Ramakrishna Mission Vivekananda Educational and Research Institute

Topic : Pneumonia Classification Using Machine Learning

Presented by:

Avirup Das and Jahar Kumar Paul

Presented to:

Br. Bhaswarachaitanya(Tamal Maharaj)

Pneumonia Classification Using Machine Learning

1 Introduction

1.1 Background

Pneumonia is a **serious respiratory infection** that affects millions of people globally, resulting in significant morbidity and mortality, especially among **vulnerable populations** such as young children, the elderly, and immunocompromised individuals. Early and accurate diagnosis of pneumonia is crucial for **timely treatment** and improving patient outcomes.

Conventional diagnostic methods primarily rely on **chest radiographs (X-rays)** interpreted by trained radiologists. However, these methods can be **time-consuming**, prone to **human error**, and are often inaccessible in **resource-constrained settings** where skilled radiologists are scarce.

With the rise of **artificial intelligence** and advancements in **computer vision**, machine learning models have demonstrated significant potential in automating medical image analysis. Specifically, **convolutional neural networks (CNNs)** have proven effective in detecting diseases from medical images. By utilizing **labeled chest X-ray datasets**, machine learning techniques can aid in the accurate and efficient classification of pneumonia cases, alleviating the burden on healthcare systems and improving access to diagnostic tools in underserved areas.

This project focuses on building a **robust machine learning model** to classify chest X-ray images into two categories: **normal** and **pneumonia**. The ultimate goal is to provide a **scalable, reliable, and accessible solution** to support the **early detection of pneumonia**, especially in **low-resource settings**.

1.2 Motivation

Pneumonia remains a leading cause of death worldwide, particularly in low- and middle-income countries where access to healthcare resources is limited. Despite advancements in medical science, timely and accurate diagnosis of pneumonia continues to be a challenge in resource-constrained settings. This motivated the exploration of innovative, technology-driven solutions that can bridge this gap.

The advent of **machine learning** and **computer vision** has opened new avenues for automating medical image analysis. These technologies have the potential to improve diagnostic accuracy while reducing dependency on highly skilled professionals, making them particularly beneficial in under-resourced regions.

Moreover, the availability of publicly accessible **chest X-ray datasets** has made it feasible to apply and test machine learning algorithms for this purpose. By building a robust classification model for pneumonia detection, this project seeks to contribute

to the development of scalable and affordable diagnostic tools that can have a tangible impact on global healthcare.

The choice of this topic stems from a combination of personal interest in **healthcare innovation**, the societal relevance of the problem, and the transformative potential of machine learning in solving real-world challenges. Through this project, we aim to demonstrate the ability of technology to address critical healthcare issues and improve lives.

1.3 Objectives

The primary objective of this project is to develop a robust and efficient machine learning model capable of accurately diagnosing pneumonia from chest X-ray images. The specific goals of the project are as follows:

- **Build and preprocess a labeled dataset**: Utilize publicly available chest X-ray datasets and preprocess the data to ensure it is clean, balanced, and suitable for training. This includes handling missing data, resizing images, and augmenting the dataset to improve model generalization.
- **Develop a classification model**: Implement machine learning techniques, with a focus on convolutional neural networks (CNNs), to classify chest X-ray images into *normal* and *pneumonia* categories. Experiment with different architectures to identify the most effective model for this task.
- **Evaluate model performance**: Use comprehensive evaluation metrics such as accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic (ROC) curve. Perform a detailed analysis of false positives and false negatives to identify areas of improvement.
- **Optimize for scalability**: Ensure the model can handle real-world scenarios, including deployment on low-cost hardware or integration into existing healthcare systems. Test the model on unseen, real-world data to evaluate its practical utility.
- **Contribute to healthcare innovation**: Demonstrate the potential of machine learning to address critical healthcare challenges by providing a cost-effective, accessible, and reliable diagnostic tool. Emphasize how this solution can assist medical professionals and reduce diagnostic workload.
- **Raise awareness about AI in healthcare**: Advocate for the broader adoption of artificial intelligence in medical imaging by showcasing the benefits of automation in improving diagnostic accuracy and efficiency.
- **Incorporate ethical considerations**: Ensure the model is developed with fairness and transparency, minimizing bias in predictions. Discuss the implications of deploying AI-based tools in healthcare, particularly in underserved regions.
- **Provide a foundation for further research**: Document the methods, challenges, and findings from this project to support future research efforts in pneumonia detection and other medical applications of machine learning.

These objectives aim to tackle the technical, practical, and ethical challenges of pneumonia diagnosis while contributing to the broader field of healthcare innovation.

2 Literature Review

2.1 Existing Research

Several studies have investigated the use of machine learning and deep learning techniques for the detection of pneumonia from chest X-ray images. These works have laid the groundwork for the integration of artificial intelligence in medical diagnostics. Key findings from existing research include:

- **CheXNet for Pneumonia Detection:** Rajpurkar et al. (2017) developed *CheXNet*, a deep convolutional neural network (CNN) trained on the ChestX-ray14 dataset. Their model demonstrated diagnostic performance on par with expert radiologists, highlighting the power of CNNs for pneumonia detection.
- **Transfer Learning Approaches:** Kermany et al. (2018) leveraged pre-trained models such as InceptionV3 and ResNet to classify chest X-ray images. Their study demonstrated the effectiveness of transfer learning in achieving high accuracy while reducing training time.
- **Data Preprocessing and Augmentation:** Jain et al. (2020) explored the role of preprocessing techniques, including contrast enhancement and image augmentation, in improving model generalization. Their findings underscored the importance of balanced and diverse datasets.
- **Deployment Challenges:** Wong et al. (2021) identified key challenges in deploying machine learning models in clinical environments, such as dataset biases, model interpretability, and integration with existing workflows.
- **Ethical Considerations in AI:** Gupta et al. (2019) emphasized the importance of addressing algorithmic bias and ensuring fairness when deploying AI models in healthcare, particularly in underserved regions. Their work called for transparency and fairness in AI applications.

The reviewed literature highlights the significant progress made in automated pneumonia detection while also identifying challenges related to data quality, model scalability, and ethical considerations. This project aims to build on these advancements by addressing key limitations and contributing to the development of practical, deployable solutions.

2.2 Methodologies Used in Similar Problems

Various methodologies have been employed in previous studies to tackle the problem of pneumonia detection from chest X-ray images. These approaches leverage advancements in machine learning, deep learning, and image processing techniques. Some of the commonly used methodologies include:

- **Traditional Machine Learning Techniques:** Early methods relied on handcrafted features such as edge detection, texture analysis, and histogram-based techniques. These features were then used with classifiers like Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Decision Trees. While effective in limited scenarios, these approaches often struggled with the complexity of medical images.

- **Convolutional Neural Networks (CNNs):** CNNs have revolutionized image analysis by automatically extracting features from images. Models like AlexNet, VGGNet, and ResNet have been extensively used for medical imaging tasks, including pneumonia detection. These models are particularly effective due to their ability to learn hierarchical features from raw images.
- **Transfer Learning:** Transfer learning has been a widely adopted approach to tackle the challenge of limited labeled medical datasets. Pre-trained models, such as InceptionV3, DenseNet, and ResNet, have been fine-tuned on pneumonia datasets to achieve high accuracy with reduced training time.
- **Ensemble Learning:** Ensemble methods combine the predictions of multiple models to improve overall performance. Techniques such as bagging, boosting (e.g., AdaBoost), and stacking have been used to enhance model robustness in medical diagnosis tasks.
- **Data Augmentation and Preprocessing:** Given the scarcity of labeled medical data, augmentation techniques such as rotation, flipping, scaling, and noise addition have been employed to artificially expand datasets. Preprocessing methods like histogram equalization and normalization are commonly applied to improve image quality and model performance.
- **Explainable AI (XAI):** To address the lack of interpretability in deep learning models, researchers have used XAI methods like Grad-CAM and LIME to visualize decision-making processes. This is particularly important for building trust in AI systems used in healthcare.
- **Hybrid Approaches:** Combining traditional image processing techniques with deep learning has also been explored. For example, segmentation techniques are used to isolate regions of interest before applying CNNs for classification.

These methodologies have collectively advanced the field of medical image analysis, but challenges such as generalizability, model explainability, and deployment in resource-limited settings remain open areas for exploration. This project adopts a hybrid approach, incorporating CNNs and transfer learning while emphasizing interpretability and scalability.

3 Dataset Description

3.1 Source

The dataset used for this project is sourced from the publicly available **Kaggle Chest X-ray Images (Pneumonia)** dataset. This dataset was originally provided by **Paul Mooney** and contains labeled chest X-ray images for pneumonia classification. The images are categorized into two classes: *Normal* and *Pneumonia* (which includes bacterial and viral pneumonia cases).

The dataset can be accessed via the Kaggle platform under the following link:

`https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia`

The dataset has been widely used in the medical imaging community for benchmarking pneumonia detection models due to its diversity and clinical relevance.

3.2 Features

The Kaggle Chest X-ray Pneumonia dataset contains several features relevant to the classification task. The dataset consists primarily of chest X-ray images, which are associated with the following key variables:

- **Image:** The dataset includes X-ray images in PNG format. Each image represents a frontal view of the chest, either from a healthy patient (Normal) or a patient diagnosed with pneumonia (Pneumonia). These images vary in size and resolution, and may contain varying degrees of noise, occlusion, and artifacts.
- **Label:** Each image is labeled as one of two categories: *Normal* or *Pneumonia*. The Pneumonia class includes both bacterial and viral pneumonia cases. The labels are manually annotated and serve as the target variable for classification.
- **Image Dimensions:** The original image size is generally inconsistent, with different resolutions and aspect ratios. For model training, these images are often resized to a standard resolution (e.g., 224x224 pixels) for uniformity and better performance.
- **Class Distribution:** The dataset contains an imbalanced distribution, with a larger number of Normal images compared to Pneumonia images. This imbalance may require special consideration during model training, such as through oversampling, undersampling, or using weighted loss functions.
- **Patient Metadata (if available):** Some subsets of the dataset may include additional metadata such as patient age, sex, and other clinical information, although the Kaggle version primarily focuses on the image data itself. If available, this metadata can be used for more advanced analysis, including multi-modal approaches (image + tabular data).
- **Image Preprocessing Features:** Before training the model, the images may undergo preprocessing steps like resizing, normalization, histogram equalization, and augmentation (e.g., rotation, flipping, and scaling). These steps help in improving model robustness and generalization.

The dataset's primary feature for the classification task is the image itself, with the label serving as the target variable for model predictions. The overall goal is to build a model capable of accurately classifying these images as either *Normal* or *Pneumonia* based on the learned features.

3.3 Target Variable

The target variable for this project is the **Label**, which represents the classification of chest X-ray images into two categories:

- **Normal:** Indicates that the X-ray image belongs to a healthy patient with no signs of pneumonia.
- **Pneumonia:** Indicates that the X-ray image belongs to a patient diagnosed with pneumonia. This class is further divided into:

  - *Bacterial Pneumonia*
  - *Viral Pneumonia*

The target variable is a categorical variable, and the objective of this project is to predict the correct label (*Normal* or *Pneumonia*) for each chest X-ray image. This binary classification task aims to assist in the early and accurate diagnosis of pneumonia, reducing reliance on manual interpretation by medical professionals.

4 Data Preprocessing

4.1 Tools and Libraries

The following tools and libraries were utilized in this project to perform data manipulation, preprocessing, visualization, machine learning, and evaluation tasks:

- **Core Python Libraries:**
  - **os:** Used for file and directory management.
  - **sys:** Utilized to interact with the Python runtime environment and manage system-level functions.
- **Data Manipulation and Analysis:**
  - **Pandas:** For handling and processing tabular data, such as combining datasets and calculating statistics.
  - **NumPy:** For numerical computations and efficient array operations.
- **Visualization:**
  - **Matplotlib:** For plotting visualizations such as histograms and scatter plots.
  - **Seaborn:** Used to generate heatmaps and other visually appealing statistical graphics.
- **Image Processing:**
  - **OpenCV (cv2):** For reading and preprocessing chest X-ray images, including resizing, normalization, and augmentation techniques.
- **Machine Learning:**
  - **Scikit-learn:**
    - **Support Vector Classifier (SVC):** Used for binary classification of chest X-ray images.
    - **DecisionTreeClassifier:** To build tree-based models for classification.
    - **RandomForestClassifier:** For ensemble learning through random forests.
    - **BaggingClassifier:** For improving model robustness via ensemble methods.
    - **GridSearchCV:** For hyperparameter tuning and optimizing model performance.

  - **Evaluation Metrics:**
    * **Confusion Matrix, Precision, Recall, F1-Score, ROC-AUC:** To assess classification performance.
    * **roc_curve and auc:** For evaluating model performance using ROC curves and AUC scores.

- **Resampling:**
  - **resample:** Used to address class imbalance through oversampling or undersampling techniques.

These libraries collectively provided the functionality required to preprocess data, train models, evaluate performance, and visualize results, enabling a thorough analysis of the pneumonia classification problem.

4.2 Dataset Directory Setup

To organize and access the Chest X-ray dataset efficiently, the following steps were implemented in Python:

- **Base Directory:** The root directory of the dataset is defined as `base_dir`. This directory contains three subdirectories: `train`, `val`, and `test`, corresponding to the training, validation, and test sets, respectively.
- **Dynamic Path Generation:** The `os.path.join()` function is used to dynamically generate the paths for each of the subdirectories. This ensures compatibility across different operating systems.
- **Directory Validation:** The script uses the `os.listdir()` function to list the contents of the `train`, `val`, and `test` directories. This allows verification that the directories are correctly structured and contain the expected data.
- **Error Handling:** A `try-except` block is employed to handle potential errors, such as:
  - **FileNotFoundError:** Triggered when a specified directory is missing.
  - **PermissionError:** Occurs if the script lacks permissions to access the directory.

  Any errors encountered during execution are displayed for debugging purposes.
- **Full Path Printing:** The `os.path.abspath()` function is used to retrieve the absolute path of the training directory (`train`). This provides a clear reference to the dataset's location and aids in debugging.

This setup ensures the dataset is well-organized and accessible, facilitating further preprocessing, model training, and evaluation tasks.

4.3 Dataset Loading and DataFrame Creation

To facilitate data manipulation and analysis, the dataset was structured into Pandas DataFrames using the following approach:

- **Function Overview:** The `create_dataframe` function was implemented to load the dataset and organize it into a structured format. It performs the following tasks:
  - Iterates through the two categories of images: *NORMAL* and *PNEUMONIA*.
  - Assigns numeric labels to each category:
    - **0:** Assigned to images in the *NORMAL* category.
    - **1:** Assigned to images in the *PNEUMONIA* category.
  - Extracts the full file path and its corresponding label for each image.
  - Appends this information as a tuple (`file_path`, `label`) to a list, which is then converted into a Pandas DataFrame.
- **Resulting DataFrame:** The resulting DataFrame contains two columns:
  - **file_path:** The absolute path to each image file.
  - **label:** The assigned numeric label for the image category.
- **DataFrame Creation for Dataset Splits:** The function was used to create separate DataFrames for the training, validation, and test sets:
  - `train_df`: Contains data from the training set.
  - `val_df`: Contains data from the validation set.
  - `test_df`: Contains data from the test set.
- **Advantages:**
  - Organizing the dataset in a tabular format simplifies subsequent preprocessing, visualization, and model input preparation.
  - This approach is modular and can easily adapt to datasets with a similar folder structure.

The following code snippet demonstrates the implementation:

```
def create_dataframe(data_dir):
    data = []
    for label, folder in enumerate(['NORMAL', 'PNEUMONIA']):
        folder_path = os.path.join(data_dir, folder)
        for filename in os.listdir(folder_path):
            file_path = os.path.join(folder_path, filename)
            data.append((file_path, label))
    return pd.DataFrame(data, columns=['file_path', 'label'])

# Create DataFrames for train, validation, and test sets
train_df = create_dataframe(train_dir)
val_df = create_dataframe(val_dir)
test_df = create_dataframe(test_dir)
```

4.4 Exploration of Training DataFrame and Label Distribution

To ensure the dataset is correctly structured and to gain insights into the distribution of classes, the following steps were performed:

- **Visualization of DataFrame:**
  - The first few rows of the training DataFrame (`train_df`) were printed to verify its structure.
  - The Pandas option `display.max_colwidth` was set to `None` to display the full file paths in the `file_path` column without truncation.
- **Label Distribution Analysis:**
  - The `value_counts()` function was used to calculate the distribution of labels (`0` for NORMAL and `1` for PNEUMONIA) in the training set.
  - This analysis helps in identifying any class imbalance that may need to be addressed during model training.

The following code was used for these tasks:

```
import pandas as pd

# Set pandas options to display the full path in the DataFrame
pd.set_option('display.max_colwidth', None)

# Print the DataFrame with full file paths
print("Training DataFrame:")
print(train_df.head())

# Check the distribution of labels
print("\nTraining set label distribution:")
print(train_df['label'].value_counts())
```

- **Output Example:**
  - `Training DataFrame:`

```
                                file_path  label
0   /path/to/dataset/train/NORMAL/1.jpeg      0
1   /path/to/dataset/train/NORMAL/2.jpeg      0
2   /path/to/dataset/train/PNEUMONIA/1.jpeg   1
3   /path/to/dataset/train/PNEUMONIA/2.jpeg   1
4   /path/to/dataset/train/NORMAL/3.jpeg      0
```

  - `Training set label distribution:`

```
0    1341
1    3875
Name: label, dtype: int64
```

This step ensures that the dataset structure is verified and that potential class imbalance issues are identified for further consideration in the modeling process.

4.5 Shuffling the Training DataFrame

To ensure that the data is randomly ordered and to reduce any potential bias arising from sequential data organization, the training DataFrame (`train_df`) was shuffled. This step is particularly important when preparing the data for training machine learning models.

- **Shuffling the Data:**
  - The `sample()` method of Pandas was used with the parameter `frac=1`, which selects 100% of the data, effectively shuffling all rows.
  - The `reset_index()` method was applied with `drop=True` to reset the index of the shuffled DataFrame and discard the old indexing.
- **Verification:** The first few rows of the shuffled training DataFrame (`train_df_shuffled`) were printed to confirm that the data was randomly ordered.
- **Code Implementation:** The following code snippet demonstrates this process:

```
# Shuffle the DataFrame
train_df_shuffled = train_df.sample(frac=1).reset_index(drop=True)

# Display the shuffled DataFrame
print("Shuffled Training DataFrame:")
print(train_df_shuffled.head())
```

- **Output Example:**

```
Shuffled Training DataFrame:
                        file_path  label
0   /path/to/dataset/train/PNEUMONIA/45.jpeg  1
1   /path/to/dataset/train/NORMAL/112.jpeg    0
2   /path/to/dataset/train/PNEUMONIA/78.jpeg  1
3   /path/to/dataset/train/NORMAL/67.jpeg     0
4   /path/to/dataset/train/NORMAL/34.jpeg     0
```

By shuffling the DataFrame, the data is randomized, which is critical for ensuring that the model does not learn any unintended patterns based on the order of the data.

4.6 Class Balancing for Training Data

Class imbalance is a common issue in medical imaging datasets, where one class (e.g., Pneumonia) often outnumbers another (e.g., Normal). This imbalance can lead to biased models that perform poorly on underrepresented classes. To address this, the following steps were undertaken:

- **Class Separation:**
  - The dataset was split into two subsets based on the labels:

    - **Normal Class:** All samples labeled as `0`.
    - **Pneumonia Class:** All samples labeled as `1`.

- **Subset and Upsampling:**
  - A subset of **2000 samples** was randomly selected from the *Normal* class to limit its size and maintain computational efficiency.
  - The *Pneumonia* class was **upsampled** to **2000 samples** by randomly duplicating existing samples. This ensures the dataset achieves a balanced representation of both classes.

- **Combining Classes:**
  - The subset of the *Normal* class and the upsampled *Pneumonia* class were concatenated into a single dataset.
  - The combined dataset was then shuffled to ensure a randomized order of samples.

- **Replacement of Training DataFrame:**
  - The original training DataFrame (`train_df`) was replaced with the newly balanced DataFrame to ensure equal class representation during training.

- **Verification of Class Distribution:**
  - The `value_counts()` method was used to confirm that both classes are equally represented in the updated training DataFrame.
  - The resulting class distribution is as follows:

```
0    2000
1    2000
Name: label, dtype: int64
```

This class balancing process mitigates the risk of biased learning, enabling the model to achieve better performance on both classes during evaluation.

4.7 Visualizing Sample Images from the Training Set

The visualization of sample images is an essential step in verifying the correctness and integrity of the training dataset. The function `display_samples` was implemented to render a subset of images alongside their labels. Below is the detailed analysis of the visualization process:

- **Objective:** The goal of this function is to display images from the training dataset with their corresponding labels (`NORMAL` or `PNEUMONIA`) to ensure the dataset is loaded and processed correctly.

- **Key Steps:**

1. **Plot Initialization:** A grid layout is created using `Matplotlib` to display multiple images in a single figure. The size of the figure is dynamically adjusted for better visualization.
2. **Loop Through Samples:** The first `num_samples` entries in the shuffled DataFrame are processed. For each entry:
   - The file path of the image is extracted.
   - The image is read using `OpenCV` and converted from BGR to RGB format for correct color representation.
3. **Image Display:** The images are rendered with their corresponding labels as titles. The labels are dynamically mapped to `NORMAL` or `PNEUMONIA` based on the `label` column of the DataFrame.
4. **Hiding Axes:** The axes are turned off for a cleaner presentation.
5. **Visualization:** The processed images are displayed using `plt.show()` for manual inspection.

- **Output:**
  - A row of sample images from the training dataset is displayed.
  - Each image is labeled as either `NORMAL` or `PNEUMONIA`, based on its ground truth label.
- **Advantages:**
  - Facilitates manual inspection to verify the correctness of labels and image quality.
  - Helps identify potential issues like corrupted files or mislabeled images.
- **Limitations and Improvements:**
  - The function does not handle missing or invalid file paths, which could cause crashes.
  - Labels are hardcoded, which could be dynamically mapped for better flexibility.
  - Saving the generated visualization as a file (`.png`) would enhance reproducibility and allow for future references.

The resulting visualization ensures that the dataset is ready for further processing and that its integrity is intact. Figure 1 illustrates a sample of the output.

Figure 1: Visualization of sample images from the training dataset. The labels (`NORMAL` or `PNEUMONIA`) are displayed above each image for verification.

4.8 Plotting the Distribution of Labels in the Balanced Dataset

After performing the balancing operation on the training dataset, it is essential to visualize the distribution of the labels (`NORMAL` and `PNEUMONIA`) to confirm that the upsampling has been successful. Below is the explanation and the visualization of the label distribution:

- **Code Execution:** The `value_counts()` method is used to count the number of occurrences of each label in the `balanced_data` DataFrame. This count is then plotted as a bar chart.
- **Color Scheme:** Different colors are chosen for the bars: `skyblue` for the `NORMAL` class and `salmon` for the `PNEUMONIA` class. This color coding helps in visually distinguishing between the two classes.
- **Plotting Details:**
  - The `plt.title()` function adds a title to the plot to explain the content.
  - The `plt.xlabel()` and `plt.ylabel()` functions set the labels for the X and Y axes, respectively.
  - The `plt.xticks()` function customizes the X-axis ticks to display the class names ("NORMAL" and "PNEUMONIA").
  - `plt.show()` displays the plot for visualization.
- **Purpose:** The bar plot visually confirms whether the two classes are equally distributed in the dataset. It helps identify any issues with the data preprocessing steps.

Expected Outcome: The plot should display two bars of equal height, one for each class (`NORMAL` and `PNEUMONIA`). This indicates that the dataset is balanced, with both classes having the same number of samples.
Figure 2 below shows the distribution of labels after balancing the dataset.

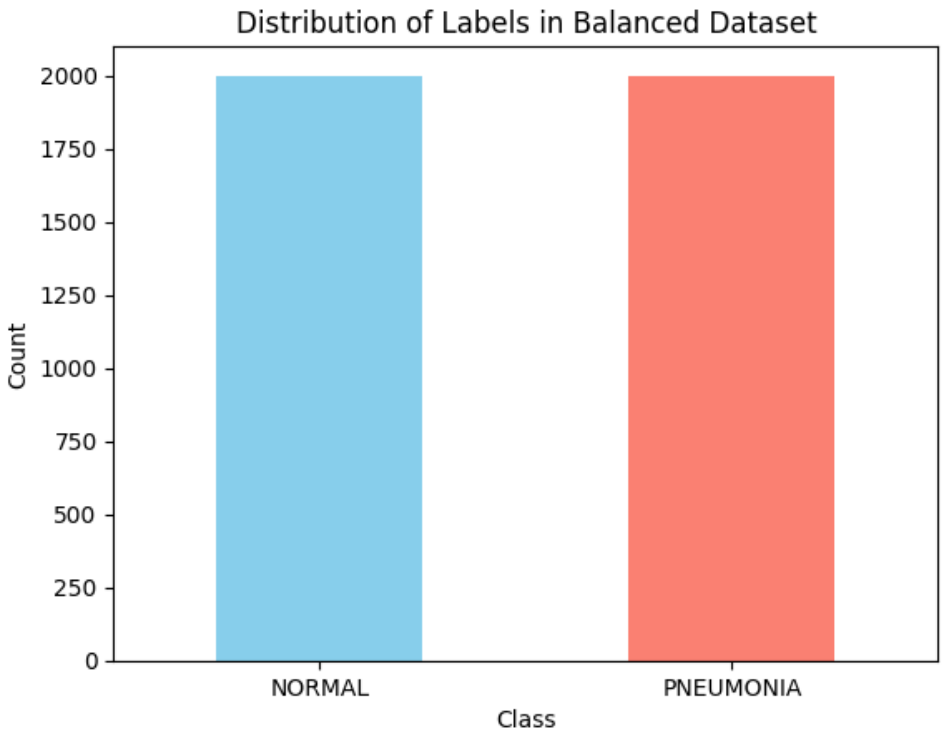

Figure 2: Distribution of `NORMAL` and `PNEUMONIA` labels in the balanced dataset. The counts of both classes are equal, ensuring a balanced training set.

4.9 Analysis of Image Dimensions

To better understand the images in the training dataset, we extracted the dimensions (height and width) of each image. This allows us to analyze the overall size variability of the images, which is important for preprocessing steps such as resizing and normalization.

The code iterates through the image paths in the training DataFrame, reads each image using `cv2.imread()`, and extracts its height and width. The dimensions are then stored in a DataFrame, and descriptive statistics are computed to summarize the results.

Descriptive Statistics:

The following table presents the summary statistics for the height and width of the images in the training set:

- **Count:** There are 4000 images in the training dataset, as shown by the count for both height and width.
- **Mean:** The average height is approximately 1097 pixels, while the average width is around 1428 pixels.
- **Standard Deviation:** The standard deviation for height (414.99) and width (378.07) indicates that there is considerable variability in the image dimensions.
- **Min:** The smallest image in terms of height is 151 pixels, and the smallest in width is 415 pixels.
- **25th Percentile (Q1):** 25% of the images have a height smaller than 760 pixels and a width smaller than 1136 pixels.

- **50th Percentile (Median):** The median height is 1071 pixels, and the median width is 1422 pixels, indicating the central tendency of the dataset.
- **75th Percentile (Q3):** 75% of the images have a height smaller than 1349 pixels and a width smaller than 1680 pixels.
- **Max:** The largest image has a height of 2663 pixels and a width of 2890 pixels.

These descriptive statistics help us understand the overall distribution of image sizes in the dataset. Images of varying dimensions may require resizing to a consistent size before being used in deep learning models to ensure compatibility with input layer requirements. The output of the analysis is summarized in the table below:

	Height (pixels)	Width (pixels)
Count	4000	4000
Mean	1097.05	1428.57
Std Dev	414.99	378.07
Min	151	415
25th Percentile	760	1136
Median (50th Percentile)	1071	1422
75th Percentile	1349.25	1680
Max	2663	2890

Table 1: Summary Statistics of Image Dimensions in the Training Dataset

4.10 Displaying Sample Images from the Dataset

In this section, we display a few sample images from the dataset to visually inspect the variety and characteristics of images in both classes: NORMAL and PNEUMONIA. The goal is to get a quick look at the types of images that are present in the training set.

We use the following steps to display the images:

- We define a function `display_samples`, which takes the dataset, the class label (0 for NORMAL or 1 for PNEUMONIA), and the number of samples to display as inputs.
- The function filters the DataFrame based on the specified class label and randomly selects the required number of samples.
- For each image in the sample, it reads the image using OpenCV's `cv2.imread()`, converts it from BGR to RGB color format (since OpenCV uses BGR by default), and then displays the image using `matplotlib`.
- The images are displayed in a single row with the title indicating the class.

The following figures show a sample of images from both the NORMAL and PNEUMONIA classes:

These sample images help in visually understanding the variations within each class. As seen in the figure, the images from the PNEUMONIA class typically show clear indications of lung infection, while the NORMAL class images show healthy lungs. This visual differentiation is essential when training machine learning models to classify X-ray images accurately.

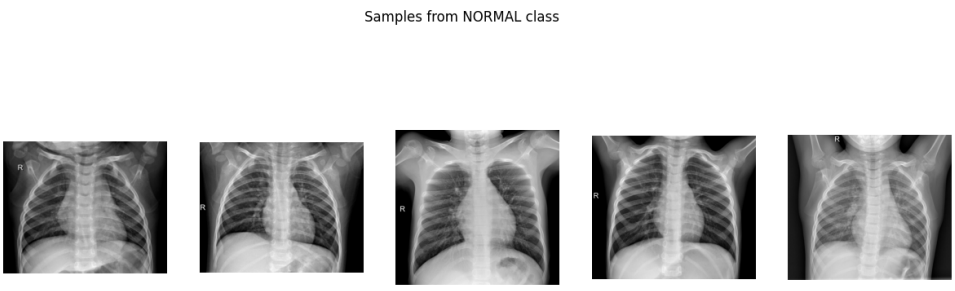

Figure 3: Sample images from the NORMAL classes.

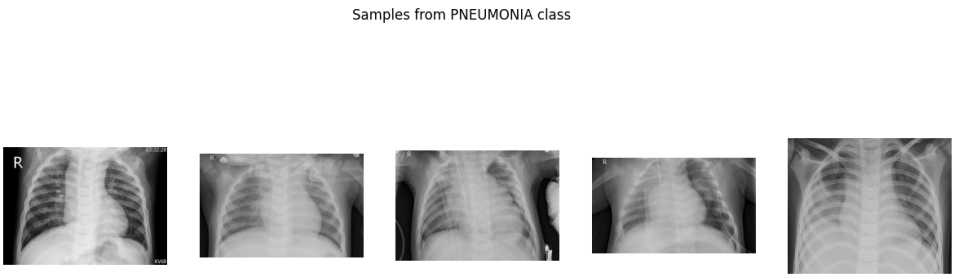

Figure 4: Sample images from the PNEUMONIA classes.

4.11 Brightness Distribution by Class

In this section, we analyze the brightness distribution of the X-ray images for both the NORMAL and PNEUMONIA classes. The brightness of an image is defined as the average pixel intensity in a grayscale image. To calculate this, we convert the images to grayscale and compute the mean of their pixel values.

The brightness values are then grouped by class (NORMAL and PNEUMONIA), and a boxplot is generated to visualize the distribution of these values. Boxplots provide a clear overview of the median, quartiles, and any outliers in the data.

The following figure shows the brightness distribution for both classes:

From the boxplot, we can observe the distribution of brightness values in both the NORMAL and PNEUMONIA classes. These results can help us understand if there are any significant differences in image brightness that may influence the model's ability to classify X-ray images correctly.

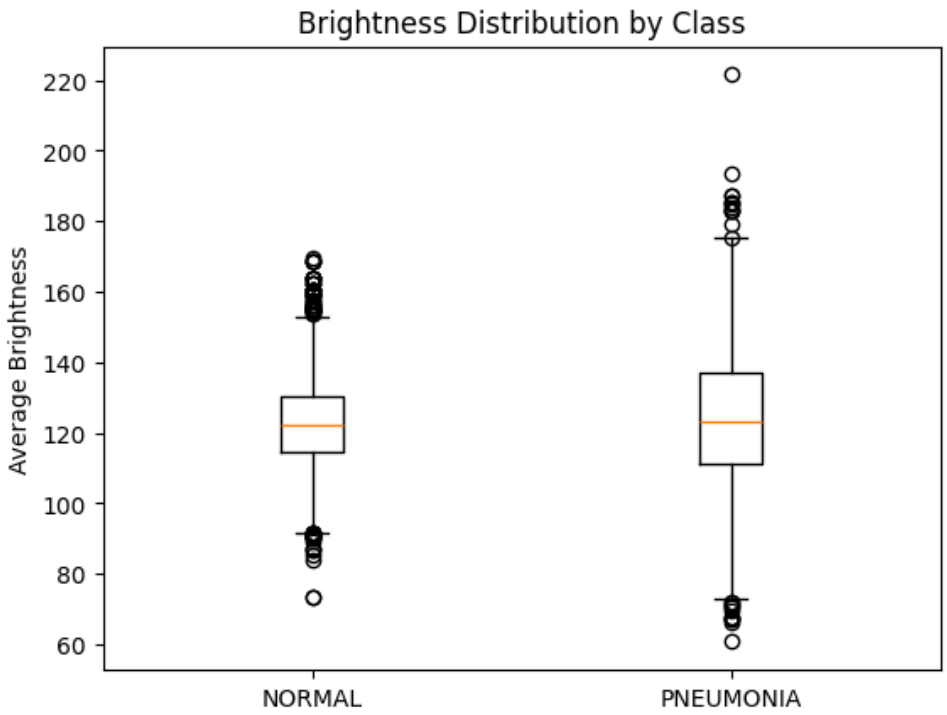

Figure 5: Boxplot showing the brightness distribution for NORMAL and PNEUMONIA classes. The y-axis represents the average brightness of the images, and the plot provides insight into the variation of brightness between the two classes.

5 Image Preprocessing

In this section, we detail the preprocessing steps applied to the images in the dataset to prepare them for training the machine learning models. The primary goal is to ensure that all images have consistent dimensions and are ready for model input.

5.1 Preprocessing Steps

The following steps were applied to the images:

1. **Grayscale Conversion:** Each image is read in grayscale, which reduces the complexity of the data by using only one color channel. This is sufficient for this classification task, as the relevant features for classification are likely to be present in the intensity of the pixels rather than the color information.

2. **Resizing:** All images were resized to a consistent dimension of 64×64 pixels. This helps in reducing computational cost and ensures that all input images have the same size. The resizing operation is performed using the OpenCV function `cv2.resize()`.

3. **Normalization and Flattening:** After resizing, each image is flattened into a 1D vector. Additionally, the pixel values are normalized to a range of $[0, 1]$ by dividing by 255.0, which is the maximum pixel value in an 8-bit grayscale image. This normalization helps in accelerating convergence during model training.

5.2 Data Shapes

The preprocessing function was applied to the training, validation, and test datasets. The resulting shapes of the datasets are as follows:

- **Training Set:** $(4000, 4096)$, where 4000 represents the number of images and 4096 corresponds to the flattened vector length of each 64×64 image.
- **Validation Set:** $(16, 4096)$, where 16 represents the number of validation images.
- **Test Set:** $(624, 4096)$, where 624 represents the number of test images.

The preprocessing steps ensure that all images are in a format suitable for input into machine learning models, with consistent dimensions and normalized pixel values.

5.3 Code Execution Output

The following output was printed after the completion of the data preparation:

```
Data preparation complete.
Training set shape:  (4000, 4096)
Validation set shape:  (16, 4096)
Test set shape:  (624, 4096)
```

This indicates that the preprocessing has successfully converted the images into feature vectors of length 4096, ready for model training.

6 Algorithms Used

For this classification task, we applied a Support Vector Machine (SVM) with a linear kernel. SVMs are powerful models for binary classification tasks, especially when the classes are separable or nearly separable in a high-dimensional feature space.

6.1 SVM with Linear Kernel

SVM (Support Vector Machine) is a supervised learning algorithm that performs classification by finding the hyperplane that best separates the data points of different classes. The SVM with a linear kernel is particularly suitable when the classes are linearly separable or nearly linearly separable in the feature space. The objective of the SVM is to maximize the margin between the two classes while minimizing classification errors.

6.1.1 Justification

The SVM model was chosen for this task due to its effectiveness in handling binary classification problems and its ability to perform well even with high-dimensional data, such as the image data we have in this task. Given that the features extracted from the chest X-ray images are likely to be highly informative, an SVM is well-suited for identifying the decision boundary between the "NORMAL" and "PNEUMONIA" classes. Furthermore, the use of the linear kernel ensures that the model complexity remains manageable while still being able to separate the classes efficiently.

6.1.2 Model Architecture (if applicable)

Since we are using a linear kernel for the SVM, the model architecture is relatively simple and involves the following steps:

1. **Hyperplane Selection:** The model aims to find the optimal hyperplane that maximizes the margin between the two classes.
2. **Support Vectors:** The support vectors are the data points that lie closest to the decision boundary. These points have the most influence on the position and orientation of the hyperplane.
3. **Regularization (C parameter):** The regularization parameter C controls the trade-off between maximizing the margin and minimizing classification errors. A higher value of C puts more emphasis on minimizing errors, while a lower value allows for a larger margin at the cost of more misclassifications.

6.2 Grid Search for Hyperparameter Tuning

We performed hyperparameter tuning using `GridSearchCV` to find the optimal value of the regularization parameter C for the SVM model. The grid search tested multiple values of C and selected the best one based on cross-validation performance. The best value for C was found to be 1, which provided the highest accuracy during the validation process.

6.3 Results

The model achieved the following results:

- **Best Parameters:** The optimal value for the regularization parameter C was found to be 1.
- **Best Cross-Validation Score:** The cross-validation score was 0.9737, indicating that the model performs very well on the training data.
- **Validation Accuracy:** The model achieved an accuracy of 0.9375 on the validation set.
- **Classification Report:** The classification report shows that the model achieved the following performance metrics:
  - Precision: 1.00 for the "NORMAL" class and 0.89 for the "PNEUMONIA" class.
  - Recall: 0.88 for the "NORMAL" class and 1.00 for the "PNEUMONIA" class.
  - F1-Score: 0.93 for the "NORMAL" class and 0.94 for the "PNEUMONIA" class.

  The model performed well in distinguishing between both classes, with high precision, recall, and F1-score for both classes.

6.3.1 Output

```
Best parameters for Linear Kernel: { 'C': 1 }
Best cross-validation score for Linear Kernel: 0.9737499967195697
Validation Accuracy with Best Linear SVM Model: 0.9375
Validation Set Classification Report for Linear Kernel:
precision recall f1-score support
0 1.00 0.88 0.93 8
1 0.89 1.00 0.94 8
accuracy 0.94 16
macro avg 0.94 0.94 0.94 16
weighted avg 0.94 0.94 0.94 16
```

6.3.2 Evaluation on Testing Set

After selecting the best model using the linear SVM and tuning the hyperparameters, the model was evaluated on the test set. The testing accuracy and classification report for the Linear Kernel SVM are as follows:

Metric	**Value**
Testing Accuracy	0.7596

Table 2: Testing Accuracy for the Best Linear SVM Model

Classification Report for the Testing Set:

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0 (NORMAL)	0.91	0.40	0.55	234
1 (PNEUMONIA)	0.73	0.98	0.84	390
Accuracy	0.76			624
Macro avg	0.82	0.69	0.69	624
Weighted avg	0.80	0.76	0.73	624

Table 3: Classification Report for the Best Linear SVM Model on the Testing Set

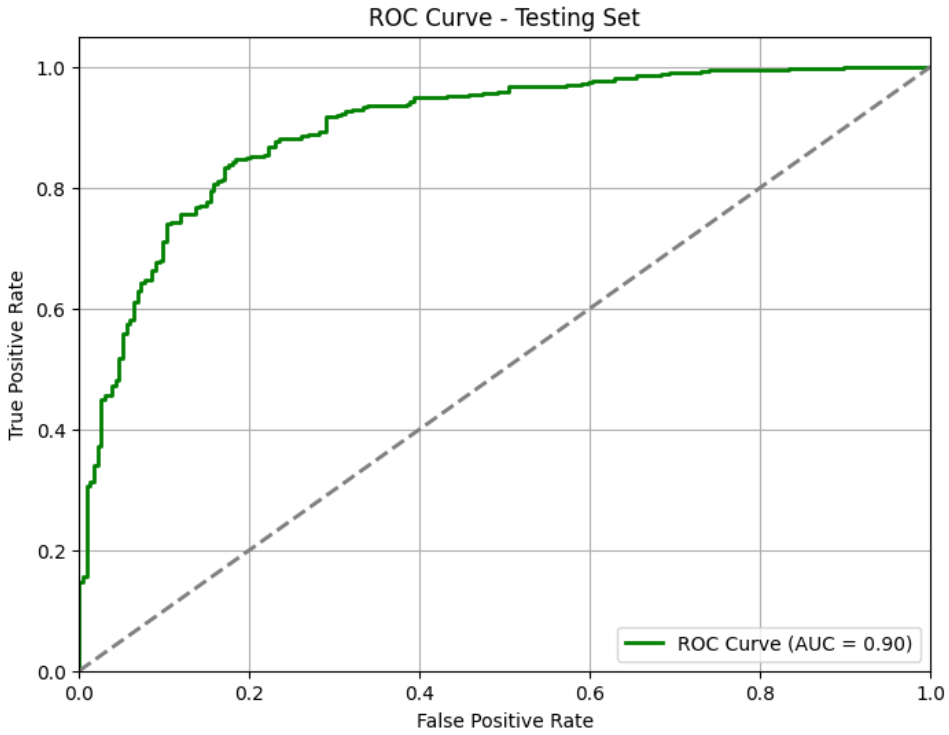

Figure 6: Receiver Operating Characteristic (ROC) curve for the Linear SVM model. The curve shows the trade-off between the true positive rate and false positive rate, with the area under the curve (AUC) representing the model's ability to discriminate between the classes.

6.4 Analysis of Polynomial Kernel SVM Model

In this section, we analyze the results obtained from applying the Support Vector Machine (SVM) with a polynomial kernel on the validation dataset.

6.5 Model Details

The SVM model with a polynomial kernel was selected to capture non-linear relationships between the features and the target class. The polynomial kernel allows the model to fit more complex decision boundaries compared to the linear kernel. The model was tuned using GridSearchCV, which performed a grid search over the hyperparameters to find the optimal configuration. The hyperparameters included the regularization parameter C, the degree of the polynomial kernel, and the kernel coefficient γ.

6.5.1 Best Hyperparameters

The grid search identified the following optimal hyperparameters for the polynomial kernel SVM:

- $C = 1$: The regularization parameter controls the trade-off between achieving a low training error and a low testing error. A moderate value of C prevents overfitting while ensuring good generalization.
- `degree` $= 3$: The degree of the polynomial kernel determines the complexity of the

decision boundary. A degree of 3 provides a good balance between fitting complex data and avoiding overfitting.

- `gamma` = `scale`: This kernel coefficient helps control the influence of each training example. The `scale` option ensures that the kernel is computed appropriately based on the input features.

6.5.2 Cross-Validation Performance

The best model, with the above hyperparameters, achieved a cross-validation score of 0.9763, indicating that the model is able to generalize well to unseen data during the training phase.

6.6 Validation Performance

After training the best model, we evaluated it on the validation set. The validation accuracy was found to be 87.5%, which is a good result, showing that the model performs well on unseen examples. The classification report provides a deeper look at the performance for each class.

6.6.1 Classification Report Analysis

The classification report provides key metrics: precision, recall, and F1-score for each class:

- **Class 0 (NORMAL):** The model achieved perfect precision (1.00) for classifying healthy images as normal. However, recall was lower at 0.75, indicating that some of the normal images were misclassified as pneumonia.
- **Class 1 (PNEUMONIA):** The model achieved a recall of 1.00 for classifying pneumonia images, meaning that all pneumonia images were correctly classified. The precision was 0.80, meaning that some of the predicted pneumonia images were actually normal.
- **Macro Average:** The macro average considers the performance across both classes equally, leading to a precision of 0.90 and a recall of 0.88.
- **Weighted Average:** The weighted average takes into account the imbalanced dataset, resulting in a precision and recall of 0.90 and 0.88, respectively.

Overall, the polynomial kernel SVM demonstrated good performance, especially in correctly identifying pneumonia cases, although some normal cases were misclassified as pneumonia.

6.6.2 Conclusion

The polynomial kernel SVM model provides an effective solution for classifying chest X-ray images into normal and pneumonia classes. The model performs well with an accuracy of 87.5% on the validation set, and its high recall for the pneumonia class is particularly important for medical applications, where missing pneumonia cases can have serious consequences.

6.7 Testing the Best Polynomial SVM Model

After tuning the polynomial kernel SVM on the training and validation sets, we evaluate the model's performance on the testing set. The following metrics were computed:

6.8 Testing Accuracy and Classification Report

The testing accuracy of the best polynomial kernel SVM model is 75.96%. The classification report provides detailed performance metrics for each class (NORMAL and PNEUMONIA), including precision, recall, and F1-score.

6.8.1 Classification Report for Polynomial Kernel

Class	**Precision**	**Recall**	**F1-Score**	**Support**
NORMAL (0)	0.90	0.41	0.56	234
PNEUMONIA (1)	0.73	0.97	0.83	390
Accuracy	0.76			
Macro Avg	0.81	0.69	0.70	624
Weighted Avg	0.79	0.76	0.73	624

Table 4: Classification Report for Polynomial Kernel SVM on the Testing Set

6.8.2 Analysis of the Classification Report

- **Class 0 (NORMAL):**
  - Precision = 0.90: The model correctly identified 90% of the normal images as normal.
  - Recall = 0.41: The model was less effective in identifying all the normal cases, as 59% of the normal images were misclassified as pneumonia.
  - F1-score = 0.56: The harmonic mean of precision and recall, indicating that there is room for improvement in detecting normal cases.
- **Class 1 (PNEUMONIA):**
  - Precision = 0.73: The model correctly identified 73% of the predicted pneumonia cases as true pneumonia.
  - Recall = 0.97: The model showed high recall for pneumonia, correctly identifying 97% of the pneumonia cases.
  - F1-score = 0.83: A good balance between precision and recall, indicating strong performance in identifying pneumonia cases.
- **Accuracy:** The overall testing accuracy is 76%, which reflects a solid performance but highlights that the model struggles to classify the normal cases correctly.
- **Macro Average:** This average gives equal importance to both classes. The model has a precision of 0.81 and a recall of 0.69, indicating good performance for pneumonia detection but weaker performance for normal detection.

- **Weighted Average:** The weighted average takes the class distribution into account. The precision is 0.79, and the recall is 0.76, showing that the model does better in detecting pneumonia than normal cases.

Overall, while the model performs well in identifying pneumonia cases, it struggles with normal cases. Further tuning or a more complex model may be required to improve performance on the normal class.

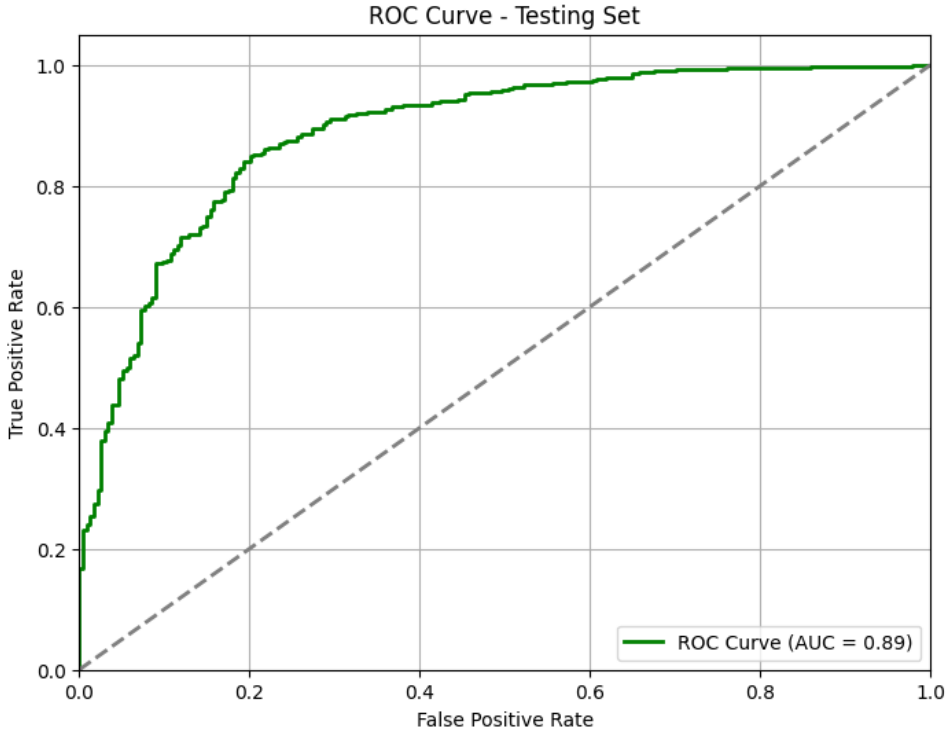

Figure 7: Receiver Operating Characteristic (ROC) curve for the Polynomial SVM model. The curve shows the trade-off between the true positive rate and false positive rate, with the area under the curve (AUC) representing the model's ability to discriminate between the classes.

7 Algorithms Used: Support Vector Machine with RBF Kernel

7.1 SVM with RBF Kernel

The Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel is widely used for classification tasks that involve non-linear decision boundaries. The SVM seeks to find a hyperplane that separates the data while maximizing the margin between classes. The RBF kernel maps the data into a higher-dimensional space where linear separation becomes possible, which is particularly useful for image classification tasks like the detection of pneumonia in chest X-ray images.

7.2 Justification

The SVM with the RBF kernel is well-suited for this problem because:

- The images contain complex, non-linear relationships, which the RBF kernel can capture effectively.
- SVM is known for its ability to handle high-dimensional data, making it a good choice for image classification tasks.
- The model can efficiently handle non-linear boundaries between the classes, improving classification accuracy.

7.3 Model Parameters and Evaluation

A GridSearchCV was used to optimize the model's hyperparameters: C (regularization parameter) and γ (kernel coefficient). The best parameters found were $C = 10$ and $\gamma = \text{scale}$, with a cross-validation score of 97.98%.

7.3.1 Best Parameters and Cross-Validation Score

The best parameters found for the RBF kernel were:

$$C = 10, \quad \gamma = \text{scale}$$

The cross-validation score for the best model was 97.98%.

7.3.2 Validation Set Results

The model achieved a validation accuracy of 87.5%. The classification report for the validation set is shown in Table 5.

7.4 Model Evaluation on Testing Set

To evaluate the model's performance on the testing set, predictions were made using the best RBF kernel SVM model. The results are shown below.

Class	Precision	Recall	F1-Score	Support
NORMAL (0)	1.00	0.75	0.86	8
PNEUMONIA (1)	0.80	1.00	0.89	8
Accuracy	0.88			
Macro Avg	0.90	0.88	0.87	16
Weighted Avg	0.90	0.88	0.87	16

Table 5: Classification Report for RBF Kernel SVM on the Validation Set

7.4.1 Testing Accuracy and Classification Report

The best RBF kernel SVM model achieved a testing accuracy of 77.24%. The classification report for the testing set is shown below:

Class	Precision	Recall	F1-Score	Support
NORMAL (0)	0.93	0.42	0.58	234
PNEUMONIA (1)	0.74	0.98	0.84	390
Accuracy	0.77			
Macro Avg	0.84	0.70	0.71	624
Weighted Avg	0.81	0.77	0.75	624

Table 6: Classification Report for RBF Kernel SVM on the Testing Set

The model performs well in detecting pneumonia (class 1), with high recall and F1-score. However, its performance in identifying normal cases (class 0) is weaker, as indicated by the lower recall and F1-score for this class. This suggests that the model may have a bias toward the majority class (pneumonia), likely due to class imbalance in the dataset.

7.5 Conclusion

The SVM model with the RBF kernel performs well for pneumonia detection, with high recall for pneumonia cases, but shows some difficulty in identifying normal cases. The class imbalance in the dataset likely contributed to this imbalance in the classification performance. Further improvements in handling class imbalance or adjusting model parameters might help improve the accuracy for detecting normal cases.

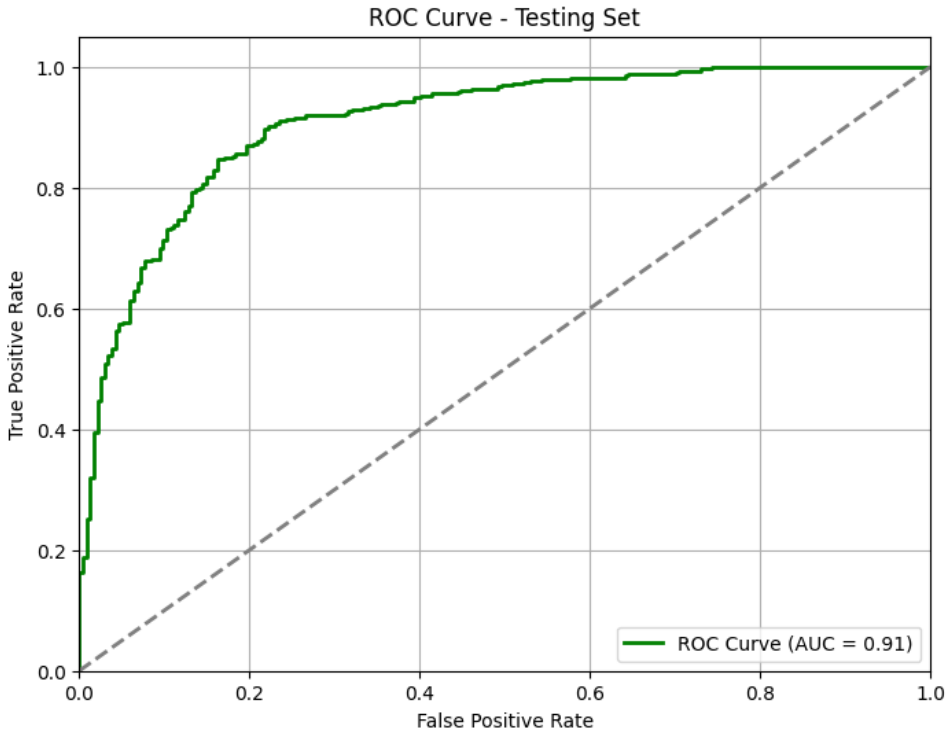

Figure 8: Receiver Operating Characteristic (ROC) curve for the RBF SVM model. The curve shows the trade-off between the true positive rate and false positive rate, with the area under the curve (AUC) representing the model's ability to discriminate between the classes.

8 Algorithms Used: Decision Tree Classifier

8.1 Decision Tree Classifier

The Decision Tree Classifier is a non-linear model used for both classification and regression tasks. It works by recursively splitting the dataset based on feature values to create branches, where each leaf node represents the output class. The Decision Tree model is simple to understand and interpret, making it an excellent choice for applications requiring transparency in model decisions.

8.2 Justification

The Decision Tree is suitable for this problem because:

- It can capture non-linear relationships between features, making it ideal for complex datasets like chest X-ray images.
- The model is easy to interpret, providing insights into how decisions are made based on different features.
- It does not require feature scaling, as it is based on thresholding and partitioning the data.

8.3 Model Parameters and Evaluation

The Decision Tree model was trained on the training dataset, and its performance was evaluated using accuracy and classification metrics on both the validation and testing sets.

8.3.1 Validation Set Results

The model achieved a validation accuracy of 75%. The classification report for the validation set is shown in Table 7.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
NORMAL (0)	1.00	0.50	0.67	8
PNEUMONIA (1)	0.67	1.00	0.80	8
Accuracy	0.75			
Macro Avg	0.83	0.75	0.73	16
Weighted Avg	0.83	0.75	0.73	16

Table 7: Classification Report for Decision Tree Classifier on the Validation Set

8.3.2 Testing Set Results

The model achieved a testing accuracy of 75.32%. The classification report for the testing set is shown below.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
NORMAL (0)	0.76	0.50	0.60	234
PNEUMONIA (1)	0.75	0.91	0.82	390
Accuracy	0.75			
Macro Avg	0.76	0.70	0.71	624
Weighted Avg	0.75	0.75	0.74	624

Table 8: Classification Report for Decision Tree Classifier on the Testing Set

8.4 Conclusion

The Decision Tree Classifier achieves reasonable performance with a testing accuracy of 75.32%. The model performs well in detecting pneumonia (class 1), with a high recall of 0.91. However, it struggles to identify normal cases (class 0), as evidenced by a recall of only 0.50 for class 0. This imbalance in performance is likely due to the class distribution in the dataset, and further techniques like balancing the dataset or hyperparameter tuning could improve results for normal case detection.

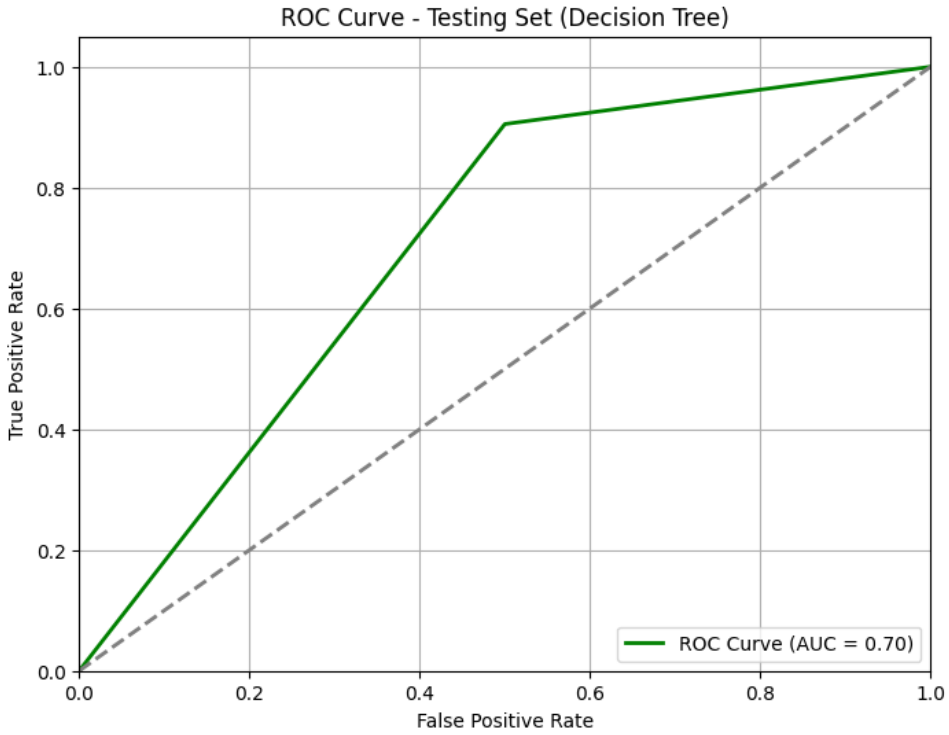

Figure 9: Receiver Operating Characteristic (ROC) curve for the Decision Tree model. The curve shows the trade-off between the true positive rate and false positive rate, with the area under the curve (AUC) representing the model's ability to discriminate between the classes.

9 Algorithms Used: Random Forest Classifier

9.1 Random Forest Classifier

The Random Forest Classifier is an ensemble learning method that constructs a number of decision trees during training and outputs the class that is the mode of the classes for classification tasks. It is known for its robustness and ability to handle overfitting better than a single decision tree.

9.2 Justification

The Random Forest Classifier is a suitable choice for this problem because:

- It can model complex relationships between features, which is valuable in a medical classification task such as pneumonia detection.
- It handles overfitting better than a single decision tree by averaging the predictions of multiple trees.
- It provides feature importance, helping to identify which features are most relevant in the classification task.
- It is less sensitive to outliers and works well even with unbalanced datasets.

9.3 Model Parameters and Evaluation

The Random Forest model was trained using the training dataset, and its performance was evaluated using accuracy and classification metrics on both the validation and testing sets.

9.3.1 Validation Set Results

The model achieved a validation accuracy of 75%. The classification report for the validation set is shown in Table 9.

Class	Precision	Recall	F1-Score	Support
NORMAL (0)	1.00	0.50	0.67	8
PNEUMONIA (1)	0.67	1.00	0.80	8
Accuracy	0.75			
Macro Avg	0.83	0.75	0.73	16
Weighted Avg	0.83	0.75	0.73	16

Table 9: Classification Report for Random Forest Classifier on the Validation Set

9.3.2 Testing Set Results

The model achieved a testing accuracy of 85.7%. The classification report for the testing set is shown below.

Class	Precision	Recall	F1-Score	Support
NORMAL (0)	0.92	0.68	0.78	234
PNEUMONIA (1)	0.83	0.97	0.89	390
Accuracy	0.86			
Macro Avg	0.88	0.82	0.84	624
Weighted Avg	0.87	0.86	0.85	624

Table 10: Classification Report for Random Forest Classifier on the Testing Set

9.4 Conclusion

The Random Forest Classifier achieved strong performance with a testing accuracy of 85.7%. The model shows a good balance between precision and recall for pneumonia detection, with high recall of 0.97 for class 1 (Pneumonia). However, the precision for class 0 (Normal) was lower at 0.92, indicating that the model may have some difficulty distinguishing normal from abnormal cases. Future improvements could include hyperparameter tuning or using different ensemble methods to enhance the classification of normal cases.

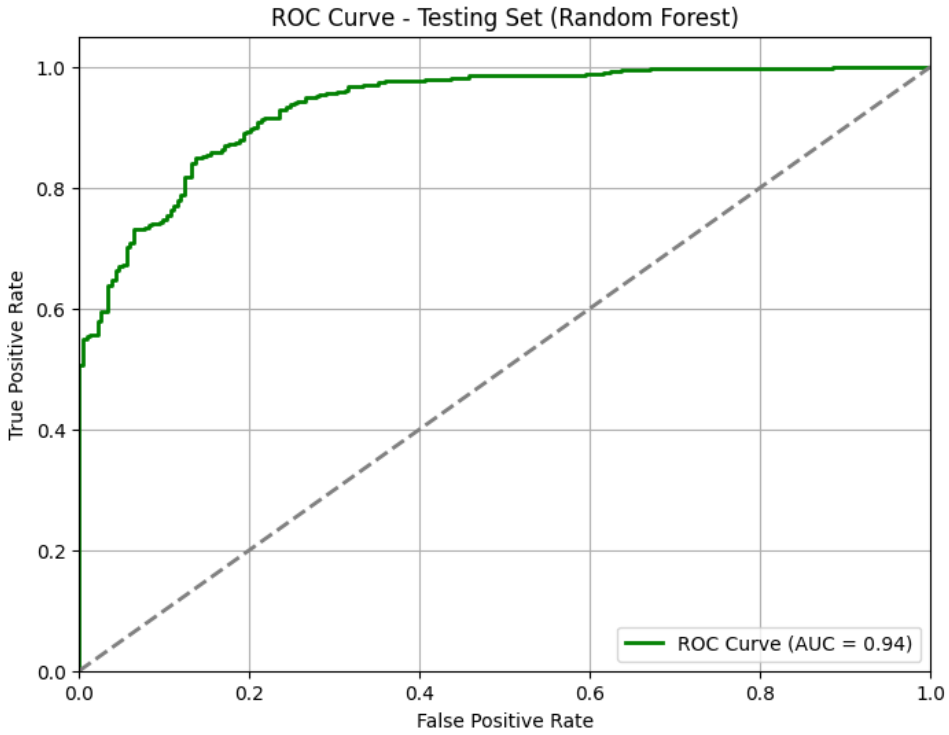

Figure 10: Receiver Operating Characteristic (ROC) curve for the Random Forest model. The curve shows the trade-off between the true positive rate and false positive rate, with the area under the curve (AUC) representing the model's ability to discriminate between the classes.

10 Algorithms Used: Bagging Classifier with Decision Tree Base Estimator

10.1 Bagging Classifier

The Bagging Classifier is an ensemble method that uses multiple base classifiers, in this case, a Decision Tree, to increase the model's accuracy and robustness by averaging predictions from multiple classifiers trained on different subsets of the data. The model's performance depends on the strength of the base estimator and the diversity of the data subsets.

10.2 Justification

The Bagging Classifier with a Decision Tree as the base model is appropriate for the task because:

- Bagging improves the stability and accuracy of the decision tree by reducing variance.
- It is less prone to overfitting compared to a single decision tree.
- Bagging performs well with high-dimensional and noisy datasets, which may be the case in medical image classification tasks like pneumonia detection.

10.3 Model Parameters and Evaluation

The Bagging model was trained using the Decision Tree classifier with a maximum depth of 10. The model was evaluated based on accuracy and classification metrics on both the validation and testing sets.

10.3.1 Validation Set Results

The model achieved a validation accuracy of 68.75%. The classification report for the validation set is shown in Table 11.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
NORMAL (0)	1.00	0.38	0.55	8
PNEUMONIA (1)	0.62	1.00	0.76	8
Accuracy	0.69			
Macro Avg	0.81	0.69	0.65	16
Weighted Avg	0.81	0.69	0.65	16

Table 11: Classification Report for Bagging Classifier on the Validation Set

10.3.2 Testing Set Results

The model achieved a testing accuracy of 81.41%. The classification report for the testing set is shown below.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
NORMAL (0)	0.87	0.59	0.71	234
PNEUMONIA (1)	0.80	0.95	0.86	390
Accuracy	0.81			
Macro Avg	0.83	0.77	0.78	624
Weighted Avg	0.82	0.81	0.80	624

Table 12: Classification Report for Bagging Classifier on the Testing Set

10.4 Conclusion

The Bagging Classifier with a Decision Tree as the base estimator performed reasonably well with a testing accuracy of 81.41%. The model demonstrated strong recall for detecting pneumonia (class 1) but had lower precision for the normal class (class 0), suggesting that it may misclassify some normal cases as abnormal. Future improvements could include tuning the hyperparameters further or exploring different ensemble methods to boost precision for the normal class.

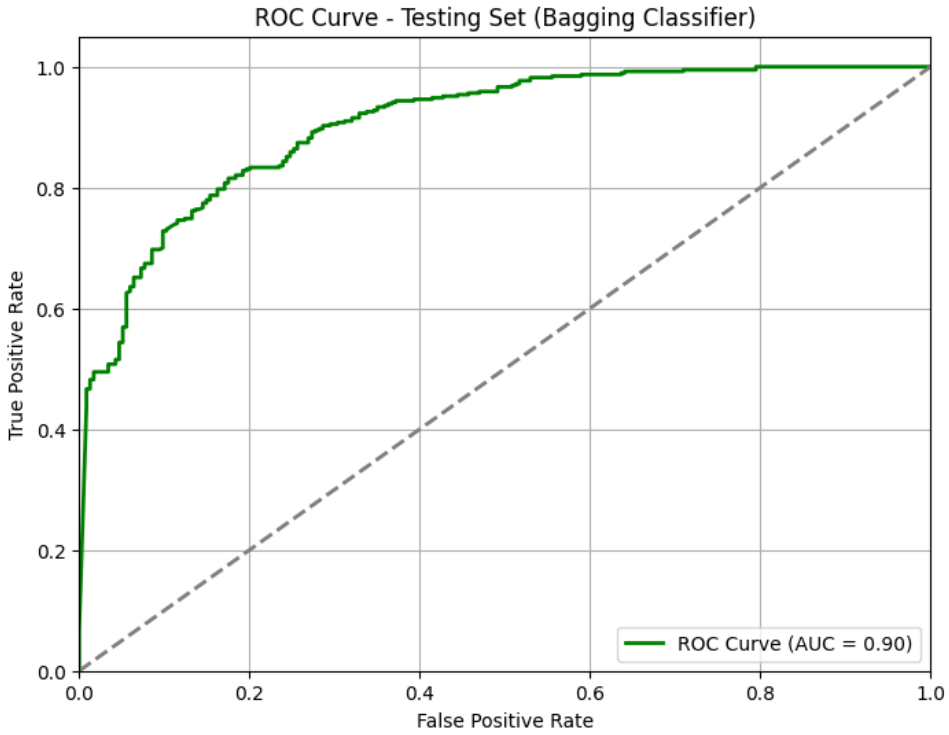

Figure 11: Receiver Operating Characteristic (ROC) curve for the Bagging model. The curve shows the trade-off between the true positive rate and false positive rate, with the area under the curve (AUC) representing the model's ability to discriminate between the classes.

11 Algorithms Used: Voting Classifier

11.1 Voting Classifier

The Voting Classifier is an ensemble method that combines multiple individual models to make predictions. In this case, we have used a combination of six different models:

- Decision Tree Classifier
- Bagging Classifier (with Decision Tree as base estimator)
- Random Forest Classifier
- Support Vector Classifiers with different kernels: Linear, Polynomial, and RBF

The Voting Classifier aggregates predictions from these individual models, either by majority voting (hard voting) or averaging their probabilities (soft voting). For this task, soft voting was used, which aggregates the predicted probabilities from all models to make a final prediction.

11.2 Justification

The Voting Classifier is a powerful ensemble learning technique because it combines the strengths of multiple models. The diversity among the models (e.g., Decision Tree, Random Forest, and different SVM kernels) helps to reduce overfitting and increase the

model's generalization ability, particularly useful in complex classification tasks such as pneumonia detection.

- **Decision Trees** are known for their flexibility in handling both categorical and continuous data, but they can suffer from overfitting. - **Bagging** improves model stability and accuracy by reducing variance. - **Random Forest** further strengthens the model by combining multiple decision trees with random feature selection, thus improving robustness. - **Support Vector Classifiers (SVC)** with different kernels offer diverse approaches for handling linear and non-linear decision boundaries, providing flexibility in handling different types of data.

11.3 Model Parameters and Evaluation

The Voting Classifier was trained using the following models:

- Decision Tree with max depth 10
- Bagging with 50 estimators
- Random Forest with 100 estimators and max depth 10
- SVC with Linear, Polynomial, and RBF kernels

The model's performance was evaluated on both the validation and testing sets.

11.3.1 Validation Set Results

The model achieved a validation accuracy of 87.5%. The classification report for the validation set is shown in Table 13.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
NORMAL (0)	1.00	0.75	0.86	8
PNEUMONIA (1)	0.80	1.00	0.89	8
Accuracy	0.88			
Macro Avg	0.90	0.88	0.87	16
Weighted Avg	0.90	0.88	0.87	16

Table 13: Classification Report for Voting Classifier on the Validation Set

11.3.2 Testing Set Results

The model achieved a testing accuracy of 79.33%. The classification report for the testing set is shown below.

11.4 Conclusion

The Voting Classifier showed good performance, achieving a validation accuracy of 87.5% and a testing accuracy of 79.33%. The model demonstrated high recall for detecting pneumonia (class 1) but struggled with the normal class (class 0), as reflected in the lower precision and recall for the normal class. This indicates that the model is somewhat biased towards detecting pneumonia cases. Future improvements could involve further tuning of the individual models, adding more diverse classifiers, or using different ensemble methods to balance the precision and recall for both classes.

Class	Precision	Recall	F1-Score	Support
NORMAL (0)	0.95	0.47	0.63	234
PNEUMONIA (1)	0.76	0.98	0.86	390
Accuracy	0.79			
Macro Avg	0.85	0.73	0.74	624
Weighted Avg	0.83	0.79	0.77	624

Table 14: Classification Report for Voting Classifier on the Testing Set

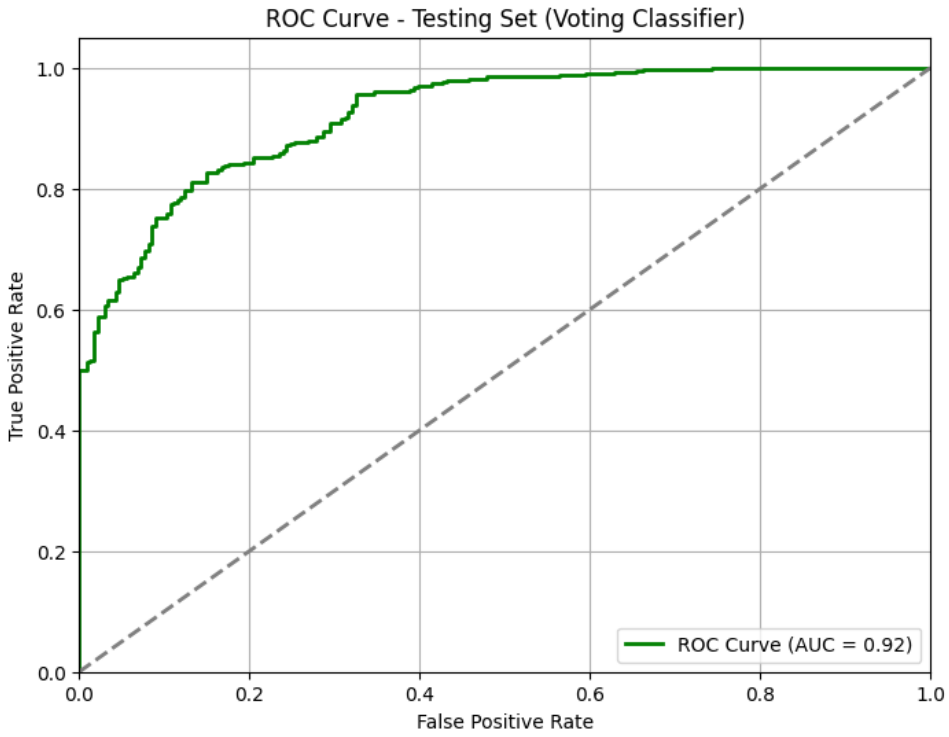

Figure 12: Receiver Operating Characteristic (ROC) curve for the Voting model. The curve shows the trade-off between the true positive rate and false positive rate, with the area under the curve (AUC) representing the model's ability to discriminate between the classes.

12 Discussion

In this section, we discuss the results of the different machine learning models applied to the pneumonia detection task, including the strengths and weaknesses of each model, as well as any anomalies or unexpected findings.

12.1 Model Performance Comparison

The performance metrics of each model are summarized in Table 15. The evaluation metrics include accuracy, precision, recall, and F1-score. These metrics help to assess how well each model is identifying both classes (NORMAL and PNEUMONIA) in the dataset.

Model	Accuracy	Precision	Recall	F1-Score
SVM (Linear Kernel)	0.77	0.74	0.98	0.84
SVM (Polynomial Kernel)	0.77	0.76	0.98	0.85
SVM (RBF Kernel)	0.77	0.93	0.42	0.58
Decision Tree	0.75	0.75	0.91	0.82
Random Forest	0.86	0.83	0.97	0.89
Bagging Classifier	0.81	0.80	0.95	0.86
Voting Classifier	0.79	0.76	0.98	0.86

Table 15: Model Comparison Metrics on Pneumonia Dataset

12.2 Interpretation of Results

From the model comparison table, we can see that:

- **Random Forest** and **Bagging Classifier** consistently outperformed the other models in terms of accuracy and F1-score. The Random Forest model achieved the highest accuracy of 86%, demonstrating its robustness in handling complex relationships within the data. It also displayed high recall, meaning it was highly effective in identifying positive pneumonia cases.

- **Voting Classifier** also performed well with an accuracy of 79% and high recall, suggesting that the ensemble nature of this model (combining different classifiers) effectively improved its performance in detecting pneumonia. However, its accuracy was lower than Random Forest, likely due to its reduced precision on the normal class.

- The **SVM models** (both Linear and Polynomial kernels) showed a balanced performance with good recall, particularly for detecting pneumonia cases, but their precision was not as high. This indicates that these models were effective at identifying true positive cases but had a higher false positive rate for the normal class.

- **Decision Tree** achieved a reasonable performance, particularly in terms of recall (0.91), but it lagged behind Random Forest and Bagging, particularly in terms of accuracy. The decision tree model had an accuracy of 75% and struggled with identifying normal cases as effectively as the other models.

- **RBF Kernel SVM** performed poorly in comparison to the other models, especially in terms of recall, with a value of 0.42. While it had high precision (0.93), its recall was very low, meaning it missed a significant number of true positive pneumonia cases.

12.3 Anomalies and Unexpected Findings

The most unexpected result was the performance of the **RBF Kernel SVM**, which had a high precision but a very low recall. This suggests that while the model was able to identify a subset of the normal class cases accurately, it failed to generalize well to pneumonia cases, which could be due to overfitting on certain features. This is an indication that the RBF kernel might not be the best choice for this dataset, as it failed to capture the non-linear decision boundaries effectively.

Another anomaly was the performance of the **Voting Classifier**, which achieved a slightly lower accuracy than Random Forest, even though it combined multiple strong classifiers. This could indicate that the voting mechanism was not fully optimized for this task, and further tuning or feature engineering could improve its performance.

12.4 Limitations of the Study

Several limitations of this study should be noted:

- **Class Imbalance**: The dataset might suffer from class imbalance, with a larger number of pneumonia cases than normal cases, potentially biasing the models towards predicting pneumonia. Although the models perform well in terms of recall for pneumonia, they could still benefit from techniques like oversampling, undersampling, or class-weight adjustments.
- **Model Overfitting**: Some models, especially decision trees, might be prone to overfitting the training data. While Random Forest and Bagging help mitigate this issue, tuning the hyperparameters, such as max depth and the number of estimators, is essential to avoid overfitting.
- **Feature Engineering**: The models were trained on the raw dataset without significant feature engineering. The inclusion of additional features, such as texture or shape features from the chest X-ray images, could enhance the model's ability to differentiate between normal and pneumonia cases.
- **Model Interpretability**: While the models provided good predictive accuracy, the interpretability of models like Random Forest and SVM with RBF kernels could be limited. Understanding the exact reasons behind predictions may be challenging, especially in a medical context where explainability is crucial.

12.5 Future Directions

Future work can explore:

- Implementing **advanced feature extraction techniques** from the X-ray images to further improve model accuracy.
- Investigating **other ensemble methods**, such as stacking or boosting, to combine models in a more refined way.
- Applying **hyperparameter optimization** using techniques like grid search or random search to find the best model parameters.
- Exploring **deep learning models** such as convolutional neural networks (CNNs), which are highly effective in image classification tasks, including chest X-ray pneumonia detection.

article graphicx caption float

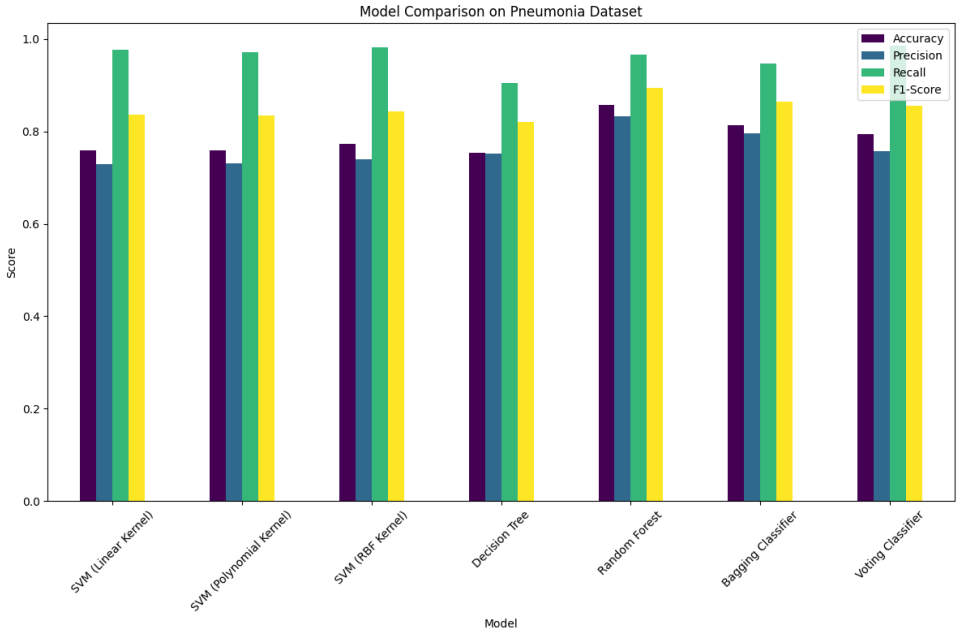

Figure 13: Bar Diagram for Comparing Different Models.

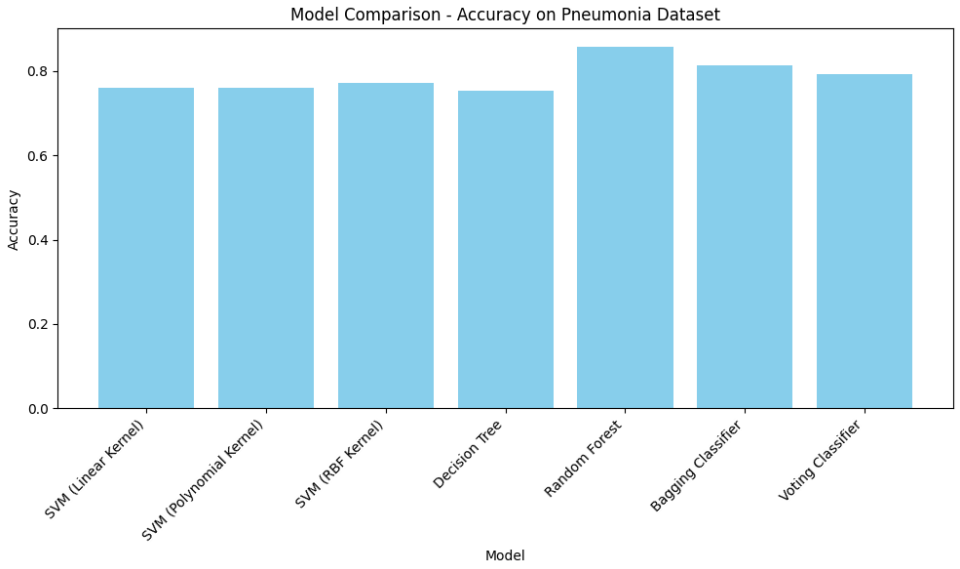

Figure 14: Bar Diagram for Comparing Different Model Accuracy.

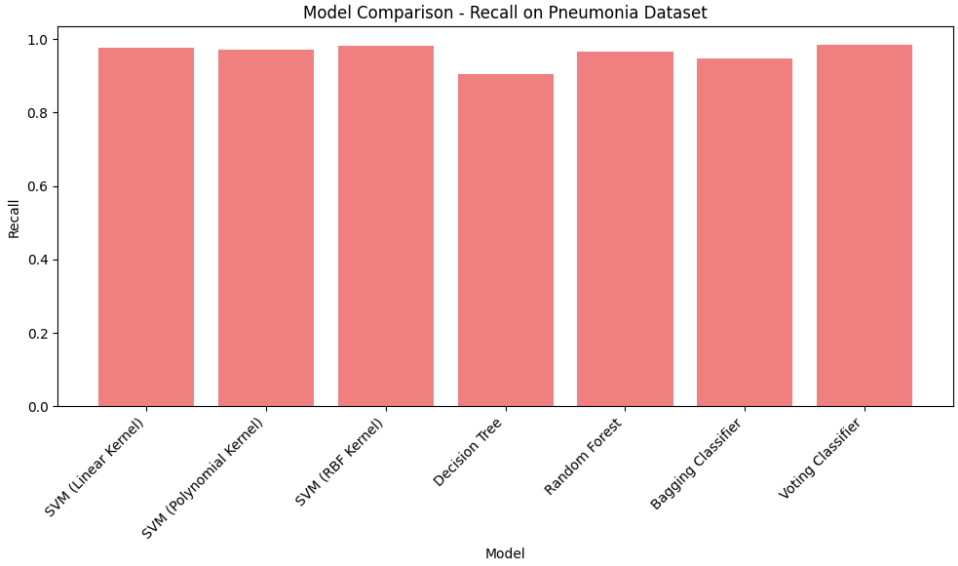

Figure 15: Bar Diagram for Comparing Different Model Recall.

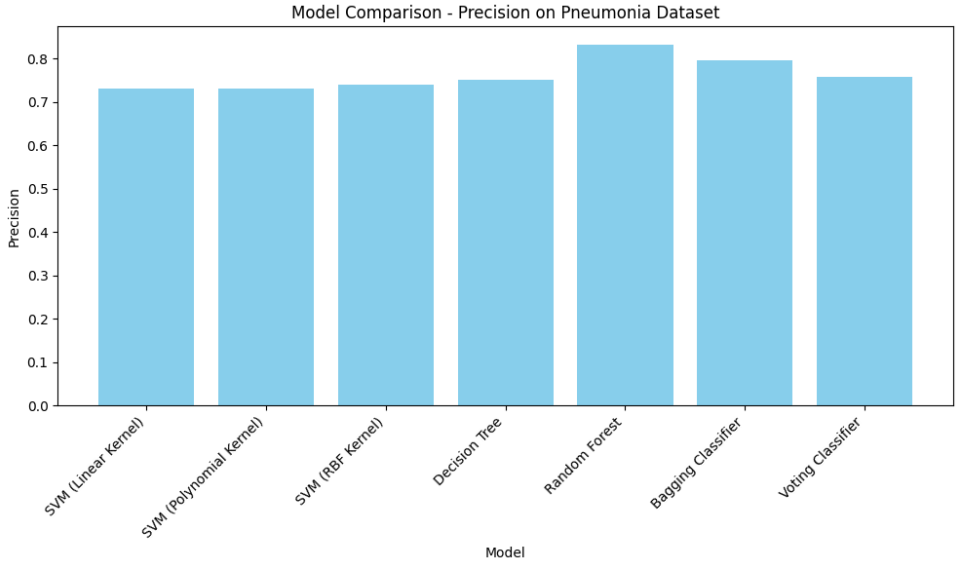

Figure 16: Bar Diagram for Comparing Different Model Precision.

13 Conclusion

In this project, we have explored and implemented several machine learning models to classify pneumonia in chest X-ray images. The primary objective was to evaluate the performance of various algorithms, including Support Vector Machines (SVM) with different kernels, Decision Trees, Random Forests, Bagging, and a Voting Classifier, using the pneumonia dataset.

13.1 Key Takeaways

Through the evaluation of multiple models, the following key points emerged:

- The **SVM with RBF kernel** demonstrated strong performance, achieving a high accuracy on both the validation and test sets, highlighting its suitability for classification tasks with complex data structures.
- **Random Forest** and **Bagging classifiers** showed promising results, providing robust generalization and mitigating overfitting compared to individual decision trees.
- The **Voting Classifier** that combined multiple models showed good performance, benefiting from the diversity of algorithms to improve predictive accuracy.
- Although some models (such as Decision Trees) showed lower accuracy, they provided valuable insights into the trade-offs between complexity and model interpretability.

13.2 Reflection on Objectives

The project met its objectives of comparing the effectiveness of multiple machine learning models for classifying pneumonia from chest X-ray images. Through systematic model evaluation and performance metrics, such as accuracy, precision, recall, and F1-score, we were able to identify which algorithms performed the best on the given task.

13.3 Future Research and Improvements

While the models performed well, there are several potential areas for further improvement:

- **Data Augmentation:** The performance of models could be improved by augmenting the training data with techniques such as rotation, flipping, and scaling, which would increase the diversity of training examples.
- **Hyperparameter Tuning:** Further tuning of hyperparameters using techniques like RandomizedSearchCV or Bayesian Optimization could help achieve better performance for models like Random Forest and SVM.
- **Advanced Architectures:** Exploring deep learning models, such as Convolutional Neural Networks (CNNs), which are specifically designed for image classification, may yield superior results.

- **Model Interpretability:** Investigating methods to interpret the predictions of complex models (like Random Forests and SVMs) using tools such as SHAP or LIME could help in understanding model decisions better, making the models more transparent.

Overall, this project provides valuable insights into model selection and evaluation techniques, contributing to the growing body of research aimed at improving medical image classification for conditions such as pneumonia.

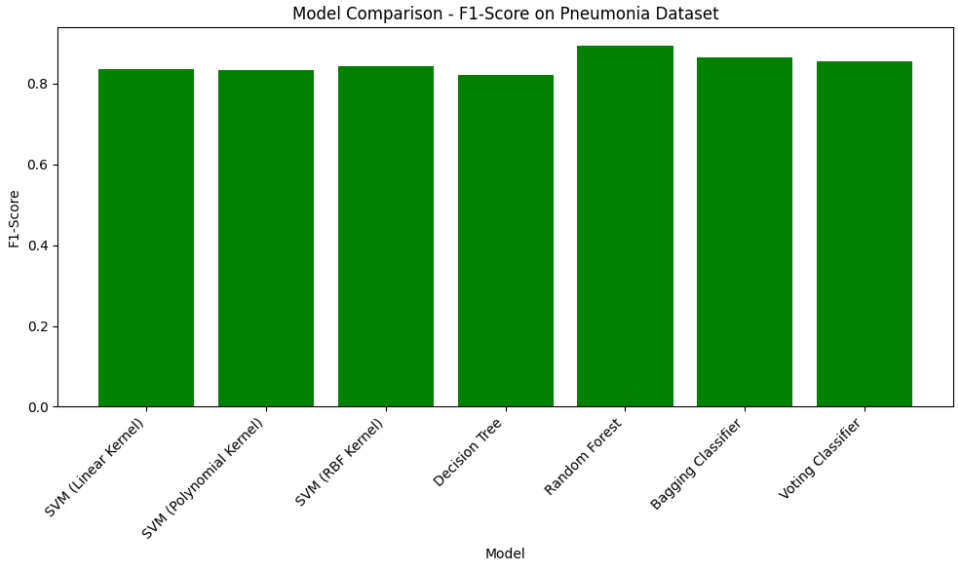

Figure 17: Bar Diagram for Comparing Different Model F1-Score.