Prediction mechanisms using k-nn and Naive Bayes algorithms

Bartosz Machniak, Bartłomiej Repeć and Łukasz Szary

Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice, Poland

Abstract
Commonly found problems in the artificial intelligence field are clustering and classification. In ever lasting search for more accurate solutions, newer and better algorithms and technologies are researched and adapted. In this paper we will examine a quite simple and obvious idea of combining the two of the easiest clustering and classification algorithms - Naive Bayes classifier and k-nn in hopes of achieving accuracy higher than found in stand alone respective methods. Modeling, testing and comparing this hybrid solution to its subparts will somewhat deepen one's knowledge of the idea and lead to forming an opinion on the proposed algorithm.

Keywords
Data classification and clustering, k-nn, Naive Bayes, Experimental Approach

1. Introduction

Artificial Intelligence has grown rapidly in recent years, driven by advances in computing power, data storage, and algorithm development. Machine learning, a subfield of AI, focuses on the development of algorithms that can learn patterns and make predictions based on data. One of the key challenges in machine learning is to develop algorithms that can perform accurate and efficient classification tasks. This is where the k-nn and Native Bayes algorithms have their applications. The k-nn algorithm is a non-parametric approach that classifies new data points based on the similarity of their features to the features of the training set. In our experiments, we used Taxicab, Euclidean, and Minkowski metrics. In contrast, the Naive Bayes algorithm is a probabilistic approach that calculates the probability of a new data point belonging to each class based on the frequency of each feature in the training set. We found Gauss distribution helpful in connection with our problems. Both algorithms have their advantages and limitations, and choosing the appropriate algorithm depends on the specific problem at hand. The rest of this paper will provide an in-depth comparison of the k-nn and Naive Bayes algorithms, examining their underlying principles, implementation, and performance on various datasets. The comparison will help researchers and practitioners to understand the strengths and weaknesses of each algorithm and make informed decisions about which algorithm to use for a particular task. The paper also focuses on a hybrid algorithm that connects Naive Bayes and k-nn to achieve more accurate results.

26th Computer Vision Winter Workshop, Robert Sablatnig and Florian Kleber (eds.), Krems, Lower Austria, Austria, Feb. 15-17, 2023

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

1.1. Used dataset

To test and examine the results of the algorithm a problem to solve is needed. For this paper, we have chosen to use a songs database with the goal of predicting whether a sample song will be a successful - hit song or not.

2. Preparing The Dataset

Firstly the dataset needs to be cleared of needless records - in the test case, it would be empty records, audiobooks, and podcasts. As finding empty entries is rather straightforward, searching for the rest requires setting some criteria. Having decided on parameters that can be used to determine whether a record is a song, we have analyzed extreme cases (examining the data sorted by a specific column) and found what parameter boundaries filter songs from audiobooks. Because the data is taken from reliable sources - Youtube and Spotify statistics, finding and deleting records with false, incorrect values was not needed.

3. Ranking The Records

To decide whether a song is popular or not, views, likes, and comments can be used. We have decided to use a simple function:

$$popularity = views + \frac{likes}{views} + \frac{comments}{views} \quad (1)$$

Making it mostly depended on the views.

Leaving us with a numerical rating of each record. Then a limit of rating is decided - how much rating does an entry need to be considered a hit, we have settled on:

4. Classification

The goal of this project is to apply and test a mix of k-nn and Naive Bayes algorithms. To "connect" those two, the first will look up for k similar to sample entries, which then will be passed to the second that will determine by the probability class of a record.

To train and test this model, data needs to be split into training and validation sets. We have decided to test it on:

The program has to get the training set and each entry from the validation set and verify if the outcome is truthful by comparing it against earlier acquired classification.

5. K - Nearest Neighbours

In this part training set gets an additional parameter for each record - the distance between each parameter of the training set entry and the current sample. It can be acquired with various metrics, we have tested it with the Taxicab and Minkowski metrics. Then the extended training set is sorted by the distance and the K top entries are chosen for the next section.

Algorithm 1: Distance measurement algorithm

```
Data:
$sample$ - record for which the algorithm searches
the k nearest neighbours
$columnNames$ – names of columns in $dataset$
which will be used to calculate distance
$f$ – metric function used to calculate distance e.g.
Taxicab, Minkowski
$list$ - array full of 0 values, $list$ length is the same
as number of records in $dataset$
Result:
$list$ – array of distances between $sample$ and
each record in dataframe

$i = 0$
for row in x do
    for col in columnNames do
        list[i] += f(x[row][col], power)
    end
    $i$++
end
Return: $list$
```

6. Naive Bayes

The argument dataset is compromised of k, similar to the sample, records. Splitting data into two by the class (in this case the boolean value of a record being a hit) allows for a simple calculation of probability for each case. Probability value can be obtained with various functions. We have decided to use Gauss's standard normal distribution function:

$$p(x = v|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu)^2}{2\sigma_k^2}} \tag{2}$$

where:

- σ^2 is the parameter column's standard deviation
- μ is the parameter column's mean
- C is the set of values of the k parameter's column
- v is the sample's value of this parameter

Algorithm 2: Naive Bayes

```
Data:
$sample$ – record for which algorithm searches
the nearest neighbours
$classCol$ – name of column used as a classifier
$columnNames$ – names of columns in $dataset$
which are used to calculate the distance
$classes$ - array holding unique values of
$classCol$ column in the $dataset$
$res$ - dictionary storing values of fields for
specific classes
Result:
0 or 1 depending on if the sample record is or is
not a hit according to Bayes

for value var in classes do
    $res[var]$ = []
    for $cl$ column name in $columnNames$ do
        $values$ = column $cl$ values from $x$ only
          records where $classCol$ column has $var$
          value
        $mean$ = mean of $values$
        $sig2$ = second power of standard
          deviation of $values$
    end
    $res[var]$ = mean of $res[var]$
end
Return: key of the max value of $res$
```

7. Connection

To feed Bayes part with appropriate data, a simple algorithm is used.

Algorithm 3: Connection

```
Data:
$stats$ - list of attributes to be used in Bayes
$classCol$ – name of column used as a classifier
Result:
Accuracy of Naive Bayes based on k-nn results

$counter = 0$
for row in testing-set do
    list= $getDistance$(trainingset, $row$, $stats$, function for specific distance metric, $power$)
    $recs$ = training-set sorted by $list$ $kRecords$ = first k records of $recs$
    if naiveBayes($kRecords$, $row$, $classCol$, $stats$) == $row$[$classCol$] then
        $counter$ += 1
    end
end
Return: $counter$ / length of testingset
```

8. Tests and outcome

8.1. Sensitivity and specificity

Table 1
Manhattan by k

k	TP	TN	FP	FN	%
100	1063	218	296	339	67
50	1094	206	308	308	68
35	1096	210	304	306	68
20	1103	205	309	299	68
10	1109	199	315	293	68
5	1109	199	315	293	68

Table 2
Euclides by k

k	TP	TN	FP	FN	%
100	653	309	205	749	50
50	840	230	284	562	56
35	969	203	311	433	61
20	1342	38	476	60	72
10	1383	12	502	19	73
5	1402	0	514	0	73

Table 3
Minkowsy 3 by k

k	TP	TN	FP	FN	%
100	3	512	2	1399	27
50	5	513	1	1397	27
35	0	514	0	1402	27
20	0	514	0	1402	27
10	0	514	0	1402	27
5	0	514	0	1402	27

8.2. Accuracy

Data showing the accuracy of the algorithm depending on chosen metric and k of the k-nn sub algorithm.

Table 4
Manhattan by training set length and k

TS	k = 20	k = 10	k = 5
75%	0.67	0.67	0.67
50%	0.67	0.67	0.67
25%	0.64	0.64	0.65

Table 5
Euclidean by training set length and k

TS	k = 20	k = 10	k = 5
75%	0.69	0.70	0.72
50%	0.63	0.71	0.72
25%	0.68	0.72	0.72

Table 6
Minkowski 3 by training set length and k

TS	k = 20	k = 10	k = 5
75%	0.28	0.28	0.28
50%	0.71	0.71	0.72
25%	0.72	0.72	0.72

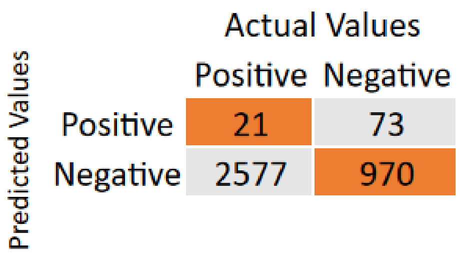
9. Testing and efficiency

9.1. Reasoning

To declare this combination of k-nn and Naive Bayes a successful and optimal solution to some problems, a comparison of sub algorithms to the whole algorithm is needed. If this project does not produce more accurate results at the same cost as corresponding k-nn (which requires the most computing power in the operation) or Naive Bayes (cheapest computing power wise but with poor accuracy), there is no point in using this hybrid as it is a needless complication to already well-known tools.

9.2. Stand alone Naive Bayes

Working on the same dataset, run on different proportions of the training set to the validation set the best accuracy achieved by Naive Bayes was barely 31%. Interestingly the low, worse than random, performance can be also seen by inspecting the sensitivity and specificity of the algorithm:



Actual Values / Predicted Values

	Positive	Negative
Positive	21	73
Negative	2577	970

Due to poor performance, there is no need to compare Naive Bayes to the studied algorithm.

9.3. Stand alone k-nn

Running the same dataset through k-nn with different metric functions and different k, same as with the combination, yielded the following results:

The graph shows the accuracy achieved by k-nn with different metric functions and different k. It is already easy to notice how high those results are. If we compare the best results of k-nn and hybrid:

We can see that even against the best-found setups k-nn performs better.

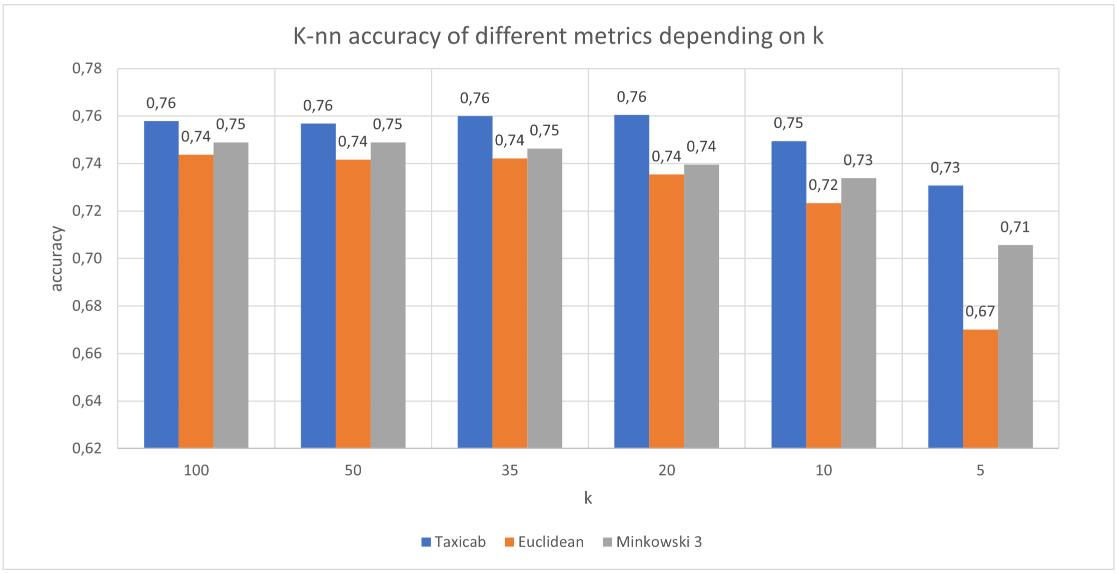
10. Conclusions

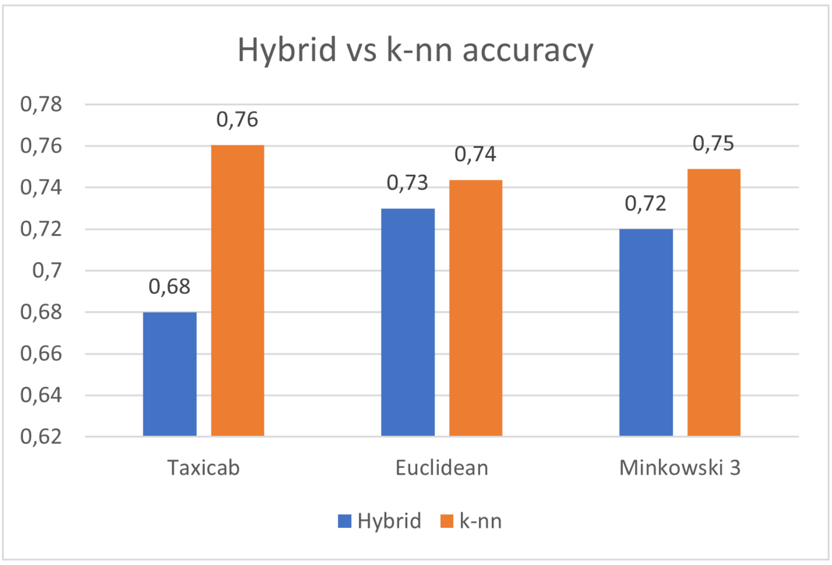
Even though the hybrid solution offers two stages of classifying, on the face of it making it more accurate, it performs worse than the simple k-nn algorithm. As noticed earlier, the Naive Bayes classifier running on the used dataset scores exceptionally low. Even though k-nn supplies the second part of the examined algorithm with quite a high percentage of realistically similar samples, Naive Bayes classifies them wrongly. So, speaking in terms of the used dataset, if a sample is a hit song, and more than half of the k closest samples are also hit songs, the whole algorithm will end up marking it as a not a hit song.

Although there may exist a combination of metric function, probability function, k, and a training set which would make a hybrid algorithm perform better than stand-alone, equally, perfectly tweaked, k-nn or Naive Bayes, the examined values did not yield positive results, making us conclude that the subject algorithm is not useful or better than the sub algorithms used.

References

- L. Moroney : Sztuczna inteligencja i uczenie maszynowe dla programistów. Praktyczny przewodnik po sztucznej inteligencji, Helion.
- R. Tadeusiewicz : Sieci neuronowe, Akademicka Oficyna Wydawnicza RM, Warszawa.
- D. Foster: Deep learning i modelowanie generatywne, Helion.
- M. Lutz : Python. Wprowadzenie. Helion, Gliwice.
- J. Krohn, G. Beyleveld, A. Bassens: Uczenie głębokie i sztuczna inteligencja, Helion.


K-nn accuracy of different metrics depending on k
accuracy
0,78
0,76
0,74
0,72
0,70
0,68
0,66
0,64
0,62
0,76
0,74
0,75
0,76
0,74
0,75
0,76
0,74
0,75
0,76
0,74
0,74
0,75
0,72
0,73
0,73
0,67
0,71
100
50
35
20
10
5
k
Taxicab
Euclidean
Minkowski 3


Hybrid vs k-nn accuracy
0,78
0,76
0,74
0,72
0,7
0,68
0,66
0,64
0,62
0,68
0,76
0,73
0,74
0,72
0,75
Taxicab
Euclidean
Minkowski 3
Hybrid
k-nn