ENGR 518 PROJECT REPORT

School of Engineering
Faculty of Applied Science
University of British Columbia

Project Title: COVID-19 Patient Intubation Prediction
Group No.: 18
Members: Sahar Sowdagar, Armita Tehranchi, Mohammadamin Lari
Date: November 28, 2023

Introduction

The COVID-19 pandemic has caused significant mortality rates, showcasing a wide array of symptoms that range from mild coughs to severe acute respiratory distress syndrome (ARDS), often leading to respiratory failure. Intubation, a medical procedure involving the insertion of a tube into the trachea for maintaining an open airway, is crucial in cases of severe respiratory distress such as ARDS. However, the pandemic's unprecedented scale uncovered critical vulnerabilities in healthcare systems, particularly revealing shortages in intubation machines and ICU beds, highlighting challenges in the medical supply chain. The 'COVID-19 Patient Intubation Prediction' project aims to develop a machine learning model that accurately predicts the need for endotracheal intubation in COVID-19 patients based on pertinent clinical data. This predictive model is crucial in optimizing resource allocation, enhancing patient care, and fortifying healthcare systems, especially during times of scarcity, and extends beyond the current crisis by laying the foundation for data-driven healthcare management.

The project's main objective is to timely and accurately identify patients requiring intubation, offering healthcare practitioners valuable insights for better resource management. By enabling healthcare professionals to anticipate patient needs and effectively manage hospitals, this initiative seeks to optimize resource utilization, improve patient outcomes, and strengthen healthcare systems. The success of this project not only addresses the pressing needs of the COVID-19 pandemic but also establishes a framework for future data-driven models, aiding healthcare professionals in addressing patient demands and managing hospitals more efficiently in various healthcare scenarios.

Theory

For this project, a COVID-19 dataset was acquired from Kaggle website. This dataset includes recorded entries for 1048573 patients. The dataset is primarily binary, with ‘1’ meaning ‘yes’ and ‘2’ meaning ‘no’ and includes 21 features. However, certain attributes such as ‘AGE’ and ‘CLASSIFICATION_FINAL’ (which describes COVID-19 severity) have non-binary values. Due to lack of space, detailed description of dataset features is not included in this report. However, one can refer to Kaggle [5] for further information.

For the purpose of this project, the ‘INTUBED’ attribute is selected as the output label. The goal of this project is developing a binary classification algorithm in order to predict whether a patient requires intubation or not based on their characteristics and medical history. Classification is a supervised machine learning process where the goal is to predict an output using labeled inputs. The full process involves data pre-processing, training the model and tuning the hyperparameters, and finally, testing the model on unseen data. Several different algorithms can be selected for binary classification, including logistic regression, decision trees, random forest, support vector machines, k-nearest neighbors, neural networks, and more. Each algorithm works in a unique way, and the ones employed in this project are described further below.

Logistic Regression models the probability of an instance belonging to the positive class using the logistic sigmoid function. It applies a linear combination of input features, transformed into

probabilities through the logistic function. Here, Gradient Descent algorithm is utilized to optimize parameters and adjust the model to best fit the observed data.

K-Nearest Neighbors classifies data points based on their nearest neighbors in the feature space. The algorithm calculates distances between samples and assigns a class label based on the most common class among the k nearest neighbors. The choice of parameter 'k' influences the model's sensitivity to local variations.

Artificial Neural Networks (ANNs) consist of interconnected neurons organized in layers, including an input layer, hidden layers, and an output layer. Neurons apply weights to input features, pass the result through an activation function, and send the output to the next layer. During training, the network adjusts weights through backpropagation, minimizing the difference between predicted and actual outcomes. ANNs can achieve very high accuracies, but figuring out the optimal neural network can be a challenge. In addition, it can be very computationally expensive.

The dataset consists of over a million entries with 18 interconnected inputs, as revealed by the correlation matrix in the project's code. To handle this high-dimensional data and interconnected features, Principal Component Analysis (PCA) is proposed as a viable technique for dimensionality reduction. PCA involves several steps: data centering, computing the covariance matrix, and eigen-decomposition to determine eigenvectors and eigenvalues. These eigenvectors represent the most critical variations in the data. By projecting onto these components, PCA creates a lower-dimensional dataset while preserving essential information. In this project, PCA can effectively reduce the inputs from 18 to 2, enabling data visualization through a feature plot.

Algorithm

A. Data Pre-Processing

Raw data often requires pre-processing before feeding it to machine learning models for several reasons, including handling inconsistencies in magnitude and missing values, encoding categorical data, and feature engineering. Here are our pre-processing steps:

1) Feature Selection: Removing irrelevant features to our project objective from the dataset
2) Handle Missing Values: Replacing missing values randomly from set {1,2}
3) Feature “Classification_Final” Binarization: Remove multiple representations for presence or absence of COVID-19 in patients
4) Feature “Age” Mapping: Map age of patients to [1,2] interval to be consistent in magnitude with other features
5) Normalization: Scale data to have a distribution with mean 0 and standard deviation 1

B. Dataset Partitioning

Performing dataset partitioning in machine learning is essential for effectively building, fine-tuning, and evaluating models. This approach divides the dataset into three subsets: the Training set, used for model parameter learning, the Validation set, employed for hyper-parameter tuning and model selection, and the Test set, used to assess the model's performance on completely unseen data. Dataset partitioning helps prevent overfitting by iteratively adjusting model hyper-parameters on the validation set, ensuring that the final model selected performs well on new, unseen data in the test set. In this project, we have randomly partitioned our dataset using the built-in *train-test split* function of *sklearn* package according to table 1.

Table 1 Data Partitioning

Partition	Train	Validation	Test
Percentage	70%	10%	20%
# of data points	734,001	104,857	209,715

C. Model Training

Now that we have the data prepared, we can feed it to a machine learning model. We have developed three binary classification algorithms from scratch. Due to lack of space, we do not provide detailed design procedures in this section, but one can refer to our submitted code that includes comments on steps. The first model is logistic regression. Here, we have used the following regularized cost functions:

1) MSE: $g(w) = \frac{1}{P}\sum_{p=1}^{P}(\sigma(\bar{x}_p^T w) - y_p)^2 + \lambda\|w'\|_2^2$
2) Binary Cross-Entropy: $g(w) = -\frac{1}{P}\sum_{p=1}^{P} y_p \log\left(\sigma\left(\bar{x}_p^T w\right)\right) + (1 - y_p)\log\left(1 - \sigma\left(\bar{x}_p^T w\right)\right) + \lambda\|w'\|_2^2$

Using regularized cost functions is effective in avoiding overfitting and better generalization properties. The models are trained using gradient descent according to algorithm 1.

Algorithm 1: Binary Classification using Logistic Regression

Input: $\{(x^{(i)}, y^{(i)})\}$,α, *num_iterations*
Output: Learned parameters w
Initialize parameters w randomly;
for *iter* $\leftarrow 1$ **to** *num_iterations* **do**
  Compute the sigmoid function: $\sigma(\bar{x}^T w) = \frac{1}{1+e^{-\bar{x}^T w}}$;
  Compute the cost function $g(w)$;
  Compute the gradient $\nabla g(\mathbf{w}^k)$;
  Update parameters: $w^{(k+1)} \leftarrow w^k - \frac{\alpha \nabla g(w^k)}{\|\nabla g(w^k)\|_2}$;

Figure 1 Logistic Regression algorithm.

The second method that we have developed for binary classification is a KNN model. Algorithm 2 demonstrates how this model classifies data points based on their k nearest neighbors:

Algorithm 2 Binary Classification using K-Nearest Neighbors

procedure KNN(X_{train}, x_{test}, k)
  for $i \leftarrow 1$ to $|X_{train}|$ **do**
    Calculate the distance between x_{test} and X_{train}
    Assign the distance to d_i
  end for
  Sort the distances d_i in ascending order
  Select the first k elements from the sorted distances
  Identify the majority class among the selected neighbors
  return Majority class
end procedure

Figure 2 K-Nearest Neighbors algorithm

Using the above algorithms, we managed to accurately perform our classification task. However, we went further and implemented an artificial neural network from scratch as well and trained it with back-propagation method using algorithm 3.

Algorithm 3 Binary Classification using Artificial Neural Network

procedure ANN($X, y, maxIter$)
  Initialize weights and biases randomly
  for *epoch* $\leftarrow 1$ to *maxIter* **do**
    for $i \leftarrow 1$ to *length*(X) **do**
      Forward pass: Compute the predicted output $\hat{y}$
      Backward pass: Compute the loss using the predicted output $\hat{y}$ and the true label
      Update weights and biases using back-propagation
    end for
  end for
  return Trained model
end procedure
procedure PREDICT(*testSample*)
  Perform forward pass through the trained network to obtain the predicted output $\hat{y}$
  return Predicted class
end procedure

Figure 3 Artificial Neural Network algorithm.

So far, we managed to perform our binary classification task on the original dataset with high accuracy. However, this dataset includes 18 input features and is not suitable for visualization purposes. Therefore, we attempted to reduce the dimensionality of our dataset using PCA algorithm that we developed from scratch. Algorithm 4 shows how we implemented PCA.

Algorithm 4 Principal Component Analysis (PCA)

procedure PCA(*NumberofComponents*)
    Center the data matrix X by subtracting the mean of each feature
    Calculate the covariance matrix $C = \frac{1}{m} X^T X$, where m is the number of samples
    Compute the eigenvalues and eigenvectors of C
    Select the top k eigenvectors corresponding to the largest eigenvalues
    Form a new matrix W with the selected eigenvectors as columns
    Project the centered data onto the subspace spanned by the columns of W to obtain the reduced-dimensional representation
    return Reduced-dimensional representation
end procedure

Figure 4 PCA algorithm.

Again, due to lack of space, the reader can refer to our submitted code for more detailed design procedures and model training, validation, and test.

Results and Discussion

In this section, we will demonstrate the effectiveness of our developed algorithms through numerical comparisons between built-in Python packages and the codes we have developed from scratch. Python offers well-crafted libraries tailored for data processing tasks such as partitioning datasets into test and train sets, dimension reduction, and creating machine learning models.

Original Dataset

First, we discuss the results of using the original dataset that consists of 18 features. We have developed a variety of algorithms for our binary classification task from scratch, including Logistic Regression, K-Nearest Neighbors, and a Fully Connected Neural Network.

A. Logistic Regression

We started with using the built-in *Logistic Regression* model from SKLearn library in order to have a reference for evaluating the performance of our algorithms. Using this model led to an accuracy of 97.004% on test dataset and the confusion matrix below:

$$confusion\ matrix = \begin{bmatrix} 173281 & 1370 \\ 4912 & 30152 \end{bmatrix}$$

In the above matrix, element $c_{ij}, i, j \in \{0,1\}$ represents the number of test samples that belong to class j, and the model has classified them as class i, i.e., columns are actual labels and rows are predicted labels. Therefore, the logistic regression algorithm is able to perform classification task on this dataset with high accuracy.

Mean Squared Error (MSE) cost

We developed a logistic regression model with MSE cost function and trained it with the gradient descent algorithm. In the first attempt, we used the whole dataset, but the training time was too large. Therefore, we chose a subset of our dataset containing 10,000 samples and fed it to our model. The model was trained for 150 epochs with learning rate $\alpha = 1$, regularization coefficient $\lambda = 0.01$, and random initialization of weights. The accuracy of this model is 92% with the confusion matrix below:

$$confusion\ matrix = \begin{bmatrix} 1409 & 62 \\ 98 & 431 \end{bmatrix}$$

A 92 percent accuracy is acceptable considering that we have approximately used only 1% of our dataset, but it is still lower than the built-in Python model. Due to high computational complexity, increasing the size of dataset is not possible. Therefore, we changed the cost function.

Binary Cross-Entropy cost

Similar to previous part, we developed a logistic regression model and trained it with gradient descent algorithm, but this time used the Binary Cross-Entropy cost function and the design parameters $\alpha = 0.3\,, \lambda = 0.01, max\_iter = 50$. Furthermore, we used the whole dataset. The accuracy of this model is 96.85% with the confusion matrix below:

$$confusion\ matrix = \begin{bmatrix} 170077 & 0 \\ 6601 & 33037 \end{bmatrix}$$

This accuracy is comparable to the built-in Python model and demonstrates the effectiveness of our approach to perform a binary classification task on COVID-19 dataset. The confusion matrix also implies that our model has captured the characteristics both of the classes effectively.

B. K-Nearest Neighbors (KNN)

Following the same approach as logistic regression, we start with using the built-in KNN function from SKLearn library of Python. Due to high computational complexity of this algorithm, we limit the size of dataset to 20,000 samples. The accuracy of this model with parameter $k = 3$ was 96.08% which is close to the logistic regression, and implies that KNN algorithm can also be used for our purpose. In the next step, we implemented the KNN algorithm from scratch using 5,000 samples, and varied parameter k from 2 to 17 to study the effect of number of neighbors on accuracy of the model and choose the best value for this parameter. Figure 5 shows the results of this experiment. As we can see in figure 5, increasing the number of nearest neighbors is not very effective in increasing the accuracy of our model, for instance, with $k = 3$ the accuracy is 88.2% and with $k = 17$, it becomes 89.9%. Therefore, we choose the parameter $k = 3$ to reduce the computational burden of this algorithm. In the next experiment, we varied the size of dataset from 3,000 to 20,000 samples with parameter $k = 3$ being fixed to study the effect of size of dataset on accuracy of our model. Figure 6 shows the results of this experiment. As we can see from the trend of figure 6, the accuracy of KNN algorithm increases as size of dataset grows. Using 20,000 samples, the accuracy of model is 95.98% which is close to the built-in model of Python.

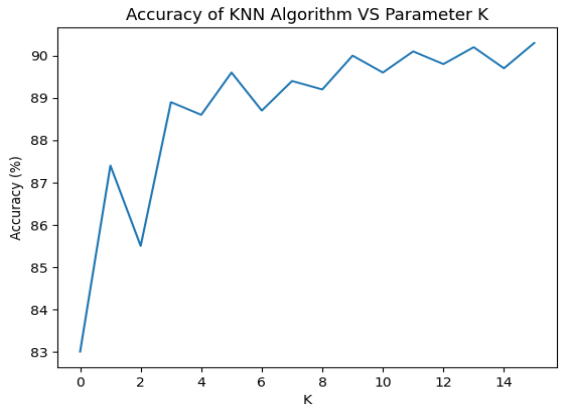

Figure 5 Accuracy of KNN algorithm vs parameter K.

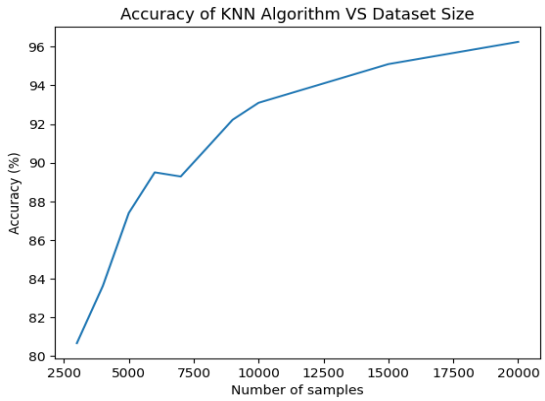

Figure 6 Accuracy of KNN algorithm vs dataset size.

C. Fully Connected Neural Network (FCNN)

Although previous approaches proved to be effective in distinguishing between the two classes of COVID-19 dataset, it was still of our interest to utilize a more complex and non-linear classifier for this task. Therefore, we used a fully connected neural network with tangent hyperbolic activation

function for our binary classification task. Here we also limited the size of dataset to 20,000 samples to reduce the computational burden of the algorithm. We tested multiple neural network architectures with different number of hidden layers and different number of neurons per layer, but the accuracy was not improving noticeably. Therefore, we decided to simplify the structure of the network as much as possible and hence, we used a single-hidden-layer neural network with 10 neurons. The accuracy of this model using the built-in *MLPClassifier* function of Python was 96.63% with the confusion matrix below:

$$confusion\ matrix = \begin{bmatrix} 3204 & 37 \\ 98 & 661 \end{bmatrix}$$

Although the dataset is not balanced, the classifier can effectively distinguish between the two classes. We also developed a single-hidden-layer neural network model from scratch. Our developed model managed to achieve an accuracy of 96.63% that is equal to the built-in Python model with the confusion matrix below:

$$confusion\ matrix = \begin{bmatrix} 3206 & 39 \\ 96 & 659 \end{bmatrix}$$

This result implies that our model is comparable to the built-in Python model. Furthermore, the accuracy is high enough to use this structure for our classification task.

Reduced-Dimension Dataset using PCA Algorithm

In the previous section, we discussed the results of classifying the patients using the original dataset which includes 18 features and managed to achieve acceptable results. However, due to the high dimensionality of this dataset, it is not suitable for visualization purposes. To overcome this issue, we utilize PCA algorithm to reduce the number of features to 2. The figure below demonstrates the simplified dataset using built-in *PCA* function of SKLearn library versus our developed PCA algorithm from scratch:

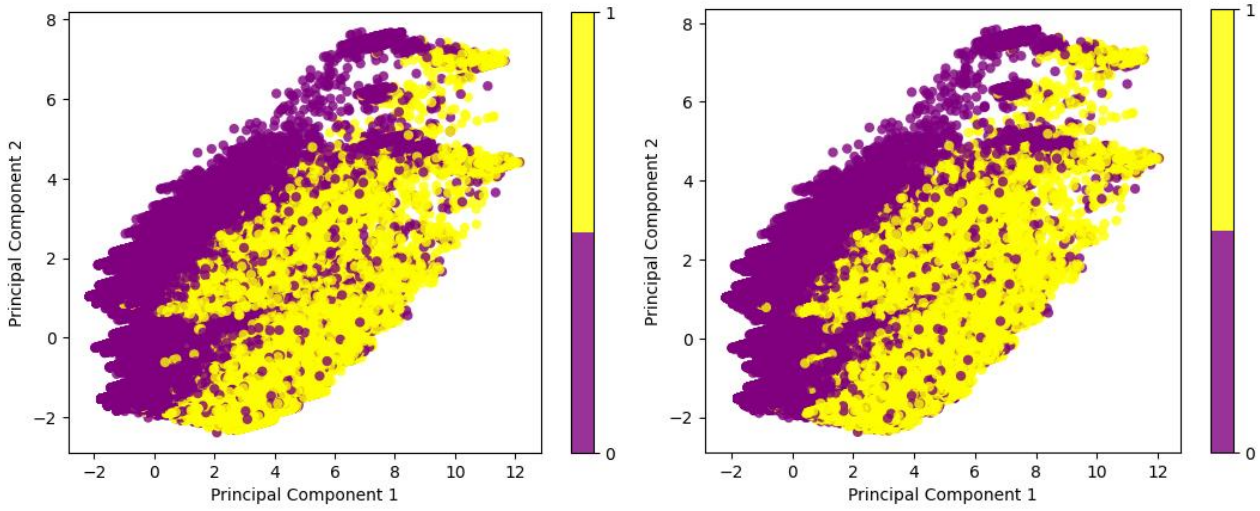

Figure 5 Dimension reduction using PCA algorithm: (a) Built-in function, (b) Our implementation.

We used the simplified dataset and ran all the algorithms developed in the previous part on it. The accuracies were less yet comparable to corresponding models trained on the original dataset. The table below summarizes the accuracy metric of our developed models.

Table 2 Accuracy metric of developed models

Classification Algorithm	Accuracy (%)	
	Original Dataset	Reduced Dataset with PCA
Built-in Logistic Regression	97.004	91.92
Logistic Regression with MSE	92	86.15
Logistic Regression with Binary Cross-Entropy	96.85	91.61
Built-in KNN	96.08	95.45
KNN	95.98	92.63
Built-in Neural Network	96.63	93.15
FCNN	96.63	86.58

Conclusions

In this project, we developed several Machine Learning models, including Logistic Regression, K-Nearest Neighbors, and Fully Connected Neural Network to accurately predict whether a patient needs to be intubated or not based on relevant clinical data. As we can see in the table above, among our developed models on the original dataset with 18 features, the logistic regression model with binary cross-entropy cost function has the most accuracy. Furthermore, among the models trained on the simplified dataset with PCA algorithm that only consists of 2 features, the KNN model is slightly better than the logistic regression with cross-entropy cost at the expense of higher computational complexity. Therefore, we claim that the logistic regression model with binary cross-entropy cost function is the best model among these algorithms for our binary classification task on COVID-19 dataset.

References

[1] World Health Organization, "WHO COVID-19 dashboard," *World Health Organization*, 2023. Available: https://covid19.who.int/

[2] J. C. Grotberg and B. Kraft, "Timing of intubation in COVID-19: When it is too early and when it is too late," Critical Care Explorations, vol. 5, no. 2, p. e0863, Feb. 2023, doi: 10.1097/cce.0000000000000863.

[3] M. M. Dar, L. Swamy, D. Gavin, and A. C. Theodore, "Mechanical-Ventilation supply and options for the COVID-19 pandemic. Leveraging all available resources for a limited resource in a crisis," Annals of the American Thoracic Society, vol. 18, no. 3, pp. 408–416, Mar. 2021, doi: 10.1513/annalsats.202004-317cme.

[4] A. T. Janke, H. Mei, C. Rothenberg, R. D. Becher, Z. Lin, and A. K. Venkatesh, "Analysis of hospital resource availability and COVID-19 mortality across the United States," Journal of Hospital Medicine, vol. 16, no. 4, pp. 211–214, Jan. 2021, doi: 10.12788/jhm.3539.

[5]"COVID-19 Dataset," Kaggle, Nov. 13, 2022. Available: https://www.kaggle.com/datasets/meirnizri/covid19-dataset

Appendices

Include in the appendices any of the following:

a. Calculations and Derivations.

b. Computer Program Listings / Information, etc