2023 – 2024
Supervisor : Sébastien Harispe

IMT Mines Alès
École Mines-Télécom

Pneumonia detection using machine learning

Le Gourrierec Titouan, Cysique Marcus, Leroy Emma, El Mannouy Aymen

March 22, 2024

Abstract

Each year, over 450 million people worldwide suffer from pneumonia, accounting for approximately 7% of the global population. Sadly, more than four million individuals die from it annually, with children under 5 years old being the most affected. In 2017, over 800,000 children under the age of five died from pneumonia, a figure 16 times higher than cancer-related deaths and 10 times higher than those related to HIV. It is estimated that more than 11 million children under the age of 5 will die from pneumonia by 2030.

Our machine learning project aims to compare different machine learning models for the detection of pneumonia in pediatric chest radiography.

Contents

1 Introduction

1.1 What is Pneumonia ?

Pneumonia is an infection of the lungs caused by pathogens such as bacteria, viruses, or fungi. It leads to inflammation of the lung air sacs, resulting in symptoms like cough, fever, and difficulty breathing. Pneumonia can be dangerous, especially for children and the elderly. Treatment depends on its cause and severity and may involve medications and supportive measures.

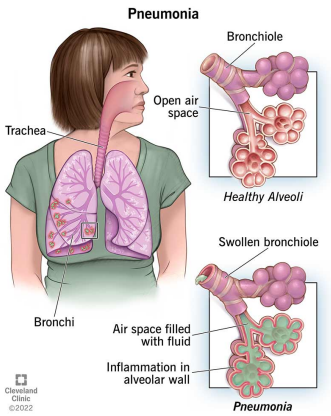

Figure 1 – Pneumonia Cle

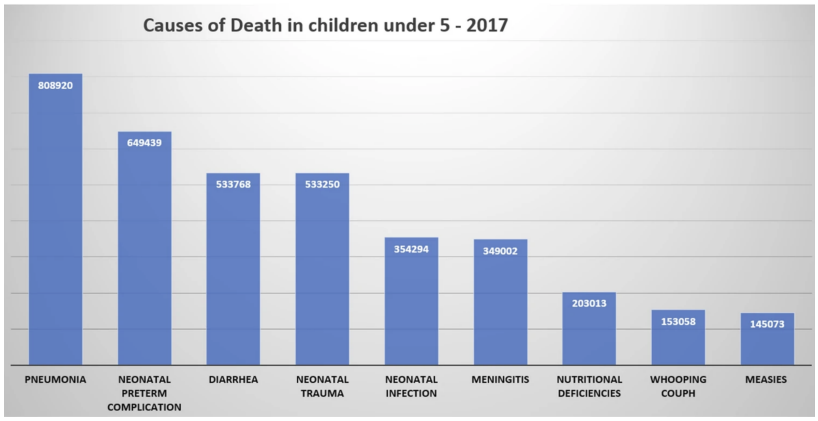

Figure 2 – Cause of Death in children under 5 - 2017 Kareem et al. [2022]

Each year, more than 450 million people worldwide suffer from pneumonia, constituting approximately 7% of the global population. Unfortunately, over four million individuals die from it annually, with children under 5 years old being the most affected. In 2017, over 800,000 children under the age of five died from pneumonia, a figure 16 times higher than cancer-related deaths and 10 times higher than those related to HIV. It is estimated that more than 11 million children under the age of 5 will die from pneumonia by 2030.

1.2 Problem formalization

Our task involves supervised binary classification, where lung images serve as input data and the output labels distinguish between healthy and pneumonia-afflicted patients. We'll assess a range of machine learning models, benchmarking their performance across multiple metrics.

1.3 Why using Machine Learning ?

The application of machine learning to pneumonia detection on lung radiographs proves suitable for several reasons. Firstly, the availability of a sufficient amount of data enables the feeding of machine learning models, thereby promoting enhanced performance. Additionally, this approach does not raise any ethical concerns as it contributes to improving patient health and aids physicians in decision-making. Moreover, given that the precise criteria for diagnosing pneumonia can be challenging to articulate algorithmically, machine learning techniques can learn to identify complex patterns associated with this pathology from data, thus providing an effective and reliable detection method.

2 The Dataset

2.1 Details

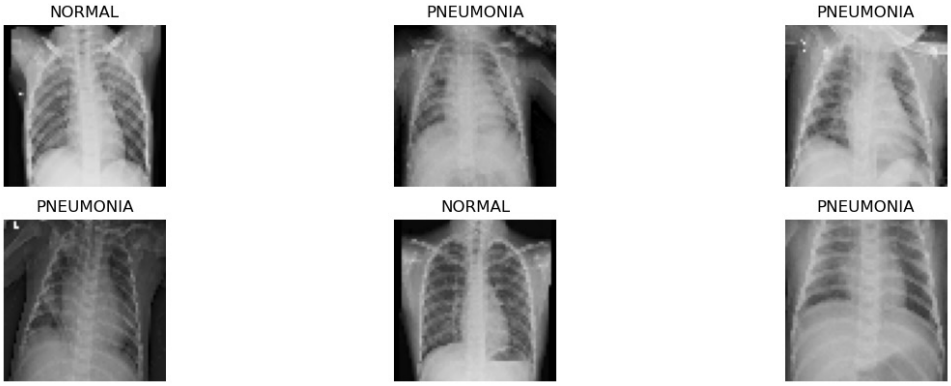

Figure 3 – Example images from the dataset

Chest X-ray images were sourced from retrospective cohorts of pediatric patients aged one to five years old, collected from Guangzhou Women and Children's Medical Center in Guangzhou. These images constituted part of routine clinical care for the patients. To ensure the reliability of the dataset, all chest radiographs underwent an initial screening process to eliminate low-quality or unreadable scans. Subsequently, the diagnoses assigned to the images were evaluated by two expert physicians before final approval. Additionally, to mitigate potential grading errors, a third expert reviewed the evaluation set as well.

The dataset is accessible at: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia.

2.2 Separation of the dataset

Initially, the dataset comprises 5856 images of varying sizes, which we resize to 64*64. It is divided into three separate folders - train, val, and test - each containing subfolders PNEUMONIA and NORMAL, which respectively hold radiographs of patients with pneumonia and those of healthy patients. To streamline management, we consolidate these images into two distinct folders - PNEUMONIA and NORMAL - before proceeding to further partition them according to our specific requirements.

We utilize the `train_test_split` function from `sklearn.model_selection` to divide our dataset into either two parts, train and test (80% train, 20% test), or three parts, train, test, and val (64% train, 16% val, and 20% test) when necessary.

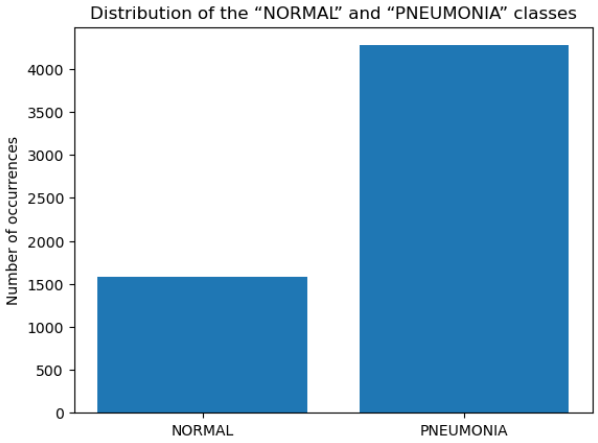

Figure 4 – Class imbalance in the dataset

It can be observed that the dataset is imbalanced, with 4273 examples of pneumonia and 1583 examples of healthy patients, a challenge that we will endeavor to address.

3 Metrics used

In this section, we will present and define the metrics employed to validate our model. Our task involves binary classification, meaning we have two possible outcomes: pneumonia or normal. In our context, these outcomes correspond to positive or negative. Consequently, following predictions, we will encounter four potential cases outlined in Figure 1. (Harispe [2024])

		Reality	
		PNEUMONIA	NORMAL
Prediction	PNEUMONIA	True Positive (TP)	False Positive (FP)
	NORMAL	False Negative (FN)	True Negative (TN)

Table 1 – Confusion Matrix

3.1 Accuracy

Accuracy in machine learning refers to the proportion of correct predictions made by a model over the total number of predictions, serving as a metric to evaluate the model's performance in classification tasks.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

3.2 Precision

Precision in machine learning measures the accuracy of positive predictions. It provides insights into the model's ability to avoid false positives.

$$Precision = \frac{TP}{TP + FP}$$

3.3 Recall

Recall, also known as sensitivity or true positive rate, evaluates the model's ability to identify all positive instances.

$$Recall = \frac{TP}{TP + FN}$$

3.4 F1 Score

The F1 Score is the harmonic mean of precision and recall, providing a single metric to evaluate the balance between precision and recall.

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

4 Preprocessing methods used

In terms of preprocessing, we experimented with three configurations to compare the models: no preprocessing, conversion from RGB to grayscale, and conversion from RGB to grayscale followed by standard scaling.

Here's a breakdown of each configuration:

1. **No preprocessing** : The original RGB images were used as input without any additional processing.

2. **Conversion from RGB to grayscale** : The RGB images were converted to grayscale, resulting in single-channel images where each pixel represents the intensity of light.

3. **Conversion from RGB to grayscale + standard scaling** : In addition to converting the images to grayscale, standard scaling was applied to normalize the pixel values. Standard scaling ensures that each feature (pixel) has a mean of 0 and a standard deviation of 1, which can help improve the performance of certain machine learning algorithms.

The formula for standard scaling is:

$$z = \frac{x - \mu}{\sigma}$$

where z is the standardized value, x is the original value, μ is the mean of the feature (pixel) values, σ is the standard deviation of the feature (pixel) values.

This formula is applied to each pixel value in the grayscale images to achieve standard scaling.

5 Models Used

For many of the models we studied, we conducted a grid search to determine the optimal parameters to use in order to maximize the results. GridSearchCV from sklearn was employed, which systematically explores a predefined grid of hyperparameters, allowing us to select the best

combination based on cross-validated performance scores. This approach aids in automating the process of hyperparameter tuning, enhancing the efficiency and effectiveness of model optimization.

5.1 K Neighbors Classifier

The **K-Nearest Neighbors (KNN)** algorithm is a supervised learning method used for classification and regression tasks. It operates by assigning a class to a data point based on the majority classes among its k nearest neighbors, where k is a predefined hyperparameter. In other words, it classifies a new instance based on the most common class among its nearest neighbors in the feature space.

Hyperparameters modified for grid search:

- `n_neighbors` : the number of nearest data points to consider when making predictions for a new instance.
- `weights` : specifies the weight function used in prediction. For 'uniform', $w_i = 1$. For 'distance', $w_i = \frac{1}{d(q,p_i)}$, where $d(q, p_i)$ represents the distance between the query point q and the i-th neighbor p_i.
- `metric` : defines the distance metric used to measure the similarity between data points.

5.2 Logistic Regression

Logistic Regression is a statistical method used for binary classification tasks, where it models the probability of a binary outcome based on predictor variables. It fits a logistic function to the observed data, transforming the output of a linear regression model into a probability value between 0 and 1, allowing it to predict the probability of occurrence of an event.

Hyperparameters modified for grid search:

- `penalty`: refers to the regularization technique applied to the model to prevent overfitting. 'l1' penalty adds the absolute values of coefficients, representing the L1 norm of coefficients: $|w|_1 = |w_1| + |w_2| + \ldots + |w_n|$. 'l2' penalty adds the squares of coefficients, representing the L2 norm of coefficients: $|w|_2 = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$.
- `C`: determines the strength of regularization, balancing between fitting the training data well and preventing overfitting.
- `solver`: specifies the algorithm used to optimize the model's parameters. `Liblinear` is efficient for small and medium-sized datasets, while `SAGA` (Stochastic Average Gradient Ascent) is suitable for large datasets.

5.3 Decision Tree Classifier

A **Decision Tree Classifier** is a machine learning model that makes decisions by splitting the data into branches based on feature values. It starts with a root node and recursively divides the data into subsets based on the feature that provides the best split, aiming to create branches that best separate the classes of the target variable. This process continues until certain stopping criteria are met, resulting in a tree-like structure where each leaf node represents a class label. During prediction, new data is traversed through the tree, following the decisions made at each node until reaching a leaf node, which then predicts the class label.

Hyperparameters modified for grid search:

— `criterion` : determines the measure used to evaluate the quality of a split at each node during the tree's construction. For 'gini' criterion, the Gini impurity formula (Ninja [2020]) is used: $I_{\text{gini}}(p) = 1 - \sum_{i=1}^{n} p_i^2$, where p_i is the probability of class i for a particular node. It aims to minimize the probability of misclassification by splitting the nodes in a way that reduces impurity. For 'entropy' criterion, the entropy formula (Sayad) is used: $I_{\text{entropy}}(p) = -\sum_{i=1}^{n} p_i \log_2(p_i)$. It splits the nodes to maximize information gain, aiming to decrease entropy and increase homogeneity in each node.

— `max_depth` : controls the maximum depth of the tree. It limits the number of levels or nodes from the root to the farthest leaf.

— `min_samples_split` : sets the minimum number of samples required to split an internal node.

— `min_samples_leaf` : sets the minimum number of samples required to be at a leaf node.

5.4 Support Vector Classifier (SVC)

Support Vector Classifier (SVC) is a classification algorithm that finds the best hyperplane to separate different classes by maximizing the margin between them, using support vectors as the closest data points to the hyperplane.

Hyperparameters modified for grid search:

— `C` : controls the regularization strength.

— `kernel` : specifies the type of kernel function used to transform the input features into a higher-dimensional space where the data may be more separable. `Linear` creates a linear decision boundary in the original feature space, `Poly` uses polynomial functions to map the data into a higher-dimensional space, allowing for more complex decision boundaries, `RBF` transforms the data into an infinite-dimensional space using Gaussian radial basis functions, effectively capturing non-linear relationships in the data.

— `gamma` : determines how far the influence of a single training example reaches, affecting the flexibility of the decision boundary.

— `degree` : specifies the degree of the polynomial used to transform the data into a higher-dimensional space (with poly kernel).

5.5 Random Forest Classifier

Random Forest Classifier is an ensemble learning method that constructs a multitude of decision trees during training. Each tree is trained on a random subset of the training data and a random subset of features. During prediction, each tree independently outputs a class prediction, and the final prediction is determined by aggregating the votes from all the trees. This ensemble approach helps improve accuracy and generalization while also providing insights into feature importance. Unlike large decision trees, Random Forests are effective at avoiding overfitting, thanks to the ensemble nature and the randomness injected into each tree's construction (Ho [1995]).

Hyperparameters modified for grid search:

— `n_estimators` : specifies the number of decision trees to be created in the forest.

— `max_depth` : same as decision tree.

— `min_samples_split` : same as decision tree.
— `min_samples_leaf` : same as decision tree.

5.6 Gradient Boosting Classifier

The Gradient Boosting Classifier is an ensemble learning technique that combines multiple weak predictive models, such as decision trees, sequentially to create a strong predictive model. It focuses on correcting the errors of the previous models in each iteration, resulting in improved overall performance. Due to time and resources reasons, we did not perform a grid search on this classifier.

5.7 Voting Classifier

Voting Classifier is an ensemble learning method that combines the predictions from multiple individual classifiers and predicts the class label by majority vote (for classification tasks) or by averaging (for regression tasks). It aggregates the predictions from different models, which may have been trained on different algorithms or subsets of data, to make a final prediction. This approach often leads to better generalization and robustness compared to using a single classifier.

`voting = 'hard' to vote for the most frequent class, 'soft' to vote based on probabilities.`

6 Models performance comparison

The best hyperparameters have been found through grid search for every model (5) and for every preprocessing method (4). We can proceed to compare the optimized models and each preprocessing method. The table below summarizes the performance of the different models with each preprocessing method.

Model	**Metrics**			
	Accuracy	**Precision**	**Recall**	**F1-score**
K Neighbors Classifier	0.9283	0.9179	**0.9894**	0.9523
Logistic Regression	0.9505	0.9561	0.9764	0.9662
Decision Tree	0.8695	0.8971	0.9257	0.9112
Support Vector Classifier	0.9505	**0.9604**	0.9717	0.9660
Random Forest Classifier	0.9488	0.9518	0.9788	0.9651
Gradient Boosting Classifier	0.9411	0.9401	0.9811	0.9602
Hard Voting Classifier	0.9531	0.9531	0.9835	0.9681
Soft Voting Classifier	**0.9565**	0.9565	0.9847	**0.9704**

Table 2 – Comparison of Models (Without Preprocessing)

Model	**Metrics**			
	Accuracy	**Precision**	**Recall**	**F1-score**
K Neighbors Classifier	0.9283	0.9179	**0.9894**	0.9523
Logistic Regression	0.9505	0.9561	0.9764	0.9662
Decision Tree	0.8549	0.8870	0.9163	0.9014
Support Vector Classifier	0.9505	**0.9604**	0.9717	0.9660
Random Forest Classifier	0.9428	0.9453	0.9776	0.9612
Gradient Boosting Classifier	0.9453	0.9424	0.9847	0.9631
Hard Voting Classifier	**0.9565**	0.9554	0.9858	**0.9704**
Soft Voting Classifier	**0.9565**	0.9543	0.9870	**0.9704**

Table 3 – Comparison of Models (With RGBtoGray)

Model	**Metrics**			
	Accuracy	**Precision**	**Recall**	**F1-score**
K Neighbors Classifier	0.9206	0.9116	**0.9858**	0.9473
Logistic Regression	0.9454	0.9527	0.9728	0.9626
Decision Tree	0.8728	0.9003	0.9269	0.9134
Support Vector Classifier	**0.9556**	**0.9606**	0.9788	**0.9696**
Random Forest Classifier	0.9488	0.9498	0.9811	0.9652
Gradient Boosting Classifier	0.9420	0.9402	0.9823	0.9608
Hard Voting Classifier	0.9522	0.9521	0.9835	0.9675
Soft Voting Classifier	0.9548	0.9553	0.9834	0.9692

Table 4 – Comparison of Models (With RGBtoGray and Standard Scaler)

In the medical context of pneumonia detection, recall is crucial to avoid missing sick patients. K-Nearest Neighbors (KNN) achieves a notably high recall of 0.9894 (without preprocessing and with RGBtoGray preprocessing), indicating its strength in identifying pneumonia cases. However, it's also essential to consider other metrics such as precision and overall accuracy. Despite KNN's high recall, the Soft Voting Classifier outperforms it with a balance of high recall (0.9870) and precision (0.9543) (RGBtoGray preprocessing). The Soft Voting Classifier's superior overall performance makes it the preferred choice, ensuring accurate pneumonia detection while maintaining a lower false positive rate.

Additionally, converting to grayscale reduces the dimensionality of the training data by one, making it faster and requiring fewer resources.

As such, the Soft Voting Classifier will be utilized for the remainder of the study, with the following parameters obtained through grid search for each model used :

Model	Hyperparameter	Value
K-Nearest Neighbors (KNN)	Metric n_neighbors weights	Manhattan 9 Distance
Logistic Regression	C Penalty Solver	0.01 l2 Saga
Decision Tree	Criterion max_depth min_samples_leaf min_samples_split	Gini 10 1 5
Support Vector Classifier (SVC)	C degree Gamma Kernel	100 2 Scale RBF
Random Forest Classifier	max_depth min_samples_leaf min_samples_split n_estimators	20 1 2 300

Table 5 – Summary of the best hyperparameters for each model used for Soft Voting Classifier

7 Refinement of the best model while addressing class imbalance

The problem of imbalanced classes arises when there is a significant difference in the number of observations between different classes of a target variable in a dataset. This imbalance can lead to biased models that prioritize the majority class, resulting in poor performance on minority classes.

Addressing class imbalance is crucial as it can affect model performance metrics such as sensitivity and specificity. While it's ideal to address imbalance before model evaluation, resource constraints may limit our ability to test every mitigation method across all models. Therefore, we focus on optimizing the soft voting classifier.

We've explored various strategies to tackle class imbalance, using the soft voting classifier with the best hyperparameters from Table 5 for subsequent analyses.

In this section, we define a training set (64%), a validation set (16%), and a test set (20%). For each method, we will explore different values for the `sampling_strategy` parameter on the validation set to determine the best parameter (*this parameter will be defined just after*), before evaluating the model on the test set.

7.1 Oversampling technique

Oversampling is a technique used to address class imbalance by increasing the representation of the minority class through duplicating or generating new examples, thus balancing the class distribution in the dataset.

7.1.1 RandomOverSampler

The RandomOverSampler in imbalanced-learn (imblearn) is a technique used to balance class distribution by randomly duplicating examples from the minority class(es) until the class distribution is approximately equal to that of the majority class.

The `sampling_strategy` corresponds to the desired ratio of the number of samples in the minority class over the number of samples in the majority class after resampling. Therefore, the ratio is expressed as $\alpha_{os} = \frac{N_{rm}}{N_M}$ where N_{rm} is the number of samples in the minority class after resampling and N_M is the number of samples in the majority class. (RandomOverSampler Documentation)

We test the values of 0.5, 0.75, and 1.0 for this parameter. The best parameter is 0.5. This means that the number of samples in the minority class after oversampling will be of the number of samples in the majority class. There are 453 new duplicate examples for the minority class.

7.1.2 SMOTE (Synthetic Minority Over-sampling Technique)

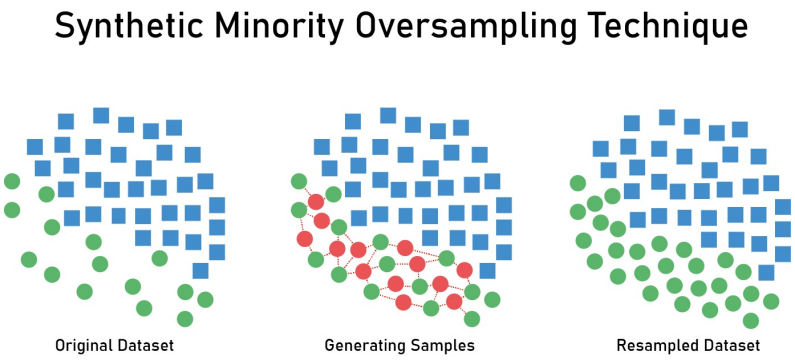

Figure 5 – How SMOTE works Dholakiya [2023]

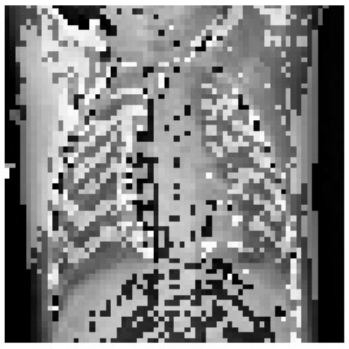

Figure 6 – SMOTE Image

SMOTE is an oversampling technique that aims to balance datasets by increasing the representation of the minority class. It works by generating new synthetic examples for the minority class by extrapolating points from the nearest neighbors in the attribute space.

As for RandomOverSampler, the `sampling_strategy` corresponds to $\alpha_{os} = \frac{N_{rm}}{N_M}$ (Smote Documentation)

We tested the values of 0.5, 0.75, and 1.0 for this parameter. The best parameter is 1. This means that the number of samples in the minority class after oversampling will be equal to the number of samples in the majority class, with 2166 new examples generated for the minority class.

7.2 Undersampling technique

Undersampling is a technique used to address class imbalance by reducing the representation of the majority class. This is typically achieved by randomly removing examples from the majority class, effectively downsampling it. By doing so, the class distribution becomes more balanced, allowing the model to give equal consideration to both classes during training.

7.2.1 RandomUnderSampler

The RandomUnderSampler in imbalanced-learn (imblearn) is a technique used to balance class distribution by randomly removing examples from the majority class(es) until the class distribution is approximately equal to that of the minority class.

The `sampling_strategy` corresponds to the desired ratio of the number of samples in the minority class over the number of samples in the majority class after resampling. Therefore, the ratio is expressed as $\alpha_{us} = \frac{N_M}{N_{rm}}$ where N_{rm} is the number of samples in the majority class after resampling and N_M is the number of samples in the minority class. (RandomUnderSampler Documentation)

We tested the values of 0.5, 0.75, and 1.0 for this parameter. The best parameter is 0.75. This means that the number of samples in the majority class after oversampling will be three-quarters of the number of samples in the minority class. There are 1747 examples removed from the majority class.

7.3 Weight modification technique

This method assigns different weights to classes based on their frequencies to handle class imbalance, enabling individual classifiers to focus more on minority classes during training. The Voting Classifier aggregates predictions from these classifiers, providing a balanced and improved performance across classes in the classification task.

7.4 Results of applying these methods to the voting classifier defined previously (5)

The table compares the performance metrics of different methods aimed at addressing class imbalance issues in the dataset. Each method is evaluated based on its accuracy, precision, recall, and F1-score. The results provide insights into the effectiveness of various approaches in handling class imbalance during classification tasks.

Methods	**Metrics**			
	Accuracy	**Precision**	**Recall**	**F1-score**
Without specific method	0.9565	0.9543	**0.9870**	0.9704
Random Over Sampling	0.9548	0.9584	0.9800	0.9691
SMOTE	0.9539	0.9584	0.9788	0.9685
Random Under Sampling	0.9565	**0.9672**	0.9729	0.9700
Weights Modification	**0.9608**	0.9609	0.9858	**0.9732**

Table 6 – Comparison of Methods to Counter Class Imbalance

In a medical context of pneumonia detection, the comparative table of different preprocessing and sampling methods shows that the weights modification method offers the best overall performance in terms of accuracy and F1-score. This method achieves an optimal balance between recall and precision, facilitating excellent pneumonia detection while minimizing false diagnoses. **Therefore, in this context, the weights modification method appears to be the most appropriate for ensuring accurate and reliable results**.

8 Final Model and Conclusion

8.1 Final Model

In data preprocessing, we employ a technique to convert color images to grayscale images. This method offers several advantages. Firstly, it reduces the number of information channels, leading to faster training and reduced resource utilization. Additionally, this conversion yields better model performance, as demonstrated in (6).

To address class imbalance in the data, we utilize class weighting, where greater emphasis is placed on errors from the minority class. This approach results in a more balanced learning process, as evidenced in section (7.4).

Subsequently, we employ a Soft Voting classifier that aggregates predictions from various models with hyperparameters explained in (Table 5) :

After performing cross-validation on the entire dataset using the chosen model, the following metrics were obtained :

	Metrics			
	Accuracy	**Precision**	**Recall**	**F1-score**
Final Model	0.9501	0.9569	0.9757	0.9662

Table 7 – Final model evaluation metrics

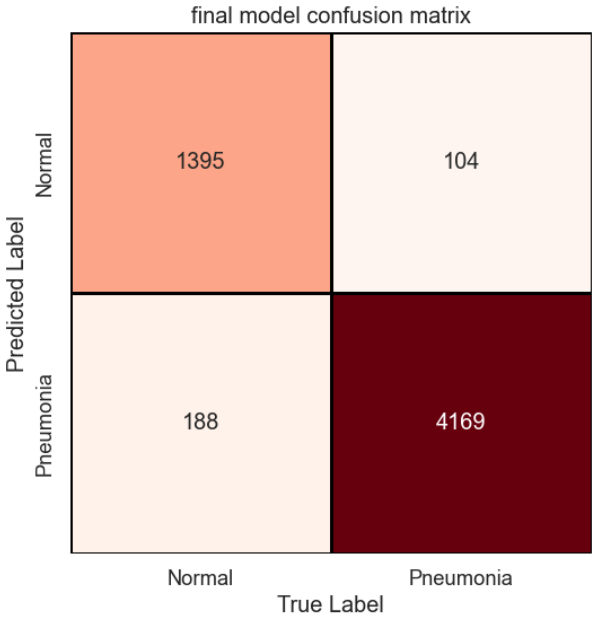

Figure 7 – Final model confusion matrix

In a real-world scenario, our model would have missed diagnosing pneumonia in 104 patients and would have incorrectly predicted that 188 healthy individuals have pneumonia. Nevertheless, it would have still accurately predicted the situation for 5564 individuals.

8.2 Conclusion and opening

8.2.1 Increase the size of the dataset

Below is the learning curve of our final model on the entire dataset. We can observe that although the score growth slows down, we do not yet see a clear plateau.

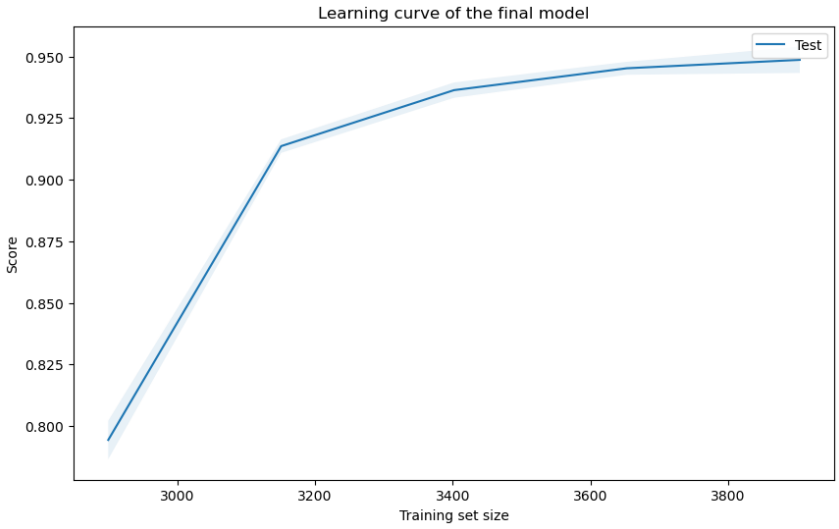

Figure 8 – Learning curve of the final model

Increasing the number of examples in the dataset could be a potential avenue for further enhancing our model's capabilities.

8.2.2 Cross-referencing the images from the dataset with other data

The Rao [2020] study evaluates the performance of four ML algorithms (KNN, Decision Tree, Logistic Regression, and Convolutional Neural Network) in detecting pneumonia from pediatric chest X-rays. Among them, the Convolutional Neural Network (CNN) achieved the highest accuracy at 90.7%. A subsequent assessment involving 14 doctors revealed a discrepancy between ML algorithms and human diagnosis, with doctors correctly diagnosing 37% of the misclassified images. This highlights the potential for ML algorithms to improve accuracy, potentially reaching up to 94%. Additionally, doctors suggested incorporating additional features like oxygen saturation level (SPO2), age, respiratory rate, and body temperature to enhance diagnosis, especially for challenging cases.

The doctors' suggestions could provide a foundation for enhancing the model by incorporating features such as oxygen saturation level (SPO2), age, respiratory rate, and body temperature.

References

Pneumonia. https://my.clevelandclinic.org/health/diseases/4471-pneumonia.

Parth Dholakiya. Smote: Synthetic minority over-sampling technique. *Medium*, 2023. URL https://medium.com/@parthdholakiya180/smote-synthetic-minority-over-sampling-technique-4d5a5d69d720.

S. Harispe. *Introduction to Machine Learning*. February 2024.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995. doi:10.1109/ICDAR.1995.598994.

Amer Kareem, Haiming Liu, and Paul Sant. Review on pneumonia image detection: A machine learning approach. *Human-Centric Intelligent Systems*, 2, 05 2022. doi:10.1007/s44230-022-00002-2.

Daniel Kermany, Kang Zhang, and Michael Goldbaum. Large dataset of labeled optical coherence tomography (oct) and chest x-ray images. https://doi.org/10.17632/rscbjbr9sj.3, 2018.

NumPy Ninja. What is gini impurity? how is it used to construct decision trees, 2020. URL https://www.numpyninja.com/post/what-is-gini-impurity-how-is-it-used-to-construct-decision-trees.

RandomOverSampler Documentation. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html#imblearn.over_sampling.RandomOverSampler.

RandomUnderSampler Documentation. https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html.

Anirudh Rao. Gap analysis of the accuracy of doctors versus machine learning models for pneumonia detection from x-rays. In *2020 IEEE International Conference on Advances and Developments in Electrical and Electronics Engineering (ICADEE)*, pages 1–3, 2020. doi:10.1109/ICADEE51157.2020.9368913.

Saed Sayad. Decision tree - classification. URL https://www.saedsayad.com/decision_tree.htm.

Smote Documentation. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html.