Medical project: Lung Cancer Prediction

Problem statement:

Predict whether the patient have lung cancer or not using the below features.

	Name	Surname	Age	Smokes	AreaQ	Alkhol	Result
0	John	Wick	35	3	5	4	1
1	John	Constantine	27	20	2	5	1
2	Camela	Anderson	30	0	5	2	0
3	Alex	Telles	28	0	8	1	0
4	Diego	Maradona	68	4	5	6	1

Result column is target variable which says whether the patient have cancer or not.

Exploratory Data Analysis – EDA:

Check for null values and data type in dataset.

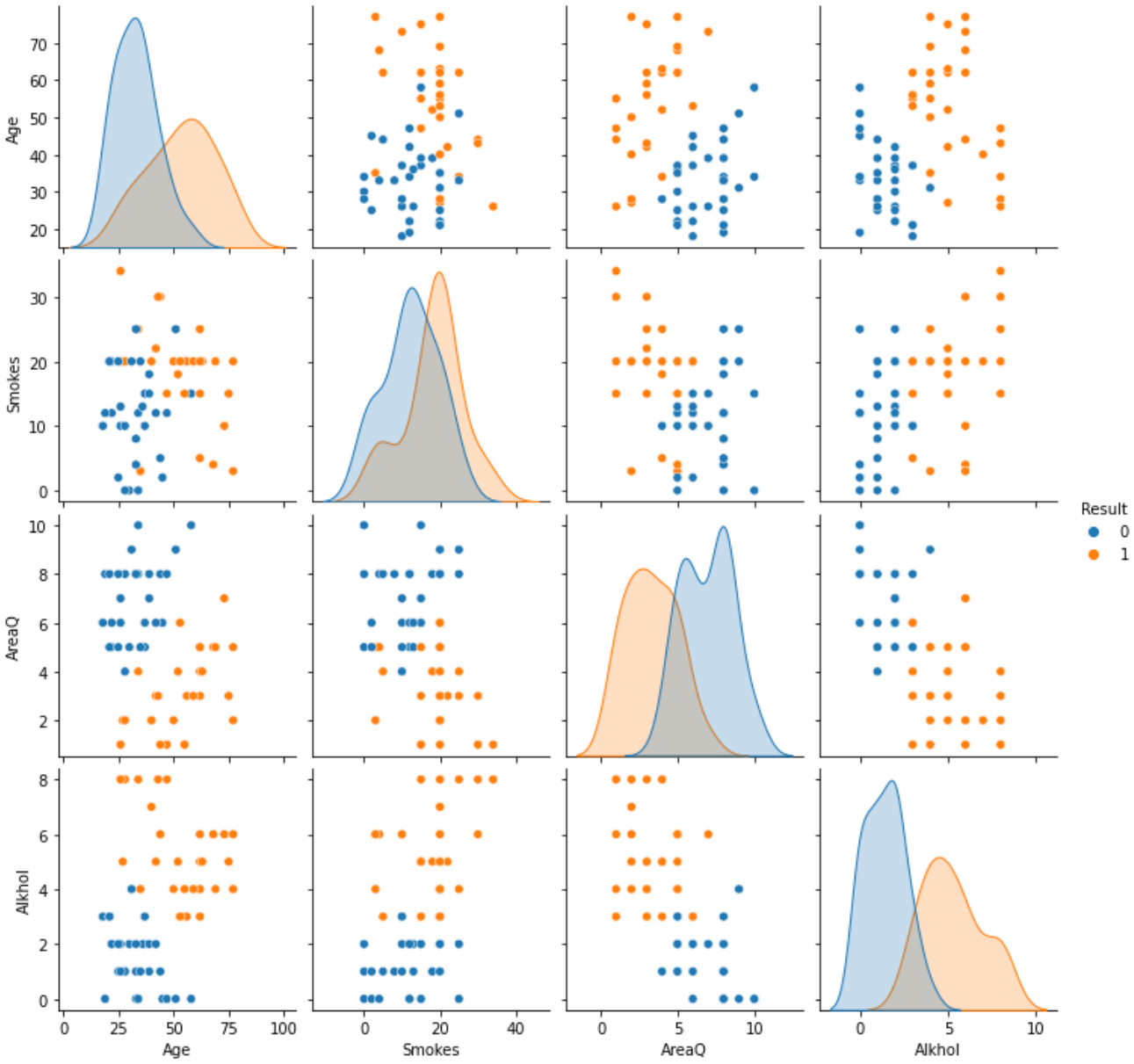
```
] data.isnull().sum()
```

```
Name       0
Surname    0
Age        0
Smokes     0
AreaQ      0
Alkhol     0
Result     0
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59 entries, 0 to 58
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Name     59 non-null     object
 1   Surname  59 non-null     object
 2   Age      59 non-null     int64
 3   Smokes   59 non-null     int64
 4   AreaQ    59 non-null     int64
 5   Alkhol   59 non-null     int64
 6   Result   59 non-null     int64
dtypes: int64(5), object(2)
memory usage: 3.4+ KB
```

- No null values
- All necessary variables are numeric
- Hence no action required in data pre-processing

Pair plot among all numeric features:



Observations:

- Lower the age and smoke, less chance of lung cancer
- Higher the alkhol and lower the areaQ, more chance of lung cancer.
- This clear colour segregation in each plot shows that each variable have significant effect on lung cancer.

Correlation heatmap:

Observations:

- Features – Age and Alkhol have strong positive correlation
- This indicates that if age and alkhol level increases, more possibility of lung cancer.
- Feature: AreaQ have strong negative correlation, this indicates that how small the values – AreaQ that much high possibility of getting lung cancer.

ANN - Model Summary:

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_16 (Dense)            (None, 128)               640

 dense_17 (Dense)            (None, 64)                8256

 dense_18 (Dense)            (None, 1)                 65

=================================================================
Total params: 8,961
Trainable params: 8,961
Non-trainable params: 0
_________________________________________________________________
```

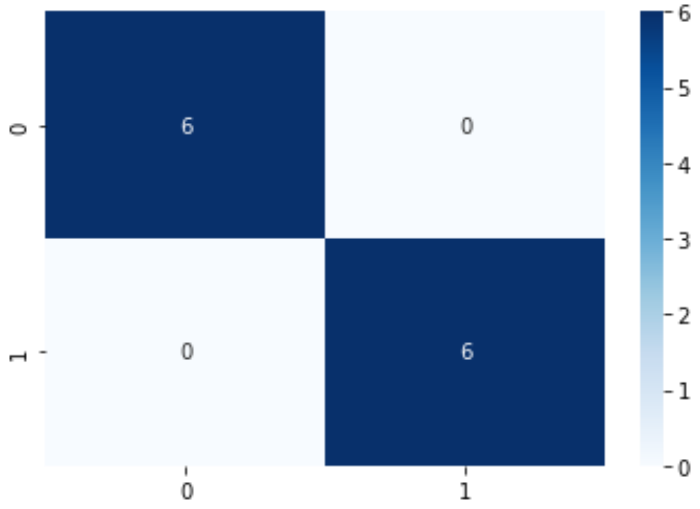
Hidden layer 1 – 128 neurons

Hidden layer 2 – 64 neurons

Output layer – 1 neuron with sigmoid activation

Performance:

Confusion matrix



No false positive and false negative which means accuracy is 100%.

Classification report:

```
[3] print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00         6

    accuracy                           1.00        12
   macro avg       1.00      1.00      1.00        12
weighted avg       1.00      1.00      1.00        12
```

All metrics are 100%, our model works in best way.

Hyper-parameter tuning:

```
Accuracy of the model with 4, 64 neurons in each hidden layers:  0.4166666666666667

Accuracy of the model with 16, 64 neurons in each hidden layers:  0.8333333333333334

Accuracy of the model with 32, 64 neurons in each hidden layers:  0.9166666666666666

Accuracy of the model with 64, 64 neurons in each hidden layers:  1.0

Accuracy of the model with 128, 64 neurons in each hidden layers:  1.0
```

As we increase the no of neurons in the first hidden layer, we obtain the best accuracy.

Different optimizers with 20 epochs:

```
Accuracy of the model with <keras.optimizer_v2.adam.Adam object at 0x7faf8810fed0>:  1.0

Accuracy of the model with <keras.optimizer_v2.gradient_descent.SGD object at 0x7faf8810fc10>:  0.8333333333333334

Accuracy of the model with <keras.optimizer_v2.rmsprop.RMSprop object at 0x7faf8810f950>:  0.9166666666666666

Accuracy of the model with <keras.optimizer_v2.adadelta.Adadelta object at 0x7faf8810fad0>:  0.5

Accuracy of the model with <keras.optimizer_v2.adagrad.Adagrad object at 0x7faf8810fe50>:  1.0
```

Adam and Adagrad is the best optimizer for our model which gives 100% accuracy.

New input is given - predict the output of your model:

```
age = int(input("Age: "))
smoke = int(input("Smokes: "))
areaq = int(input("AreaQ: "))
alkhol = int(input("Alkhol: "))

xnew = [[age,smoke,areaq,alkhol]]
pred = model.predict(xnew)
y_pred=(pred>0.5)
if y_pred==True:
  print("The person has Lung cancer")
else:
  print("Free from lung cancer")
```

```
Age: 77
Smokes: 20
AreaQ: 5
Alkhol: 4
The person has Lung cancer
```

Based on the independent variables our model predicted that the patient has lung cancer.

Conclusion:

Our model performed well with below parameters

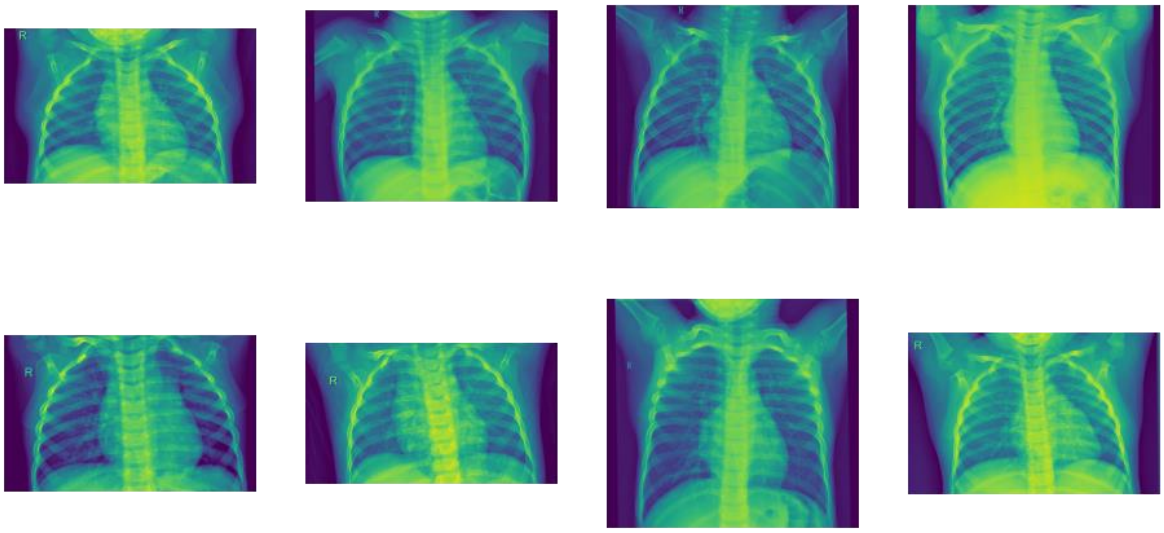
- 20 epochs
- Two hidden layers
- First hidden layers: 128 neurons
- Second hidden layers: 64 neurons.
- Optimizer: Adam or Adagrad

Covid X-Ray Classification

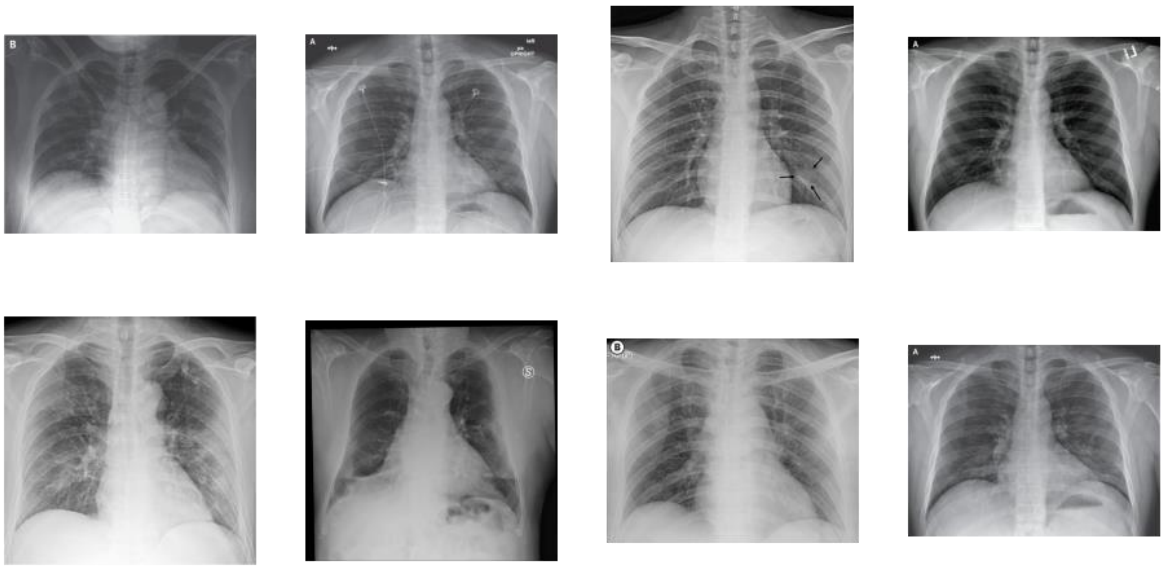
Problem statement:

Given with the X-Ray images of the covid-19 patients, predict whether the person has Covid or not.

Normal person X-ray images

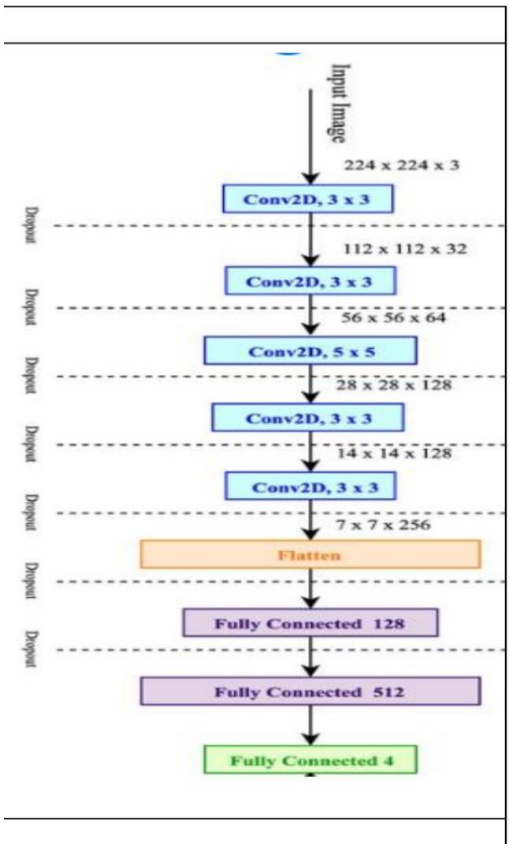


Covid patient - X-ray images



Building the Models:

Model 1 - CNN model given in the below figure.



We implemented the above architecture.

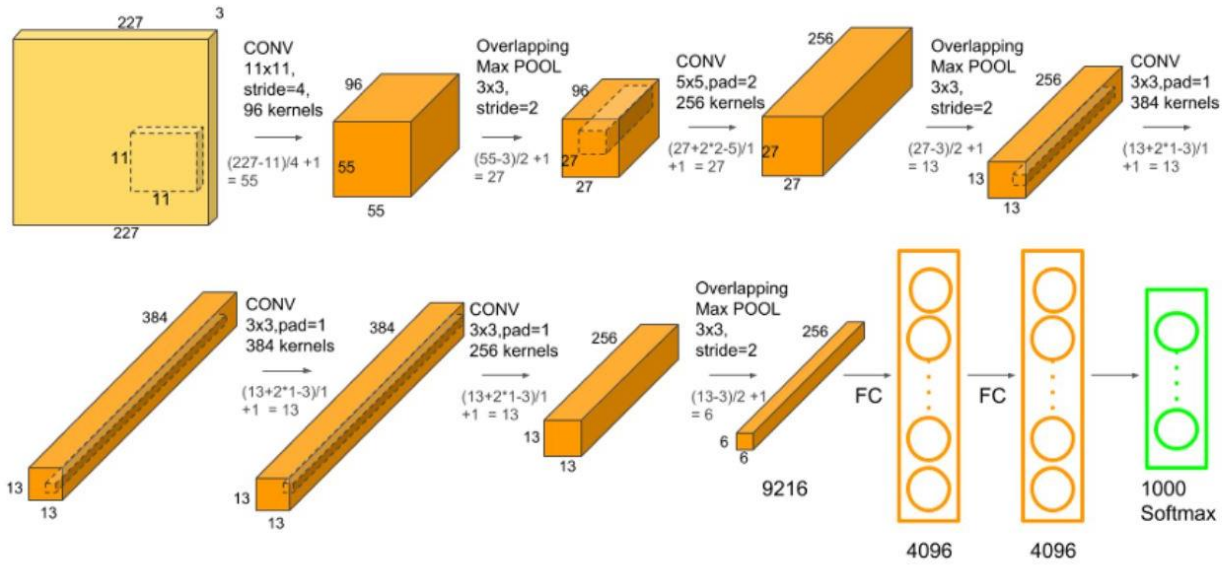
Model summary:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 112, 112, 32)      896

 dropout (Dropout)           (None, 112, 112, 32)      0

 max_pooling2d (MaxPooling2D  (None, 56, 56, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 56, 56, 64)        18496

 dropout_1 (Dropout)         (None, 56, 56, 64)        0

 max_pooling2d_1 (MaxPooling  (None, 28, 28, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 28, 28, 128)       204928

 dropout_2 (Dropout)         (None, 28, 28, 128)       0

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 14, 14, 128)       147584

 dropout_3 (Dropout)         (None, 14, 14, 128)       0

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
```

```
conv2d_4 (Conv2D)            (None, 7, 7, 256)         295168

dropout_4 (Dropout)          (None, 7, 7, 256)         0

max_pooling2d_4 (MaxPooling  (None, 3, 3, 256)         0
2D)

flatten (Flatten)            (None, 2304)              0

dropout_5 (Dropout)          (None, 2304)              0

dense (Dense)                (None, 128)               295040

dropout_6 (Dropout)          (None, 128)               0

dense_1 (Dense)              (None, 512)               66048

dense_2 (Dense)              (None, 4)                 2052

dense_3 (Dense)              (None, 1)                 5

=================================================================
Total params: 1,030,217
Trainable params: 1,030,217
Non-trainable params: 0
_________________________________________________________________
```

Model – 2 – AlexNet architecture:

We implemented the below structure.



Changes in Output layer:

1 neuron and sigmoid – Activation function.

Model summary:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 54, 54, 96)        34944

max_pooling2d_5 (MaxPooling  (None, 26, 26, 96)        0
2D)

conv2d_6 (Conv2D)            (None, 26, 26, 256)       614656

max_pooling2d_6 (MaxPooling  (None, 12, 12, 256)       0
2D)

conv2d_7 (Conv2D)            (None, 12, 12, 384)       885120

conv2d_8 (Conv2D)            (None, 12, 12, 384)       1327488

conv2d_9 (Conv2D)            (None, 12, 12, 256)       884992

max_pooling2d_7 (MaxPooling  (None, 5, 5, 256)         0
2D)

flatten_1 (Flatten)          (None, 6400)              0

dense_4 (Dense)              (None, 4096)              26218496

dropout_7 (Dropout)          (None, 4096)              0

 dense_5 (Dense)             (None, 4096)              16781312

 dropout_8 (Dropout)         (None, 4096)              0

 dense_6 (Dense)             (None, 1)                 4097

=================================================================
Total params: 46,751,105
Trainable params: 46,751,105
Non-trainable params: 0
_________________________________________________________________
```

Performance:

```
4/4 [==============================] - 19s 5s/step - loss: 0.2680 - accuracy: 0.8917 - val_loss
```

```
[ ] cnn_model.evaluate(test_dataset)

2/2 [==============================] - 4s 3s/step - loss: 0.5372 - accuracy: 0.9750
[0.5371925234794617, 0.9750000238418579]
```

```
[ ] alex_model.evaluate(test_dataset)

2/2 [==============================] - 2s 273ms/step - loss: 0.0719 - accuracy: 0.9750
[0.07185763120651245, 0.9750000238418579]
```

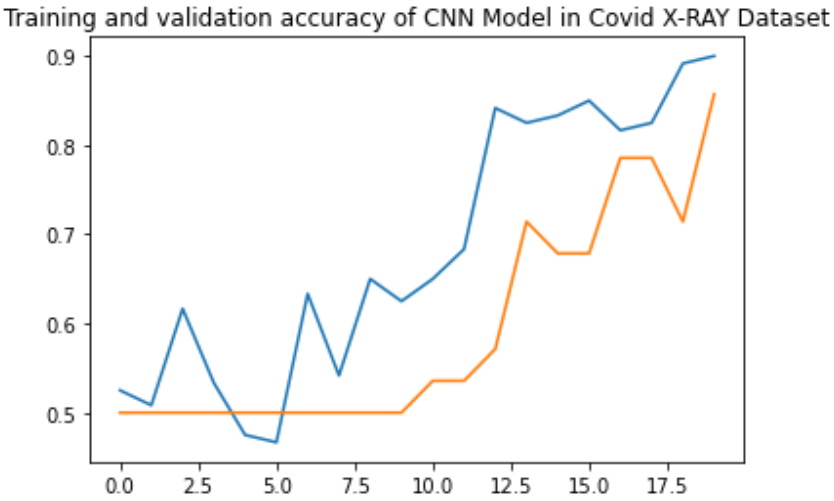
Figure based CNN model – Model 1:

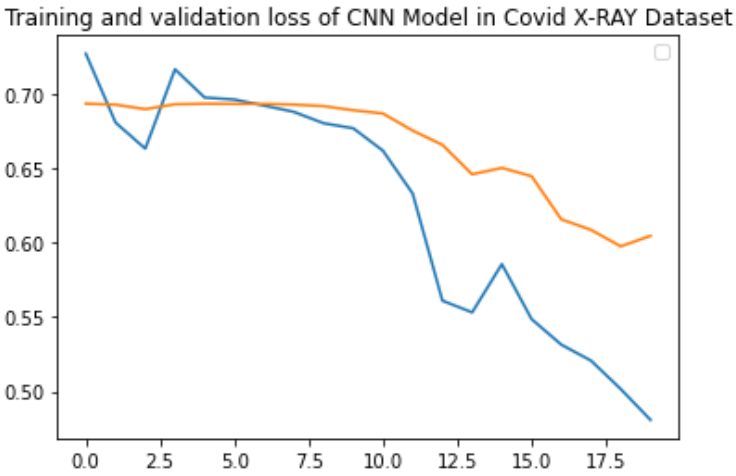
- Accuracy: 97.5%
- Loss: 0.53

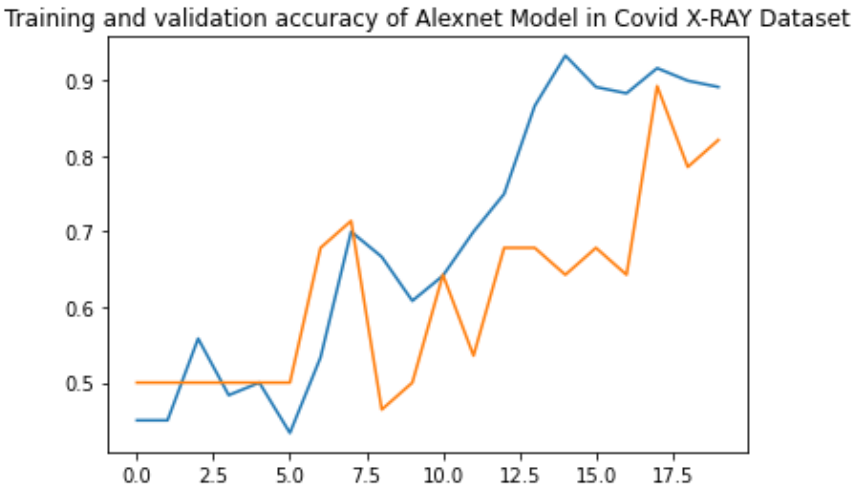
AlexNet Model -2 :

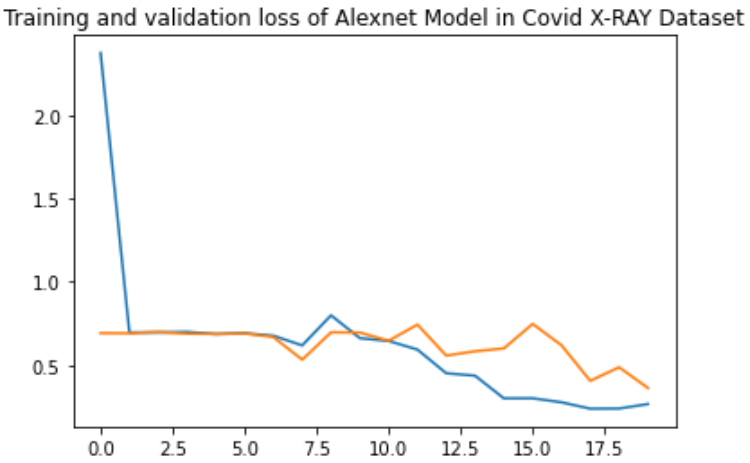
- Accuracy: 97.5%
- Loss: 0.07

Accuracy and Loss plots:









Conclusion:

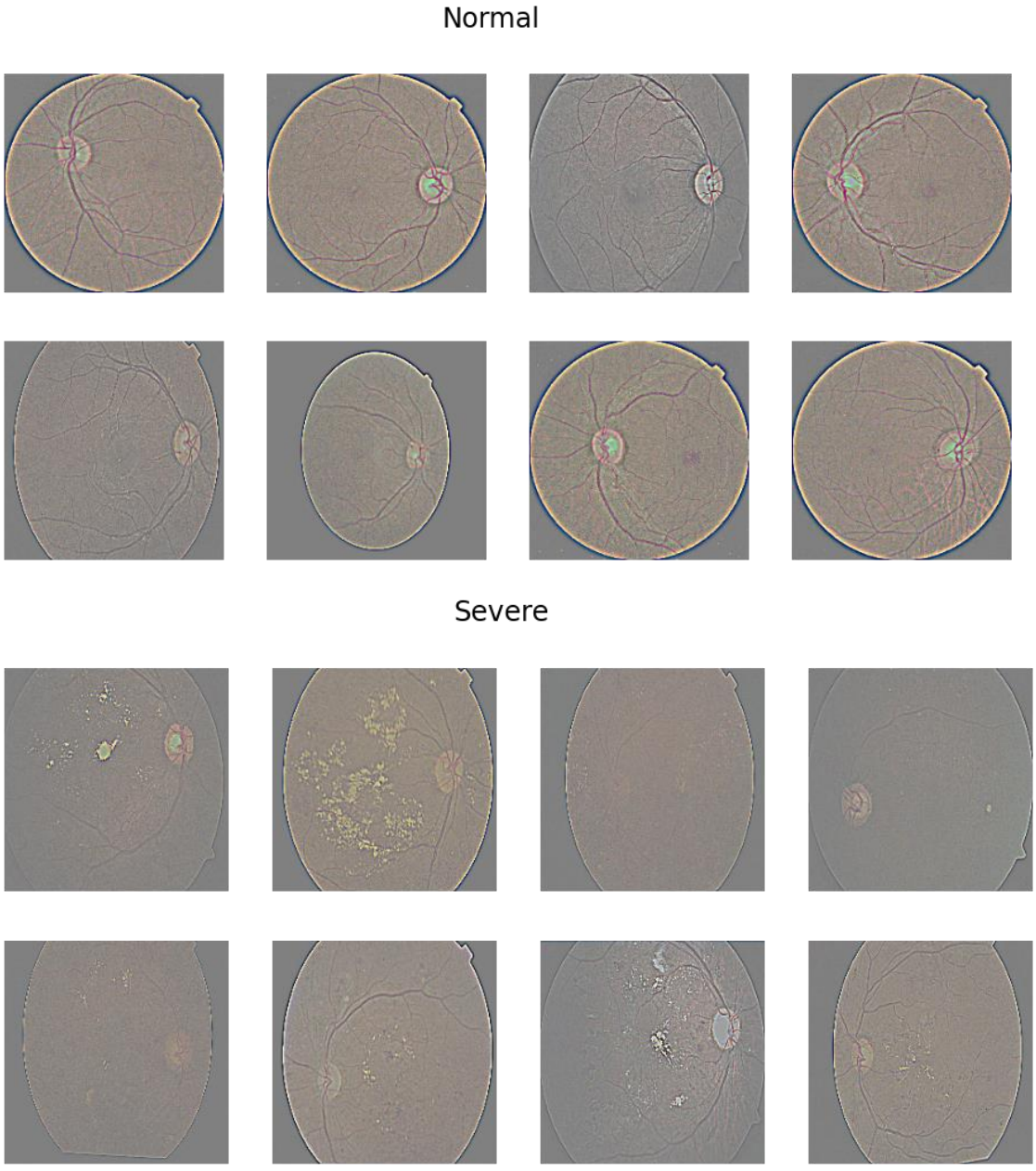
- In both models, training and validation – loss is decreasing as epoch increases
- Training and validation – accuracy is increasing as epoch increases
- This indicates the model is best fit, there's no underfit and overfit.
- Both a models performed well equally with same high accuracy (97.5%)
- However, model 2 is elected as best one because it has lesser loss (0.07<0.53)
- Therefore AlexNet model is best.

Diabetic Retinopathy – Retina images classification

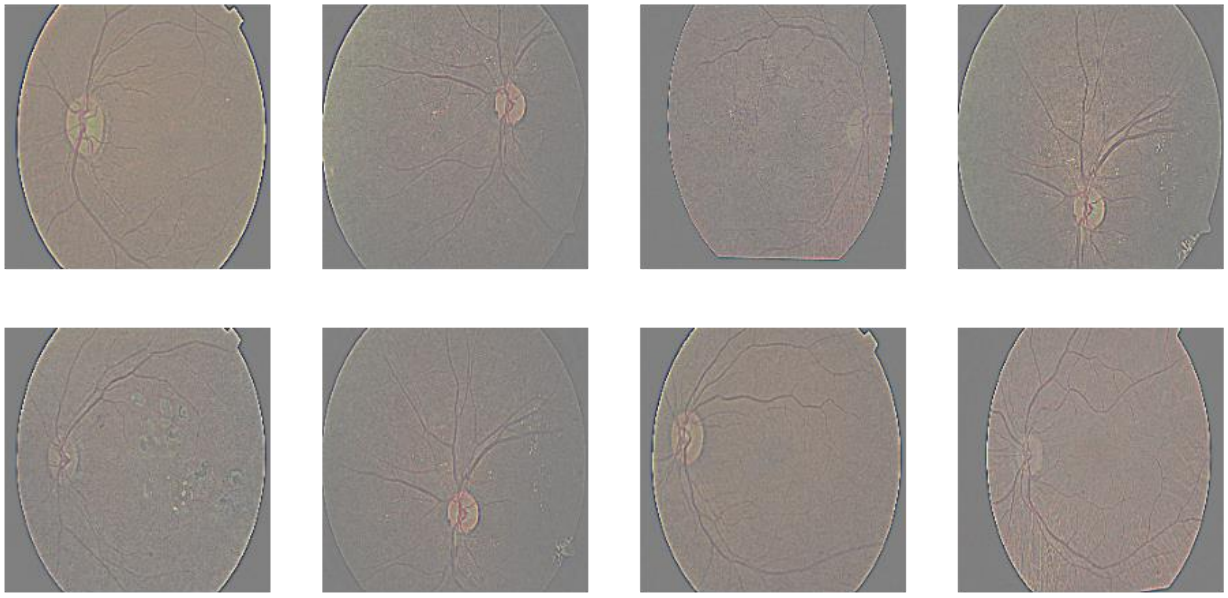
Problem statement:

Given with the retina images of patients, classify the patients based on the below 5 conditions:

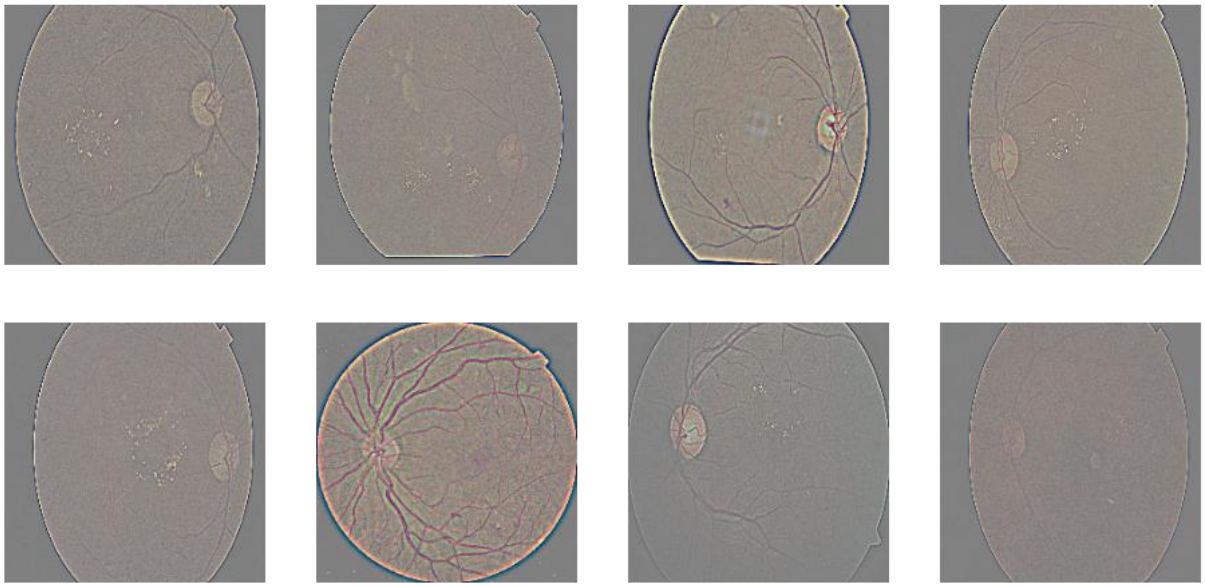
- **No DR** : normal person
- **Severe**: Having extreme diabetic retinopathy
- **Mild**: starting stage of DR
- **Moderate**: next stage of mild
- **Proliferate**: rapidly growing DR.

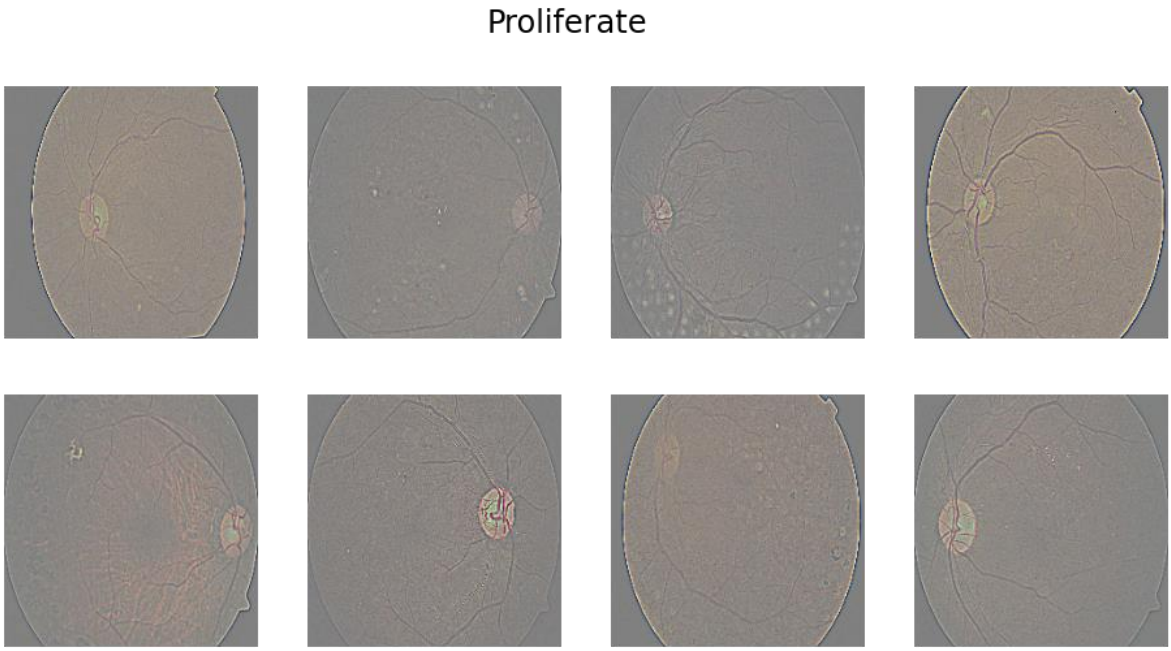


Mild



Moderate





Models:

We implemented the same figure based CNN and AlexNet model for this dataset.

However, the output layer neurons will be 5 because we have 5 class variables and activation function is Softmax.

Performance:

```
cnn_model.evaluate(test_dataset)
```

```
6/6 [==============================] - 2s 363ms/step - loss: 1.0436 - accuracy: 0.6222
[1.04362154006958, 0.6222222447395325]
```

```
alex_model.evaluate(test_dataset)
```

```
6/6 [==============================] - 6s 945ms/step - loss: 1.2136 - accuracy: 0.5000
[1.2135668992996216, 0.5]
```

CNN model:

Accuracy: 62%

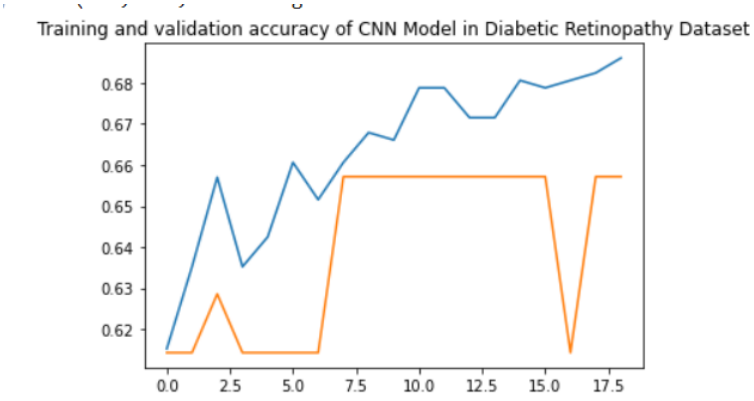
Loss: 1

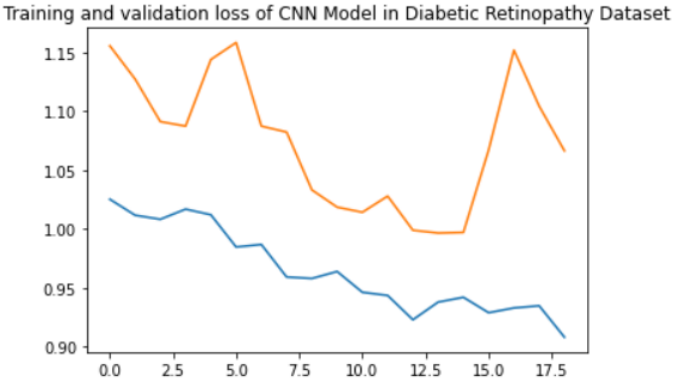
AlexNet model:

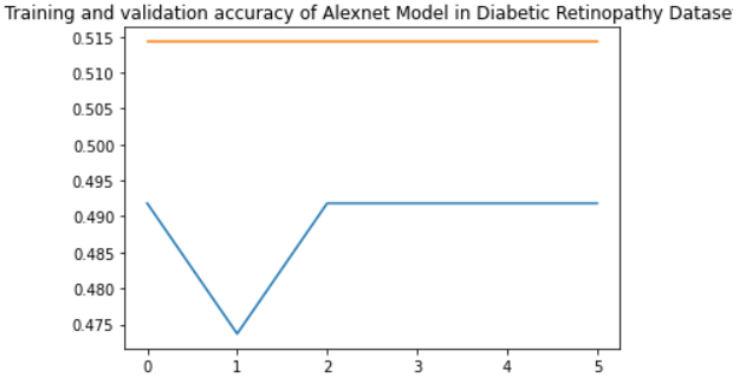
Accuracy: 50%

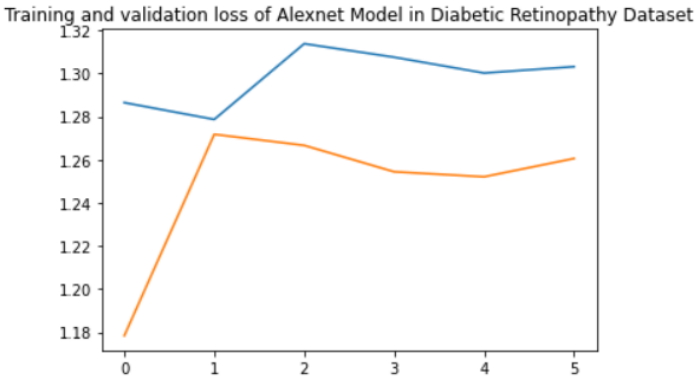
Loss: 1.21

Training and Validation – Accuracy and loss plot:









Conclusion:

- Training and test - Accuracy is increasing better in CNN model.
- Training and test – Loss are decreasing well in CNN model.
- Flat line in accuracy plot of AlexNet model shows that accuracy haven’t increase though the epochs are increase.
- This indicates the weakness of the model, the training itself insufficient.
- CNN model have higher accuracy and lower loss in comparison of AlexNet.
- Therefore, we can conclude that CNN model performed well in DR dataset.

Saving the weights of the best model

```
[ ]  model_final.save("/content/MyDrive/models/diabeticretino.h5")
     print("Saved model to disk")
     model_final.save_weights("/content/MyDrive/models/diabeticretino.h5")
     print("Saved weights to disk")
```