International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869 (O) 2454-4698 (P) Volume-7, Issue-11, November 2017

Machine Learning Algorithms Comparison

Kostandina Veljanovska

***Abstract* - The paper compares performance of two machine learning algorithms, Naive Bayes and k-Nearest Neighbor in real world problem. We selected problem of choosing the car or public transportation for traveling from home to work. The aim of the paper is to help the local government in making influence of choosing the way of travel to the citizens. This will help reduce the environmental pollution, reduce the fuel consumption and also, improve air quality. The results are feasible for implementation in urban planning for various environments since the machine learning algorithms are used and the model is capable of learning from presented data.**

***Index Terms*—artificial intelligence, k-Nearest Neighbor algorithm, machine learning, Naive Bayes algorithm.**

I. INTRODUCTION

Air pollution solution is currently top priority for the government in the Republic of Macedonia. Many studies and research are undertaken and they show that not only heating, but also traffic and transportation is main cause for air pollution. In that direction we would like to contribute to the solution of this problem. This study was undertaken in order to show how transportation means selection could be predicted in order to undertake measures to influence citizens in their way of travel selection. The problem is interesting for the local government since addresses the way of behavior of potentially drivers and/or users of public transport.

For the purpose of fulfilling the aim, we select two different algorithms from artificial intelligence and set the problem as solving a classification problem. This type of research has not been done in our country using artificial intelligence, yet.

Since machine learning is taken as the practice of using algorithms to parse data, learn from data, and then make a determination or prediction about something in the world, we could say we use machine learning. Instead of hand-coding software routines with a specific set of instructions to accomplish a particular task, the machine was "trained" using large amounts of data and algorithms that give it the ability to learn how to perform the task. The algorithmic approaches over the years included decision tree learning, inductive logic programming, reinforcement learning, and Bayesian networks among many others.

Machine learning have been widely used for classification and pattern recognition. Some of the implementations are commercially very popular.

Some techniques perform well on numerical and text data like Naive Bayes [1, 2, 3]. There is a research for determining if smart environment sensor data can be used to predict air quality levels implementing NB [4]. Naïve Bayes is potentially good at serving as a document classification model due to its simplicity [2]. Neural Networks can handle both discrete and continuous data [1]. k – Nearest Neighbors (k-NN) is a time consuming method and finding the optimal value is always an issue [1, 5] even if regarding air quality prediction some specified software is used for the implementation of the algorithm [6]. Decision tree can reduce the complexity but is unable to handle continuous data.

In the second section of the paper, the model of the problem is presented. Naïve Bayes classifier is discussed in detail, and also, k-NN is discussed in terms of theoretical basics. The model of prediction for our concept is described as well. The third and fourth sections discuss the results of the Naïve Byes and k-NN algorithms and their performances are compared.

There are many implementations of Naive Bayes and k – Nearest Neighbors algorithms in the real world. The strength of Naive Bayes is that it is parametric, it assumes probability distribution for data, and learns parameters from data. Naïve Bayes along with its simplicity is computationally cheap also. On the other side, k-NN as a non-parametric algorithm has no fixed number of parameters.

II. MODELLING THE REAL WORLD PROBLEM

A. *The Naive Bayes Classifier algorithm*

Learning Bayes classifiers typically requires very huge, almost an unrealistic number of training examples (sometimes more than |X| training examples where X is the instance space) unless some form of prior assumption is made about the form of probability P(X|Y). The Naive Bayes classifier algorithm assumes that all attributes describing X are conditionally independent given Y. This assumption dramatically reduces the number of parameters that have to be estimated in order to learn the classifier. Naive Bayes is a widely used learning algorithm, for both discrete and continuous X. [7]

This gives a Naive Bayes classifier feature of an algorithm that is simple probabilistic classifier based on implementation of Bayes theorem with strong (naive) independence assumptions. An advantage of the classifier that we use is that it only requires a small amount of training data to estimate the parameters necessary for classification.

A Naive Bayes algorithm assumes that the presence of a particular feature of a class is unrelated to the presence of any other feature.

B. *k - Nearest Neighbor algorithm*

One of the simplest decision methods that can be used for classification is the rule of nearest neighbor. It classifies a sample based on the category of its nearest neighbor. When large samples are involved, it can be shown that this rule has an error probability less than twice the optimum error. The algorithms based on the nearest neighbor classification use some or all the patterns available in the training set for the aim of classifying or recognizing a test pattern. These classifier algorithms involve finding the similarity between the test pattern and every pattern in the training set. [8]

Kostandina Veljanovska, Department of Intelligent Systems, Faculty of ICT, University "St. Kliment Ohridski", Bitola, Republic of Macedonia,

The nearest neighbor decision rule assigns to an unclassified sample point the classification of the nearest of a set of previously classified points. [9]

If we try to explain how the k-NN algorithm works we could say that given a query vector x_0 and a set of N labeled instances $\{x_i,y_i\}_1^N$, the task of the classifier should be to predict the class label of x_0 on the predefined P classes. The k-NN classification algorithm tries to find the k nearest neighbors of x_0 using a majority vote in order to determine the class label of x_0. The k-NN classifier usually applies Euclidean distances as the distance metric. In spite of being simple and easy-to-implement algorithm, k-NN can still yield competitive results even compared to the most sophisticated machine learning algorithms. The performance of a k-NN classifier algorithm is primarily determined by the choice of k as well as the distance metric applied. [10, 11]

C. *Modelling the database*

Database was created based on the real-world concept going to work with car or not (by car and public transport or by public transport).

We tried to construct at least 50 positive instances of our concept and at least 50 negative ones. Our negative instances were 'near misses' but we also have several 'far misses' as it is in real world when we choose way of travel. Each instance was represented by many attributes and each of them have on average four possible values.

The meaning of attributes and concept is shown in Table1.

Table1. Database concept

Attributes (x_i)	Possible values	Meaning
Ttwcar	1,2,3,4	travel time to/from work by car (15, 30, 45, 60 minutes)
Ppay	1,2,3,4	parking fee (20,30,40,50 in denars)
Pplace	0,1,2,3	parking search (2, 5, 10, 15 minutes)
Pwdistance	0,1,2,3	parking to work distance (2, 5, 10, 15 minutes)
Ptttime	1,2,3,4	public transport travel time (15, 30, 45, 60 minutes)
Transfer	1,2,3,4	how many times transfer is needed (1,2,3,4)
Walk	1,2,3,4	walking time from home/work to public transport station (2,5,10, 15 minutes)
Weather	1,2,3,4	currently is sunny,windy, raining, snowing (s,w,r,x)
Job	1,2,3	type of job (flex time, flex place, must be on time)
R1	1,2,3,4	Salary (less than 15000, 20000, 30000, more than 30000 denars)
R2	1,2,3,4,5,6,7	Years of travelling with public transportation
R3	1,2,3,4	Number of persons in the family
R4	1,2,3,4	Number of cars
go	1,0	Go by car or otherwise (yes/no)

In order to get nominal attributes, mapping nominal value to integer values is done. This helps us to get distance between values:

the values of "S(unny)", "W(indy)", "R(aining)" and "X(snowing)" are set as 1,2,3,4, respectively

the values of "flex time", "flex place", must be on time" are set as 1,2, 3

the values of "less than 1500", "2000", "3000" and "more than 3000" are set as 1,2,3,4 respectively.

Value of attribute R1 was created using Rand().

```
If rand()<0.25, R1=1;
  Else, if  rand()<0.33, R1=2,
        Else if rand()<0.5, R1=3
           Else R1=4.
```

R3 was created the same way as R1.

```
R2=int(R1+rand()*3);
R4=int(R3+rand()*4);
```

So, attributes R1 and R2, R3 and R4, are two pairs of random and independent features.

D. *Hypothesis evaluation*

The total number of records in the database was 110. Records were randomly divided into 10 disjoint subsets of 11 records. This was done by shuffling the data and then dividing the records according to the sequence.

9 of the 10 subsets were used as training data and the one left was tested. This was done 10 times so that every subset was tested.

E. *The Naive Bayes Classifier algorithm*

We implement NB algorithm this way: training and testing data were taken from the database. The learning rule was created according to the Bayes Theorem.

$$P(H|E)=(p(E|H)*P(H)/P(E)$$

Where, P(H) is the probability of hypothesis H being true (known as the prior probability). P(E) is the probability of the evidence (regardless of the hypothesis). P(E|H) is the probability of the evidence given that hypothesis is true. P(H|E) is the probability of the hypothesis given that the evidence is there.

F. *1-Nearest Neighbor algorithm*

For the NN algorithm we started first with 1-NN. Training and testing data were taken from the same database. The instances (hypotheses) were divided into 10 subsets. Every instance in one subset was taken as query instance and the instances in other 9 groups were used to test this query instance.

For each instance in each test set, algorithm finds one nearest neighbor among the instances in other 9 training sets. The distance between two instances of x1 and x2 was:

$$d(x1,x2)=(\sum_{r=1}^{n}(a_r(x_1)-a_r(x_2))^2)^{0.5}$$

where a_r(r=1,2,…n) were attributes of an instance.

For the tie-breaking: when two nearest neighbors has the same distances with the test instance it is required to decide which one should be taken as the nearest neighbor. The Tie-breaking method will discard the last attribute and calculate the distance again (r=n-1). If the distance is the same, algorithm continues to discard the second attribute and compare the new distance (r=n-2)…If all attributes are the same, the new one is replacing the older one.

So, the 1-NN algorithm classifies the query instance according to the concept value of the nearest neighbor.

III. Discussion of the Results

Based on the database created for this research, the accuracy of test results of the Naive Bayes algorithm and 1-Nearest Neighbor algorithm were as follows.

Test Results for the accuracies by the Naive Bayes algorithm for each subset were: 1.0; 0.818; 0.818; 1.0; 1.0; 0.818; 0.818; 0.636; 0.909; 0.818; respectively.
Summary of results is shown in Table 2.

Table2. Naive Bayes algorithm results summary

Measure	Result
Mean Accuracy	0.863
Standard Deviation of Accuracy	0.115
Number of total tests	110
Number of test correct	95
Percent of correctness	86

Results of 1-NN algorithm regarding accuracies were: 0.909; 0.818; 0.909; 0.909; 1.0; 0.818; 1.0; 0.818; 0.909; 0.636; respectively for each subset.
Summary of results is shown in Table 3.

Table3. 1-NN algorithm results summary

Measure	Result
Mean Accuracy	0.872
Standard Deviation of Accuracy	0.106
Number of total tests	110
Number of test correct	96
Percent of correctness	87

A. *Tuning for k-NN algorithm*

In order to judge which value of k is best for the nearest-neighbor algorithm the leave-one-out method was used. We consider several values for k: 1, 3, 5 and 7. We use odd values in order to avoid ties in classification. For each value of k, iterations were done through all the training examples and inspections were performed to see how the current example would be classified. For each example, its k nearest neighbors were collected and then the most common category among these k neighbors was taken. In case of ties when collecting the nearest neighbors a tie-breaking policy was applied.

Using Leave-one-out method by repeating n (n=110) times for n instances (cases), each time leaving one case out for testing and the rest (109 cases) for training, we get the accuracies of the tests shown in Table 4.

Table 4. The results after tuning for k-NN algorithm

Number of Neighbors (K)	Result (%)
1	85
3	79
5	80
7	82

As it can be seen the result for k =1 is the best.

We choose the k that does best in the leave-one-out experiment (breaking any ties by choosing the largest k). After tuning the k parameter, we did classification of the corresponding test set using the complete training set.

We have repeated this parameter tuning procedure for each of our ten train/set partitions and have recorded the test set accuracy for each of the ten test sets. k is separately set for each test set and is not necessary the same over all ten test sets.

The values of k were set as: 1,3,5,7, respectively. For every k, k-NN algorithm have been used. 9 groups as training instances and another as test instances were taken and repeated for 10 times to test every group. The results of accuracy are as shown in Table 5.

Table 5. The accuracies after tuning for k-NN algorithm

Number of Neighbors (K)	Accuracies
1	0.872
3	0.845
5	0.809
7	0.836

IV. Comparing Learning Algorithms and Conclusions

Comparison among 1-NN, NB and k-NN algorithms was done regarding our database. To test for statistical significance in the difference of generalization performance we have applied three paired tests. It was found that because $t0 > -t_{.05,9} = -1.833$, in 90% of confidence, there is no significant difference between the performance of algorithms of 1-NN and Bayes.

We can conclude that 1-NN perform almost the same as Naive Bayes.

Regarding comparison between k-NN and 1-NN, because $t0 > -t_{.05,9} = -1.833$, in 90% of confidence, also, there is no significant difference between the performance of algorithms of 1-NN and 7-NN.

After comparing k-NN and Naive Bayes algorithm, it can be concluded that because $t0 < t_{.05,9} = 1.833$, in 90% of confidence, there is no significant difference between the performance of algorithms of 1-NN and Bayes.

A. *Generalization Performance after Injecting Randomness and Conditional Dependence*

In order to improve generalization we add two pairs of random features to the dataset of our concept. The features of each pair were conditionally dependent given the class. The same steps were performed on the new database.

The following Table 6, compares the results of accuracy between before and after injecting randomness and conditional dependence (CP) of 4 features R1, R2, R3, R4.

Table 6. Accuracy before injecting randomness and CP

Algorithm	NoTests	NoTest Corr	%
1-NN-algorithm	110	96	87
Naive-Bayes-algorithm	110	95	86
1-NN-algorithm(Leave-one-out)	110	94	85
3-NN-algorithm(Leave-one-out)	110	87	79
5-NN-algorithm(Leave-one-out)	110	88	80
7-NN-algorithm(Leave-one-out)	110	91	82
1-NN-algorithm	110	96	87
3-NN-algorithm	110	93	84
5-NN-algorithm	110	89	80
7-NN-algorithm	110	92	83

After Injecting randomness and conditional dependence the results were as follows (Table 7.):

Table 7. Accuracy after injecting randomness and CP

Algorithm	**No Tests**	**No Test Corr**	**%**
1-NN-algorithm	110	89	80
Naive-Bayes-algorithm	110	95	86
1-NN-algorithm(Leave-one-out)	110	88	80
3-NN-algorithm(Leave-one-out)	110	82	74
5-NN-algorithm(Leave-one-out)	110	80	72
7-NN-algorithm(Leave-one-out)	110	82	74
1-NN-algorithm	110	89	80
3-NN-algorithm	110	82	74
5-NN-algorithm	110	82	74
7-NN-algorithm	110	79	71

It can be seen that tuning for k =1 is the best.

After comparison between 1-NN and Naive Bayes algorithm (Table 8.), because $t0<t_{.05,9}= 1.833$, in 90% of confidence, it can be concluded that there is no significant difference between the performance of algorithms of 1-NN and Bayes.

Table 8. Comparison 1-NN and NB algorithm

Test set	1	2	3	4	5	6	7	8	9	10	Ave	sum
1-NN	0.909	0.727	0.909	1	1	0.636	0.636	0.636	0.818	0.818	0.81	
Bayes	1	0.727	0.818	1	1	0.818	0.909	0.636	0.909	0.818	0.86	
E(1NN)	0.09	0.27	0.09	0.00	0.00	0.36	0.36	0.36	0.18	0.18	0.19	
E(B)	0.00	0.27	0.18	0.00	0.00	0.18	0.09	0.36	0.09	0.18	0.14	
Diff	0.09	0.00	-0.09	0.00	0.00	0.18	0.27	0.00	0.09	0.00	0.05	
Square	0.00	0.00	0.02	0.00	0.00	0.02	0.05	0.00	0.00	0.00		0.10
										S=	0.03	
										t0=	1.62	

When compared k-NN and 1-NN (Table 9.), because $t0>t_{.05,9}=1.833$, in 90% of confidence, we conclude that there is significant difference between the performance of algorithms of 1-NN and 3-NN. So, it was concluded that 1-NN is better than 3-NN.

Table 9. Comparison k-NN and 1-NN algorithm

Test set	1	2	3	4	5	6	7	8	9	10	Ave	sum
3-NN	0.818	0.636	0.909	0.909	0.909	0.727	0.545	0.545	0.909	0.545	0.75	
1-NN	0.909	0.727	0.909	1	1	0.636	0.636	0.636	0.818	0.818	0.81	
E(3NN)	0.18	0.36	0.09	0.09	0.09	0.27	0.46	0.46	0.09	0.46	0.25	
E(1NN)	0.09	0.27	0.09	0.00	0.00	0.36	0.36	0.36	0.18	0.18	0.19	
Diff	0.09	0.09	0.00	0.09	0.09	-0.09	0.09	0.09	-0.09	0.27	0.06	
Square	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.02	0.04		0.10
										S=	0.03	
										t0=	1.91	

Finally, k-NN was compared with Naive Bayes (Table 10.). Because $t0>t_{.01,\ 9}= 2.821$, in 98% of confidence, it was concluded that there is significant difference between the performance of algorithms of 3-NN and Bayes. It can be concluded that Naive Bayes algorithm is better than 3-NN or k-NN.

Table 10. Comparison k-NN and NB algorithm

Test set	1	2	3	4	5	6	7	8	9	10	Ave	sum
3-NN	0.818	0.636	0.909	0.909	0.909	0.727	0.545	0.545	0.909	0.55	0.75	
B	1	0.727	0.818	1	1	0.818	0.909	0.636	0.909	0.82	0.86	
E(3NN)	0.18	0.36	0.09	0.09	0.09	0.27	0.46	0.46	0.09	0.46	0.25	
E(Bayes)	0.00	0.27	0.18	0.00	0.00	0.18	0.09	0.36	0.09	0.18	0.14	
Diff	0.18	0.09	-0.09	0.09	0.09	0.09	0.36	0.09	0.00	0.27	0.12	
Square	0.00	0.00	0.04	0.00	0.00	0.00	0.06	0.00	0.01	0.02		0.15
										S=	0.04	
										t0=	2.90	

As final conclusion, it can be seen that performance of the algorithms changes after inserting two pairs of random and conditionally dependent features. There is no significant changes in Naive Bayes algorithms, but for k-NN, the performance decreases for about 5% after inserting two pairs of random and conditionally dependent features.

REFERENCES

[1] G. Kaur et al, *A Review Article on Naive Bayes Classifier with Various Smoothing Techniques*, International Journal of Computer Science and Mobile Computing, Vol.3 Issue.10, October- 2014, pg. 864-868

[2] S.L. Ting, W.H. Ip, Albert H.C. Tsang, *Is Naïve Bayes a Good Classifier for Document Classification?*, International Journal of Software Engineering and Its Applications Vol. 5, No. 3, July, 2011

[3] D. Soria, et al., *A 'non-parametric' version of the naive Bayes classifier*, Knowledge-Based Systems, Elsevier, Vol.24, Issue 6, 2011, pages 775-784.

[4] S. Deleawe, et al., Predicting Air Quality in Smart Environments, J Ambient Intell Smart Environ. 2010; 2(2): 145–152.

[5] Y. Zhao, et al. *Comparison of Three Classification Algorithms for Predicting pm2.5 in Hong Kong Rural Area*, Journal of Asian Scientific Research, 2013, 3(7):715-728

[6] Elia Georgiana Dragomir, *Air Quality Index Prediction using K-Nearest Neighbor Technique*, BULETINUL Universităţii Petrol – Gaze din Ploieşti, Vol. LXII No. 1/2010, pages 103 – 108.

[7] T. Mitchell, Machine Learning, draft for second edition, 2016

[8] M. N. Murty, et al. Nearest Neighbour Based Classifiers, Springer's Pattern Recognition, Universities Press (India) Pvt. Ltd. 2011, pp 48-85

[9] Cover, T.M. and P.E. Hart, 1967. *Nearest neighbor pattern classification,* Institute of electrical and electronics engineers transactions on information theory. 13: 21-27

[10] Y. Song et al., *Ik-NN: Informative K-Nearest Neighbor Pattern Classification*, Department of Computer Science and Engineering, The Pennsylvania State University, PA, 2006

[11] *D. R. Wilson and T. R. Martinez (1997) "Improved Heterogeneous Distance Functions", Journal of Artificial Intelligence Research, Volume 6, pages 1-34*

Prof. d-r Kostandina Veljanovska. D-r Kostandina Veljanovska completed her education at the University "Sts. Kiril i Metodi", Skopje (BSc in Computer Science), at the University of Toronto, Toronto (MASc in Applied Engineering) and got her MSc and also her PhD in Technical Sciences at the University "St. Kliment Ohridski", Bitola, R. Macedonia. She has completed postdoc in Artificial Intelligence at the Laboratory of Informatics, Robotics and Microelectronics at the University of Montpellier, Montpellier, France. She worked as a Research assistant at the ITS Centre and Testbed at the Faculty of Applied Science, University of Toronto, Canada. She also, worked at research team for Constraints, Learning and Agents at LIRMM, University of Montpellier. Currently, she works as an Associate Professor in Information Systems and Networks, Artificial Intelligence and Systems and Data Processing at the Faculty of Information and Communication Technologies, University "St. Kliment Ohridski" -Bitola, Republic of Macedonia. Her research work is focused on artificial intelligence, machine learning techniques and intelligent systems. She has published numerous scientific papers in the area of interest, as well as several monographic items. She is a reviewing referee for well-known publishing house, journals with significant impact factor in science and also, member of editorial board of several international conferences.