Disease Detection using Machine and Deep Learning Models

Siddhesh M. Bhabad
Student, M.C.A. Department, VJTI college, Mumbai.
Under guidance of prof. Surekha Kohle

Abstract –

Machine Learning and deep learning are modern and highly sophisticated technologies. They are widely used in many fields like finance, medical science and security. The application of machine learning in the field of medical diagnosis is increasing gradually. ML and DL models are used to discover patterns from medical data sources and provide excellent capabilities to predict diseases. In this paper, I have used various machine learning and deep learning algorithms on different medical datasets for developing efficient ML and DL models. I have applied different classification algorithms for diseases like Heart/Diabetes to classify whether a patient is affected by the disease or not. Along with Deep learning algorithms for diseases like Covid19 for classifying the type of infection present in the organ. Then based on the accuracy of each model, best one is selected. Finally, the web app 'Disease Detection' is being deployed using Heroku platform.

Keywords –

Healthcare, machine learning, deep learning, classification, disease detection

Introduction –

According to the World Health Organization, yearly, 28% of all deaths in India are related to heart disease, over 98 million people may have diabetes in India. India has the highest amounts of diabetes of any country in the world and India is now the country with highest number of covid19 cases after US.

The early detection of common diseases such as heart disease, diabetes and the newly discovered covid19, could control and reduce the chance of these diseases to be fatal for the patient. With the advancement in machine learning and artificial intelligence, several ML and DL algorithms can be used to achieve this.

Following paper presents the use of machine learning and deep learning algorithms for prediction of diseases including heart diseases, which are the leading cause of deaths in the India, diabetes, in which blood glucose or blood sugar levels are too high, and covid19 which is the current pandemic around the globe.

I have applied different classification algorithms like Logistic regression, Random Forest, Decision tree, SVM, KNN, naive Bayes etc. for diseases like Heart/Diabetes to classify whether a patient is affected by the disease or not. Along with Deep learning algorithm like CNN, Transfer Learning on X-rays of lungs for classifying whether a person is covid positive or not. Then based on the accuracy of each model's performance, best one is selected.

The paper includes three different experiments on three different diseases - Heart, Diabetes and Covid19. The explanation and models for each one is described in this paper. Paper also includes entire project methodology of developing this whole project all the way from data collection to the app deployment phase.

Literature Review –

A. Detection of Heart Disease

[1] K. Deepika and S. Seema proposed predictive analytics to prevent and control the chronic disease with the help of machine learning techniques such as naive Bayes, support vector machine, decision tree, and artificial neural network and they have used UCI machine learning repository datasets to calculate the accuracy. Among them, Support vector machine gives the best accuracy of 95.55%.

[2] Parthiban and Srivatsa proposed a machine learning algorithm for detection and analysis of heart disease by utilizing Naive Bayes algorithm, Support vector machine. By utilizing Naïve Bayes algorithm provides 74% accuracy and SVM offer 94.60%.

B. Detection of Diabetes Disease

[3] Iyer proposed a machine learning algorithm to predict diabetic disorder by using Classification algorithms like Naive Bayes and Decision trees. Naive Bayes gives 79.56% accuracy and decision tree provides 76.95% accuracy. Clearly Naïve bayes performed better for them.

[4] Dash and Sen performed Machine learning algorithms for diagnosing diabetes disease. Logiboost, CART algorithms are used and Logiboost provides the correctness of 77.479% which is higher of them.

D. Detection of Covid19

[5] Baranwal, S. K., Jaiswal, K observed that Polynomial SVM is marginally better than Linear SVM but CNN perform much better than Polynomial SVM. So it is concluded that CNN is the best option for the most precise and reliable classification, owing to its significantly better performance under different metrics like ROC-AUC.

[6] Hossam H. Sultan has proposed a brain tumor image classification model using a 16-layer convolution neural network to classify three types of brain tumors using two publicly available dataset. First DL model classifies tumors into meningioma, Glioma and pituitary tumor and the three Glioma grades are classified by second model into Grade II, III and IV. The accuracy for first and second models is 96.13% and 98.7% respectively

[7] Saul and Can Jozef proposed deep learning architecture for the classification of pneumonia using chest x-rays. The model used convolutional neural networks and residual network architecture for classifying the images. An accuracy of 78.73% was obtained, [10] which surpassed the previously topscoring accuracy of 76.8%.

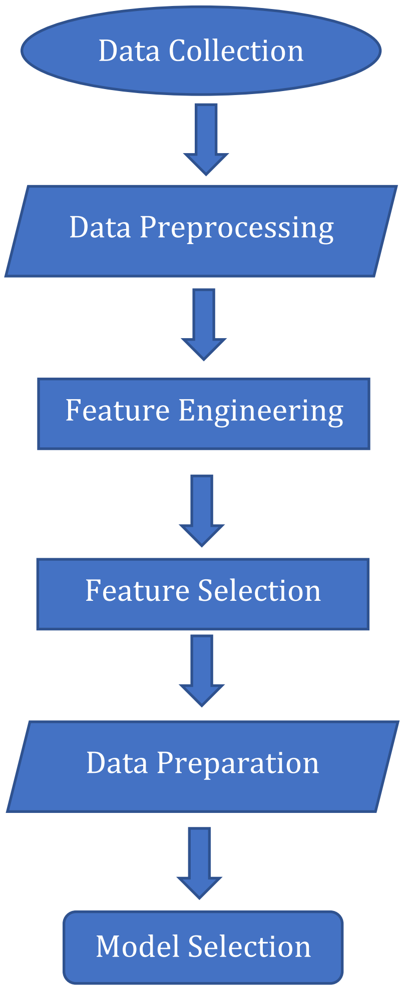
[8] The models proposed in this research for the detection of pediatric pneumonia on the frontal chest x-ray images using CNN and Transfer learning approaches have significant results. CNN gave 97% accuracy while Transfer learning algorithms like VGG16 and Inception V3 gave 98% each.

Research Methodology –

The methodology for two different techniques is discussed below.

I. Machine Learning

For detection of heart and diabetics the ML classification models are used.



Data Collection –

The datasets which I used are taken from Kaggle website. The two datasets, one for heart disease and other for diabetes disease.

Data Pre-processing –

After collecting the datasets, the next step was to clean it. Python language and Jupyter notebook tools were used to do the cleaning.

Feature Engineering –

The redundancies were removed like missing values. Then outliers were removed using z-scores and quartiles.

Feature Selection –

By observing the heatmaps non-correlated features were removed. Also the most important features for prediction were identified.

Data Preparation –

To run the models on the dataset following methods were applied-

- Features and Target values were separated and stored in X and y variables respectively.

- Encoding: The columns containing categorical values then encoded using dummy variables method. The datatypes of other numerical features were checked to be int.

- The datasets were divided into Training and Testing sets using train-test-split().

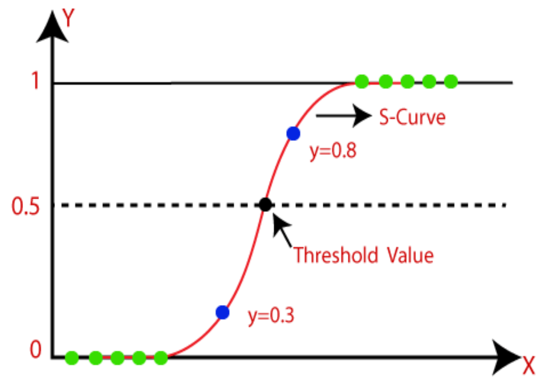
Model Selection –

Different Classification algorithms like Logistic Regression(LR), Random Forest(RF), Decision tree(DT), Support Vector Machine(SVM), K-Nearest Neighbors(KNN), Naïve Bayes(NB) were applied on our training set using sklearn library. Based on the measures like Confusion Matrix, Classification report, Accuracy and ROC curve, the best model was selected. And the performance of the model was checked for the test set.

Machine Learning Models –

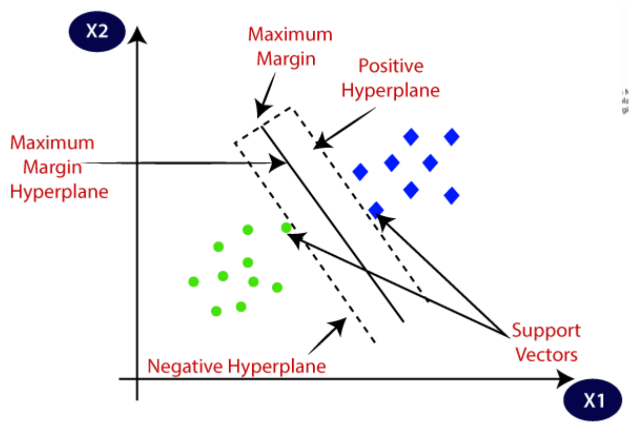
1. Logistic Regression

It is a calculation used to predict a binary outcome, either 0 or 1. It is used for predicting the categorical dependent variable using a given set of independent variables.



2. Support Vector Machine

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

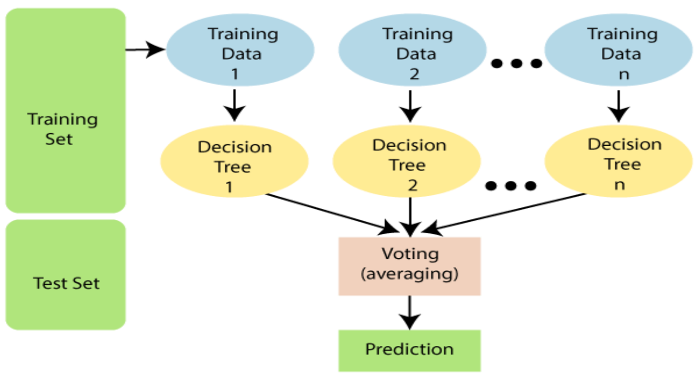


3. Decision Trees

It works like a flow chart, separating data points into two similar categories at a time from the "tree trunk" to "branches," to "leaves," where the categories become more finitely similar. This creates categories within categories, allowing for organic classification with limited human supervision.

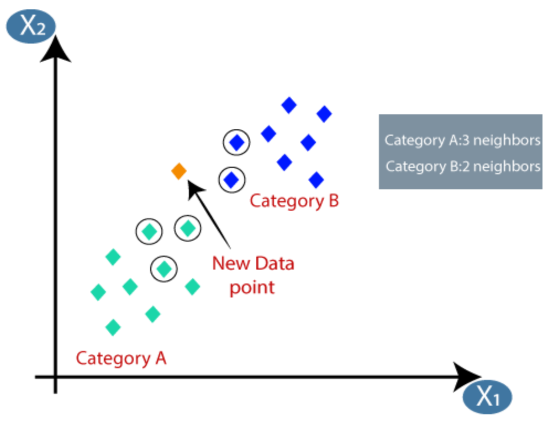
4. Random Forest

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



5. K-nearest Neighbor

K-nearest neighbors (k-NN) is a pattern recognition algorithm that uses training datasets to find the *k* closest relatives in future examples. If *k* = 1, then it would be placed in the class nearest 1. *K* is classified by a plurality poll of its neighbors.



6. Naïve Bayes

Naive Bayes calculates the possibility of whether a data point belongs within a certain category or does not. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

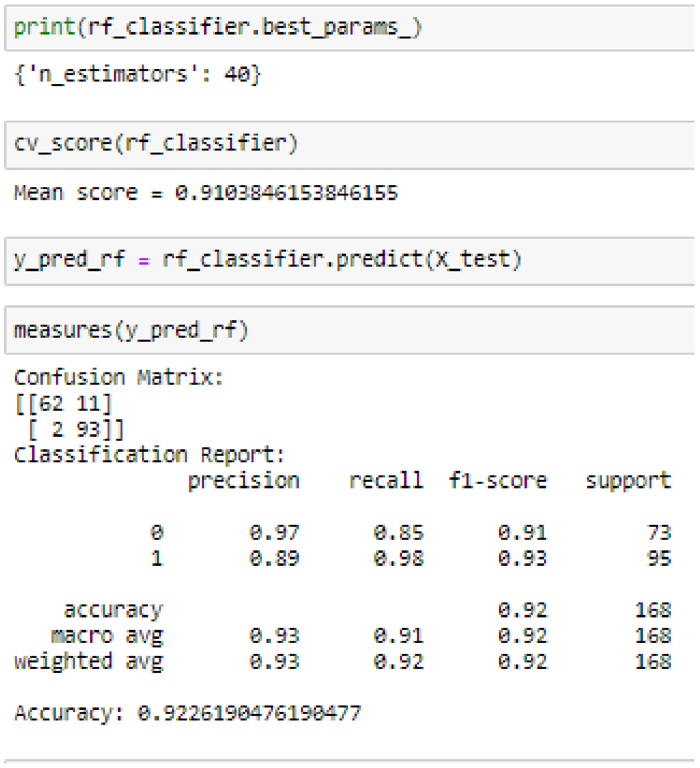
A. Detection of Heart Disease

The Heart Disease Dataset consists of 11 input attributes including age, gender, type of chest pain, blood pressure, cholesterol, blood sugar level, electrocardiograph result, maximum heart rate, exercise-induced angina, old peak, ST Slope and one target variable containing binary data 0 for affected and 1 for not affected. The dataset contains 1190 instances. No null values were present in the dataset. After removing outliers 559 instances remained and sex, blood sugar features came out to be non-correlated ad hence were dropped.
Categorical features like chest pain, ECG result, exercise angina, slope of ST was encoded using dummy variables. Data then split into train as X_train and test as X_test set. Feature scaling was performed using StandardScalar() of sklearn library. Finally, model selection is done and the best model came out to be Random Forest with the accuracy of 96%.

Performance Measures:

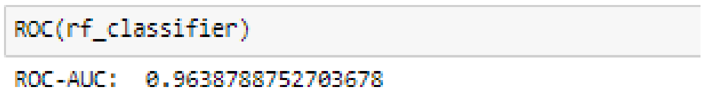
Model	Mean Cross-validation	Accuracy	ROC-AUC
Logistic Regression	86%	82%	90%
Decision tree	85%	89%	88%
Random forest	91%	92%	96%
K-nearest neighbor	84%	80%	92%
Naïve bayes	83%	83%	90%
Support vector machine	86%	83%	85%

Random Forest Performance –

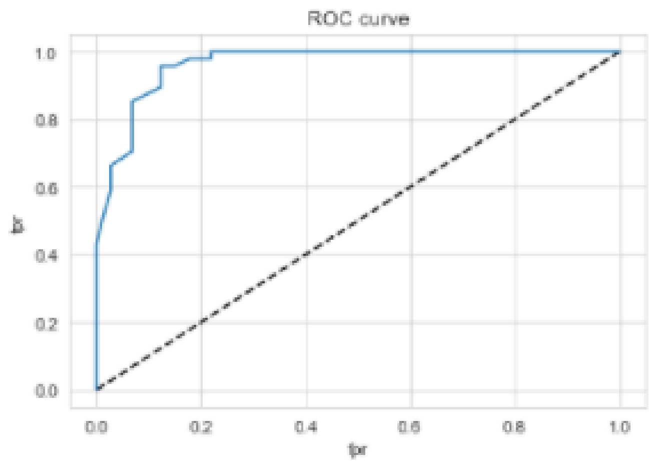


```
print(rf_classifier.best_params_)
{'n_estimators': 40}

cv_score(rf_classifier)
Mean score = 0.9103846153846155

y_pred_rf = rf_classifier.predict(X_test)

measures(y_pred_rf)
Confusion Matrix:
[[62 11]
 [ 2 93]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.85      0.91        73
           1       0.89      0.98      0.93        95

    accuracy                           0.92       168
   macro avg       0.93      0.91      0.92       168
weighted avg       0.93      0.92      0.92       168

Accuracy: 0.9226190476190477
```

Area Under the Curve -



```
ROC(rf_classifier)
ROC-AUC:  0.9638788752703678
```



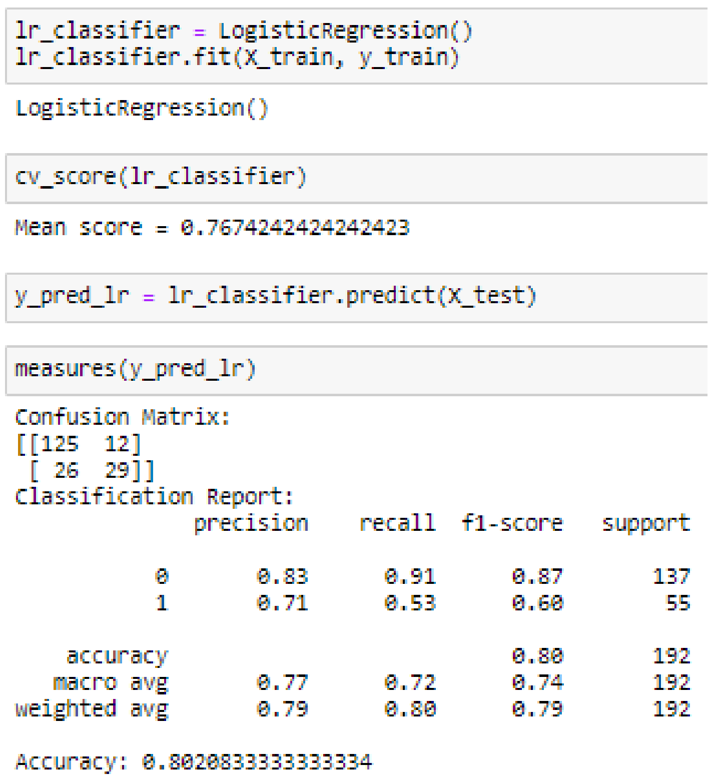
B. Detection of Diabetes disease

The Diabetics Disease Dataset consists of 8 input attributes including age, BMI, pregnancies, blood pressure, glucose, skin thickness, insulin, Diabetes Pedigree Function and one target variable containing binary data 0 for affected and 1 for not affected. The dataset contains 768 instances. No null values were present in the dataset. After removing outliers 639 instances remained all the features came out to be correlated and important for the prediction.
No categorical features were present in the dataset. Data then split into train as X_train and test as X_test set. Feature scaling was performed using StandardScalar() of sklearn library. Finally, model selection is done and the best model came out to be Logistic regression with the accuracy of 86%.

Performance Measures:

Model	Mean Cross-validation	Accuracy	ROC-AUC
Logistic Regression	77%	80%	86%
Decision tree	70%	74%	68%
Random forest	77%	77%	80%
K-nearest neighbor	75%	75%	80%
Naïve bayes	74%	78%	82%
Support vector machine	77%	80%	80%

Logistic Regression performance-



```
lr_classifier = LogisticRegression()
lr_classifier.fit(X_train, y_train)
```

```
LogisticRegression()
```

```
cv_score(lr_classifier)
```

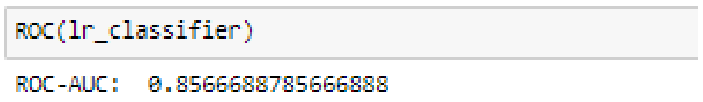
```
Mean score = 0.7674242424242423
```

```
y_pred_lr = lr_classifier.predict(X_test)
```

```
measures(y_pred_lr)
```

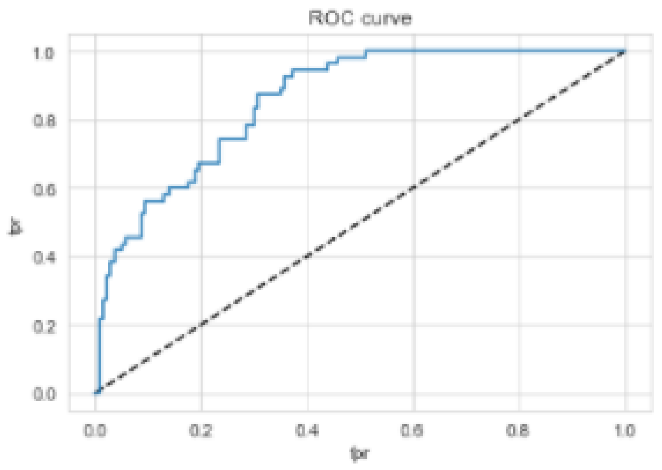
```
Confusion Matrix:
[[125  12]
 [ 26  29]]
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.91      0.87       137
           1       0.71      0.53      0.60        55

    accuracy                           0.80       192
   macro avg       0.77      0.72      0.74       192
weighted avg       0.79      0.80      0.79       192

Accuracy: 0.8020833333333334
```

Area Under the Curve –



```
ROC(lr_classifier)
```

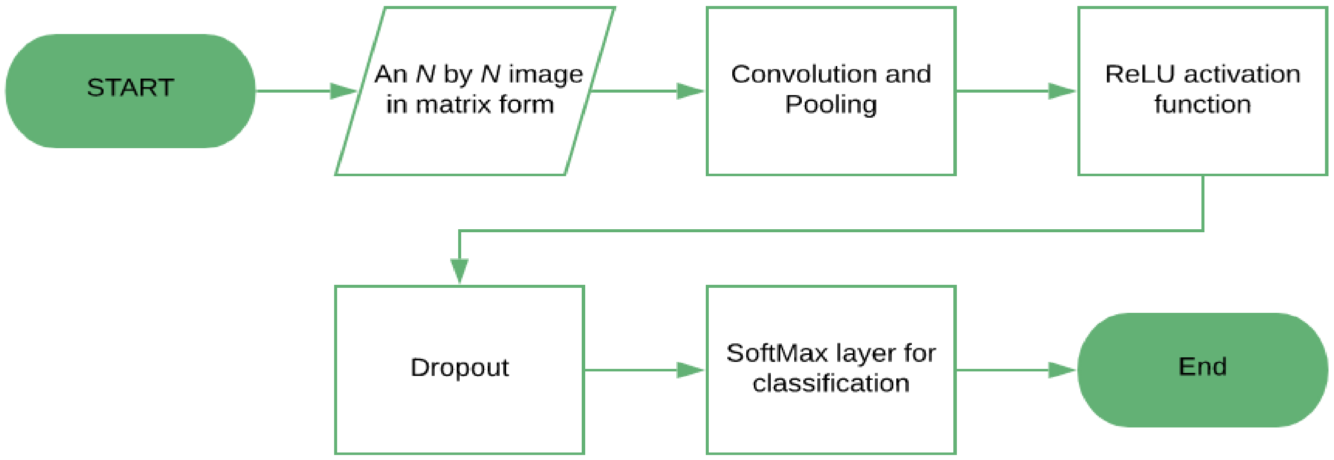
```
ROC-AUC:  0.8566688785666888
```



II. Deep Learning

For classifying X-ray images to detect Covid19 infection the Deep Learning models are used.
I have used Convolutional Neural Network (CNN) and InceptionV3 - Transfer Learning Models for Covid19 detection.

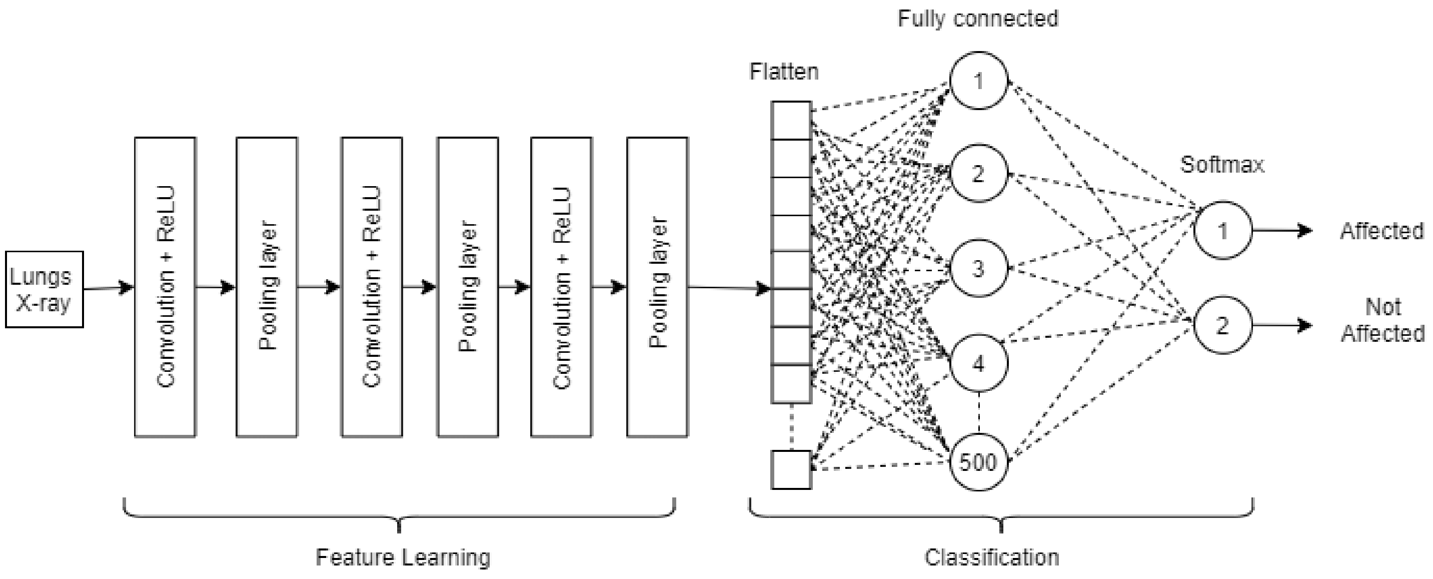
CNN Architecture –


START
An N by N image in matrix form
Convolution and Pooling
ReLU activation function
Dropout
SoftMax layer for classification
End

Convolutional Neural Network (CNN) is a well-known neural network specially for image recognition and classification. CNN is highly excellent in extracting complex features for classifications. CNN consist of neurons where weight and bias can be learned. Each neuron receives some input; weighted sum is taken then given to the activation function. CNN uses successive convolution layer and nonlinear ReLU function to extract valuable feature with specific dimension. MaxPooling layer is used to max pool the value from given size matrix and same is used for the next layers. Then flatten is used to flatten the dimensions of the image obtained after convolving it. Dense is used to make this fully connected model. In Fully connected layer, each neuron is connected to every other neuron of previous dense layer. Back-propagation and gradient descent are used while training the network. Dropout layer is used to deal with the overfitting. Softmax function is probability distribution to limit all class output value between 0 and 1. CNN provides feature maps which helps neural network to learn small features of the image depending on the depth of hidden layers. Dense is the output layer containing number of neurons equal to the number of output categories.

C. Detection of Covid19

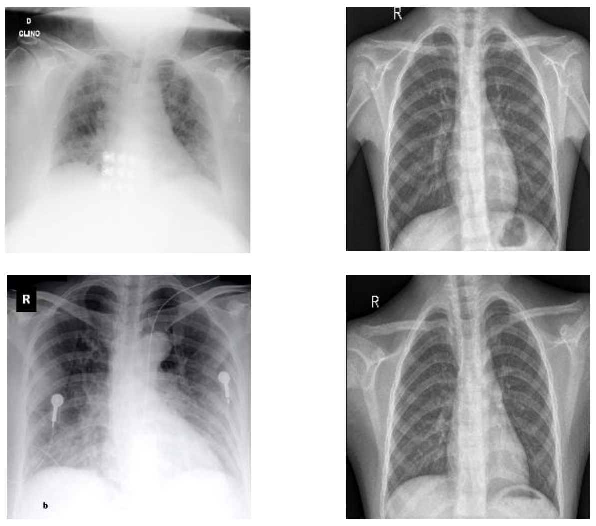
Proposed CNN Architecture:



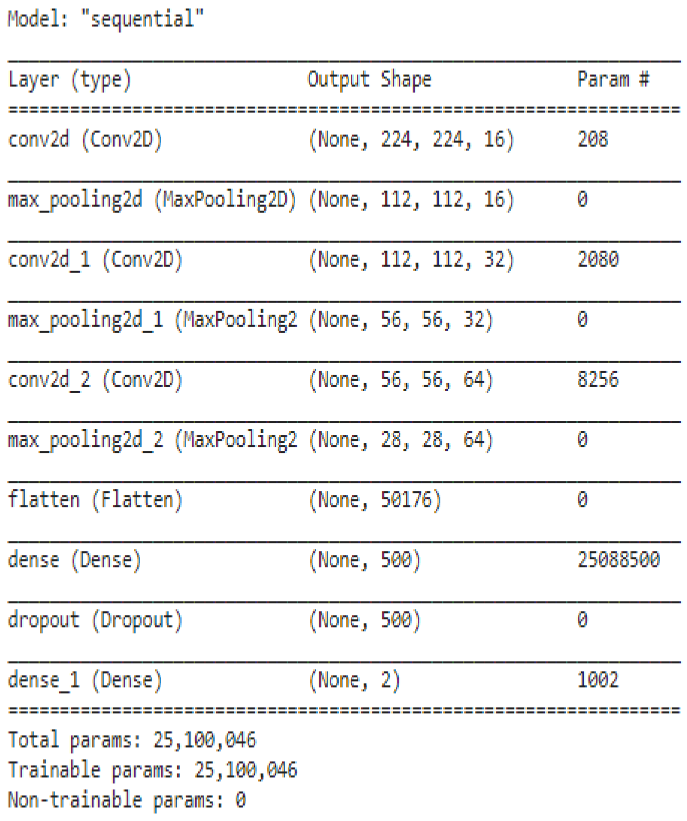
The dataset employed in this work is picked from Kaggle. It consists of 3000 images of chest X-ray. The dataset is split into two portions namely training dataset and testing dataset. The train and test set further have two categories Affected and Not affected.

Eg.

Affected Not affected



Model summary

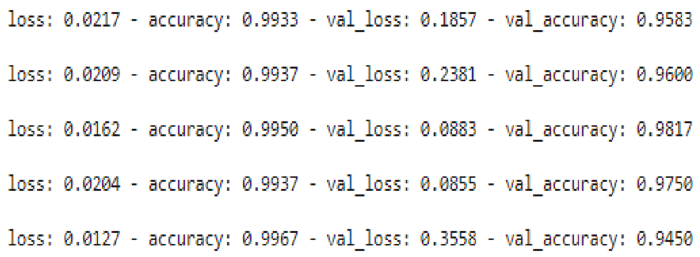


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 16)      208
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 32)      2080
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 64)        8256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 50176)             0
_________________________________________________________________
dense (Dense)                (None, 500)               25088500
_________________________________________________________________
dropout (Dropout)            (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1002
=================================================================
Total params: 25,100,046
Trainable params: 25,100,046
Non-trainable params: 0
```

After getting the dataset next step is to create a CNN model from scratch.

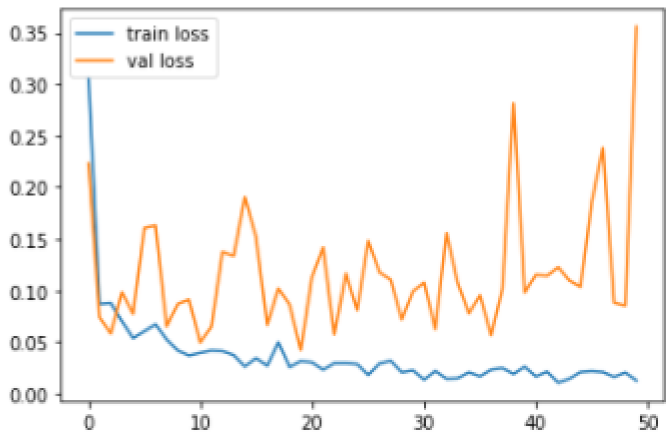
CNN consist of convolution, max pooling and a classification layer. A series of convolution and max-pooling layers act as a feature extractor that is known as feature learning part.
In CNN, 3 Conv2D layers with 16,32 and 64 filters each, along with 3 MaxPooling2D layers of size 2 and a ReLU activator are used. ReLU is a popular activation function which is generally used in neural networks especially in CNNs. ReLU function introduces nonlinearity to the model.
Features extracted from the feature extractor part of the CNN are given as input to the dense layer which classifies the image. As the dense layer takes 1-Dimensional input, the flatten layer flattens the feature data and gives a 1-Dimensional output which is fed to the dense layer.
While training a CNN it might be possible that output through a certain layer is more dependent on a few selected neurons. To reduce this dependency and prevent overfitting, the concept of dropout is introduced. During training, in each epoch, a neuron is momentarily dropped with a dropout probability p. Due to this, all the inputs and outputs to this neuron become disabled in the current epoch which results in some loss of data, enhancing regularization in the model so that at times it would predict with much higher accuracy. Here, one dropout of p=0.2 is used at the dense layer.
Initially, the images in the dataset were of varying sizes. However, all the input images given to CNN should be of same size. To solve this problem, all the images in the dataset are resized to 224 × 224.
To avoid overfitting and to enhance the generalization capabilities of the proposed CNN architectures, various in-place data augmentation techniques such as rescale, shear, zoom, Horizontal-flip are used.
The model is then trained for 50 epochs with Adam optimizer. The batch size is 16. During training and validation, loss is calculated using categorical cross-entropy. Training accuracy and validation accuracy is calculated and it came out as, 99.67% and 97.50%, with very less loss as well, 0.0127 and 0.3558 respectively.

Output – Last 5 epochs

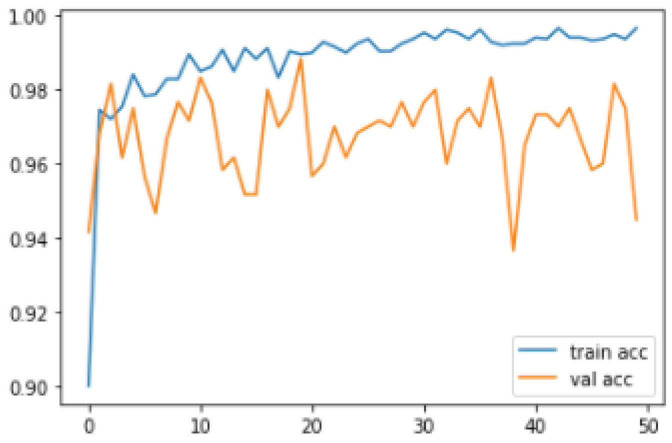


```
loss: 0.0217 - accuracy: 0.9933 - val_loss: 0.1857 - val_accuracy: 0.9583

loss: 0.0209 - accuracy: 0.9937 - val_loss: 0.2381 - val_accuracy: 0.9600

loss: 0.0162 - accuracy: 0.9950 - val_loss: 0.0883 - val_accuracy: 0.9817

loss: 0.0204 - accuracy: 0.9937 - val_loss: 0.0855 - val_accuracy: 0.9750

loss: 0.0127 - accuracy: 0.9967 - val_loss: 0.3558 - val_accuracy: 0.9450
```

Train loss vs Validation loss



Train accuracy vs Validation accuracy



Transfer Learning – InceptionV3

Though training a CNN from scratch is possible for small projects, most applications require the training of very large CNN's and this takes extremely huge amounts of processed data and computational power. And both of these are not found so easily these days. That's where transfer learning comes into play.

Transfer learning reduces the training time and data needed to achieve a custom task. It takes a CNN that has been pre-trained(one that has been trained on millions of images belonging to 1000's of classes, on several high power GPU's for several days e.g. imagenet), removes the last fully-connected layer and replaces it with our custom fully-connected layer, treating the original CNN as a feature extractor for the new dataset. Once replaced the last fully-connected layer we train the classifier for the new dataset.

The advantages of transfer learning are,
1: There is no need of an extremely large training dataset.
2: Not much computational power is required. As we are using pre-trained weights and only have to learn the weights of the last few layers.

There are several models that have been trained on the image net dataset and have been open sourced. For example, VGG-16, ResNet50, Inception-V3 etc.

The building of a model is a 3-step process:

1: Importing the pre-trained model and adding the dense layers.
2: Loading train data into ImageDataGenerators.
3: Training and Evaluating model.

Import the pre-trained InceptionV3 model.

```
from tensorflow.keras.applications.inception_v3  import InceptionV3
```

The InceptionV3 will have a last layer consisting of 1000 neurons (one for each class). We want as many neurons in the last layer of the network as the number of classes we wish to identify. So we discard the 1000 neuron layer and add our own last layer for the network which have two neurons for two classes namely 'affected' and 'not affected'. Next added a few dense layers so that our model can learn more complex functions. The last layer which contains two neurons has softmax activation function.

```
prediction = Dense(len(folders), activation='softmax')(x)
```

As we will be using the pre-trained weights, that our model has been trained on, we have to set all the weights to be non-trainable. We will only be training the last Dense layers that we have added previously.

```
for layer in inception.layers:
    layer.trainable = False
```

loading the training data into the ImageDataGenerator. ImageDataGenerators are inbuilt in keras and help us to train our model. We just have to specify the path to our training data and it automatically sends the data for training, in batches.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Finally, training the model on the Covid dataset. The model is then trained for 20 epochs with Adam optimizer. The batch size is 16. During training and validation, loss is calculated using categorical cross-entropy. Training accuracy and validation accuracy is calculated and is given as, 99.29% and 97.67% along with loss 0.2290 and 0.7242 respectively.

Output - Last 5 Epochs:

```
loss: 0.2373 - accuracy: 0.9912 - val_loss: 0.1878 - val_accuracy: 0.9867

loss: 0.4101 - accuracy: 0.9867 - val_loss: 1.3386 - val_accuracy: 0.9550

loss: 0.0782 - accuracy: 0.9950 - val_loss: 2.8086 - val_accuracy: 0.9300

loss: 0.2270 - accuracy: 0.9917 - val_loss: 0.5634 - val_accuracy: 0.9783

loss: 0.2290 - accuracy: 0.9929 - val_loss: 0.7242 - val_accuracy: 0.9767
```

Tools and Requirements –

- Kaggle for data collection
- Python for Data pre-processing
- Jupyter notebook and google Collaboratory for analysis
- Sklearn and TensorFlow for model selection
- Pickle for storing and loading the model
- VS Code and PyCharm for building web app
- Flask framework for coding of web app
- CSS and JavaScript for UI Design.
- Heroku for deployment of the app.

Conclusion –

Machine learning (ML) and Deep Learning (DL) techniques are crucial in the medical field. This paper presents various ML and DL techniques for detection of various diseases like heart disease, diabetes disease and covid19. To my best understanding, the models built according to the proposed methods exhibits better accuracy than the existing ones from the earlier study. It is recognized that Random Forest provides 96% of accuracy for the diagnosis of heart disease, Logistic regression provides 86% of accuracy for the detection of diabetic disease, and CNN gives 99.67% of accuracy for the covid19 detection. After all this analysis a proper platform has been designed named “Disease Detection” integrating all these different disease at one place.

Future scope –

Current system only gives prediction of whether the person is diagnosed with the disease or not, so in future work user can be suggested with diagnosis or preventive measures based on severity of the disease. Detection of more diseases like breast cancer and Brain tumor can be done. The models have been applied on small datasets due to less processing power, but large datasets can be used for making these models more robust.

References –

[1] *K. Deepika and S. Seema, “Predictive analytics to prevent and control chronic diseases,” Proc. 2016 2nd Int. Conf. Appl. Theor. Comput. Commun. Technol. iCATccT 2016, no. January 2016, pp. 381–386, 2017, doi: 10.1109/ICATCCT.2016.7912028.*

[2] *Parthiban, G. and Srivatsa, S.K., “Applying Machine Learning Methods in Diagnosing Heart Disease for Diabetic Patients”, International Journal of Applied Information Systems, vol.3, pp.25-30, 2012*

[3] *Iyer A., Jeyalatha S. and Sumbaly R, “Diagnosis of Diabetes Using Classification Mining Techniques”, International Journal of Data Mining & Knowledge Management Process (IJDKP), vol.5, pp. 1-14,2015*

[4] *Sen S.K. and Dash S, “Application of Meta Learning Algorithms for the Prediction of Diabetes Disease”, International Journal of Advance Research in Computer Science and Management Studies, vol.2, pp.396- 401, 2014.*

[5] *Baranwal, S. K., Jaiswal, K., Vaibhav, K., Kumar, A., & Srikantaswamy, R. (2020). Performance analysis of Brain Tumor Image Classification using CNN and SVM. 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA).*

[6] *Hossam H. Sultan, Nancy M. Salem and Walid Al-Atabany, ”Multiclassification of Brain Tumor Images using Deep Neural Networks,” IEEE Special Section on Deep Learning for Computer-Aided Medical Diagnosis, IEEE Access, June 2019.*

[7] *Saul, Can Jozef, Deniz Yagmur Urey, and Can DorukTaktakoglu. "Early Diagnosis of Pneumonia with Deep Learning." arXiv preprint arXiv:1904.00937 (2019)*

[8] *Labhane, G., Pansare, R., Maheshwari, S., Tiwari, R., & Shukla, A. (2020). Detection of Pediatric Pneumonia from Chest X-Ray Images using CNN and Transfer Learning. 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE).*