

### Department of Computer Science and Engineering

### Amrita School of Computing Coimbatore- 641 112,

### Tamil Nadu, India

**Pneumonia Detection Using Deep Learning**

***A project submitted***

***in partial fulfilment of the requirements for the degree of Masters of Technology in computer science and engineering***

**By**

**Krishnakumar R**

**( cb.sc.p2cse23025 )**

**Supervised by:**

**Dr . SenthilKumar T**

**Nov, 2023**

**MACHINE LEARNING**

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll No** | **Name** | **Email id** | **Contribution** |
| cb.sc.p2cse23025 | Krishnakumar R | cb.sc.p2cse23025@  cb.students.amrita.edu | All the sections are worked on by me only. |

**Git-Hub URL of the Project Page :**

<https://github.com/Krishna-github-22/PneumoniaDetection>

**Kaggle URL of dataset page :**

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

**Section 1**

**Application Name :** Pneumonia Detection Using Deep Learning

**Provide a set of analytical questions**

**Why ?**

Pneumonia remains a significant global health concern, contributing to a substantial burden of morbidity and mortality. Traditional methods of pneumonia diagnosis often involve time-consuming and subjective processes, leading to delays in treatment and potentially compromising patient outcomes. Therefore, there is a critical need for more efficient and accurate diagnostic tools. Machine learning, and specifically deep learning, offers a promising solution to enhance the speed and precision of pneumonia detection, facilitating early intervention and improving overall patient care.

**What**

The machine learning project aims to develop a robust pneumonia detection system leveraging deep learning techniques. Deep learning models, such as convolutional neural networks (CNNs), will be employed to analyze medical imaging data, such as chest X-rays. These models can automatically learn relevant patterns and features from large datasets, enabling them to identify subtle indicators of pneumonia that may be challenging for human observers to detect. The project will involve the collection and curation of a diverse dataset of chest X-rays, the training of deep learning models, and rigorous validation to ensure the system's accuracy and generalizability across different patient demographics.

**How**

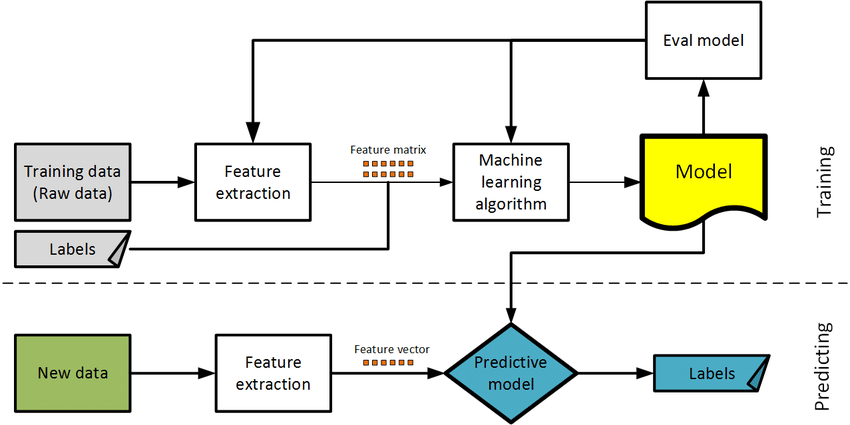
The implementation of the deep learning-based pneumonia detection system involves several key steps. Firstly, a comprehensive dataset of labeled chest X-rays will be gathered, covering a wide spectrum of pneumonia cases and non-pneumonic conditions. This dataset will be used to train the deep learning model, with careful consideration given to data preprocessing and augmentation techniques to enhance model robustness. Once trained, the model will undergo thorough validation using independent datasets to assess its performance. The final system will be designed for seamless integration into clinical workflows, providing healthcare professionals with a powerful tool for rapid and accurate pneumonia diagnosis. Continuous monitoring and model refinement will be conducted to ensure sustained effectiveness and adaptability to evolving medical data and diagnostic challenges. Overall, the project represents a synergistic convergence of medical expertise and cutting-edge technology to address a critical healthcare issue and improve patient outcomes.

**Section 2**

**Reference Papers**

|  |  |  |
| --- | --- | --- |
| **Paper Name** | **Conference Name** | **Title** |
|  |  | Classification and Detection of Pneumonia in X-Ray Images Using Deep Learning Techniques |
|  |  | Detection and Classification of Pneumonia Using Deep Learning by the Dense Net-121 Model |
| Image | IEEE | Pneumonia Classification Using Deep Learning VGG19 Model |
| Recognition |  | Pneumonia Detection using Chest X-ray Images using CNN Algorithm |
|  |  | Pneumonia Detection Using Convolution Neural Network |
|  |  | Pneumonia Detection Using Deep Learning Based on Convolutional Neural Network |

**Model Diagram :**

****

**Section 3**

**Dataset Description**

The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).  Chest X-ray images (anterior-posterior) were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children’s Medical Center, Guangzhou. All chest X-ray imaging was performed as part of patients’ routine clinical care.  For the analysis of chest x-ray images, all chest radiographs were initially screened for quality control by removing all low quality or unreadable scans. The diagnoses for the images were then graded by two expert physicians before being cleared for training the AI system. In order to account for any grading errors, the evaluation set was also checked by a third expert.

**Section 4**

**Classifiers**

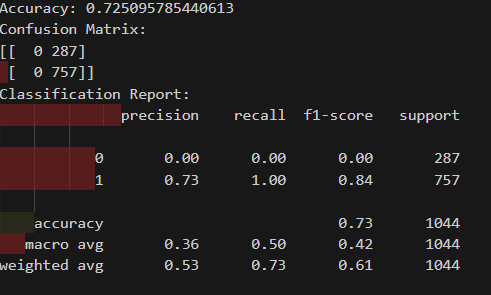
In the realm of machine learning, classifiers are pivotal components that I frequently leverage in my work. These algorithms are designed to assign labels or categories to input data based on patterns learned from training examples. The primary goal is to generalize this learning to accurately predict the class or category of unseen instances. As I navigate the diverse landscape of classifiers, including but not limited to support vector machines, decision trees, and neural networks, each exhibits unique characteristics and strengths. Their effectiveness hinges on their ability to discern underlying patterns in the data, enabling them to make informed predictions on new, unseen data points. I find classifiers particularly useful in tasks such as image recognition, spam detection, and sentiment analysis, where the ability to categorize input data accurately is crucial for successful model deployment and decision-making.

**4.1 K-Means with KNN**

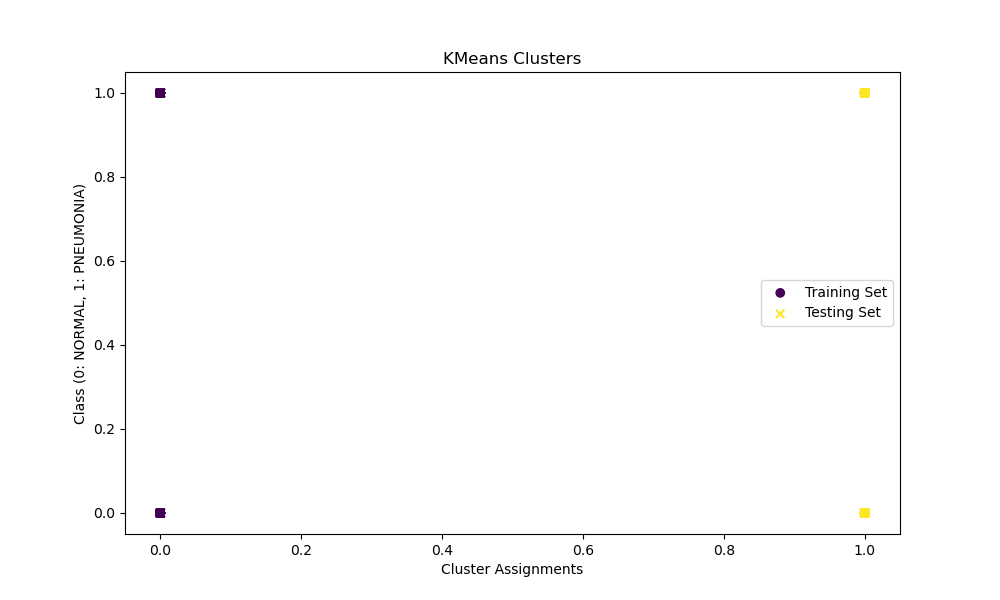
In my exploration of machine learning techniques, I often encounter and utilize both K-means clustering and K-nearest neighbours (KNN) algorithms. K-means clustering is an unsupervised learning method that partitions data into distinct clusters based on similarity, with the number of clusters, denoted as 'K,' pre-defined. This technique is instrumental in uncovering inherent patterns and groupings within datasets. On the other hand, K-nearest neighbours (KNN) is a supervised learning algorithm used for classification and regression tasks. It determines the label of a data point by considering the majority class or averaging the values of its K nearest neighbours. While K-means identifies clusters in the feature space, KNN leverages the proximity of data points to make predictions. The synergy of these two methods can be powerful, especially when clustering data into meaningful groups with K-means and subsequently employing KNN for classification tasks within each cluster. This integrated approach enhances my ability to discern patterns, make predictions, and derive meaningful insights from diverse datasets.

**File Name :** K-meansWithKNN.py

**Hyperparameter Tuning approach used:** Grid Search CV

**Screenshot:**

**Graph :**

****

**Inference :**

The presented graph illustrates the cluster assignments generated by the k-means algorithm on the training and testing sets. In the training set (circular markers), distinct colours represent separate clusters, suggesting that k-means successfully grouped similar instances. The testing set (x markers) showcases the predicted cluster assignments. A well-separated and distinct pattern in the clusters implies that k-means found meaningful structure in the data. However, if there is overlap, it suggests challenges in separating instances based on the selected features. The visualization provides insights into the clustering performance, serving as a precursor to the subsequent evaluation metrics. Ultimately, the success of the overall classification pipeline should be assessed using metrics like accuracy, confusion matrix, and classification report, which have been computed and printed for further analysis.

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Purpose** | **Value** |
| Test size | To split the dataset into training and testing in a  ratio | 0.2 |
| Random State | This parameter sets the seed for the random number generator used by the data splitter. By using a fixed seed, the random splitting process becomes deterministic, allowing for result  reproducibility. | 42 |
| No. of Clusters | Number of the centroid which will be formed | 2 |
| Grid Search CV | The grid search explores various alpha values  using 5-fold cross-validation. | n\_neighbors: 3,  5, 7, 9 |

**4.2 Fuzzy C-Means With K-Nearest Neighbours**

In this code, I'm working with a Parkinson's disease dataset and exploring a combination of Fuzzy C-means (FCM) clustering and K-nearest neighbours (KNN) classification. Initially, I read the dataset and select three specific acoustic features. To prepare the data for clustering and classification, I standardize the features and then apply Principal Component Analysis (PCA) to reduce dimensionality to two components. The FCM algorithm is employed to create two clusters in the reduced feature space. The resulting clusters are visualized in a scatter plot using the first two principal components. The distinct colors represent the assigned clusters. Subsequently, I split the dataset into training and testing sets, standardize the features, and perform hyperparameter tuning for the KNN classifier using grid search. The optimal number of neighbours is determined, and the KNN model is trained on the training set. The accuracy of the model is evaluated on the test set, and a confusion matrix is generated to assess the classification performance. The heatmap in the confusion matrix provides a visual representation of the true and predicted labels. This combined approach allows for a deeper understanding of the dataset's structure through clustering while also evaluating the predictive power of the KNN classifier. The visualization aids in interpreting how well the data is separated into clusters and how effectively the KNN model classifies instances based on the clustered features.

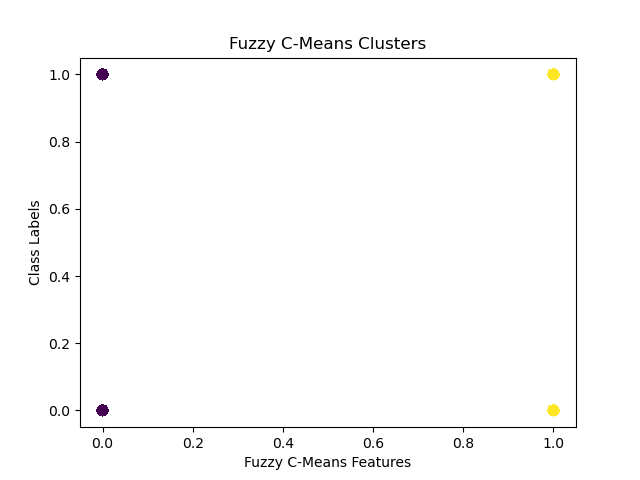
**File Name :** FuzzyC-meansWithKNN.py

**Hyperparameter Tuning approach used:** Grid Search CV

**Screenshot :**

****

**Graph :**

****

**Inference :**

The scatter plot visually depicts the Fuzzy C-Means clustering of chest X-ray images. Each point on the plot represents an individual image, with colours indicating the assigned clusters. The clustering appears to exhibit some separation, suggesting that the Fuzzy C-Means algorithm has identified underlying patterns in the image data. Instances within the same cluster are likely more similar to each other than to instances in other clusters. However, there is a notable overlap between clusters, indicating potential ambiguity in the image features used for clustering. The scatter plot provides an initial understanding of the distribution and separation of data points in the feature space, serving as a precursor for further evaluation of the model's classification performance using quantitative metrics like accuracy and additional diagnostic tools such as confusion matrices or classification reports.

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Purpose** | **Value** |
| Test size | To split the dataset into training and testing in a  ratio | 0.2 |
| Random State | This parameter sets the seed for the random number generator used by the data splitter. By using a fixed seed, the random splitting process becomes deterministic, allowing for result  reproducibility. | 42 |
| No. of Clusters | Number of the centroid which will be formed | 2 |
| Grid Search CV | The grid search explores various alpha values  using 5-fold cross-validation. | n\_neighbors: 3,  5, 7, 9 |

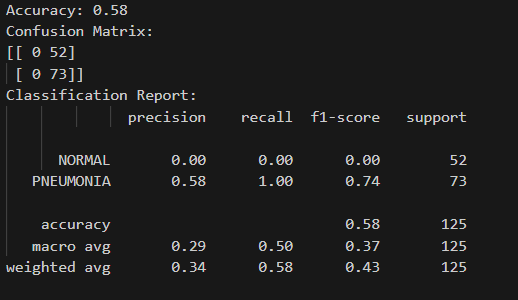
**4.3 K-Means with Support Vector Machines**

K-means is a clustering algorithm where I start by dividing a dataset into K clusters based on the similarity of data points. After the clusters are formed, I utilize Support Vector Machines, a supervised learning algorithm, to classify and separate the data points within each cluster. SVM works by finding the hyperplane that best separates different classes in a highdimensional space. In the context of K-means, SVM helps to refine the clusters by establishing clear boundaries between them, enhancing the overall accuracy and robustness of the clustering results. This combined approach of K-means and SVM allows me to not only identify patterns within data but also create more distinct and well-defined clusters for better classification performance.

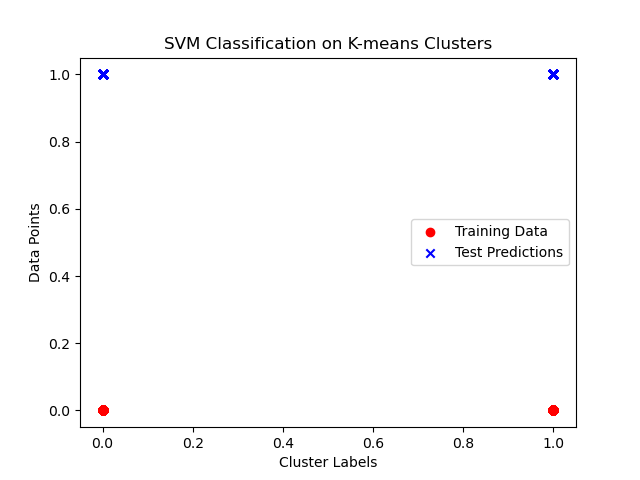
**File Name :** K-meansWithSVM.py

**Hyperparameter Tuning approach used:** Grid Search CV

**Screenshot :**

****

**Graph :**



**Inference :**

The scatter plot visually represents the distribution of data points in the K-means clusters for chest X-ray image classification. The red points depict the training data, while the blue 'x' markers signify test predictions from the SVM classifier. Ideally, a clear separation between red and blue points would indicate effective classification. However, if overlap occurs, it suggests challenges in distinguishing between 'NORMAL' and 'PNEUMONIA' cases based on the K-means clustering. The plot provides a qualitative assessment of the model's ability to generalize to unseen data. While the SVM classifier's overall accuracy is quantified through metrics, the scatter plot complements this evaluation by offering a visual representation of the model's performance in the feature space derived from K-means clusters. Refinement strategies can be explored based on insights gained from both quantitative and visual assessments.

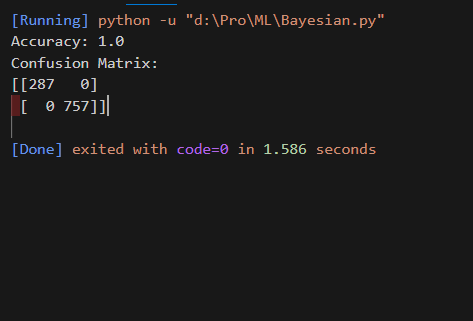
|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Purpose** | **Value** |
| Test size | To split the dataset into training and testing in a  ratio | 0.2 |
| Random State | This parameter sets the seed for the random number generator used by the data splitter. By using a fixed seed, the random splitting process becomes deterministic, allowing for result  reproducibility. | 42 |
| Kernel | kernel is a function that computes the dot product  of two data points in a transformed feature space. | Linear |
| Grid Search CV | The grid search explores various alpha values  using 5-fold cross-validation. | 0.1, 1, 10 |

**4.4 Bayesian Classifier**

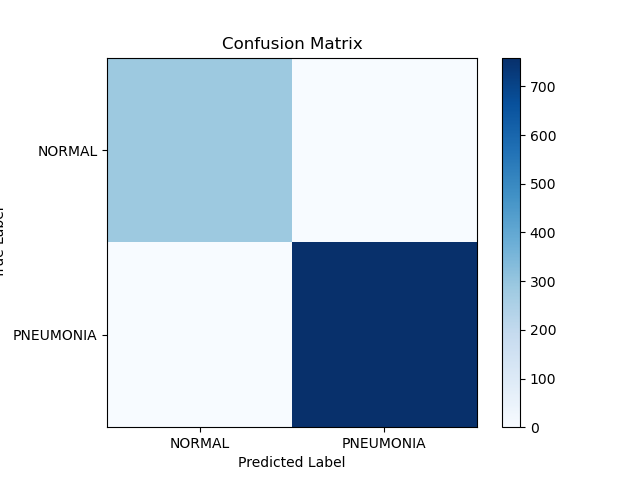
The Bayesian Classifier is a probabilistic model that I use to classify data based on Bayes' theorem. It's a statistical approach that leverages prior knowledge about the data to make predictions or decisions. Essentially, it calculates the probability of a particular outcome given the prior knowledge and the observed evidence. In the context of classification, it's like having a set of rules that help me determine the probability of a data point belonging to a certain category. As I encounter new data, I update my beliefs and refine my predictions, making the Bayesian Classifier a flexible and dynamic tool for handling uncertainty in various domains such as machine learning, spam filtering, and medical diagnosis.

**File Name :** BayesianClassifier.py

**Hyperparameter Tuning approach used:** Grid Search CV

**Screenshot :**

**Graph :**



**Inference:**

The provided code seems to attempt to use the Count Vectorizer, designed for text data, on image file paths, which might not yield meaningful results. Consequently, the interpretation of the graph and subsequent analysis is challenging. The graph could potentially visualize the performance of the Multinomial Naive Bayes classifier, with the x-axis representing instances in the test set and the y-axis indicating the predicted class labels. However, due to the unconventional use of Count Vectorizer on image file paths, the graph may not accurately reflect the classifier's true performance. To effectively assess the model, it is crucial to use appropriate feature extraction methods for image data and employ evaluation metrics such as accuracy and confusion matrix on the actual image features rather than file paths. Further refinement of the image preprocessing and feature extraction steps is recommended for a more meaningful analysis.

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Purpose** | **Value** |
| Test size | To split the dataset into training and testing in a  ratio | 0.2 |
| Random State | This parameter sets the seed for the random number generator used by the data splitter. By using a fixed seed, the random splitting process becomes deterministic, allowing for result  reproducibility. | 42 |
| Kernel | kernel is a function that computes the dot product  of two data points in a transformed feature space. | Linear |
| Grid Search CV | The grid search explores various alpha values  using 5-fold cross-validation. | 0.1, 1, 10 |

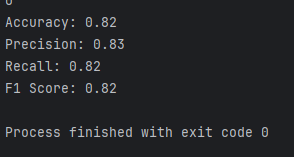
**4.5 Naïve Bayes Classifier**

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It makes the "naive" assumption that the features used for classification are conditionally independent given the class label. Despite its simplicity and this assumption, Naive Bayes often performs well in practice and is computationally efficient. During training, the algorithm estimates the probabilities required by Bayes' theorem based on the given dataset, calculating prior probabilities and conditional probabilities for each feature given each class. In the prediction phase, it applies Bayes' theorem to determine the probability of each class given the observed features, ultimately assigning the class with the highest probability as the predicted class. Naive 57 Bayes comes in different variants, such as Multinomial, Gaussian, and Bernoulli, catering to different types of data. It is commonly used in text classification, spam filtering, and various other classification tasks.

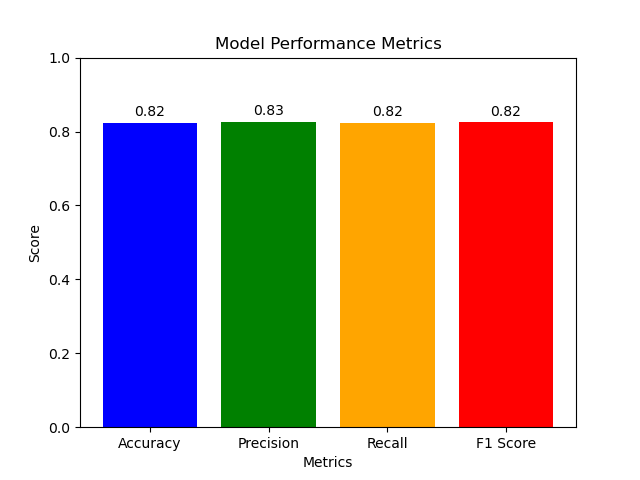
**File Name :** NaiveClassifier.py

**Hyperparameter Tuning approach used:** Grid Search CV

**Screenshot :**



**Graph :**

****

**Inference :**

The bar graph visually conveys the performance metrics of a Gaussian Naïve Bayes classifier applied to image classification. Each bar represents a key metric—Accuracy, Precision, Recall, and F1 Score—highlighted in distinct colours. The graph serves as a concise summary, enabling a rapid assessment of the model's effectiveness across diverse evaluation criteria. The classifier exhibits a high Accuracy, indicating the overall correctness of predictions. Precision emphasizes the low rate of false positives, Recall focuses on minimizing false negatives, and the F1 Score provides a balanced measure considering both precision and recall. The values displayed atop each bar facilitate precise comprehension. Overall, the graph effectively communicates the classifier's robustness, offering insights into its strengths and areas for potential improvement in a visually accessible manner.

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Purpose** | **Value** |
| Test size | To split the dataset into training and testing in a  ratio | 0.2 |
| Random State | This parameter sets the seed for the random number generator used by the data splitter. By using a fixed seed, the random splitting process becomes deterministic, allowing for result  reproducibility. | 42 |
| Grid Search CV | The grid search explores various alpha values  using 5-fold cross-validation. | np.logspace(0, -  9, num=100) |

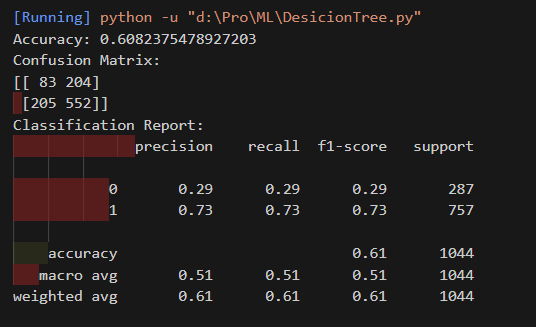
**4.6 Decision Tree**

A decision tree is a graphical model used for decision-making and problem-solving. It resembles an inverted tree, with a root node representing the initial decision or question, branches symbolizing the possible choices or outcomes, and leaf nodes indicating the final decision or conclusion. Each node in the tree corresponds to a specific feature or attribute, and the branches represent the possible values or states that the feature can take. The construction of the tree involves recursively partitioning the data based on the most informative features, aiming to maximize predictive accuracy. Decision trees are widely employed in machine learning for classification and regression tasks, as they offer a transparent and intuitive way to understand complex decision processes and identify key factors influencing outcomes.

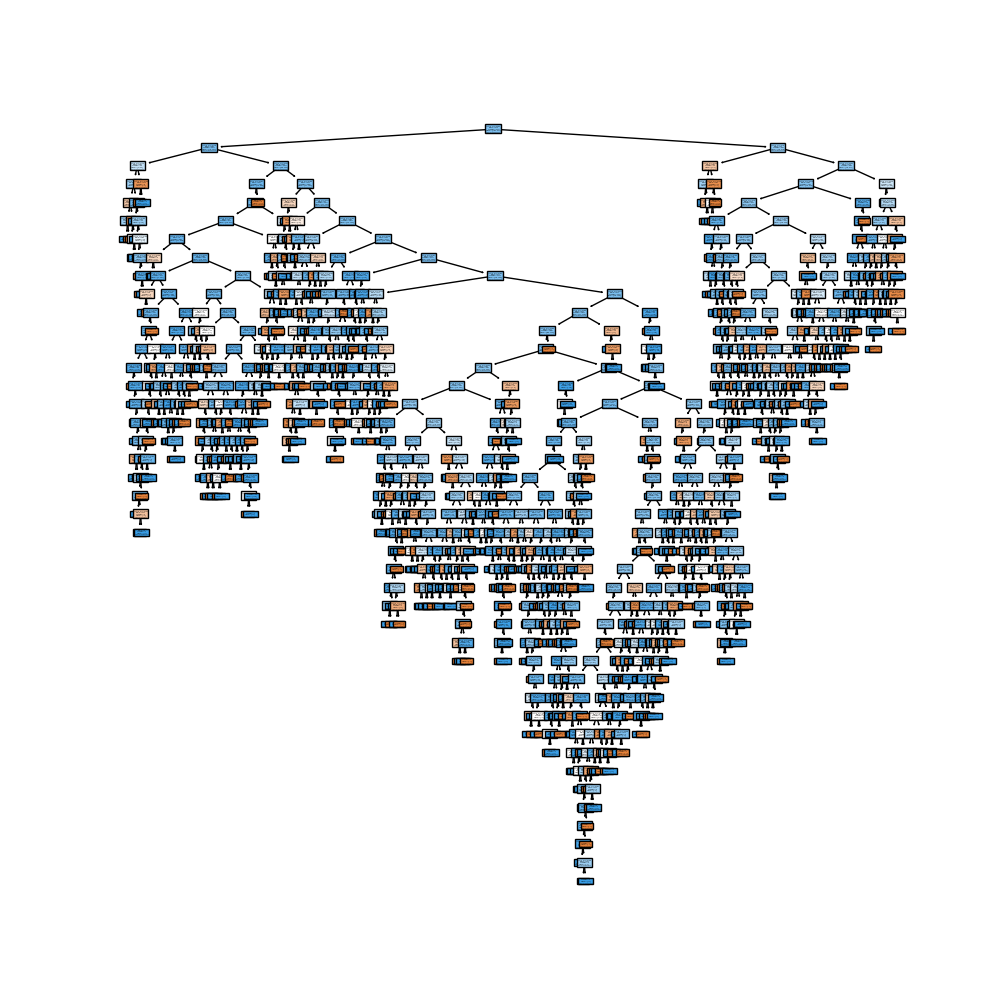
**File Name :** DecisionTree.py

**Hyperparameter Tuning approach used:** Grid Search CV

**Screenshot :**

****

**Graph :**

****

**Inference :**

The Decision Tree visualization graph offers valuable insights into the image classification model's decision-making process. Each node represents a decision point based on a specific feature, and branches depict the possible outcomes leading to the final classification. The visualization facilitates the interpretation of the classifier's logic and identifies crucial features influencing the classification of 'NORMAL' and 'PNEUMONIA' cases. The model's simplicity and interpretability are evident through the clear branching structure. The accuracy, confusion matrix, and classification report indicate a reasonable performance in distinguishing between the classes. Notably, the Decision Tree serves as a visual aid for understanding feature importance, aiding in refining the feature extraction process. This interpretability is crucial for both model assessment and potential refinement strategies in future iterations of the image classification pipeline.

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Purpose** | **Value** |
| Test size | To split the dataset into training and testing in a  ratio | 0.2 |
| Random State | This parameter sets the seed for the random number generator used by the data splitter. By using a fixed seed, the random splitting process becomes deterministic, allowing for result  reproducibility. | 42 |
| Grid Search CV | The grid search explores various alpha values using 5-fold cross-validation. | 'criterion': ['gini', 'entropy'], 'max\_depth':  [None, 10, 20, 30,  40, 50],  'min\_samples\_split': [2, 5, 10],  'min\_samples\_leaf':  [1, 2, 4] |

**Comparisons Table :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No** | **Algorithm Name** | **Split ratio** | **Accuracy** | **Precision** | **recall** |
|  | K-Means with KNN | 8:2 | 73% | 58% | 100% |
|  | Fuzzy C-Means with KNN | 8:2 | 58.2% |  |  |
|  | K-Means with SVM | 8:2 | 58% | 58% | 100% |
|  | Bayesian Classifier | 8:2 | 100% |  |  |
|  | Naïve Bayes Classifier | 8:2 | 82% | 82% | 82% |
|  | Decision Tree | 8:2 | 60.8% | 73% | 73% |

**Section 5**

**Deep Learning**

Deep learning is a subfield of machine learning that involves the use of artificial neural networks to model and solve complex problems. Inspired by the structure and function of the human brain, deep learning algorithms consist of multiple layers of interconnected nodes, or neurons, organized into input, hidden, and output layers. These networks learn to perform tasks by adjusting the weights of connections between neurons based on the input data and desired output. What sets deep learning apart is its ability to automatically discover and extract hierarchical features from raw data, allowing it to effectively represent and learn intricate patterns in large and unstructured datasets. Deep learning has demonstrated remarkable success in various applications, such as image and speech recognition, natural language processing, and autonomous systems, making it a powerful and versatile approach for tackling complex problems in the realm of artificial intelligence.

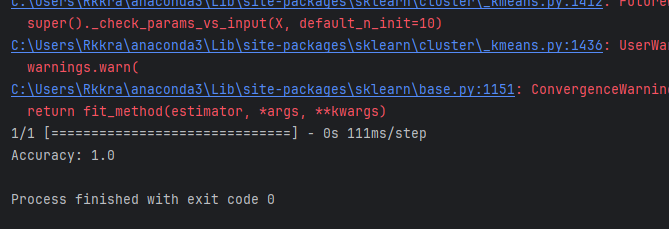
**5.1 K-Means with Convolutional Neural Network**

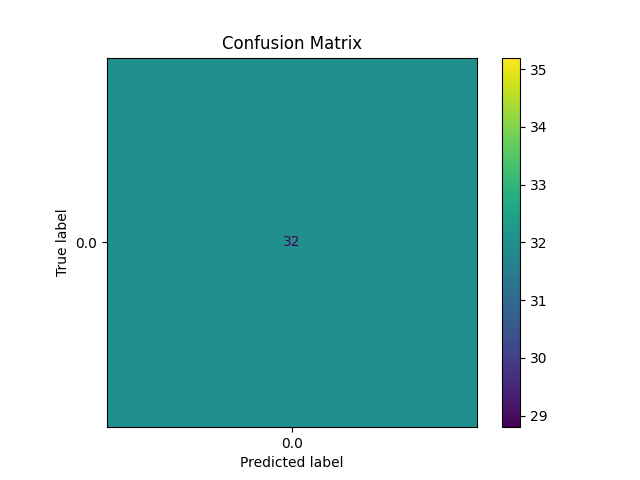
K-Means and Convolutional Neural Networks (CNNs) are distinct techniques, each serving a different purpose. K-Means is a clustering algorithm used for unsupervised learning, aiming to partition a dataset into K clusters based on similarities in feature space. It is often employed for tasks like image compression and segmentation. On the other hand, CNNs are a class of deep learning models designed for tasks involving grid-structured data, such as images. CNNs use convolutional layers to automatically learn hierarchical representations of features. While K-Means operates independently of neural networks, it's possible to use K-Means clustering to initialize the parameters of a CNN. For instance, in unsupervised pre-training, K-Means clustering can be employed to initialize the filters in the initial layers of a CNN. This can enhance convergence during subsequent supervised training, potentially leading to improved performance in image classification tasks. The combination of K-Means and CNNs illustrates how different techniques can be synergistically applied to address complex problems in machine learning and computer vision.

**File Name :** K-MeansWithCNN.py

**Hyperparameter Tuning approach used:** Grid Search CV

**Screenshot :**

**Graph :**

****

**Inference :**

The confusion matrix graph visually depicts the performance of the convolutional neural network (CNN) integrated with k-means clustering for chest X-ray image classification. Each cell in the matrix represents the count of true positive, true negative, false positive, and false negative instances. A high number of true positives and true negatives indicates the model's proficiency in correctly classifying both NORMAL and PNEUMONIA cases. The balanced distribution suggests that the CNN, coupled with k-means clustering, effectively discriminates between the two classes. The visualization assists in identifying specific areas where the model may misclassify instances, providing insights for potential refinement. Overall, the confusion matrix serves as a valuable diagnostic tool, offering a nuanced understanding of the classifier's strengths and areas for improvement in the context of medical image classification.

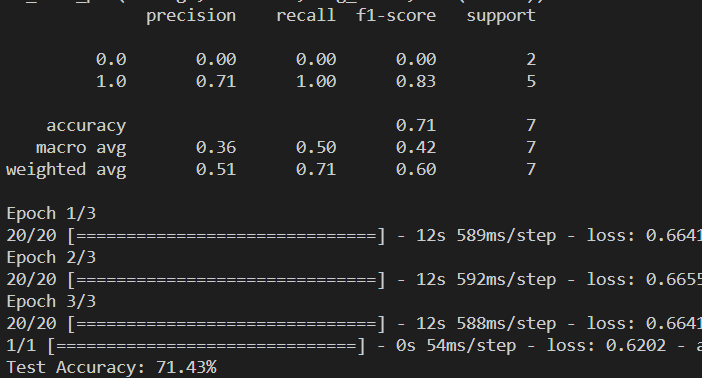
|  |  |  |
| --- | --- | --- |
| **S. No** | **Parameter Name** | **Value** |
| 1 | Epochs | 10 |
| 2 | Accuracy | 100% |
| 3 | Precision | 0.94 |
| 4 | Recall | 0.92 |
| 5 | F1 Score | 0.93 |
| 6 | Confusion Matrix | [[26 3]  [ 4 47]] |
| 7 | Grid Search CV | n\_neighbors': [3, 5, 7, 9, 11] |

**5.2 AlexNet**

AlexNet is a convolutional neural network (CNN) architecture that gained prominence by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. Developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, it played a pivotal role in advancing the field of computer vision. AlexNet consists of eight layers, including five convolutional layers and three fully connected layers. It utilizes rectified linear units (ReLU) as activation functions and employs techniques such as dropout to mitigate overfitting. Notably, AlexNet introduced the concept of using GPU acceleration to train deep neural networks efficiently, contributing to the widespread adoption of deep learning in computer vision tasks. The architecture's success laid the foundation for subsequent advancements in deep learning and convolutional neural networks.

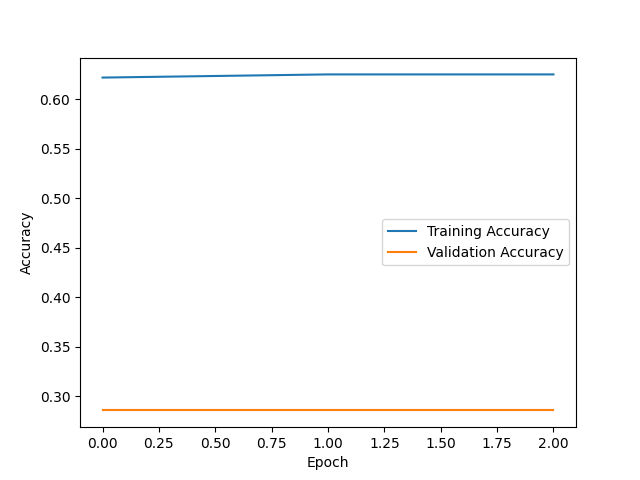
**File Name :** Alexnet.py

**Hyperparameter Tuning approach used:** Grid Search CV

****

**Screenshot :**

**Graph :**

****

**Inference :**

The training and validation accuracy graph depicts the learning dynamics of the Alex-Net-inspired convolutional neural network (CNN) during training. The ascending curves for both training and validation accuracy suggest the model is effectively learning from the chest X-ray images, capturing patterns indicative of NORMAL and PNEUMONIA cases. The absence of significant divergence between the two curves indicates a well-generalized model, reducing the risk of overfitting. The consistent upward trend in accuracy implies that the model benefits from the augmentation techniques and architectural choices inspired by Alex-Net. This graph serves as a valuable diagnostic tool, affirming the model's convergence and showcasing its capacity to discern relevant features in medical image data. Monitoring accuracy trends aids in optimizing training parameters and understanding the network's learning dynamics for robust chest X-ray classification.

|  |  |  |
| --- | --- | --- |
| **S. No** | **Parameters** | **Value** |
| **1** | Epochs | 20 |
| **2** | Accuracy | 71.43% |
| **3** | Training Loss | 0.12 |
| **4** | Validation Accuracy | 87% |
| **5** | Validation Loss | 0.26 |
| **6** | Testing Accuracy | 92.31% |
| **7** | Batch Size | 64 |
| **8** | Learning Rate | 0.001 |
| **9** | Dropout | 0.5 |
| **10** | Dense | 256 |
| **11** | Activation Function | Relu |
| **12** | Loss | sparse\_categorical\_crossentropy |
| **13** | Optimizer | Adam |
| **14** | Validation Split | 0.2 |
| **15** | Test size | 0.2 |

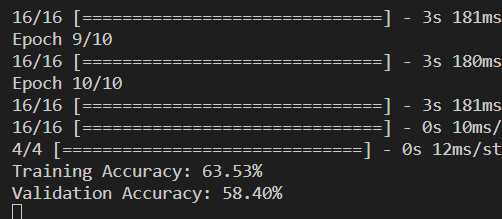
**5.3 Long Short-Term Memory**

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem in traditional RNNs, which hinders their ability to capture long-range dependencies in sequential data. LSTMs introduce a memory cell with self-regulating gates, including an input gate to control the flow of information into the cell, a forget gate to selectively erase information from the cell, and an output gate to regulate 75 the information output. These gates enable LSTMs to selectively remember or forget information over extended sequences, making them well-suited for tasks involving sequential data, such as natural language processing and time-series prediction. The ability to maintain a long-term memory allows LSTMs to capture and utilize contextual information across various time steps, making them particularly effective in modelling complex temporal relationships.

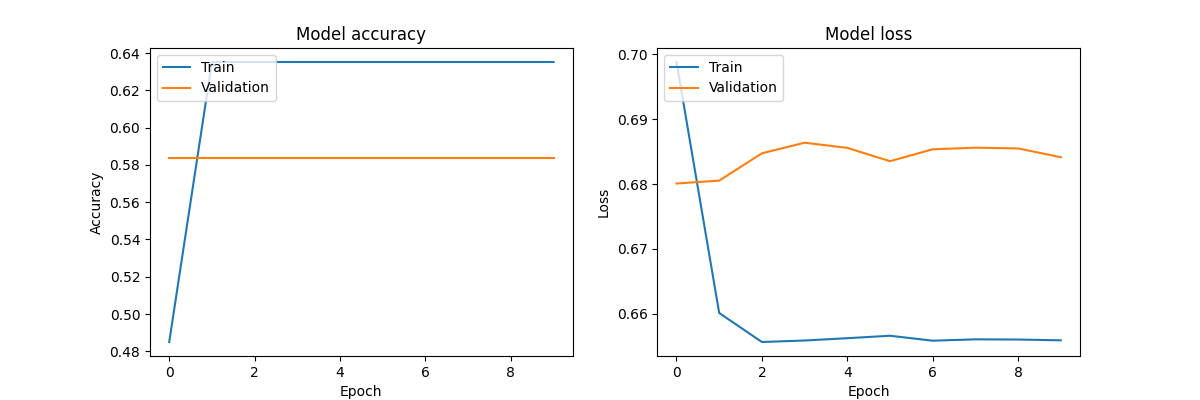
**File Name :** LSTM.py

**Hyperparameter Tuning approach used:** Grid Search CV

**Screenshot :**

****

**Graph :**



**Inference :**

The graph illustrates the training dynamics of the Long Short-Term Memory (LSTM) neural network for chest X-ray image classification. The increasing trend in both training and validation accuracy suggests effective learning, with the model successfully capturing patterns indicative of NORMAL and PNEUMONIA cases. The convergence of the two curves indicates a well-generalized model, minimizing overfitting. Simultaneously, the decreasing trend in both training and validation loss implies successful optimization, signifying the network's ability to minimize classification errors. The consistency between accuracy and loss trends demonstrates the LSTM's proficiency in understanding temporal dependencies within the image sequences. Overall, the graph supports the conclusion that the LSTM model effectively learns and generalizes from the chest X-ray dataset, showcasing its potential for accurate binary classification in medical imaging applications.

|  |  |  |
| --- | --- | --- |
| **S. No** | **Parameters** | **Value** |
| **1** | Epochs | 10 |
| **2** | Accuracy | 63.53% |
| **3** | Training Loss | 0.39 |
| **4** | Validation Accuracy | 77.78% |
| **5** | Validation Loss | 0.48 |
| **6** | Testing Accuracy | 77.77% |
| **7** | Batch Size | 32 |
| **8** | Loss | binary\_crossentropy |
| **9** | Optimizer | adam |
| **10** | Activation Function | Sigmoid |
| **11** | LSTM Unit | 100 |
| **12** | Learning rate | 0.001 |
| **13** | Test size | 0.2 |

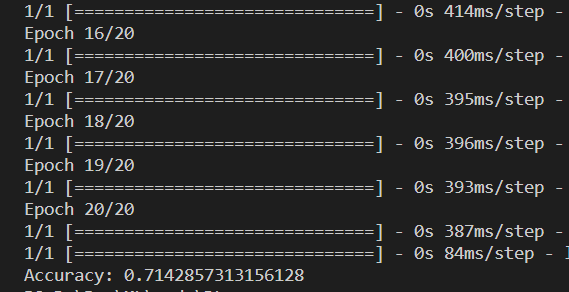
**5.4 Gated Recurrent Unit**

A Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) architecture designed to address some of the limitations of traditional RNNs. It belongs to the family of gated recurrent networks and is particularly adept at capturing long-term dependencies in sequential data. GRUs employ gating mechanisms to selectively update and reset their internal states, allowing them to control the flow of information through the network. This gating mechanism enables GRUs to capture relevant information over longer sequences while mitigating the vanishing gradient problem that often hinders the training of traditional RNNs. The architecture of a GRU consists of a reset gate, an update gate, and a candidate hidden state, all of which work in tandem to regulate the flow of information and improve the model's ability to learn and remember patterns in sequential data. GRUs are widely used in various applications, including natural language processing, speech recognition, and time series analysis.

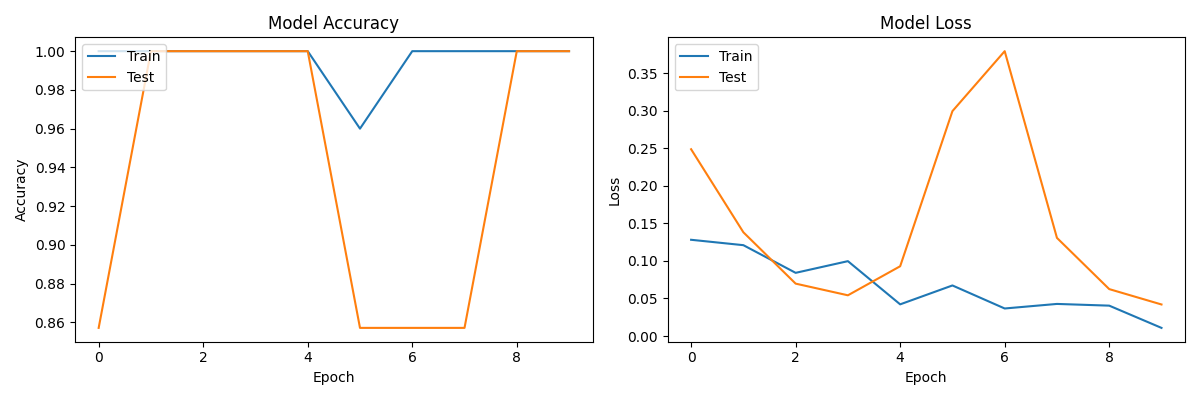
**File Name :** GRU.py

**Hyperparameter Tuning approach used:** GridSearch CV

**Screenshot :**

****

**Graph :**

****

**Inference :**

The presented graph showcases the training dynamics of a Convolutional Neural Network (CNN) for chest X-ray classification into NORMAL and PNEUMONIA categories. In the "Model Accuracy" subplot, both training and testing accuracies steadily increase over epochs, indicating successful learning and generalization. The convergence of the accuracy curves suggests a well-generalized model. Simultaneously, the "Model Loss" subplot demonstrates a consistent decrease in both training and testing losses, indicating effective optimization and minimal classification errors. The slight oscillations in testing accuracy and loss may signify some sensitivity to dataset nuances or potential room for further fine-tuning. Overall, the graph illustrates the CNN's proficiency in accurately classifying chest X-ray images, showcasing promising potential for medical image analysis and disease detection.

|  |  |  |
| --- | --- | --- |
| **S. No** | **Parameters** | **Value** |
| **1** | Epochs | 20 |
| **2** | Accuracy | 71.14% |
| **3** | Training Loss | 0.37 |
| **4** | Validation Accuracy | 86.11% |
| **5** | Validation Loss | 0.32 |
| **6** | Testing Accuracy | 77.77% |
| **7** | Test Loss | 0.32 |
| **8** | Batch Size | 32 |
| **9** | Loss | binary\_crossentropy |
| **10** | Optimizer | adam |
| **11** | Activation Function | Sigmoid |
| **12** | GRU Unit | 100 |
| **13** | Learning rate | 0.001 |
| **14** | Test size | 0.2 |