Covid-19 classification with pattern recognition system

Abstract

The COVID-19 pandemic has highlighted the critical need for effective diagnostic tools to rapidly identify infected individuals. This project aims to develop a comprehensive pattern recognition system to analyze COVID-19-related data and accurately predict infection status. Leveraging machine learning algorithms, the system processes a diverse set of features derived from clinical data, laboratory tests, and imaging stud- ies. The primary goal is to enhance the accuracy and speed of COVID-19 diagnosis, providing a valuable tool for healthcare professionals in managing the pandemic. This documentation outlines the methodology, implementation, and evaluation of the developed system, demonstrating its potential to contribute significantly to public health efforts.

1. Introduction

The COVID-19 pandemic has presented unprecedented challenges to global healthcare systems, necessitating the development of rapid and reliable diagnostic tools. Early and accurate detection of COVID-19 is crucial for effective treatment, quarantine measures, and controlling the spread of the virus. Traditional diagnostic methods, while effective, can be time-consuming and resource-intensive. As a result, there is a growing interest in employing machine learning techniques to enhance diagnostic accuracy and efficiency.

This project focuses on building a pattern recognition system specifically designed to address the diagnostic challenges posed by COVID-19. The system utilizes the COVID-19 Dataset, which includes a wide range of patient data such as demographic information, symptoms, laboratory results, and radiographic images. By applying advanced machine learning algorithms, the system identifies patterns and correlations within the data that are indicative of COVID-19 infection.

The objectives of this project are as follows:

1. To preprocess and analyze the COVID-19 Dataset to extract meaningful features relevant to diagnosis.
2. To design and implement a machine learning model capable of accurately classifying COVID-19 infection status based on the extracted features.
3. To evaluate the performance of the model using standard metrics such as accuracy, sensitivity, specificity, and F1 score.

The subsequent sections of this documentation will provide a detailed account of the data preprocessing techniques, model selection and training process, performance evaluation, and the final system implementation. Through this project, we aim to demonstrate the feasibility and effectiveness of machine learning-based pattern recognition systems in the context of COVID-19 diagnostics, contributing to the broader efforts to combat the pandemic.

2. Dataset "COVID-19"

The dataset was provided by the Mexican government (link). This dataset contains an enormous number of anonymized patient-related information including pre-conditions. The raw dataset consists of 21 unique features and 1,048,576 unique patients. In the Boolean features, 1 means "yes" and 2 means "no".

Features:

- **sex:** 1 for female and 2 for male.
- **age:** of the patient.
- **classification:** covid test findings. Values 1-3 mean that the patient was diagnosed with covid in different
- **degrees:** 4 or higher means that the patient is not a carrier of covid or that the test is inconclusive.
- **patient type:** type of care the patient received in the unit. 1 for returned home and 2 for hospitalization.
- **pneumonia:** whether the patient already have air sacs inflammation or not.
- **pregnancy:** whether the patient is pregnant or not.

- **diabetes:** whether the patient has diabetes or not.
- **copd:** Indicates whether the patient has Chronic obstructive pulmonary disease or not.
- **asthma:** whether the patient has asthma or not.
- **inmsupr:** whether the patient is immunosuppressed or not.
- **hypertension:** whether the patient has hypertension or not.
- **cardiovascular:** whether the patient has heart or blood vessels related disease.
- **renal chronic:** whether the patient has chronic renal disease or not.
- **other disease:** whether the patient has other disease or not.
- **obesity:** whether the patient is obese or not.
- **tobacco:** whether the patient is a tobacco user.
- **usmr:** Indicates whether the patient treated medical units of the first, second or third level.
- **medical unit:** type of institution of the National Health System that provided the care.
- **intubed:** whether the patient was connected to the ventilator.
- **icu:** Indicates whether the patient had been admitted to an Intensive Care Unit.
- **date died:** If the patient died indicate the date of death, and 9999-99-99 otherwise.

3. Data Preprocessing

3.1. Data cleaning

The COVID-19 dataset required several cleaning and preprocessing steps to prepare it for analysis and modeling. This section outlines the key steps taken to clean the data, transform variables, and prepare the features for subsequent machine learning tasks.

- **Handling Classifications**

The CLASIFFICATION FINAL column in the dataset represents different classifications of COVID-19 patients. The initial value counts for this column were as follows:

CLASIFFICATION FINAL

- 7 -> 499250
- 3 -> 381527
- 6 -> 128133
- 5 -> 26091
- 1 -> 8601
- 4 -> 3122
- 2 -> 1851

To simplify the classification and create a binary outcome variable for modeling, we implemented a binarization step. Patients with a classification value below 4 were considered COVID-19 carriers, while those with values 4 and above were not carriers After binarization, the value counts for the CLASIFFICATION FINAL column were:

CLASIFFICATION FINAL

- 0 -> 656596
- 1 -> 391979

3.2. Feature Selection

3.2.1. Droping

The dataset contains various columns, among which CLASIFFICATION FINAL is the target variable and DATE DIED is not relevant for scaling purposes.Therefore, these columns were excluded from the feature set.The feature matrix X was created by dropping the CLASIFFICATION FINAL and DATE.DIED columns from the dataset.

After preprocessing, the shape of the feature matrix X was (1,048,575, 19), indicating that there are 1,048,575 samples and 19 features. The target vector y had a shape of (1,048,575,), corresponding to the binary classification labels for each sample.

3.2.2. Particle Swarm Optimization (PSO)

Feature-selection-with-Particle-Swarm-Optimization-PSO
Swarm Optimization Algorithm: An Overview.
Swarm Optimization Algorithm is a type of computational algorithm inspired by the collective behavior of decentralized, self-organized systems found in nature, such as bird flocks, fish schools, and insect colonies. The most common forms of swarm optimization algorithms include Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO).

Particle Swarm Optimization (PSO):

1. Inspiration: Derived from the social behavior of birds flocking or fish schooling.
2. Mechanism: Uses a population of candidate solutions called particles. Each particle moves through the solution space influenced by its own best-known position and the best-known positions of other particles in the swarm.
3. Objective: Find the optimal solution by **iteratively improving candidate solutions** with regard to a given measure of quality or fitness.

How It Works: Particle Swarm Optimization (PSO)

1. Initialization: Initialize a population of particles with **random positions** and **velocities** in the solution space.
2. Evaluation: Evaluate the fitness of each particle according to the objective function.
3. Update Velocities and Positions: Each particle updates its velocity based on three factors:

- *Inertia:* The particle's previous velocity.
- *Personal Best:* The distance to the particle's best-known position.
- *Global Best:* The distance to the swarm's best-known position. Update the particle's position by **adding its new velocity to its current position**.

4. Iteration: Repeat the evaluation and update steps until a termination criterion is met (e.g., a maximum number of iterations or a satisfactory fitness level).

Particle Swarm Optimization (PSO) for feature selection:

Particle Swarm Optimization (PSO) can be adapted for feature selection in a tabular dataset. Feature selection aims to choose a subset of relevant features for model building, which can improve the performance and interpretability of a machine learning model.

Feature selection with (PSO) step-by-step:

1. Define the Problem and Objective Function

The objective function could be the performance metric of a machine learning model (e.g., accuracy, F1-score) using a subset of features.

2. Encode Particles

Each particle represents a subset of features. This can be encoded as a binary vector where 1 means the feature is selected and 0 means it is not.

3. Initialize Particles

Initialize particle positions (binary vectors) and velocities (continuous values).

4. Evaluate Fitness

Evaluate the fitness of each particle by training and validating a machine learning model using the selected features.

5. Update Velocities and Positions

Update velocities and positions using PSO equations. Use a sigmoid function to convert continuous velocities into binary positions.

6. Update Personal and Global Bests

Update the personal best positions and the global best position based on the fitness evaluations.

7. Iterate

Repeat the update steps until the stopping criterion is met.

8. Return the Best Feature Subset

Return the best featuresubset found during the iterations.

- Defining the Problem and Objective Function
- Encoding Particles
- Initializing Particles
- Evaluating Fitness
- Updating Velocities and Positions
- Updating Personal and Global Bests
- Iteration Process
- Returning the Best Feature Subset

In our project on COVID-19, PSO selected the following features from a medical dataset:

- MEDICAL UNIT
- SEX
- PATIENT TYPE
- INTUBED
- PNEUMONIA
- AGE
- COPD
- HIPERTENSION
- OTHER DISEASE
- OBESITY
- ICU

This subset of features resulted in a model **accuracy of 0.66**, demonstrating the effectiveness of PSO in identifying a relevant feature set that enhances model performance.

3.3. Feature Extraction

3.3.1. Minmax Scaling

We employ min-max scaling as a feature extraction method in our pattern recognition project. This technique normalizes the features to a specified range, ensuring uniformity in feature scales and preserving relationships within the data. By scaling features using min-max scaling, we enhance the stability and performance of our machine learning models, particularly in algorithms sensitive to feature scales such as k-nearest neighbors (KNN) and support vector machines (SVM). This preprocessing step is integral to our data pipeline, contributing to improved pattern recognition accuracy and convergence speed.

Min-max scaling transforms feature values to a fixed range (often 0 to 1) by subtracting the minimum value and dividing by the range. This ensures all features are on a similar scale, which can benefit machine learning algorithms sensitive to feature scales.

4. Dimentionality Reduction

4.1 What's the Method for LDA?

LDA is a technique used primarily for dimensionality reduction and classification. It increases interpretability and minimizes loss of information.

The goal of LDA is to project a dataset onto a lower-dimensional space with good class-separability in order to avoid overfitting ("curse of dimensionality") and also to reduce computational costs.

It works by calculating the directions ("linear discriminants") that will represent the axes that maximize the separation between multiple classes.

4.2 How Does LDA Work?

LDA works by calculating the mean and variance for each class. Then, it computes the "between-class variance" and the "within-class variance".

The goal of LDA is to maximize the between-class variance (i.e., the distance between the means of different classes) and minimize the within-class variance (i.e., the scatter of each individual class).

4.3 Comparison: LDA vs PCA

While both Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) are popular methods for dimensionality reduction, they have significant differences and are used in different contexts. Let's dive deeper into these distinctions:

Supervised vs Unsupervised: LDA is a supervised method, which means it takes into account the output labels during the training process. On the other hand, PCA is an unsupervised method and ignores the output labels, focusing only on the features in the dataset. This fundamental difference often dictates when you'd use one over the other.

Dimensionality Reduction vs Class Separation: PCA is primarily used for dimensionality reduction and data compression, seeking to find the axes that capture the most variance in the data. In contrast, LDA aims to find the axes that maximize the separation between multiple classes. This makes PCA more suitable for tasks where the goal is to reduce the size or complexity of the dataset, while LDA is more suitable for tasks where the goal is to separate different classes or categories.

Assumptions: PCA makes fewer assumptions about the data than LDA. PCA does not make assumptions about the class labels or the distribution of data, while LDA assumes that the data for each class is drawn from a Gaussian distribution and that all classes have the same covariance matrix. This can make PCA more robust to different kinds of datasets, while LDA might perform poorly if its assumptions are not met.

4.4 when should we use each method?

Use PCA when: We are dealing with a high-dimensional dataset and your primary goal is to reduce its dimensionality (data compression) without considering any class labels. PCA can also be used as a preliminary step to reduce the dimensionality before applying another method.

Use LDA when: We have labelled data and your goal is to maximize the separability between different classes, especially in the context of a classification problem. Keep in mind that LDA's assumptions about the data must be met for it to perform well.

5. Modeling

- **RandomForestClassifier:**
  The RandomForestClassifier is an ensemble learning method that builds multiple decision trees during training and merges their predictions to improve accuracy and control overfitting. It leverages the concept of bagging, where each tree is trained on a random subset of the data, and uses averaging to reduce variance, making it robust against overfitting.

- **LogisticRegression:**
  LogisticRegression is a linear model used for binary classification tasks. It estimates the probability that a given input belongs to a particular class using the logistic function. Despite its name, Logistic Regression is a classification algorithm, not a regression algorithm. It is simple to implement and interpret, and it performs well on linearly separable data.

- **GaussianNB:**
  GaussianNB, or Gaussian Naive Bayes, is a probabilistic classifier based on Bayes' theorem. It assumes that the features follow a Gaussian distribution. This model is particularly effective when the independence assumption holds and is known for its computational efficiency, making it suitable for large datasets.

- **GradientBoostingClassifier:**
  The GradientBoostingClassifier is an ensemble technique that builds models sequentially, with each new model attempting to correct the errors of the previous ones. It uses a gradient descent algorithm to minimize the loss function. This approach makes it highly effective for a variety of predictive modeling tasks, particularly where the relationship between features and the target is complex and non-linear.

6. Results

Accuracy: 0.7552593451173573
Classification Report:

	precision	recall	f1-score	support
0	0.54	0.45	0.49	9106
1	0.81	0.87	0.84	25404
accuracy			0.76	34510
macro avg	0.68	0.66	0.66	34510
weighted avg	0.74	0.76	0.75	34510

Figure1: Results Of KNN Classifier

Accuracy: 0.7877426832802087
Classification Report:

	precision	recall	f1-score	support
0	0.67	0.39	0.49	9106
1	0.81	0.93	0.87	25404
accuracy			0.79	34510
macro avg	0.74	0.66	0.68	34510
weighted avg	0.77	0.79	0.77	34510

Figure2: Results Of Randomforest

Accuracy: 0.794986960301362
Classification Report:

	precision	recall	f1-score	support
0	0.71	0.37	0.49	9106
1	0.81	0.95	0.87	25404
accuracy			0.79	34510
macro avg	0.76	0.66	0.68	34510
weighted avg	0.78	0.79	0.77	34510

Figure 3: Results Of Logistic :

Accuracy: 0.7944074181396696
Classification Report:

	precision	recall	f1-score	support
0	0.70	0.38	0.49	9106
1	0.81	0.94	0.87	25404
accuracy			0.79	34510
macro avg	0.76	0.66	0.68	34510
weighted avg	0.78	0.79	0.77	34510

Figure 4:GaussianNB

Accuracy: 0.7958562735439003
Classification Report:

	precision	recall	f1-score	support
0	0.72	0.37	0.49	9106
1	0.81	0.95	0.87	25404
accuracy			0.80	34510
macro avg	0.76	0.66	0.68	34510
weighted avg	0.78	0.80	0.77	34510

Figure5: GradientBoostingClassifier