**Classification of Microorganisms Using Various Machine Learning Methods**

Gaurav Kumar

**Abstract**

The mortality rate of some bacterial infections increases as treatment is delayed. Therefore, quick identification of a particular gene or genetic species decreases the time it takes to develop an antibiotic for bacterial infection. Multiple machine learning methods have been applied to the classification problem of ten different bacterias in order to compare their performance. K-nearest neighbor, multilayer perceptron, and gradient boosting were applied to the dataset. Due to the large number of duplicate instances in the selected dataset, the performance of these machine learning methods were compared using the entire dataset and then a subset of only unique instances. Both the accuracy and runtime were looked at when comparing the different methods and datasets.

**Introduction**

Machine learning is a branch of artificial intelligence that helps the computer make predictions by learning from training data and previous experiences without any explicit programming. It has been applied in many fields of research and work, including biomedical and genetics. DNA sequencing has become a major part in both biomedical and genetic research. By identifying the DNA sequence of an infectious bacteria, an antibiotic can be developed or synthesized for the identified bacteria. Recently, machine learning models have been applied to help quickly identify a particular gene or genetic species in order to decrease the time it takes to develop an antibiotic and increase treatment and diagnostic rates. In this work, K-nearest neighbor, multilayer perceptron, and gradient boosting are applied to the classification problem of ten different bacterias within the dataset. The dataset used is the microbes dataset provided by Kaggle [1] which has been adapted from the dataset provided by Dhindsa et al [2]. It contains twenty five attributes and 30,527 instances, most of which are duplicates. The listed machine learning methods are applied to both the entire dataset and a subset containing only unique instances and their accuracy and runtimes are compared.

**Motivation and Related Work**

The problem of classifying bacteria based on certain attributes was chosen due to its relation with the medical and pharmaceutical industries. As previously stated, it is important to be able to quickly classify certain bacteria via their DNA or a segment of their DNA so that the infected individual can be treated. The longer it takes to classify the bacteria and synthesize an antibiotic, the higher the risk of the patient. It is for that reason multiple machine learning methods are to be applied to this classification set inorder to identify which model performs the fastest with relatively good accuracy.

A considerable amount of research has been devoted to machine learning methods and their applications in the field of genetic sequencing. Dhinda et al. performed exhaustive hyperparameter tuning of multilayer perceptron, k-nearest neighbors, quadratic discriminant analysis, logistic regression, and support vector machine to accurately classify the original dataset used in this study [3]. Chen and Boggess looked at a variety of neural network models when using genome signatures for gene classification [4]. Wang proposed a new ensemble method consisting of a sequential self-optimization structure and data-bootstrapping technique for classification and prediction on gene expression data. Wang applied his model to twelve different gene expression datasets, compared the accuracy to existing ensembles, and showed that the sequential self-optimization structure greatly improved the ensembles accuracy [5].

**Problem Description**

The goal of this work is to investigate the accuracy and efficiency of multiple machine learning techniques applied to a classification problem. The techniques applied in this present work consist of K-nearest neighbors, multilinear perceptrons, and gradient boosting which will be used to classify different microbes from the microbe dataset. Since the data set contains many duplicate instances, the machine learning techniques will also be applied to both the full dataset and a subset containing only unique instances in order to see how the duplicates affect the accuracy and efficiency. The accuracy and efficiency of the models is used for evaluation since both would be important as described in the motivation.

**Dataset**

The microbes dataset from Kaggle is used for the classification problem. The dataset contains twenty five continuous attributes and ten target values. The attributes are described in Table A1 in Appendix A. There are 30,527 instances which have about 84% duplicate instances. Removing these instances leaves 4,874 unique instances. Looking at the target values, the majority of instances for the entire dataset are classified as either Ulorthrix, Volvox, or Aspergillus sp, while the majority of instances for the subset are classified as either Ulorthrix or Aspergillus sp. The distributions of the target values for both the entire dataset and the subset are shown in Figure 1a and Figure 1b respectively. The correlation matrix between the attributes are also looked at for both datasets and is shown in Figure 2a and Figure 2b for both the entire dataset and the subset respectively. Looking at the correlation matrices, a few attributes are shown to be highly correlated and there is not much difference between the attribute correlations in the full dataset compared to the subset. Due to limited time, the effect of this correlation will not be investigated.

|  |  |
| --- | --- |
|  |  |
| 1. Full dataset | 1. Subset containing unique instances |
| Figure 1. Distribution of target values within the dataset. | |

|  |  |
| --- | --- |
|  |  |
| 1. Full dataset | 1. Subset containing unique instances |
| Figure 2. Correlation matrix for both datasets. | |

**Experimentation**

Before building and running the models, the dataset needed to be processed. The target value labels were encoded to values between zero and nine using a label encoding technique. The original data was imbalanced as shown in Figure 1, so SMOTE was used to balance the data in the subset dataset. SMOTE (Synthetic Minority Oversampling Technique) works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors. The dataset was then splitted into training and testing proportions with a ratio of 0.8 and 0.2 using a stratify parameter, where training and test subsets have the same proportions of class labels as the input dataset. Lastly, each set of training and testing were scaled, each by the same scaler. The processed datasets, the full dataset and the subset containing only unique instances, were used in a K-nearest neighbor, multilayer perceptron, and gradient boosting model.

K-nearest neighbors is a type of supervised learning algorithm used for both regression and classification by using the “k” nearest neighbors when making a prediction. The K-nearest neighbor model that was built for the full dataset and for the subset was the scikit-learn KNeighbors classifier. Two parameters, the number of neighbors and the weight function used, were tested to determine which combination gave better results with respect to accuracy and time. The number of neighbors used ranged from three to thirty three in increments of three. Each number of neighbors was run ten times for both models in order to get an accurate timing. Once an appropriate number of neighbors was determined, that number was used to compare the weight functions. The two weight functions were “uniform,” same weight for every instance, and “distance,” weights were adjusted by the inverse of their distance to an instance.

Multilayer perceptron is a feedforward neural network. It is composed of at least three layers including an input layer to receive the signal, an output layer that decides or predicts about the input, and an arbitrary number of hidden layers in between that are the true computational engine of the multilayer perceptron. A multilayer perceptron model was built for both the full dataset and the subset using scikit-learn MLPClassifier. For this method, the number of hidden layers was tested to determine an appropriate model, ranging from ten to fifty in increments of ten. Each model was run five times to measure the runtime of the model given the number of hidden layers on both datasets. The maximum number of iterations was set to one thousand.

Gradient boosting is an ensemble method that applies typical boosting to the training set, builds decision trees as the base learners, and optimizes a loss function. It is one of the most well-known and powerful algorithms. The gradient boosting algorithm used in this work was the scikit-learn GradientBoostingClassifier. For this machine learning technique, the parameter for the number of estimators, the number of boosting stages performed, was tested to determine an appropriate model. The models were run one time for each number of estimators due to limited time. A learning rate of 0.1 was used, and the max depth of the base learners was set to three.

**Results** **and Discussion**

K-nearest neighbor, multilayer perceptron, and gradient boosting methods were applied to the microbes dataset. The results from the K-nearest neighbor models are given in Table 1. The analysis of the K-nearest neighbor method indicates that the highest accuracy for both the full and subset dataset was obtained using three neighbors. This was expected since adding more neighbors may result in the prediction being shifted further away from its true value, depending upon the weight function used and is seen by the lower accuracy for the higher values of n. However, using fewer neighbors does leave the prediction more susceptible to noise. The subset dataset was about fifteen times faster compared to the full dataset. This may be because there were much fewer data points to compare to identify the n nearest neighbors. After identifying the optimal number of neighbors, the subset dataset was evaluated using different weight functions, and it found that the distance weight function produces the highest accuracy of 90.3% compared to 87.2% when using uniform weights.

|  |
| --- |
| Table 1. Accuracy and runtime (seconds) of K-nearest neighbor models on the full dataset and subset dataset using n neighbors with uniform weights. |
|  |

The results from the multilayer perceptron models are given in Table 2. The analysis of the multilayer perceptron method indicates that higher accuracy for both the full and subset dataset is obtained using more hidden layers. By adding more hidden layers to the model, more nonlinearity can be introduced to better capture the boundaries between the classes, but the risk of overfitting increases. Like in the K-nearest neighbor method, the subset dataset was faster than the full dataset but had a lower accuracy. A multilayer perceptron model was run using five hundred hidden layers on the full dataset and achieved an accuracy of 97.8% in about four minutes and twenty six seconds.

|  |
| --- |
| Table 2. Accuracy and runtime (seconds) of multilayer perceptron models on the full dataset and subset dataset using L hidden layers and one thousand iterations. |
|  |

There was a mistake in the code for the results from the gradient boosting models so table for this method is not given. The Gradient Boosting models were evaluated on the full dataset using different numbers of estimators ranging from 250 to 1000 in increments of 250, all with a learning rate of 1, which was meant to be 0.1, and base learners of depth 3. Along with the high learning rate, the line for stopping the timer was missing from the code. Due to these mistakes, the gradient boosting method was run using one thousand estimators, a learning rate of 0.1, and a depth of 3 for the base learners on both datasets using xgboost. For the full dataset, an accuracy of 98.5% in about 200 seconds. The subset dataset reached an accuracy of 85.4% in about 316 seconds. Surprisingly, the gradient boosting method ran faster for the full dataset compared to the subset dataset.

**Summary**

The project analyzed the performance of three machine learning methods, K-nearest neighbor, multilayer perceptron, and gradient boosting, on the microbes dataset from Kaggle. The K-nearest neighbor method was found to have the highest accuracy for the shortest runtime. For K-nearest neighbors, using more neighbors decreased the accuracy, but using fewer leaves the model susceptible to noise. The faster runtime for the K-nearest neighbor model may be credited to not having to train the model. Each model was timed from start to finish, including building, training, and testing. When comparing the full dataset to the subset dataset, the subset dataset was much faster for K-nearest neighbor and multilayer perceptron but not for gradient boosting. The multilayer perceptron method analysis showed that a higher accuracy for both datasets can be achieved by increasing the number of hidden layers, although overfitting risk also increases. Removing duplicate values from the dataset was found to decrease runtime, and the accuracy of the subset dataset was lower than the full dataset. The higher accuracy in the full dataset may be attributed to overfitting. The gradient boosting method showed high accuracy using one thousand estimators. To address the need for an accurate but fast model for gene identification a K-nearest neighbor model with three neighbors would be suggested, as well as using the subset dataset over the full dataset.

**References**

[1] Dhindsa, Anaahat, et al. (2020), “Dataset for Efficient Microbes Classification System ”, Mendeley Data, V3, doi: 10.17632/f9m85ptmvc.3

[2] Sayan Saha, “Microbes Dataset.” Kaggle, 2020. https://www.kaggle.com/datasets/ sayansh001/microbes-dataset

[3] Dhindsa, Anaahat, et al. “An Improvised Machine Learning Model Based on Mutual Information Feature Selection Approach for Microbes Classification.” *Entropy*, vol. 23, no. 2, Feb. 2021, p. 257. *Crossref*, https://doi.org/10.3390/e23020257.[4] C. W. Wang, "New Ensemble Machine Learning Method for Classification and Prediction on Gene Expression Data," *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, New York, NY, USA, 2006, pp. 3478-3481, doi: 10.1109/IEMBS.2006.259893.

[5] Liangyou Chen and L. Boggess, "Neural networks for genome signature analysis," *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02.*, Singapore, 2002, pp. 1554-1558 vol.3, doi: 10.1109/ICONIP.2002.1202882.

**Appendix A: Dataset Attribute Description**

Table A1. Microbes dataset attributes and their descriptions.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| Solidity | float | Ratio of area of the object to the area of its convex hull |
| Eccentricity | float | Ratio of length of major to minor axis |
| EquivDiameter | float | Diameter of a circle with the same area as the object |
| Extrema | float | Extrema points in the region |
| Filled Area | float | Number of on pixels in an image |
| Extent | float | Ratio of the pixel area of a region with respect to bounding box area of the object |
| Orientation | float | Overall direction of the shape from [-90, 90] |
| EulerNumber | float | Number of objects in the region minus the number of holes in those objects |
| BoundingBox1 | float | Position and size of the smallest box which bounds the object |
| BoundingBox2 | float | Position and size of the smallest box which bounds the object |
| BoundingBox3 | float | Position and size of the smallest box which bounds the object |
| BoundingBox4 | float | Position and size of the smallest box which bounds the object |
| ConvexHull1 | float | Smallest convex shape/polygon that contains the object |
| ConvexHull2 | float | No Description Provided |
| ConvexHull3 | float | No Description Provided |
| ConvexHull4 | float | No Description Provided |
| MajorAxisLength | float | The longest line that can be drawn through the object. Length (in pixels) of the major axis is the largest dimension of the object |
| MinorAxisLength | float | The major axis is the endpoints of the longest line that can be drawn through the object. Length (in pixels) of the major axis is the largest dimension of the object |
| Perimeter | float | Number of pixels around the border of the region. |

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Description** |
| ConvexArea | float | No Description Provided |
| Centroid1 | float | Centre of mass of the region. It is a measure of the object’s location in the image. |
| Centroid2 | float | No Description Provided |
| Area | float | Total number of pixels in a region/shape. |
| Radi | float | No Description Provided |