrison, 2018).

1.2. Naïve Bayes Algorithm

Naïve Bayes is a probabilistic supervised machine learning algorithm based on the Bayes' theorem – where the "naïve" assumption is being made that the features/variables being used for classification are independent of each other (IBM, 2022).

How Naïve Bayes works:

1. **Training:** The Naïve Bayes algorithm starts with a training dataset containing labeled examples. It calculates the probabilities of each feature occurring with each class based on the training data.
2. **Conditional Probability:** For a new data point to be classified, the algorithm calculates the conditional probability of it belonging to each class given its features. It uses Bayes' theorem to calculate these probabilities.
3. **Class Selection:** The algorithm selects the class with the highest conditional probability as the predicted class for the new data point (GeeksforGeeks, 2023).

Some real-life uses of Naïve Bayes include email spam detection, medical diagnosis, document classification, sentiment analysis and recommendation systems (IBM, 2022).

2. Data Understanding and Preparation

2.1. Iris Dataset

The dataset was chosen as it has been widely used and applied to machine learning problems. It is a small dataset that contains measurements of four features (sepal length, sepal width, petal length, and petal width) for three different species of iris flowers (setosa, versicolor, and virginica), with 50 samples per species.

It offers a manageable size, clear class separation, and is a perfect benchmark for testing the performance of various classification techniques. Additionally, the Iris dataset's simplicity allows focusing on understanding the algorithms' mechanics and their application in real-world scenarios.

As mentioned above, due to the manageable size and clear class separation there was no need for extensive data preparation such as normalizing the data or dropping certain variables. The only preparation done was the separation of seven different rows (found in the file "test"), to be used for later testing and validation of the model. Hence, it is possible to move into data understanding straight away.

2.2. Data Understanding

The basis of the data set is that the different species are characterized by the dimensions of the sepal and the petal of the flower. Hence, firstly to get a general overview of how the different species are grouped a pair plot was created as seen in Figure 1 below (for the script please refer to file "DataUnderstanding.py").

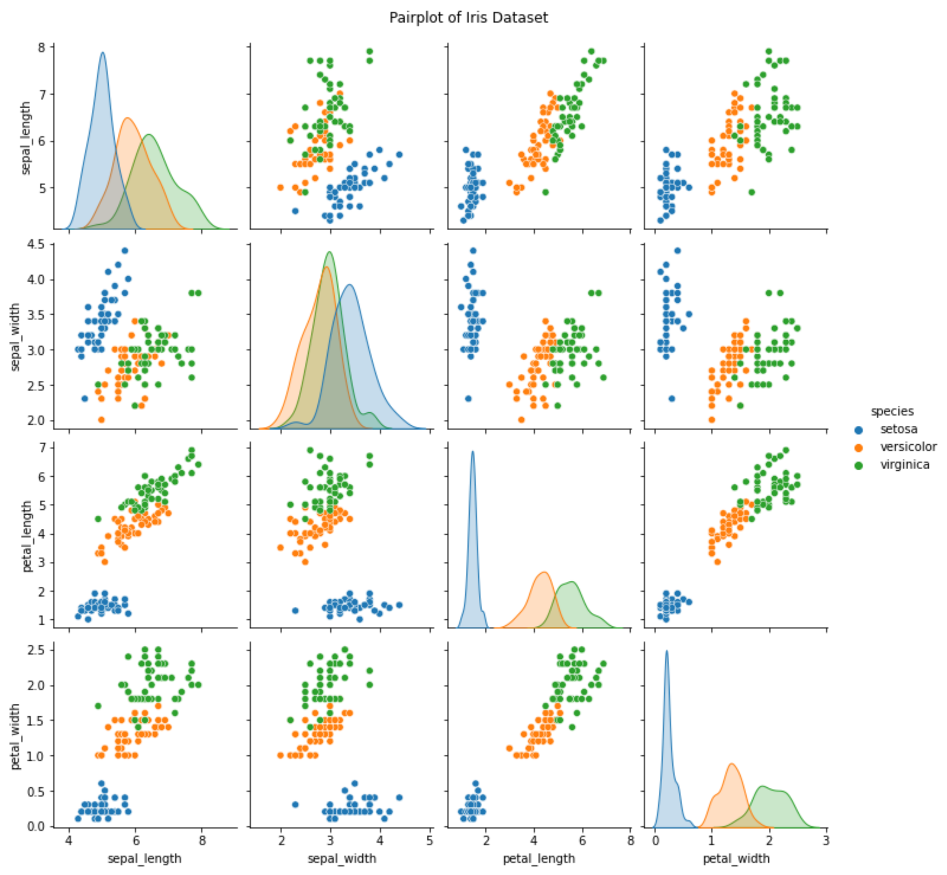


Figure 1 Pair plot of Iris Dataset

The pair plot clearly indicates that setosa is the species with the most distinct characteristics as it does not overlap with the other two species. Whereas when it comes to versicolor and virginica the distinction is still present, however not as clean-cut. The category where all three species overlap the most is the sepal-width. The pair plot illustrates that the dataset is well suited for both the algorithms due to the well-defined nature of the different species.

Furthermore, a second visual was created, where the “width” and “length dimensions were clustered together – both the width/length of the sepal and petal were plotted on the same axis. Ergo, producing a more concise version of the visual above – where the same findings were reinforced including the species and data separation. Overall, it is possible to conclude that length of the setosa species tends to be between 3 to 4.5 cm while its length only around 6. The width for versicolor is quite similar, however its lengths vary between 8 to 12 cm. While virginica displays both the largest width and length.

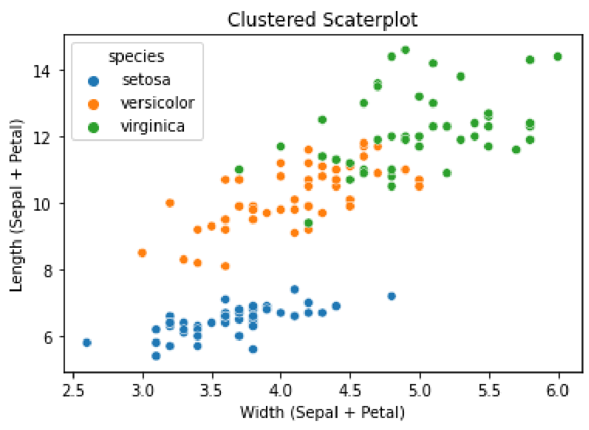


Figure 2 Clustered Scatter plot

Lastly a third 3D visual was created, where the overlaps between the different categories can be seen in more detail. PCA creates three new features, providing a visualization of how the Iris dataset can be reduced to three principal components while still preserving the most important information about the differences between the species. Each point in the plot represents an individual iris flower, and its position along these three axes reflects how it differs from other flowers based on the most significant variations in the original feature space. This visualization further helps in understanding the separability and clustering of the different species as now the overlap between versicolor and virginica is minimal.

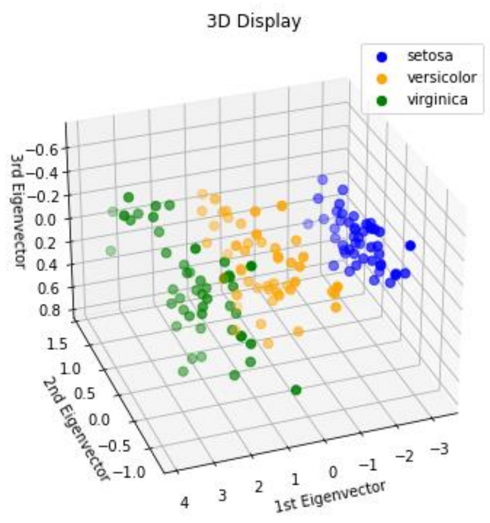


Figure 3D PCA Display

3. Testing

3.1 Implementation of code and the use of AI within testing

During the testing phase, the models were trained on a portion of the dataset and then evaluated on the remaining data to assess their accuracy. This process was repeated multiple times with different portions of the data used for training and testing to ensure that the models were robust and not overfitting to the training data.

In terms of AI implementation, various AI powered chat models were used to perfect and understand the code in more depth. Primarily, AI model from OpenAI, ChatGPT, was used to improve the code.

Overall, testing and implementation of AI were critical components in the development of kNN and Naive Bayes models for the Iris dataset, helping to ensure that the models were accurate, robust, and optimized for classification as well as further testing.

3.2 Implementation of k Nearest Neighbors model

Application of the kNN algorithm to the data set is done in multiple steps. First, the dataset must be loaded into memory and inspected to ensure that it is in a suitable format for analysis. Secondly, research of suitable and available libraries and packages was performed. As a result of this research, a selection of libraries and packages was chosen which contained functions related to the assignment and specifically related to the kNN algorithm. These libraries and packages were used to further clean the data and prepare the dataset for following operations. Here is the list of the chosen libraries and packages, and their functions within the script (note that not all libraries/packages used within the assignment are listed here):

- Pandas (basic functions)
- Sklearn (advanced functions)
  - sklearn.neighbors (KNeighborsClassifier)
    - used as a primary function to run the model
  - sklearn.model_selection (train_test_split)
    - used as a function to split the dataset into two; training & testing
  - sklearn.metrics (accuracy_score)
    - used to calculate, measure, and compare the accuracy scores when using different number of neighbors

Once the dataset has been cleaned, it can be used to train and evaluate the kNN algorithm. This involves tasks such as splitting the dataset into training and testing sets and selecting appropriate features for use in the model to evaluate the performance and results. This involves selecting an appropriate value for the number of neighbors to use in the algorithm, as well as a distance metric to measure the similarity between data points. The algorithm then searches through the training data to find the k nearest neighbors to a new data point, and uses these neighbors to classify the new data point.

Finally, the performance of the kNN algorithm is evaluated using a separate test dataset. The separate dataset originates from the original dataset. Evaluation of the model is based on the accuracy of the results. More about accuracy will be introduced in the following chapters.

3.3 Implementation of Naïve Bayes

When implementing Naive Bayes on the Iris dataset, the first step is to load the dataset into memory and inspect it to ensure that it is in a suitable format for analysis. This step is similar to the first step of implementation of the kNN model. Furthermore, suitable packages and libraries were selected. These packages and libraries will be used to further clean and preprocess the data. Among all packages and libraries, these were chosen for their functionality and relatedness to the assignment:

- Pandas (basic functions)
- sklearn.model_selection (train_test_split)
    - used as a function to split the dataset into two; training & testing
- sklearn.naive_bayes (GaussianNB)
    - used as a primary function to run the model
- sklearn.metrics (accuracy_score)
    - used to calculate, measure, and compare the accuracy scores when using different number of neighbors

After the dataset has been cleaned and preprocessed, it can be used to train and evaluate the Naive Bayes model. Using the “train_test_split” function the dataset is split into training and testing sets. The Naive Bayes algorithm assumes that features are independent of one another, so it is important to select features that are not highly correlated. Once the model has been trained, it can be used to classify new data points based on their feature values.

Lastly, the performance of the Naive Bayes model is evaluated using a separate test dataset. The test dataset is typically a subset of the original dataset that was not used during training. The accuracy of the model is calculated using variety of functions that will be introduced in later chapters. Overall, the implementation of Naive Bayes involves several key steps, including data preprocessing, feature selection, model training, and performance evaluation.

4. Evaluation

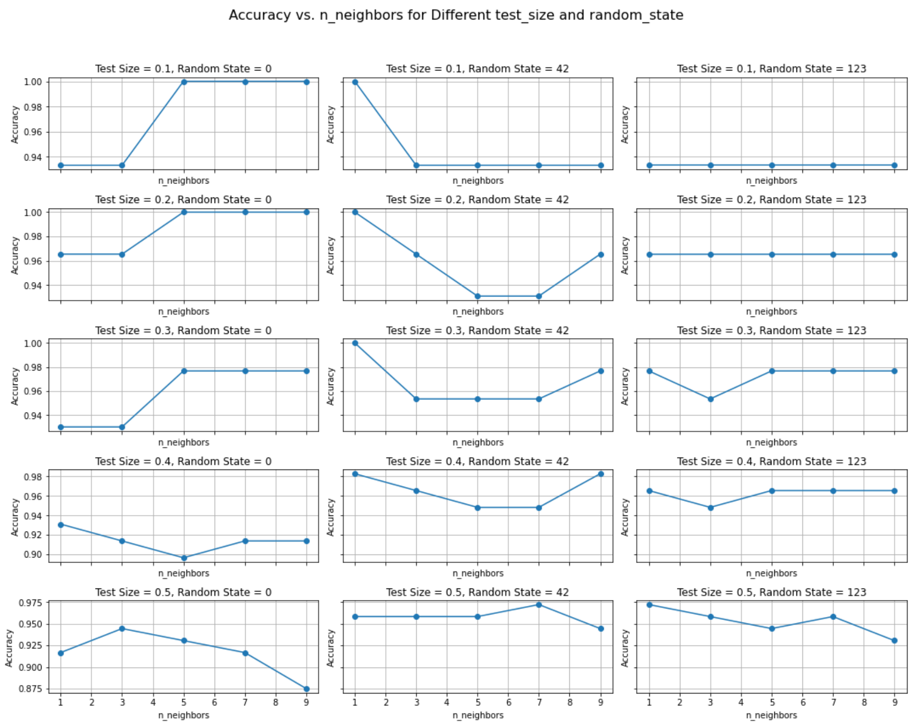
In this chapter the final step – evaluation of the models is to be discussed. Evaluation is a crucial step in the machine learning pipeline, as it helps to assess the model's performance and make informed decisions regarding its suitability for the task at hand. The following is going to be discussed: how testing was conducted, what aspects were tested, the purpose of the generated graphs, and the insights into their significance.

The script for kNN and Naïve Bayes differs as to showcase the different possibilities of finding the optimal parameters. In the case of kNN the parameters to be tested were specifically defined (more interesting for cases when specific circumstances are to be tested) and kept in a newly created data frame for clarity purposes. Whereas for Naïve Bayes a range was used and new variables initialized with “None” in order to cycle through the options and keep track of the best option.

The primary goal was to determine the optimal combination of the hyperparameters that maximize the model’s predictive power and accuracy. The following aspects were tested: test size, random state and number of neighbors (for Naïve Bayes only the first two were considered). In order to maximize the clarity of outputs both of the evaluations were also graphically plotted. Naturally, the procedure for utilizing the models to get the different accuracies is the same as described in the previous chapter.

3.1. kNN

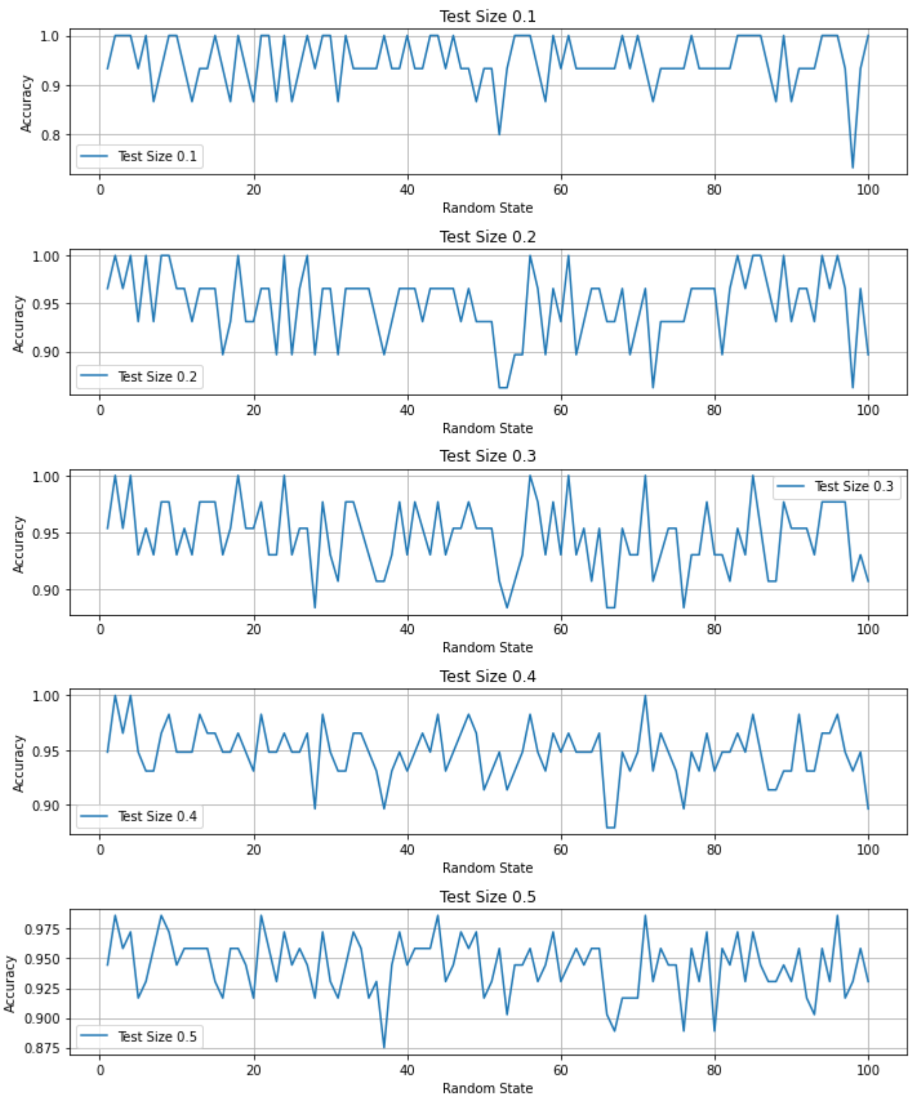
In the case of kNN the optimal accuracy was evaluated against the test sizes, random state and the number of neighbors, for the entire script please refer to the file “KnnEvaluation.py”. The Figure below displays an overview of the different tests conducted. Where the highest accuracy (100%) was achieved in seven different cases out of which one can be disregarded – the one being, where only one neighbor was being utilized. All the other instances have five plus neighbors, ergo suggesting that a higher value of k (neighbors) is necessary to achieve higher accuracy most likely due to the slight overlap between the versicolor and virginica species as discussed in the second chapter.



Overall, kNN is a suitable algorithm for classifying the Iris dataset due to its simplicity and effectiveness. Furthermore, lending itself well to kNN intuitive approach. Some pros of utilizing kNN with the Iris dataset include its ability to handle multi-class classification tasks effectively and its ease of implementation. Additionally, kNN also works well when the decision boundaries between classes are relatively simple, as is the case with Iris. However, there are some cons to consider. kNN can be sensitive to the choice of hyperparameters, such as the number of neighbors (k) (as it can be clearly seen in the visual above), and it may struggle with high-dimensional datasets where the curse of dimensionality can affect performance. Furthermore, it assumes that data points close in feature space have similar class labels, which may not always hold true in more complex datasets. The algorithm serves as a good example for the Iris dataset, but its suitability should be carefully assessed based on the specific characteristics and requirements of the data at hand.

3.2. Naïve Bayes

For Naïve Bayes the optimal accuracy was evaluated against the test sizes, random state and, for the entire script please refer to the file “BaysEvaluation.py”. The Figure below displays an overview of the different tests conducted. Where the highest accuracy (100%) was achieved several times in all of the test sizes, except 0.5 – most likely due to the large chunk of the data set being used for training and thus not enough being left over for the actual testing, resulting in lower overall accuracy.



The Naive Bayes algorithm is also a choice for classifying the Iris dataset, albeit with its own set of advantages and limitations. The algorithm is particularly well-suited for text classification tasks, but it can perform in other domains such as the Iris dataset. Pros of utilizing Naive Bayes with the Iris dataset are simplicity and speed (easy to implement and computationally efficient, a great choice when a quick and reliable classification is needed), efficiency with small data sets (as it does not require a lot of data to be trained), multiclass classification (Naive Bayes handles multiclass classification effortlessly, thus suitable to the well-defined classes of the Iris data set).

Some cons include the assumption of independence (meaning that the features are to be conditionally independent, which is usually an oversimplification. Hence, some interdependencies might have been overlooked in the Iris data set.), lack of probabilistic interpretation (while Naive Bayes provides class predictions, it does not offer a probabilistic interpretation of these predictions. In some applications, a probabilistic output might be desired). In summary, Naive Bayes is a reasonable choice for classifying the Iris dataset, especially when simplicity and efficiency are priorities. Its ability to handle multiclass classification and small datasets makes it a practical option. However, one should be aware of its assumptions and limitations, particularly when dealing with more complex datasets or when a probabilistic interpretation of results is necessary. The suitability of Naive Bayes for the Iris dataset should be evaluated in the context of the specific objectives and characteristics of the data.

4. Reflection

4.1. Jakub Marcinek

Firstly, regarding the learning goals: which included exploration and preparation, training and evaluation. All of these have been acquired in my opinion – for the exploration and preparation, not much actual data cleaning was needed. However, due to the choice of our data set it made me realize and think about how the preparation is a crucial step as it directly impacts the performance of the models.

The core included implementing the two algorithms. Initially, it was hard to grasp the innerworkings of the models as the combination of both coding and machine learning seemed daunting. However, with the help of available resources, including Chat GPT I can now confidently say that I do understand how these models work as well as their application in real life.

Lastly, there was the evaluation, a part which I probably enjoyed the most. As with the help of brainstorming we were able to come up with a lot of different little tweaks and functions that helped us measure the performance of the model as well as visualize and translate what that meant in reality (such as the importance of leaving enough of the data set available for training purposes).

Overall, I would say this project has helped in understanding the general basis of machine learning models and applying that to a dataset. However, the biggest take-away for me personally is my progress in coding. Thanks to this project I am now able to search for solutions better as well as not be as intimidated by a coding task, more so interested in how to tackle it.

4.2. Andrej Zidek

Throughout my learning journey, I had set various goals for myself. With this project, I wanted to understand how programming world functions and what are the capabilities of python. While I understood the importance of data cleaning already prior to the assignment, the importance of data preparation was unknown to me. During the exploration and preparation phases, I gained valuable experience and, hence, managed to participate in the code writing far further than I would have original thought I could. Lastly, I learned that preparation is crucial for the performance of machine learning models, especially when working with our chosen dataset.

When it came to implementing the two algorithms, I initially found it difficult to understand the coding and machine learning concepts. However, I found various resources that helped me understand how these models work and their real-life applications. To actually understand the code, I have taken advantage of chat assistant Monica, a plug-in for Chrome that combines Chat GPT (version 3.5), Chat GPT (version 4), Bard, and Claude-2.

The evaluation phase was probably the most complicated part of the project for me. In our group, we brainstormed the various functionalities and visualizations of the model, however, I struggled with internal visualization, and, therefore, relied on the explanation of my fellow group member. In joined forces, we managed to come up with great ideas and now I have gained further understanding of coding.