# Text classification algorithms*

Katarzyna Czernik[1,*,†], Karolina Kamela[1,†]

[1] *Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND*

**Abstract**

In the article, we will describe and compare the operation of two classifiers: K-Nearest Neighbors and Naive Bayes Classifier. We will focus primarily on the application of these algorithms in text analysis. The division of texts is made into three classes of abstraction: SPORTS, FOOD & DRINK, and HOME & LIVING, which correspond to the categories of the texts we selected. We evaluate the classifiers based on key metrics such as accuracy and execution time, providing a detailed analysis of their performance across different parameter settings and dataset sizes. The experimental setup involved multiple runs to ensure the robustness of the results, and the findings were averaged for consistency. Overall, this comparison provides valuable insights into the practical applications of KNN and Naive Bayes Classifiers in text classification tasks, guiding the choice of algorithm based on specific needs such as accuracy, speed, and computational resources. For our study we used programs written in Python, using libriares: pandas, numby, seaborn, matplotlib.pyplot and sklearn

Average results of accuracy is 99.0222% for KNN and 91.3333% for Naive Bayes classifier. The advantage in accuracy lies with KNN; however, the operational time required to achieve such a result amounts to as much as 173.8866 s, whereas the Bayesian classifier is capable of analyzing a dataset of the same size in an average of 0.2897 s.

**Keywords**

KNN, Naive Bayes Classfier, Text analisys, Comparision, Machine Learning

## 1. Introduction

Text classification is a crucial task in natural language processing (NLP), enabling the automated categorization of text into predefined categories. In this study, we explore and compare the effectiveness of two popular classifiers: Naïve Bayes Classifier and K-Nearest Neighbor(KNN), within the context of text analysis.

In text classification KNN uses all the training data sets which makes process of measuring and sorting beacome more complex and time-consuming. It often shows different results from different samples [3]: „KNN still suffers from inductive biases or model misfits that result from its assumptions, such as the presumption that training data are evenly distributed among all categories". K-Nearest Neighbor algorithm has been developed by adding and modyfying various improvement schemes [9].

The second method we will use is Naive Bayes Classifier which is most often used as a baseline intext classification because it is fast and easy to implement [4]. Some may say that the Naive Bayes classifier is currently experiencing a renaissance in machine learning[5]: in numerous head-to-head classification papers [6] [7][8] it has been earning nearly las tor even last places.



*IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania

[1,*] Corresponding author

[†] These author contributed equally.

kc307856@student.polsl.pl (K. Czernik); kk307874@student.polsl.pl (K. Kamela)

Existing solutions offer different trade-offs in terms of accuracy, computational complexity, and scalability. KNN is known for its simplicity and effectiveness in various domains, while Naive Bayes is appreciated for its strong probabilistic foundation and efficiency. By comparing these classifiers, we aim to provide insights into their relative strengths and weaknesses, particularly in handling text data. There were some attempts to, similarly to us, compare those two classifiers [2] [10]. Based on obtained data we can analyze both algorythms advantages and disadvantages, which could result with new and innovaitve ideas of potencial application of classifiers.

Authors of that article[1] had the idea to construct a new classifier that combines the distance-based algorithm K-Nearest Neighbor and statistical based Naïve Bayes Classifier in order to increase effectivnes and accuracy . As it is noticed in this article[1] both alghorithms have their weeknesses. In case of K Nearest Neighbor algorithm the issues is caused by problems regarding categorical attributes, whereas Naïve Bayes Classifier have issue handling numerical atributes.

## 2. Methodology of KNN

### Description of Operation

KNN (k-Nearest Neighbors) identifies k neighbors to which the examined element is closest to. We can use different metrics to measure the similarity between data points.
In the project we used the Euclidean metric. For two points $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and $\mathbf{q} = (q_1, q_2, \ldots, q_n)$, the Euclidean distance $d(\mathbf{p}, \mathbf{q})$ is given by the formula:

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

W wyniku analizy identycznego zbioru danych różne metryki mogą zwracać różne rezultaty[11].
Formulas for KNN's most popular metrics:
Minkowski Metric:

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

where:

- $d(\mathbf{p}, \mathbf{q})$ is the distance between points $\mathbf{p}$ and $\mathbf{q}$,
- $p_i$ and $q_i$ are the coordinates of points in the i-th dimension
- $r$ is the Minkowski metric parameter.

Manhattan Metric:

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

where:

- $d(\mathbf{p}, \mathbf{q})$ is the distance between points $\mathbf{p}$ and $\mathbf{q}$,
- $p_i$ and $q_i$ are the coordinates of points in the $i$-th dimention.

After measuring distances beetwen the element and all elements from training set, the points are sorted based on their distance from the examined element. The k nearest elements are selected. Then classifier assigns the class label to the element based on the majority class among its k nearest neighbors.

### Calculation Example:

Determine, for k=3, to which set the element * belongs among the sets of elements shown in the chart:

The example (Figure 1.) includes three classes of abstraction: 'green', 'yellow', and 'pink'. Looking at the chart or calculating the distances from point * to the other elements, we can determine that the k nearest elements to * are: 'pink', 'yellow', 'pink'.
Then, through voting, the classifier determines which class of abstraction is most common among the k nearest neighbors. In this case, it is 'pink'.

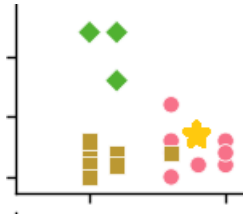

**Figure 1:** Ilustration of example

## Algorithm

Pseudocode:

**Algorithm 1:** K-Nearest Neighbors (KNN) Algorithm

**Data:** Training set $D$, test point $x$, number of neighbors $k$
**Result:** Predicted class for point $x$
**for** *each point* $p$ *in* $D$ **do**
    Calculate the distance $d(p, x)$;
Sort points $p$ by distance $d(p, x)$;
Select the $k$ nearest neighbors;
Obtain the class labels of the neighbors;
Choose the most frequent class label as the predicted class;
**return** *Predicted class*;

In the project we divided our database into training set and validation set. We used 70 : 30 proportion, so for the base consisting 1500 elements, validation set has 450 elements and training set has 1050 elements. Alghorithm predicts class of the element taken from validation set, when it checks accuracy of the guess. This action is reapeated for every element in validation set.

$$\text{Accuracy} = \left( \frac{\text{number of correctly guessed elements}}{\text{total number of elements in validation set}} \right) \times 100\%$$

# 3. Methotology of Naive Bayes Classifier

## Description of Operation

The Bayesian classifier operates based on three key elements:

- **Prior probability (initial probability)**: This is our initial belief about the chances that a given text belongs to each category. In our case, these probabilities are calculated based

on the ratio of the number of words in the dictionary of each category to the total number of words in all dictionaries. Since each dictionary has 150 words, this probability will be the same for all classes and will be $\frac{1}{3}$.

- **Conditional probability of words in the class**: This refers to the probability that a given word will appear in a text that belongs to a specific category. This is calculated based on the frequency of words in a given category. To avoid a scenario where a word has a probability of 0, we use Laplace smoothing, which in this case means adding 1 to the occurrence of each word in the text.
- **Conditional probability of text in the class**: This is what we actually want to find out. It is the probability that a given text belongs to a specific category, based on the words that appear in it. It is calculated based on the conditional probability of words in the class, for all the words in the text.

**The conditional probability of a word $w$ for a class $C$** is calculated as:

$$P(w|C) = \frac{N_{w,C} + 1}{N_C + V}$$

where:

- $P(w|C)$ - denotes the conditional probability of the word $w$ in the class $C$
- $N_{w,C}$ - is the number of occurrences of the word $w$ in the class $C$
- $N_C$ - is the total number of words in the class $C$
- $V$ - denotes the number of unique words in all classes

**The conditional probability of a text $T$ for a class $C$** is calculated as:

$$P(C|T) = \log(P(C)) + \sum_{w \in T} \log(P(w|C))$$

where:

- $P(C)$ is the prior probability of the class $C$.
- $P(w|C)$ is the probability of the word $w$ occurring in the class $C$.

Logarithmization allows for summing the logarithms of probabilities instead of multiplying the probabilities themselves, which is numerically more stable.

With Laplace smoothing, if a word $w$ does not appear in the variable storing the probabilities of word occurrences for the class $C$, we use:

$$P(w|C) = \frac{1}{\text{total\_count} + \text{L(v)}}$$

where:

- total_count is the sum of all word occurrences in that class plus 1 for each word (smoothing).
- len(self.vocab) is the number of unique words in all dictionaries.

- v is a vector
- L(v) is a length of the vector

The conditional probability

- $P(C \wedge W)$ - the joint probability of $C$ and $W$ occurring.
- $P(C \mid W) = \frac{P(C \wedge W)}{P(W)}$ - conditional probability: if $W$ has occurred, then the probability that $C$ has also occurred is $P(C \mid W)$.

where:

- $C$ denotes the class to which we assign the text.
- $W$ denotes the features of the text, i.e., the words in the text.

## Algorithms

**Algorithm 2:** Calculating conditional probabilities of words.

**Data:** Data dictionaries: *dictionaries*
**Result:** Dictionary of word probabilities: *word_probs*

*word_probs* ← {};
**foreach** *class cls, words words in dictionaries* **do**
    *word_counts* ← defaultdict with default value 1;
    **foreach** *word word in words* **do**
        *word_counts*[*word*] += 1;
    *total_count* ← sum of values in *word_counts*;
    *word_probs*[*cls*] ← {word *word*: $\frac{count}{total\_count}$ for *word, count* in *word_counts*};
**return** *word_probs*

**Algorithm 3:** Predicting the class of a text.

**Data:** Text: *text*
**Result:** Predicted class: *predicted_class*

*text* ← *text* replace all punctuation with spaces, remove quotes, hyphens, exclamation marks, question marks, and apostrophes;
*words* ← split *text* into words, convert to lowercase;
*class_scores* ← dictionary with initial values equal to the logarithm of prior probabilities from *class_probs*;
**foreach** *class cls, word probabilities word_prob in word_probs* **do**
    **foreach** *word word in words* **do**
        **if** *word is in vocab* **then**
            *class_scores*[*cls*] += log(word_prob.get(*word*, $\frac{1}{\text{sum of values in } word\_prob + \lvert vocab \rvert}$));
*predicted_class* ← class with the highest score in *class_scores*;
**return** *predicted_class*

# 4. Experiments

## 4.1. Database

The database was constructed based on the News Category Dataset available on the Kaggle platform. From the original database, the following were utilized: headline category and headline + short description - combined into one column of the table named "Text".
Then we created 150 words long dictonaries consisting most commonly used vocabulary in texts, divided by categories: SPORTS, HOME&LIVING, and FOOD&DRINK.
Using a Python program, the number of words occurring in the text matching the selected categories was calculated. Subsequently, appropriate columns describing these numbers were created (Table 1)
Final database is made out of 1500 records.
Here are some of the records.

| No. | Text | Category | Words |
|---|---|---|---|
| 1 | maury wills... | SPORTS | 183 |
| 2 | boston marathon... | SPORTS | 168 |
| 3 | nfl rookie... | SPORTS | 186 |
| 4 | 10 movies... | HOME & LIVING | 102 |
| 5 | organic gardening... | HOME & LIVING | 112 |
| 6 | hiring a cleaning... | HOME & LIVING | 123 |

**Table 1**
Graphic representation of our database - first part

| No. | Food words | Sports words | Home words |
|---|---|---|---|
| 1 | 2 | 22 | 2 |
| 2 | 0 | 14 | 1 |
| 3 | 0 | 21 | 0 |
| 4 | 1 | 2 | 14 |
| 5 | 3 | 2 | 12 |
| 6 | 2 | 0 | 13 |

**Table 2**
Graphic representation of our database - second part

## 4.2. Tests

### Test for KNN:

The charts(Figure 2.) depict the relationships between the characteristic features of individual abstraction classes and their intensity in the case of elements from other classes. The dispersion of elements has been presented, with colors corresponding to the following classes -

text categories analyzed in this project: 'green' - 'SPORTS', 'yellow' - 'HOME & LIVING', and 'pink' - 'FOOD & DRINK'.

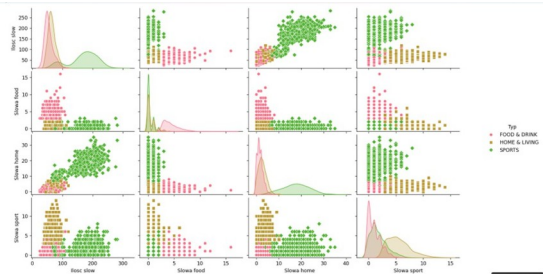

Figure 2: Graphic reprezentation of database

Five tests of the program's accuracy were conducted (Figure 3.) for values of k in the range <2;8>. The results were then averaged (Figure 4.).

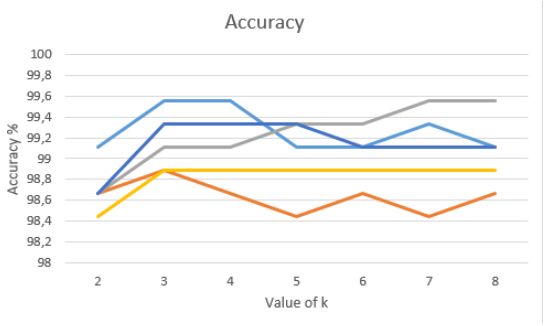


Figure 3: Accuracy for k in {2,3,4,5,6,7,8}

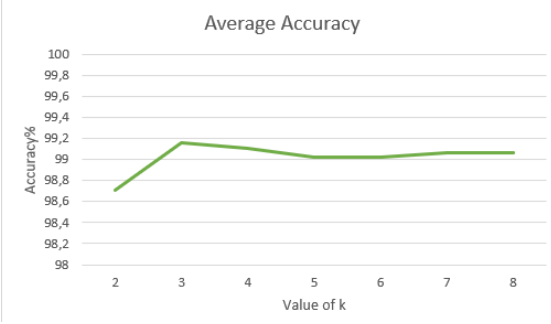


Figure 4: Average accuracy for k in {2,3,4,5,6,7,8}

**Results:**

Average Accuracy for KNN = 99,0222 %

Average time needed to analyse whole base for chosen k = 173,8866626 s

**Conclusion:**

Depending on the content of the training set, the k-nearest neighbors algorithm demonstrates variable accuracy in text analysis [3]. This stems from the uneven distribution of elements

across each abstraction class in the training set, as well as the varying intensity of training attributes among the elements. Therefore, even when operating on the same database, with each new partition of elements into training sets, the results will differ slightly (Figure 3.).

The tests were conducted on the entire database consisting of 1500 records. The ratio of the training set to the validation set is 70:30.

**Test for Naive Bayes Classifier:**

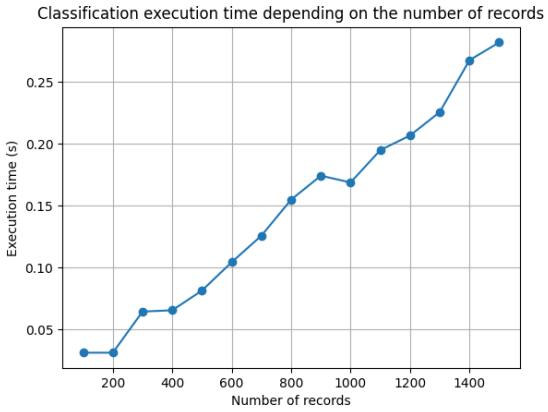


Figure 5: Time depending of number of records

**Conclusions of Figure 5.:**

Figure 5, which shows the classification execution time depending on the number of data rows, shows the expected dependence of the time on the number of rows. The more rows the algorithm tells you to classify, the longer it takes it. Measurements were performed every 100 added rows. The shortest time was for 100 and 200 records, and the longest time was for all of 1500 records of database.

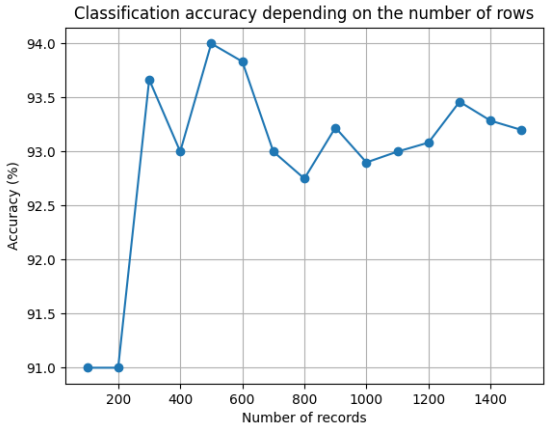


Figure 6: Accuracy depending of number of records

**Conclusions of Figure 6.:**

Figure 4 shows an interesting relationship. The lowest accuracy was recorded for data consisting of 100 rows - 89%, while the highest for 400 rows - almost 95%. From data consisting

of 500 rows, the accuracy slowly normalized until, with 1500 rows, it reached a level of just over 93%. Average Accuracy for Naive Bayes Classifier: 93% Average time needed to analyse database: 0,15 s

### 4.3. Experiments with KNN

**Experiment 1:**

Examine the accuracy and operation time of the program depending on k:

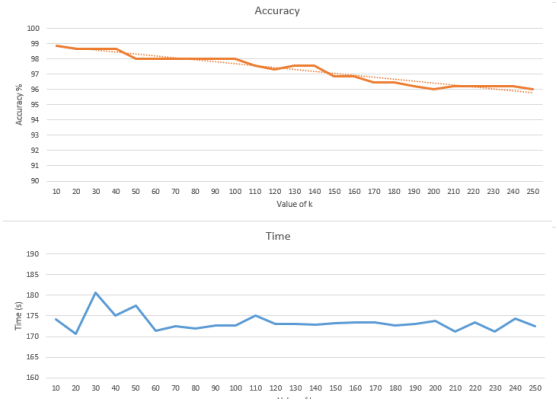


Figure 7: Accuracy and operation time for k in {10,20,30,...,250}

**Conclusions of Figure 7.:**

As the value of k increases, the accuracy of the program decreases.

The trend line can be described by the formula: $-\frac{1}{100}x + 99$. The decrease in accuracy for k belonging to {10,20,30...,250} is slight - about 3%.

The time required to perform the classification for most k values oscillates between 170 s and 175 s.

The standard deviation is: 2.065 s

Thus, it can be stated that the time required to execute the test(k) function is independent of the value of k.

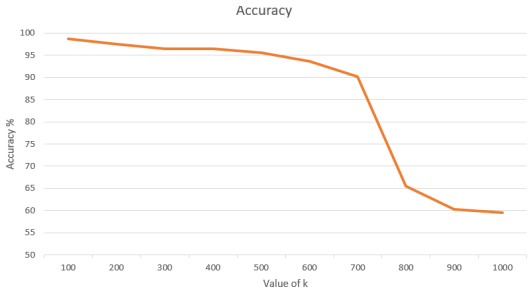


Figure 8: Accuracy for k in {100,200,...,1000}

**Conclusions of Figure 8.:**

Considering larger k values, a sharp drop in classifier accuracy can be observed for k in the range (700,800).

When k is too large, the algorithm begins to average predictions based on a very large number

of neighbors, which leads to a loss of the model's ability to capture local patterns in the data.

### 4.4. Comparing KNN with Naive Bayes Classifier

**Experiment 1:**

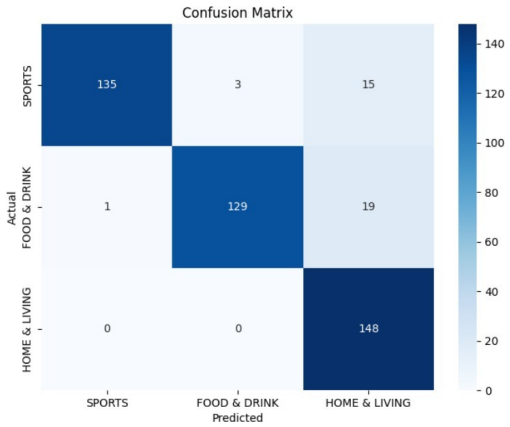


Figure 9: Confusion matrix for Naive Bayes Classifier

**Conclusions of Figure 9.:**

It is noticeable that the algorithm correctly classifies texts in the vast majority of cases. Most often, it confuses the FOOD & DRINK and SPORTS categories with the HOME & LIVING category. This may be due to the dictionaries being insufficiently long or the data being too imprecise.

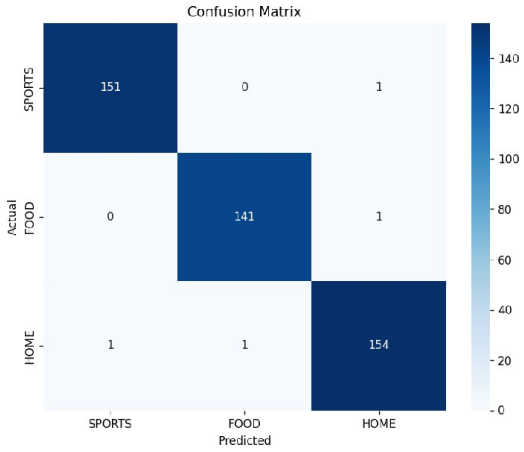


Figure 10: Confusion matrix for KNN

**Conclusions of Figure 9.:**

It is noticeable that the algorithm correctly classifies almost all of the texts. From 450 texts it did not classified only 4.

**Conclusion:**

Most problematic category for both classifiers is category HOME&LIVING. KNN's accuracy is overall a little better than Bayes's.

**Experiment 2:**
Examine the accuracy and operation time of the program depending on the number of analyzed elements:
The entire database is taken, then it is randomly shuffled. The validation and training sets are created from the first n elements of the shuffled database. In KNN k=10.


Figure 11: Accuracy for n(size of dataset) in {100,200,...,1500}

**Conclusions of Figure 11.:**
The accuracy of the classifier for a database size from 100 to 300 elements is lower than for larger n in case of both classifier. The functions shown in the chart are not monotonic; their values, regardless of the database size, slightly increase or decrease. However for bigger values of n accuracy begin to stabilize.

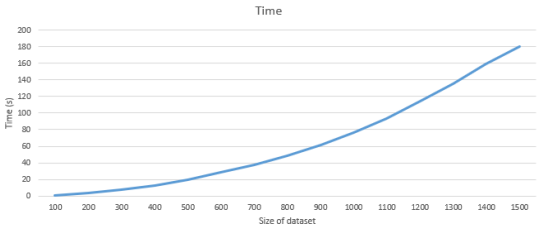


Figure 12: KNN operation time for n(size of dataset) in {100,200,...,1500}

The time required to perform KNN operations increases with the size of the database. Using Lagrange interpolation, the formula can be determined:

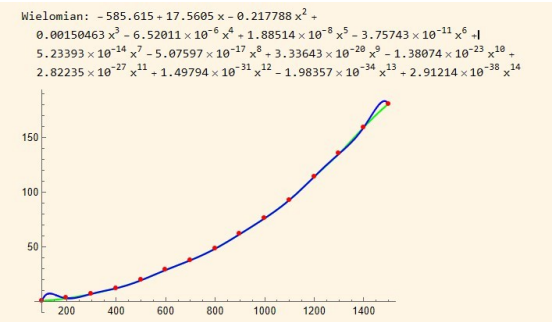


Figure 13: Green line - chart of dependency of time on n
Blue line - polynomial obtained by interpolation

Time needed to perform Naive Bayes Classifier also increases, but comparing to KNN scale of rise is inconsiderable: 0.02 s to 0.29 s.

## 5. Conclusion

In conclusion, our comparative study of K-Nearest Neighbors (KNN) and Naive Bayes Classifier for text classification reveals distinct advantages and limitations of each method. KNN demonstrated superior accuracy, achieving an average accuracy of 99.02%, which significantly outperformed the Naive Bayes Classifier's 91.33%. However, this accuracy came at a cost, with KNN requiring considerably more computational time (173.89 seconds on average) compared to the much faster Naive Bayes (0.29 seconds on average).
Overall, while KNN provides higher accuracy, its computational demands make it less suitable for large-scale applications compared to Naive Bayes, which offers a good balance of speed and accuracy. For applications where speed is critical and slight accuracy trade-offs are acceptable, Naive Bayes is preferable. Conversely, for scenarios demanding the highest possible accuracy and where computational resources are ample, KNN is the better choice. This study underscores the importance of selecting the appropriate classifier based on the specific requirements and constraints of the application at hand.

## References


1. Ferdousy, Elma & Islam, Md & Matin, M.. (2013). Combination of Naïve Bayes Classifier and K-Nearest Neighbor (cNK) in the Classification Based Predictive Models. Computer and Information Science. 6. 10.5539/cis.v6n3p48.
2. Muliono, Yohan & Tanzil, Fidelson. (2018). A Comparison of Text Classification Methods k-NN, Naïve Bayes, and Support Vector Machine for News Classification. Jurnal Informatika: Jurnal Pengembangan IT. 3. 157-160. 10.30591/jpit.v3i2.828.
3. Tan, Songbo. (2006). An effective refinement strategy for KNN text classifier. Expert Systems with Applications. 30. 290-298. 10.1016/j.eswa.2005.07.019.
4. J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers," no. 1973, 2003.
5. Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. Proceedings of ECML '98.
6. Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. Proceedings of SIGIR '99
7. Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. Proceedings of ECML '98.
8. Zhang, T., & Oles, F. J. (2001). Text categorization based on regularized linear classification methods. Information Retrieval, 4, 5–31.
9. Sun, Jingwen & Du, Weixing & Shi, Niancai. (2018). A Survey of kNN Algorithm. Information Engineering and Applied Computing. 1. 10.18063/ieac.v1i1.770.

10. Jayaprakash, Sreemathy & Balamurugan, P. (2022). AN EFFICIENT TEXT CLASSIFICATION USING KNN AND NAIVE BAYESIAN. International Journal on Computer Science and Engineering. 4. 392-396.
11. Chomboon, Kittipong & Chujai, Pasapitch & Teerarassammee, Pongsakorn & Kerdprasop, Kittisak & Kerdprasop, Nittaya. (2015). An Empirical Study of Distance Metrics for k-Nearest Neighbor Algorithm. 280-285. 10.12792/iciae2015.051.