CS 545: Machine Learning, Fall 2022

DETECTION OF LUNG TUBERCULOSIS

Bhanoday Sai Kastala| Harshitha Depuru| Sai Sindhu Muppaneni| Siva Sai Kumar Paritala| Soham Nandi

Dec 9, 2022

Abstract:

Tuberculosis is a type of bacterial disease that attacks the human body, mostly the lungs. According to a recent study, most cases are detected in the regions of South-East Asia and more than 50% of the cases are diagnosed in seven countries where India and China are at the top. According to the WHO report, over 1.2 million people died from tuberculosis. If the disease is detected in the early stages, there is a possibility for a cure through proper medication. So here, we use some of the machine learning algorithms and techniques to detect tuberculosis.

A severe human disease concern, Mycobacterium TB exhibits a complicated evolution of antibiotic resistance (AMR). Mycobacterium, which causes the disease known as tuberculosis, can damage all the organs along via the lungs. HIV-positive people are much more likely than HIV-uninfected people to die from the disease infection. Pulmonary TB diagnosis has never been easy. The K Nearest Neighbors algorithm and a group of classifiers' performance on data pertaining to tuberculosis are compared in this article using a machine learning technique. To determine which classifier has the highest prediction accuracy, the k-fold Cross-Validation method was used to analyze the classifiers' predictions. The outcomes show that the ensemble Boosting Algorithm is effective, and performs well when compared to weighted KNN, local KNN, and basic KNN.

1. Introduction

In the biomedical field, data growth is currently enormous. We can perform fruitful research on these types of fatal diseases using the information that has been gathered. The quick evolution of data mining research has resulted in the creation of effective strategies to extract knowledge from these data. Medical data mining is an efficient method for collecting medical data, such as patient and clinical data. Applications for mining medical data are effective tools that can be helpful in identifying the factors involved in this life-threatening illness.

One of the topics that have received the most attention in statistics, decision science, and computer science is the classification of data using information derived from known historical data. Large amounts of data are currently being analyzed by ML applications in healthcare to help doctors and other medical professionals make better judgments. This technology can help medical professionals spot abnormalities, patterns, and trends while also reducing human error. Researchers may now analyze and uncover hidden information inside data that

can improve forecast results using techniques from the fields of artificial intelligence/machine learning, statistics, and other visualization tools. Therefore, improvements in ML might aid in closing the gap between the diagnosis of tuberculosis and access to healthcare in TB-endemic nations. However, the lack of familiarity and accessibility with these potent instruments in the medical community poses a difficulty to their application in diagnostic medicine.

To acquire more accurate findings and to get the data into the same size (600*600) for this data set, we used preprocessing techniques such as trimming, inverting, and compressing methods before applying the algorithms. The number of total images in the dataset is 155, among those 50% is for patients who are affected by TB, other 50 % is for non-TB patients, variable image size is approximately around 60 KB moreover the dataset consists of two major columns i.e., Study ID and TB findings.

2. Work Distribution

2.1. Pre-processing or scaling the data:

The dataset is downloaded from the resource website called SourceForge. Import the necessary libraries to work on image-related data. Some examples could be NumPy, image, pyplot libraries. ImagesChops library is used for processing and manipulation of image data. In the first step, this library is used to add the background for the images and the final images are stored in a different folder of the dataset. In the second step, the images need to resize to a manageable size (1024x1024) resolution as the size of the original images is quite large. Another reason for the compression of images is to avoid the matrix size to be large. Images are further compressed by resizing all the images to a fixed window size. Using the same library images are also inverted.

2.2. **Applied local k-nearest neighbor algorithm**

After pre-processing the data, the final scaled images are divided into train and test datasets. KNN algorithm is applied to the training dataset. Once the algorithm is computed, results are printed out for accuracy, precision, reclass, and f1 score.

2.3. Applied distance weight K-nearest neighbor (KNN) algorithm

Applied weighted KNN is applied to the initial scaled training data. Performance results are posted once the algorithm is computed. The result values are shown for accuracy, precision, reclass, and f1 score.

2.4. Applied Gradient Boosting algorithm

The initial dataset is used to apply the gradient boosting algorithm. We have initialized the values for the learning rate in between the range of 0.05 to 0.5. Accuracy is calculated based on the output. Other performance results are also derived such as precision, reclass, and f1 score. To determine which method has the highest accuracy, the outputs

of all three are compared. The Gradient Boosting method has the highest accuracy of all, according to the derived output.

3. Methods

3.1. K-nearest algorithm

The K-nearest Neighbors algorithm mostly known as the KNN algorithm uses the nearest training feature space as a classification approach for objects. KNN is a form of instance-based learning or lazy learning where all computations are postponed until classification and only local approximations of the function are made. Objects are classed here by a majority vote of their neighbors, with the object being given to the class most prevalent among its k closest neighbors. The value of k is always positive and typically small. K value could differ based on the dataset, so the initial assumption could be always tricky. Based on the output, the value of k could be initialized again and again to find the optimal value.

3.2. Local mean based KNN

Local mean-based KNN is a variant of the k-nearest neighbors (KNN) algorithm that uses local means to make predictions. In this variant, the mean of the k nearest neighbors is calculated for each data point, and this means is used as the prediction for that point. This can be useful when working with datasets that have many outliers, as it can help to smooth out the predictions and reduce the influence of the outliers. LMKNN algorithmic approach is straightforward, efficient, and robust. Especially for smaller datasets, it has been shown that the algorithm improves classification performance and minimizes the impact of current outliers.

To compute the LMKNN, we need to initialize the value of k. From the training and testing data output compute the Euclidean distance using the equation.

$$(x, y) = \|x - y\| = \sqrt{\sum N * |x - y|^2}$$

Based on the output values of the equation, the data points are sorted based on the distance, starting from the smallest, and it assigns as many values as possible for k for each data class. The local mean vector of each class can be derived from the equation.

$$mean(k) = i \sum kyN^2$$

Using the equation, determine which data class's distance from the local mean vector is closest to define the test data class.

$$w = argmin\ (x, mkcwjwj) = 1, 2, 3, \dots\dots\dots, M$$

3.3. Distance Weighted KNN

A variation of the KNN algorithm known as distance-weighted k-nearest neighbors (DWN-KNN) uses distances to weigh the contributions made by each of the k-nearest neighbors to the final prediction. Based on the weight value derived from the distance between two points, the DWNKNN method defines a new data point. Further, we can avoid misclassifying data by paying attention to the proximity of the data. When working with datasets that contain a lot of outliers, this can be helpful because it can help to smooth out the predictions and lessen the impact of the outliers.

For computing the DWNKNN algorithm, so initially we need to determine the value of k. From the training and testing data output compute the Euclidean distance using the equation. Based on the output values of the equation, the data points are sorted based on the distance, starting from the smallest, and it assigns as many values as possible for k for each data class. Calculate the new weights for the sorted data based on the equation.

$$wi = (4)(xq, xi)$$

3.4. Gradient Boosting

A supervised machine learning algorithm called gradient boosting can be applied to both classification and regression problems. As an ensemble method, it combines the results of various weaker models to produce a more reliable and accurate result.

Iteratively training weak models, like decision trees, and adding them to the ensemble is how the gradient-boosting algorithm functions. Every new model is trained to fix the mistakes made by the ensemble's earlier models. Until a predetermined number of models have been trained and added to the ensemble, or until the ensemble's performance on the training data stops improving, this process is repeated.

The fact that gradient boosting can handle a variety of data types, including numerical, categorical, and ordinal data, is one of its main benefits. It is a popular option for working with large and complex datasets because it is also resistant to overfitting. Gradient boosting involves three elements: A loss function, a weak learner to make predictions, and a model that is additive in nature.

Loss Function: In gradient boosting, a loss function is used to measure the error between the predicted value and the true value for each data point. The loss function is minimized during the training process, which allows the gradient-boosting algorithm to produce

accurate predictions. For classification problems, the logarithmic loss (also known as the cross-entropy loss or log loss) is often used as the loss function.

Weak Learner: In gradient boosting, a weak learner is a simple model that is trained on a subset of the data and used to make predictions. The predictions made by the weak learner are then combined with the predictions of other weak learners to produce a more accurate and robust prediction. Decision trees are commonly used as weak learners in gradient boosting. This is because decision trees are simple to train and can handle a wide range of data types, including numerical, categorical, and ordinal data.

Additive Model: An additive model in gradient boosting is one that is created by gradually adding weak learners to the ensemble. Each new weak learner in the ensemble is taught how to fix the mistakes made by the earlier weak learners. This enables the ensemble's overall performance to increase with the addition of each new weak learner. The specific algorithm used to train the weak learners in an additive model will depend on the type of problem being solved and the characteristics of the data. The algorithm can learn complex and non-linear relationships in the data, which is the main benefit of using an additive model in gradient boosting. The algorithm can learn a more adaptable and precise model that can capture the underlying structure of the data by gradually adding weak learners to the ensemble.

4. Results

4.1. Local KNN

precision recall f1-score support
0 0.63 0.85 0.72 26
1 0.76 0.50 0.60 26
accuracy 0.67 52
macro avg 0.70 0.67 0.66 52
weightedavg 0.70. 0.67 0.66 52

4.2. Weighted KNN

precision recall f1-score support
0 0.66 0.96 0.78 26
1 0.93 0.50 0.65 26
accuracy 0.69 52
macro avg 0.79 0.73 0.72 52
weighted avg 0.79 0.73 0.72 52

4.3. Gradient Boosting

Learning rate: 0.05 - Accuracy score (validation): 0.668

Learning rate: 0.075 - Accuracy score (validation): 0.637
Learning rate: 0.1 - Accuracy score (validation): 0.594
Learning rate: 0.25 - Accuracy score (validation): 0.693
Learning rate: 0.5 - Accuracy score (validation): 0.652
Learning rate: 0.75 - Accuracy score (validation): 0.738
Learning rate: 1 - Accuracy score (validation): 0. 681

Conclusion

The results of this work enable us to draw the conclusion to categorize tuberculosis using both simple and ensemble classifiers by sensitive ML algorithms. Intelligent techniques, such as Artificial Neural Networks (ANN), have recently a lot of time has been spent on classification activities. Given that it is linked to other illnesses, tuberculosis is a serious health issue. Retrospective studies on tuberculosis indicate that HIV infection progresses more quickly when active tuberculosis is present. There are still difficulties in locating, treating, and controlling the disease in a number of communities, and TB is still the world's greatest cause of death. The effectiveness of the strategy was evaluated by comprehensive experiments utilizing real datasets and the recommended algorithms.

References

1. Panicker, R.O., Soman, B., Saini, G. and Rajan, J., 2016. A review of automatic methods based on image processing techniques for tuberculosis detection from microscopic sputum smear images. *Journal of medical systems*, *40*(1), pp.1-13.
2. Poornimadevi, C.S. and Sulochana, H., 2016, March. Automatic detection of pulmonary tuberculosis using image processing techniques. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)* (pp. 798-802). IEEE.
3. Leibstein, J.M. and Nel, A.L., 2006. Detecting tuberculosis in chest radiographs using image processing techniques. *University of Johannesburg*.
4. Hrizi, O., Gasmi, K., Ben Ltaifa, I., Alshammari, H., Karamti, H., Krichen, M., Ben Ammar, L. and Mahmood, M.A., 2022. Tuberculosis Disease Diagnosis Based on an Optimized Machine Learning Model. *Journal of Healthcare Engineering*, *2022*.