ISSN No:-2456-2165

Survey on ML Algorithms for Medical Guidance System

[1]Ayush Sanklecha, [2]Gowtham L, [3] Harshitha D J, [4] Krishna B M, [5]Poornima N
Assistant Professor
Vidya Vardhaka College of Engineering, Mysuru

Abstract:- Health is very important to each and every person on this world no matter where the individual is from. People at distant places with lack of medical assistance and cities with high demand of medications need quick and affordable medical assistance with higher accuracy. A few health issues can be resolved by themselves with a considerable amount of legit assistance. If technology can help them solve those problems, it'll help people to save time with reduced effort. Everyone who uses internet and technology can do self-checking of their health based on certain symptoms with their queries. It can help in pre-diagnosing the possible diseases and provide alerts on the risks and help prevent it.

Keywords:- *Machine Learning, Decision Tree, Random Forest, Naive Bayes, KNN.*

I. INTRODUCTION

No matter how much people achieve and grow, they can do anything only if they are healthy. People do anything to keep their health in a good condition when they gets to know there is some problem with their health. As we are now in the 4th industrial revolution, we are moving towards digitisation and internet is the new future. Internet is the only means through which ends of the globe can stay connected all the time without much cost and effort. Most of the people nowadays use smart phones and as soon as they encounter some health issues, they first go with the online remedies available to get it cured. As it is available at the fingertip of user, they use it the most and it is really important to provide the correct information. It is one of the important research concern.[2]

The world is adapting the mode of automation in almost every field, but when it comes to diagnosing of health people mostly rely on doctors than to use any automation techniques. This paper is a survey on various machine learning algorithms that can be used to automate the predictions and evaluation of a doctor while diagnosing a patient, more precisely the problem statement will be "How can we turn patients' data into useful information that can enable healthcare practitioners to make effective clinical decisions?" .Our objective is to provide a human-like, exhaustive and informative medical consultation model. Analysing the symptoms and questions asked by patients and detecting the disease of person is conundrum because most of them when they become sick, as a result of lack of medical erudition and experience, will describe their symptoms inaccurately in medical terms.[3]

II. RELATED WORK

We have surveyed on various supervised machine learning to refer to creating and using models that are learned from data, i.e., there is a set of data labeled with the correct answer for the model to learn from.This work aims to explore several competing supervised machine learning classification techniques namely:

- Decision Trees
- Random Forests
- k-Nearest Neighbors (kNN)
- Naive Bayes

A. Decision Trees

Decision trees are built for classification tasks where we need to classify a data point saying that it belongs to a particular class. We are given a training set on which we build our model which in this case in a decision tree. Then we use the test set in order to test the accuracy of the model and see whether it can predict the class of each tuple in the test set correctly. We implemented the decision tree algorithm called as CART [9] which stands for Classification and Regression Trees.The tree which is built is a binary tree where, at each node we first check if all the records belong to the same class label. If not, we try to find the best split at each column by computing impurity for each split in the column using the gini index. The best splits for each of these columns are taken and the one which gives lowest impurity among them is taken as the split for that node and the data is divided according to that split. This process is carried on repeatedly until either we reach a point where all records belong to the same class or when all records have similar attribute values . Computing the best split has to be handled differently for categorical and continuous data.[7]

In case of categorical features first we are mapping these categories into numbers. So if there are two categories 'present' and 'absent', they become 1 and 0. This is basically done in order to convert the entire string numpy data matrix into a float data matrix so that future computations can be made easy. We also keep a record of which columns are categorical and which columns are continuous so it becomes easy to determine which data is categorical and which data is continuous. Then we find all the combinations of the available categories. For example,

if the categories were 'Sunny', 'Cold' and 'Humid' they would first be mapped to 0, 1, 2 and then we would try to find all the possible combinations which are [0, 1] [0, 2] [1, 2] etc. They are considered to be splits. For each of these splits we create two sets. Then we traverse each row in the training data and check if the attribute value is in the split. For example, if we are computing impurity for split [1, 2] we check each row for that attribute value. If the value is 1 or 2 we add that row to set1 else set0.

In case of continuous features we take that specific continuous feature value of each row as a split and compute impurity using that split. So whichever rows have feature value less than the split are considered to be in set0 and whichever rows have a feature value >= split are considered to be in set1. Gini index is computed for both sets and then overall gini (impurity) is calculated for the split. We try to minimize this impurity by choosing the split which gives least impurity. handle_numerical_data() method in our code is used to handle this. In our algorithm, we sorted the entire matrix given to this method using the argsort() method such that the matrix is sorted according to the column of the feature value. Then we traverse through each row of the matrix and select that feature value as a split and the two sets are rows which are above and below the current row as we have sorted the matrix. Once we have our two sets we do the remaining method the same way as done for categorical data.[8]

➢ *Pros*
- Simple to understand and interpret
- Pretty useful and accurate when the dataset is small.
- This model allows us to consider as many different consequences of a decision and weigh the tradeoffs of one decision against another.

➢ *Cons*
- Tree creation becomes difficult and convoluted if there are a lot of uncertain values in the dataset.
- Decision trees are not that particularly good with handling continuous variables
- Decision tree suffers from instability. Slight change in the input can cause a completely different tree to be created. Hence it lacks robustness.
- Causes overfitting issue resulting in reduced accuracy. Needs pre-pruning and post-pruning to improve the accuracy of this classifier.

B. Random Forests

Random Forests is a part of ensemble learning wherein we compute a set of t classifiers for t subsets of training data. The testing phase involves a voting by each of these classifiers to determine the class of the training data and depending on the majority the class of the test data is chosen. We used bootstrap sampling with replacement in order to sample data from the training set each time for creating a classifier. Thus the data used for creating a classifier each time may contain duplicates. A hyperparameter t is chosen and that many classifiers or trees are constructed where each time, random samples are chosen from the training data with replacement for construction of the tree. Thus each tree will be different in some way from the other as each of them have been built using different data. Another important hyperparameter in this case is m which needs to be set at a predefined value which has to be quite low (For example, 20% of the number of features in the data). We have chosen m to be the square root of total no. of features in the data.[10]

During construction of the trees, at each node we choose m random features and find the best split among these m random features. A continuous feature can be chosen multiple times however a categorical feature can be chosen only once along that path starting from root to the current node. Stopping conditions for a tree in random forests are when a node becomes too small (<= 3 records) or when height of the tree exceeds the number of features. After construction of t such trees we run the same test data through each of these 't' trees and gather votes from them. For example, if there are 5 trees and 3 of them classify a sample test point as 1 while 2 of them classify the same point as 0, then we assign the class label 1 to that test data point due to majority. This is the concept of ensemble learning.[11]

➢ *Pros*
- Handles the overfitting problem of decision trees since here averaging of several trees is performed.
- It is one of the most accurate learning algorithms producing a highly accurate classifier.
- It handles large datasets very well since we do not consider all the attributes while creating random trees.
- It can handle several input variables without having to delete them.

➢ *Cons*
- This model is slower because it takes into account several trees to classify the test data.

C. k-Nearest Neighbors (kNN)

The K-nearest neighbor (k-NN) algorithm is a lazy learning algorithm that learns by analogy. That is, it compares a tuple with other training tuples that are most similar to it.Each tuple is a point in an n-dimensional space, where n is the number of attributes in the data set. The training dataset consists of tuples whose classes are known to be correct. To classify a new point, k-NN calculates k closest tuples to it in n-dimensional space using some distance metric. In our implementation, this metric is taken to be the Euclidean distance. Then, the new point is assigned a class label corresponding to the majority class label amongst these k training tuples.[12]

Data is typically normalized before feeding it to k-NN algorithm to prevent attributes with larger ranges from outweighing those with lower ranges . Several forms of normalizations can be performed on the input data. A popular method is the Z-score normalization. If x is the value for the nth attribute, then the scaled attribute value(x') is obtained by:

$$x' = (x - meann) / stdn$$

Where meann and stdn are the mean and standard deviation for the nth attribute respectively. While classifying a test tuple, the same normalization can be performed on it using the values for mean and standard deviation obtained earlier for the training data.[13]

➢ *Pros*
- It is straightforward to implement and it skips the training phase.
- The distance metric is configurable and can be tweaked to get better results.

➢ *Cons*
- Nearest Neighbor classifiers use distance as their defining metric and thus give equal weights to every attribute. Hence, their accuracy can drop significantly when the data set contains noisy or irrelevant attributes.
- It can be computationally expensive for large training sets because of the distance calculation and linear searching
- If k is very small then model is sensitive to noise points and if k is too large then it may include points from other classes.

D. Naive Bayes

Naive Bayes is a statistical classifier which is based on Bayes Theorem. This classifier predicts the class membership probabilities such as the probability that a tuple belongs to a particular class. This algorithm assumes that all the attributes of a tuple are independent of each other and have no correlation whatsoever. This assumption is called class-conditional independence. Let H be the hypothesis that a tuple X belongs to a class label C. For classification, we look for the probability that the tuple X belongs to class C given that we know the attributes of that tuple. This is also called as the posterior probability and is given by P(H|X).[14]

➢ *Terms*
- P(H|X): Probability of hypothesisH holding true that a tupleX belongs to classC given that we know the attributes of X. (Posterior probability)
- P(H) :It is the prior probability of hypothesis H holding true which is independent of the data tuples.
- P(X|H) : It is the probability that there exists a tuple X such that it satisfies the hypothesis H of belonging to class C . (Posterior probability of X conditioned on H)
- P(X):It is the prior probability of existence of tuple X.

This is the formula we are going to use to perform classification:

$$P(H|X)=(P(X|H)*P(H)) / P(X)$$

As explained in the algorithm, we handle categorical columns and numerical columns differently. In case of training data, we have maintained a dictionary that keeps a track of our class labels. If we find that the column is categorical we simply append the value as keyword 'Categorical' to the key of our dictionary which is the class label corresponding to our tuple. In case of test data, we divide the count of occurrence of our categorical value for class label Ci by the total count of Ci present in the dataset.

In case of training data, we have maintained a dictionary that keeps track of our class labels. If we find that the column is numeric, we calculate the mean and standard deviation over all the values in that column and append the calculated mean and standard deviation to the key (corresponding class label) of the dictionary. This calculation is done only once per column. In case of test data, for each tuple, we calculate the probability density for each column of that tuple using the mean and standard deviation values that were already stored in the dictionary for that column.[15]

➢ *Pros*
- Simple and easy to understand and implement.
- If the conditional independence assumption actually holds true, then this classifier converges faster than other classifiers, hence will require less training data.
- Handles continuous as well as discrete data.
- Makes probabilistic predictions.
- Efficient when dealing with large databases.

➢ *Cons*
- This classifier doesn't perform well if the dataset has correlated attributes. Hence called 'Naive'.
- Faces a 'zero-probability' issue if you do not have a particular attribute value present in the training data, it will give a zero probability while calculating the posterior probability on the tuple that has that attribute value. This drawback can be handled using 'Laplacian Correction' however.

III. CONCLUSION

It is an approach where various machine learning algorithms is being studied and will be used for classification of the data and to provide a human-like, exhaustive and informative medical consultation model. Analysing the symptoms and questions asked by patients and detecting the disease of person is conundrum because most of them when they become sick, as a result of lack of medical erudition and experience, will describe their symptoms inaccurately in medical terms.

REFERENCES

[1]. Brian Godsey, Think Like a Data Scientist Manning, ISBN: 9781633430273
[2]. H. Brink, J. Richards, M. Fetherolf, Real World Machine Learning, Manning, ISBN: 9781617291920
[3]. D. Cielen, A. Meysman, M. Ali, Introducing Data Science, Manning ISBN: 9781633430037.
[4]. J. Han, M. Kamber, and J. Pei, Data mining: concepts and techniques. Elsevier, 2011
[5]. T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Second Edition, Springer
[6]. G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning with Applications in R, Springer, ISBN 9781461471370

[7]. Chen Jin, Luo De-lin and mu Fen-xiang An improve ID3 Decision tree algorithm. IEEE 4th International Conference on computer Science & Education
[8]. Quinlan J. R. (1986). Induction of decision trees. Machine Learning, Vol.1-1, pp. 81-106.
[9]. Prof. Nilima Patil and Prof. Rekha Lathi(2012), Comparison of C5.0 & CART Classification algorithms using pruning technique
[10]. Bernard S, Heutte L, Adam S, (2008) Forest-RK : A New Random Forest Induction Method, Proceedings of 4th International Conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications – with Aspects of Artificial Intelligence, Springer-Verlag
[11]. Abdulsalam H, Skillicorn B, and Martin P, (2007): Streaming Random Forests, Proceedings of 11th International Database and Engineering Applications Symposium, Banff, Alta pp 225-232.
[12]. R. Kumar and R. Verma, "Classification algorithms for datamining: A survey," Int. J. Innov. Eng. Technol. IJIET, vol. 1, no. 2,pp. 7–14, 2012.
[13]. M. Shouman, T. Turner, and Stocker, "Applying k-nearest neighbour in diagnosing heart disease patients," Int. J. Inf. Educ. Technol., vol. 2, no. 3, p. 220, Jun. 2012.
[14]. Hetal Doshi and Maruti Zalte, "Performance of Naïve Bayes Classifier-Multinomial model on different categories of documents" National Conference on Emerging Trends in Computer Science and
Information Technology,IJCA,2011.
[15]. Ajay S. Patil and B.V. Pawar, "Automated Classification of Naïve Bayesian Algorithm" ,Proceedings of International Multi-Conference of Engineers and Computer Scientists, March 14-16,2012.