Predicting Brain Tumor
Using
MRI Scanned Image

HAP 880 Final Project Report
Professor: Dr. Janusz Wojtusiak

By,

Mayank Dubey
G01220775

Introduction:

"A brain tumor is a collection, or mass, of abnormal cells in your brain" [1]. The Brain is enclosed by the skull, if something unusual happens in this area it may cause problem. Brain tumors can be categorized into benign (non - cancerous) and malign (cancerous) [1]. When this tumor grows it causes pressure in the skull and may cause brain damage [1].

The most common way of detecting the brain tumor is Magnetic resonance imaging (MRI) [2]. MRI measures the tumor's size [2]. These MRI images are presented to the neurologist who decides the tumor is benign or malign. The aim of the project is to build a model on MRI scan images and given a new image which classifies whether the person has tumor or not.

Problem Statement:

1) Implementation of machine learning algorithm to detect brain tumor on MRI scanned images on the dataset taken from Kaggle.
2) It is estimated that 700,00 people are living with brain tumor, out of those 69.8% has benign brain tumor and 30.2% has malign [3]. According to google there are 200,000 cases of brain tumor every year which makes this disease common, not everyone has the luxury to go to doctor by building this model, detecting brain tumor will be less expensive.
3) Exploratory data analysis: Producing relevant statistics and visualization for the data.
4) Data preprocessing: Opencv is very rich library to work with image dataset, noise is added in the data by blurring the images, rotating the images, resizing the images, Flatting the images (Converting images in 1D from 2D) etc.

Machine Learning Algorithm:

In the dataset the labels are given, so the supervised algorithms like Logistic regression, Random Forest, Deep Neural Network and Convolutional models are implemented. Images are present in two folders yes and no, Yes means the associate label is 1 (tumor) and No means the associate label with the image is 0 (not a tumor).

Model evaluation:

It is a binary class classification problem, In the medical use case accuracy is wrong metric to evaluate the model as it is very misleading (It is important to detect everyone who has disease) so in this use case false negative should be 0 or Recall should be close to 1.

Other evaluation matrix are confusion matrix, precision, specificity, AUC and ROC curve.

- Confusion Matrix = $\begin{bmatrix} True\ Positive & False\ Negative \\ False\ Positive & True\ Negative \end{bmatrix}$
- Precision = $\frac{True\ Positive}{True\ Positive + False\ Positive}$

- Recall = $\frac{True\ Positive}{True\ Positive+False\ Negative}$
- Accuracy = $\frac{True\ Positive + TrueNegative}{True\ Positive + True\ Negative+False\ Positive+False\ Negative}$
- ROC curve is a plot between TPR and FPR.
- AUC is area under curve, Higher the AUC (close to 1) better the model.

EDA:

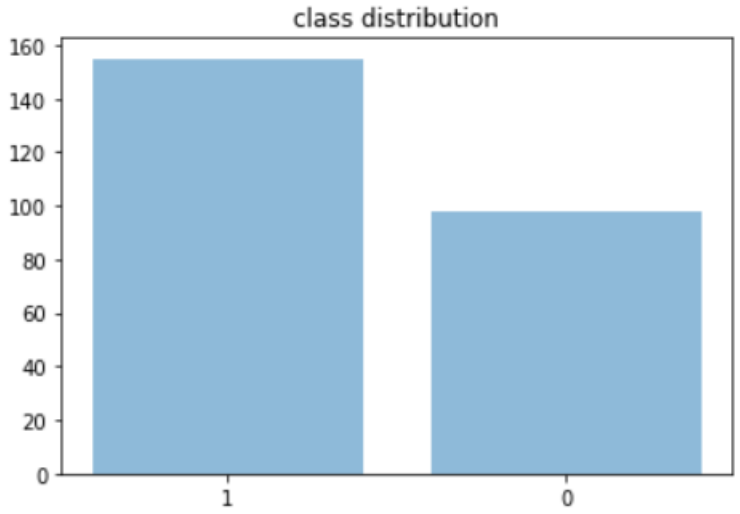


Figure – 1

The graph shows (fig. 1) the class distribution of the data. Total number of images we have is 253 in that 155 images we have has tumor and 98 images we have does not has tumor.

- **Brain Tumor:**

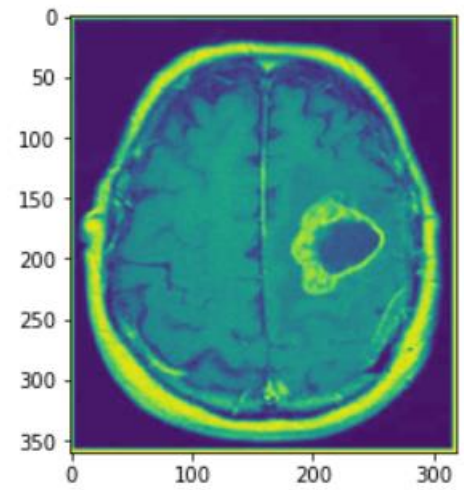


Figure – 2

```
array([[255, 255, 255, ..., 252, 247, 250],
       [255, 255, 255, ..., 249, 250, 248],
       [255, 253, 234, ..., 233, 252, 252],
       ...,
       [255, 253, 234, ..., 233, 254, 255],
       [255, 255, 255, ..., 249, 251, 251],
       [255, 255, 255, ..., 252, 249, 253]], dtype=uint8)
```

Figure – 3

Fig. 3 shows, the example of brain tumor and Fig. 4 is its mathematical representation (pixel value). The image has size 360 * 319.

- **Not Brain Tumor.**

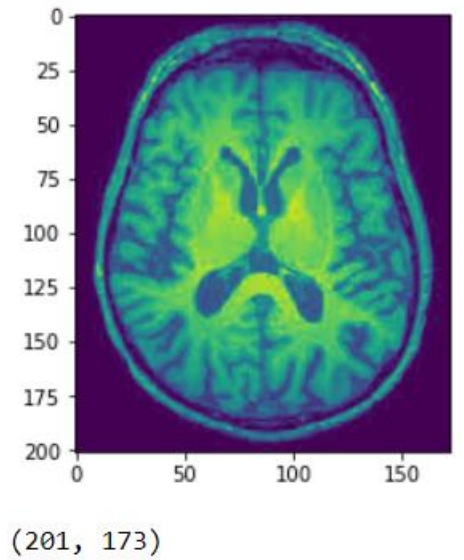


Figure – 4

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

Figure – 5

Fig. 4 shows, the example of not brain tumor and Fig. 5 is its mathematical representation (pixel value). The image has size 201 * 173.

Data Preprocessing:

As, Image are of different size we have to resize the image. By doing so we might loose some information or else we will be unable to train model. I converted image into 100 * 100 pixel.

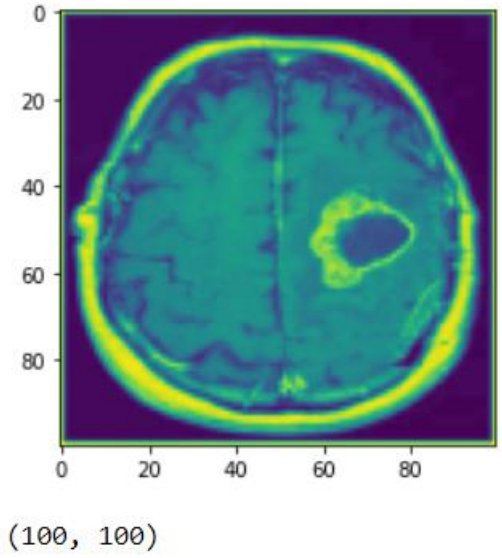


Figure – 6

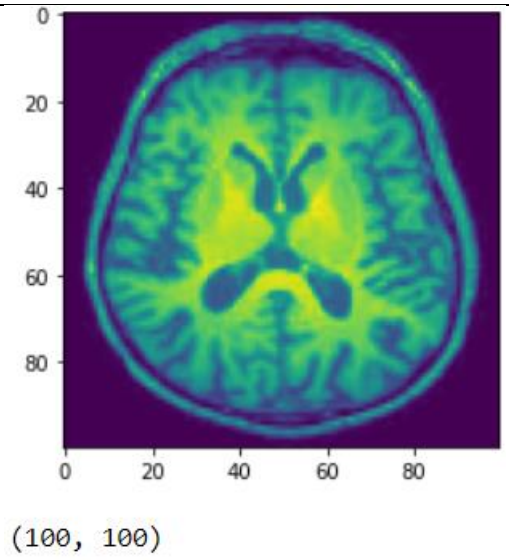


Figure – 7

Fig. 6 is resized version of Fig. 3, i.e. no tumor. Fig. 9 is resized version of Fig. 5

- **Train Test Spilt:**
  I divided data into train and test to see the model performance.
  The percent which I took is 80% for training and 20% for testing.
- **Flattening the image:**
  ML algorithms like 1-D data we have 2D data we have to fatten it, by flattening it loose original property, 100 * 100 image will become 1 D array of 10000.

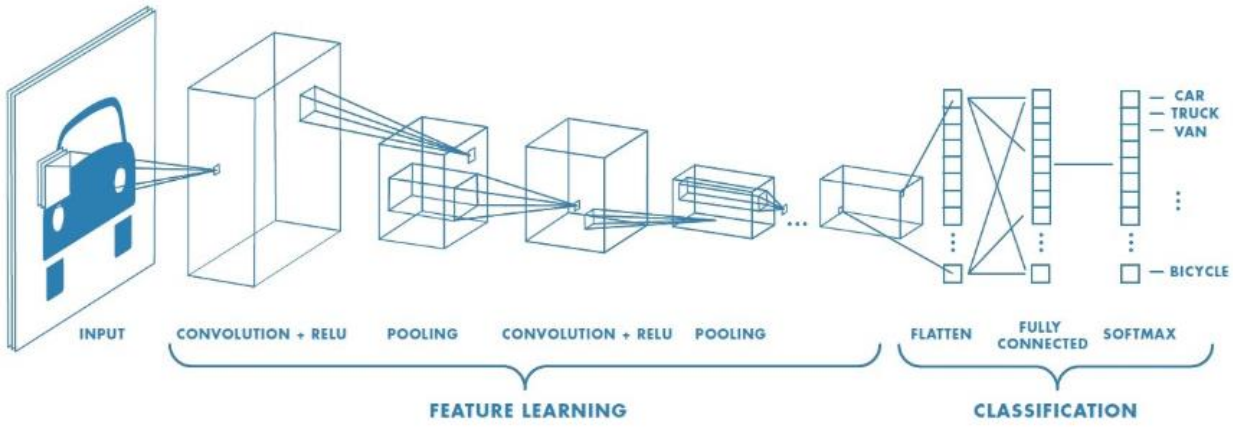
```
X_train = []
for img in X_train2d:
    X_train.append(img.flatten())
```

```
X_test = []
for img in X_test2d:
    X_test.append(img.flatten())
```

Figure – 8

Figure – 8 is the code snippet for image flattening which is required for Machine algorithms.

CNN:



CNN is just couple of layers added on the neural network. The problem with other machine learning algorithms and neural network is that they require 1 – D data which dissolves the relationship between pixels, however CNN can take 2 – D (Black & White) or 3 – D (RGB) data. The first layer in CNN is convolution and ReLu and second layer is Pooling layer.

Figure – 9

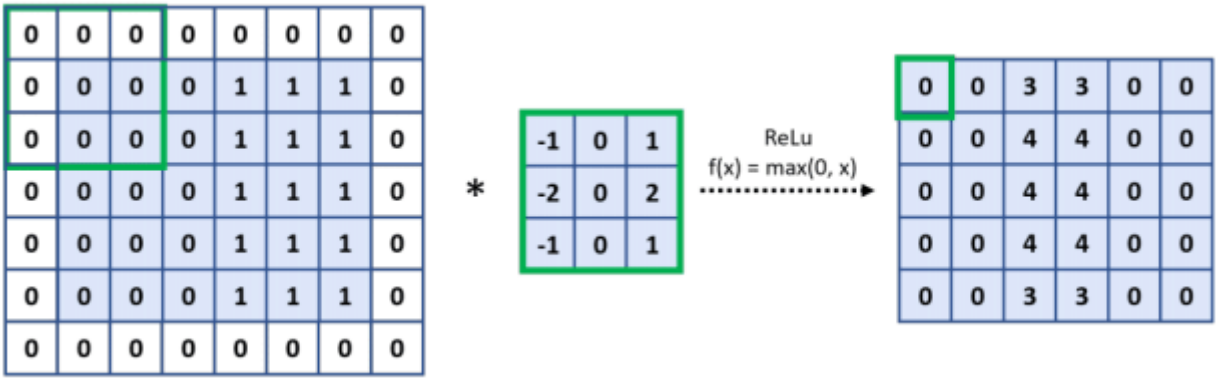


Figure – 10

The operation for first layer is explained in figure 10. On left most side that is image pixel which performs convolution operation with kernel (randomly generated 2 – D matrix), we have to give the kernel size (here 3*3). Convolution operation is nothing but dot product between image pixel and kernel. That output is passed through ReLu activation function which gives output as same value if it is non - negative. 0 if value is negative.

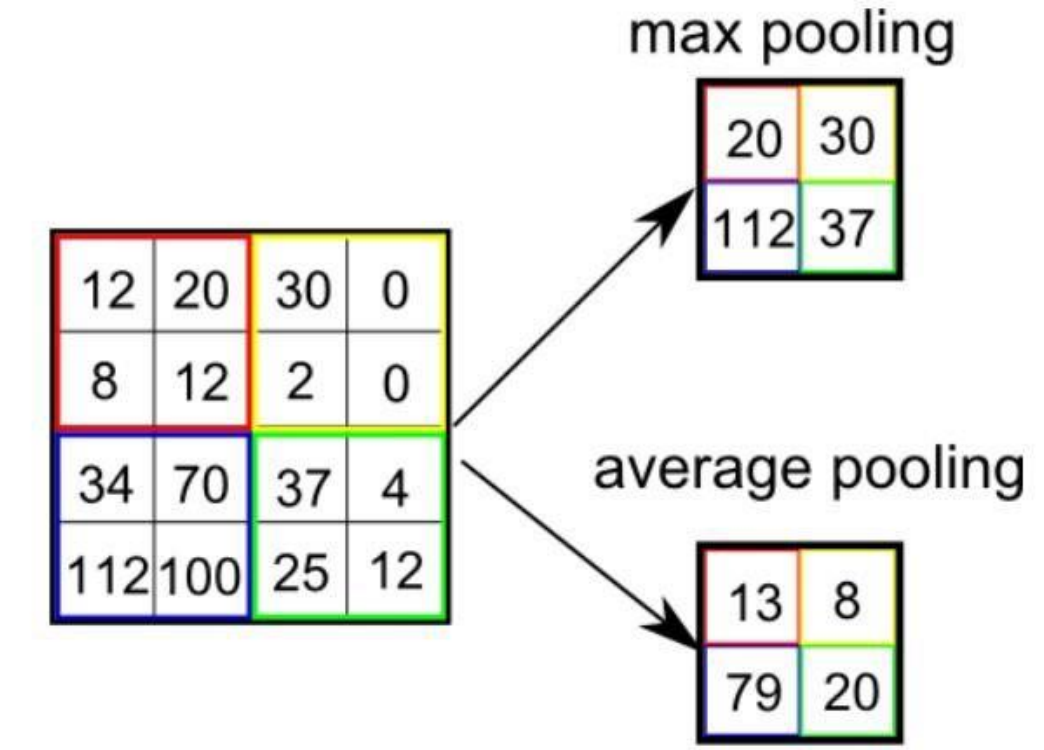


Figure – 11

The second operation is pooling operation. The output of ReLu is passed to the pooling layer. Basically, there are two type of polling operation. First is max pooling and second is average pooling. We have to provide the pooling size which is 2 – D matrix and type pf pooling operation. Here in figure 10 the pooling size is 2 * 2 and shown two types of pooling max and average pooling. If it's a max pooling it takes maximum value from 2*2 matrix or if it's a average pooling it takes average value from 2*2 matrix.

```
def getIndexPositions(listOfElements, element):
    ''' Returns the indexes of all occurrences of give element in
    the list- listOfElements '''
    indexPosList = []
    indexPos = 0
    while True:
        try:
            # Search for item in list from indexPos to the end of list
            indexPos = listOfElements.index(element, indexPos)
            # Add the index position in list
            indexPosList.append(indexPos)
            indexPos += 1
        except ValueError as e:
            break

    return indexPosList
```

Figure – 12

Figure – 12 is a code snippet to get indexes for some data, here I used it to get index False Negative images in X_test.

Modeling 1:

- For us False Negative should be less ideally 0 and recall should be more ideally 1.
- How to do that?
  Easiest way to make FN = 0 and recall = 1, to say everyone has brain. Although we have problem with that
  1) It will increase FP and TN will be 0
  2) Accuracy will be too bad.
  3) Model is not learning

The results of four algorithms are given as follows:

ALGORITHM	ACCURACY	AUC	PREC	RECALL	TP	TN	FP	FN
Base model	0.52	-	0.52	1.0	27	0	24	0
Logistic Regression	0.78	0.81	0.75	0.88	24	16	8	3
Random Forest	0.82	0.89	0.80	0.88	24	18	6	3
Neural Network	0.83	0.78	0.75	0.88	24	16	8	3
CNN	0.66	0.72	0.92	25	9	15	2	2

Every algorithm performs well then CNN for original data.

Indexes of the images in the X-test which gave false negative cases:

ALGORITHM	INDEX	INDEX	INDEX
LOGISTIC REGRESSION	8	18	36
RANDOM FOREST	8	18	46
NEURAL NETWORK	8	18	56
CNN	8	18	

8, 18 number of indexes in X_test gives False Negative result in every algorithm.

AUC:

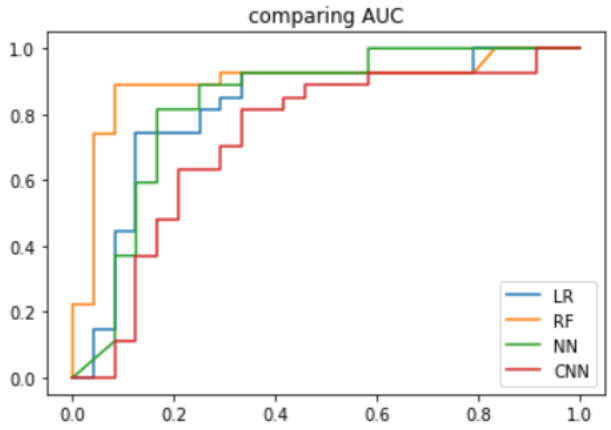


Figure – 13

AUC of random forest is highest.

Calibration curve and Probability distribution:

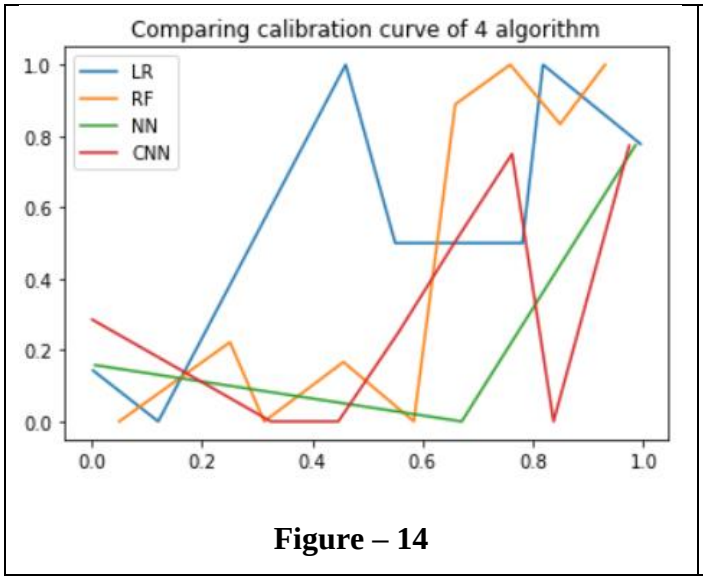


Figure – 14

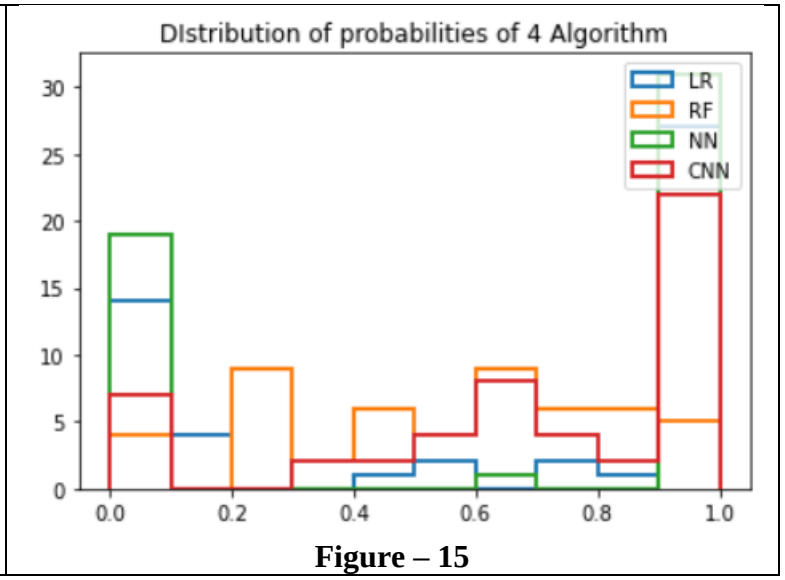


Figure – 15

Figure 14 and 15 shows the calibration curve and probability of prediction respectively. As we clearly see, no model is calibrated and no model is confident when predicting the class of an image.

Modeling 2:

Preprocessing:

1) Normalizing every image (Both train and test). Every value is divided by maximum value. Normalization is done to remove outliers
2) Every pixel value is divided by 255, In Black & White images maximum value for pixel is 255, where 255 represents black and 0 represents White.
3) By Normalizing images every pixel value is converted between 0 and 1.
4) Flattening the data for modeling.

```
X_train_norm2D = []
for i in range(0, len(X_train2d)):
    X_train_norm2D.append(X_train2d[i]/255)

X_test_norm2D = []
for i in range(0, len(X_test2d)):
    X_test_norm2D.append(X_test2d[i]/255)
```

Figure – 16

```
X_train = []
for img in X_train_norm2D:
    X_train.append(img.flatten())

X_test = []
for img in X_test_norm2D:
    X_test.append(img.flatten())
```

Figure – 17

Figure 16 and 17 is the code snippet for preprocessing which is required for modelling 2. Figure 16 is the normalization of images whereas figure 17 is flattening of images.

The results of four algorithms are given as follows:

ALGORITHM	ACCURACY	AUC	PREC	RECALL	TP	TN	FP	FN
Logistic Regression	0.80	0.83	0.77	0.88	24	17	7	3
Random Forest	0.78	0.88	0.75	0.88	24	16	8	3
Neural Network	0.78	0.86	0.76	0.85	23	17	7	4
CNN	0.82	0.82	0.82	0.85	23	19	5	4

From the above model performance we can clearly see that CNN is improved significantly and all algorithms are performing about same.

Indexes of the images in the X-test which gave false negative cases:

ALGORITHM	INDEX	INDEX	INDEX	INDEX
LOGISTIC REGRESSION	8	15	18	
RANDOM FOREST	8	18	46	
NEURAL NETWORK	8	15	18	36

8, 18 number of indexes in X_test gives False Negative result in every algorithm.

AUC:

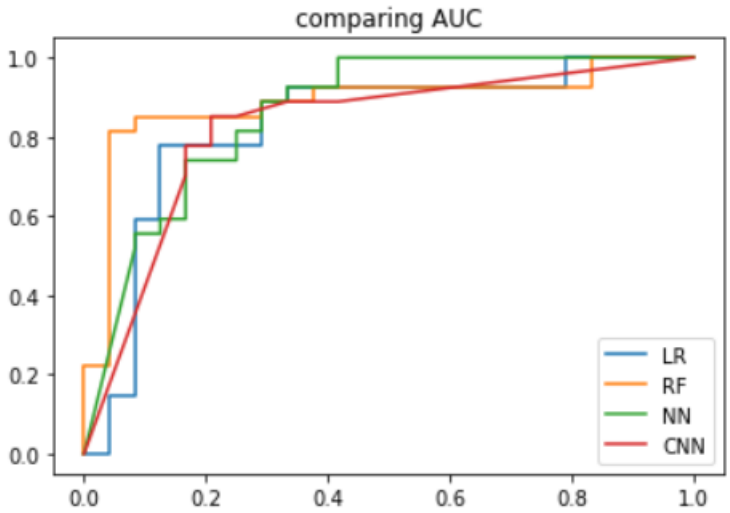


Figure – 18

AUC of random forest is highest.

Calibration curve and Probability distribution:

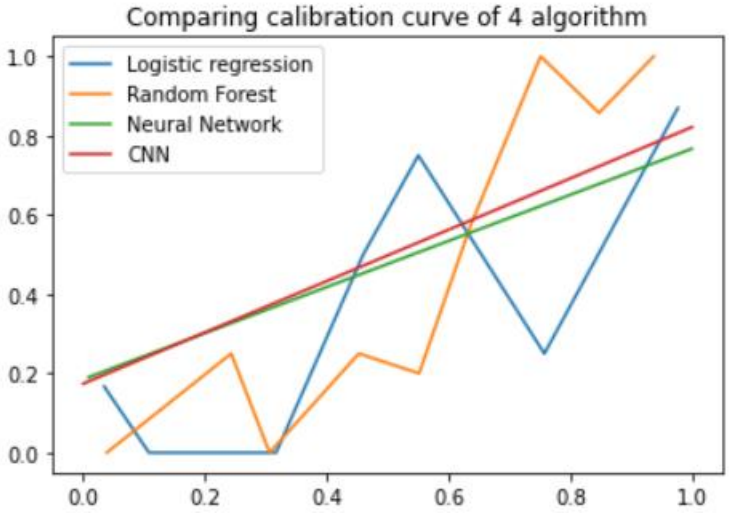


Figure – 19

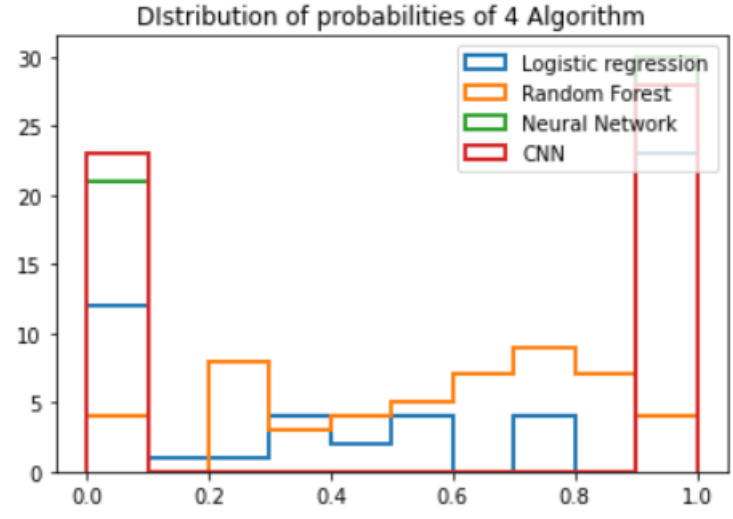


Figure – 20

Figure 19 and 20 shows the calibration curve and probability of prediction respectively. As we clearly see, CNN and Neural network is most calibrated and CNN model is confident when predicting the class of an image.

Modeling 3:

Preprocessing:

1) Blurring the images, only training images were blurred.
2) I used Gaussian blur, which does 100 % blurring.
3) Testing is done on unblur data.
4) Flattening the data for modeling.

```
X_train_blur2D = []
for i in range(0, len(X_train_norm2D)):
    X_train_blur2D.append(cv2.GaussianBlur(X_train_norm2D[i],(5,5),0))
```

Figure – 21

```
X_train_blur = []
for img in X_train_blur2D:
    X_train_blur.append(img.flatten())
```

Figure – 22

Figure – 21 and 22 is the preprocessing for modeling 3. Figure – 21 is the code snippet which does the gaussian blur which does the 100% blurring of the image. Figure – 22 is code snippet for image flattening.

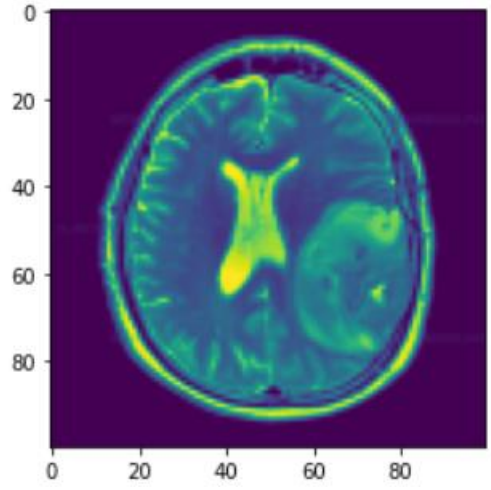


Figure – 23

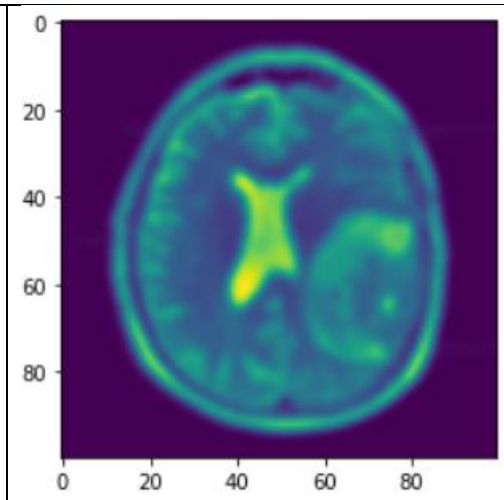


Figure – 24

Figure 23 shows the original image, whereas Figure 24 is blur image.

The results of four algorithms are given as follows:

ALGORITHM	ACCURACY	AUC	PREC	RECALL	TP	TN	FP	FN
Logistic Regression	0.72	0.79	0.69	0.85	23	14	10	4
Random Forest	0.80	0.88	0.79	0.85	23	18	6	4
Neural Network	0.74	0.81	0.71	0.85	23	15	9	4
CNN	0.82	0.80	0.80	0.88	24	18	6	3

After blurring the image, CNN and Random forest performs better than logistic regression and Neural network, which shows they were unable to capture noise.

Indexes of the images in the X-test which gave false negative cases:

ALGORITHM	INDEX	INDEX	INDEX	INDEX
LOGISTIC REGRESSION	1	8	18	46
RANDOM FOREST	8	15	18	46
NEURAL NETWORK	8	8	18	36
CNN	8	18	38	

8, 18 number of indexes in X_test gives False Negative result in every algorithm.

AUC:

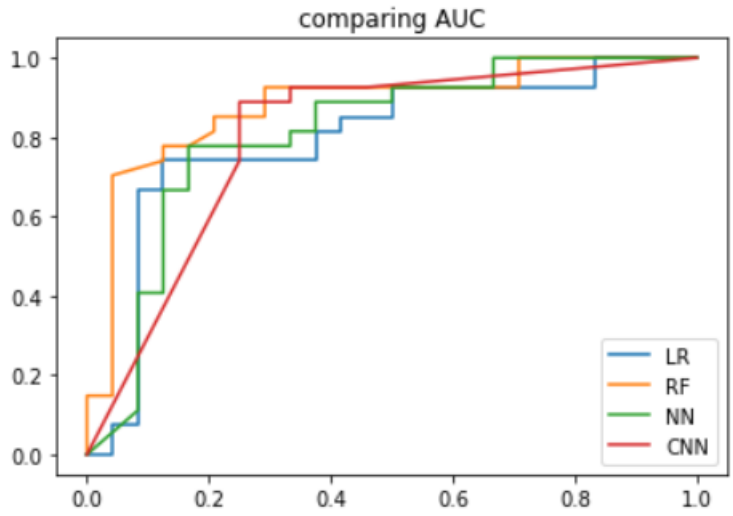


Figure – 25

AUC of random forest is highest.

Calibration curve and Probability distribution:

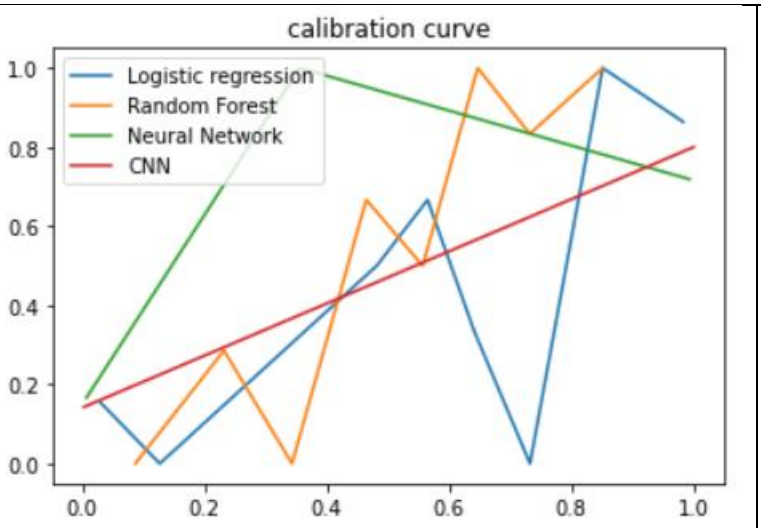


Figure – 26

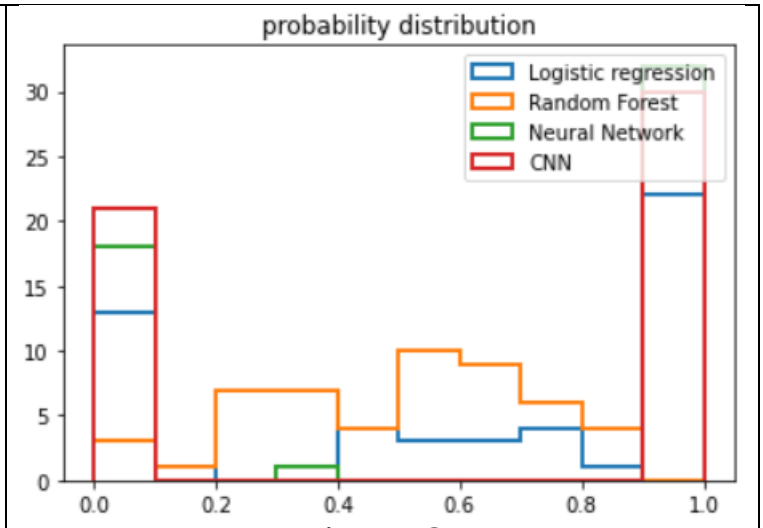


Figure – 27

Figure 26 and 27 shows the calibration curve and probability of prediction respectively. As we clearly see, CNN is most calibrated and is confident when predicting the class of an image.

Modeling 4:

1) Rotating the blur images, which creates more data.
2) Blur image is rotated 90, 180, 270 degrees.
3) Total training example is 808 images.
4) However, the model is trained on noise not on actual data.
5) Testing is done on the original images.

```
h =100
w =100
center = (h/2, w/2)

scale =1.0
```

```
rotate = []
rotate_label = []
for i in range(0, len(X_train_blur2D),1):
    M = cv2.getRotationMatrix2D(center, 0, scale)
    rotate.append(cv2.warpAffine(X_train_blur2D[i], M, (h, w)))
    rotate_label.append(y_train[i])
    M = cv2.getRotationMatrix2D(center, 90, scale)
    rotate.append(cv2.warpAffine(X_train_blur2D[i], M, (h, w)))
    rotate_label.append(y_train[i])

    M = cv2.getRotationMatrix2D(center, 180, scale)
    rotate.append(cv2.warpAffine(X_train_blur2D[i], M, (h, w)))
    rotate_label.append(y_train[i])

    M = cv2.getRotationMatrix2D(center, 270, scale)
    rotate.append(cv2.warpAffine(X_train_blur2D[i], M, (h, w)))
    rotate_label.append(y_train[i])
```

Figure – 28

Figure – 28 is code snippet for creating 90, 180, 270 degrees rotation.

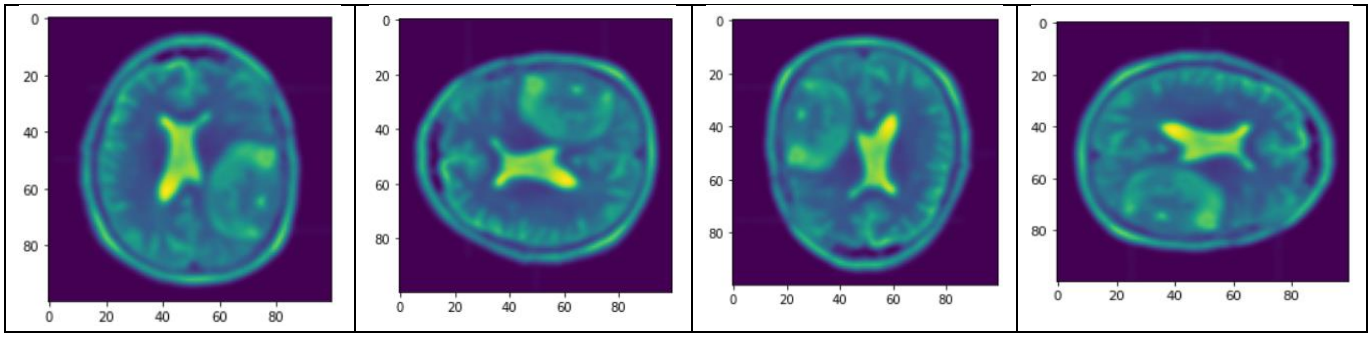


Figure – 29

Figure 29 shows, original blur, 90 degrees rotation, 180 degrees rotation, 270 degrees rotation respectively.

The results of four algorithms are given as follows:

ALGORITHM	ACCURACY	AUC	PREC	RECALL	TP	TN	FP	FN
Logistic Regression	0.62	0.71	0.62	0.74	20	12	12	7
Random Forest	0.76	0.87	0.75	0.81	22	17	7	5
Neural Network	0.74	0.78	0.70	0.88	24	14	10	3
CNN	0.80	0.82	0.77	0.88	24	17	7	3

After blurring the image, CNN model is the best performer, other algorithms such as Random forest, Logistic regression, Neural networks were unable to capture noise.

Indexes of the images in the X-test which gave false negative cases:

ALGORITHM	INDEX	INDEX	INDEX	INDEX	INDEX	INDEX	INDEX
LOGISTIC REGRESSION	8	17	18	22	23	36	46
RANDOM FOREST	8	15	18	38	46		
NEURAL NETWORK	8	18	36				
CNN	8	18	37				

8, 18 number of indexes in X_test gives False Negative result in every algorithm.

AUC:

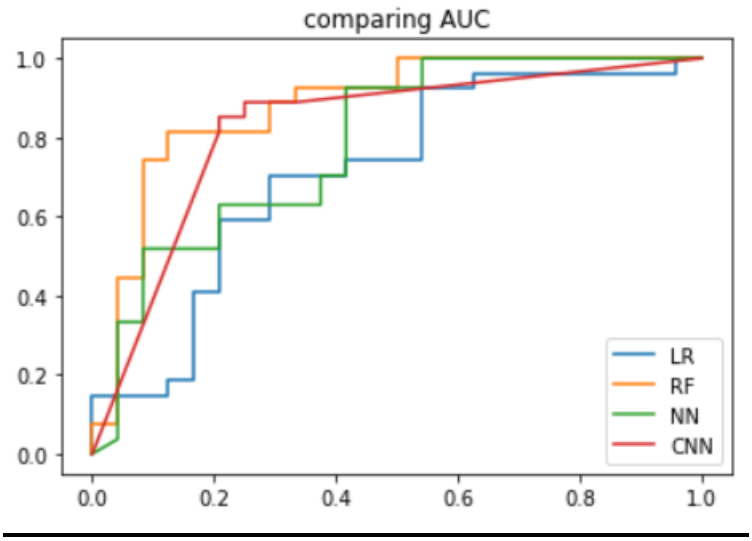


Figure – 30

AUC of random forest is highest.

Calibration curve and Probability distribution:

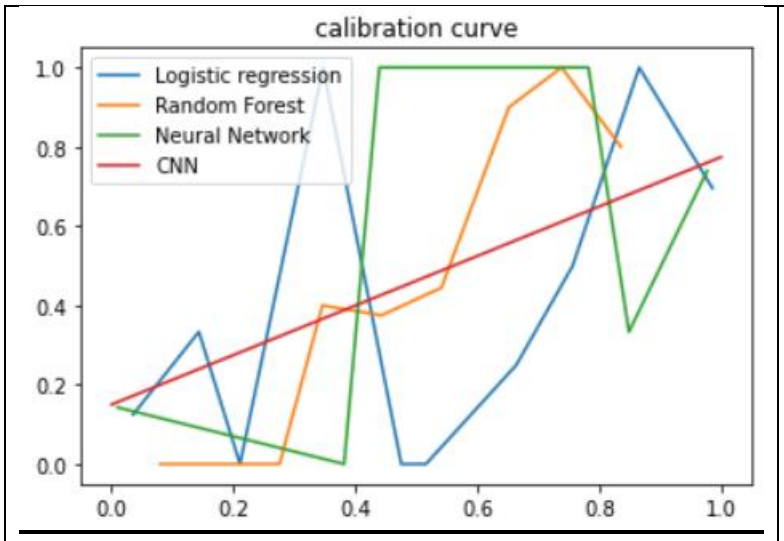


Figure – 31

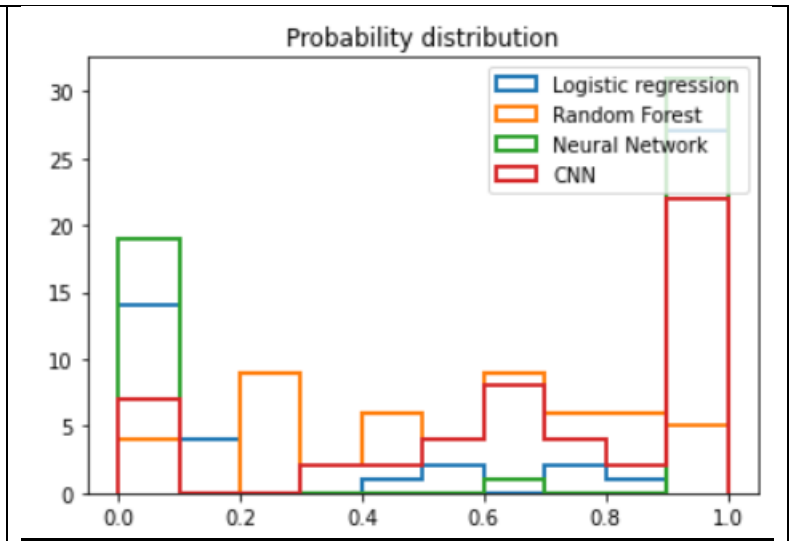


Figure – 32

Figure 31 and 32 shows the calibration curve and probability of prediction respectively. As we clearly see, CNN is most calibrated and is confident when predicting the class of an image, as CNN performs better than any other algorithm.

Modeling 5:

1) Rotating Original and Blur images, for more training data.
2) Rotating 90, 180, 270 degrees blur and original images.
3) Total training example for training is 1616 images.
4) Only 202 images are original images, rest are just noise in the data.
5) Testing is done on the original data.

```
h =100
w =100
center = (h/2, w/2)

scale =1.0
```

```
rotate_clear = []
rotate_label_clear = []
for i in range(0, len(X_train2d),1):

    M = cv2.getRotationMatrix2D(center, 0, scale)
    rotate_clear.append(cv2.warpAffine(X_train2d[i], M, (h, w)))
    rotate_label_clear.append(y_train[i])

    M = cv2.getRotationMatrix2D(center, 90, scale)
    rotate_clear.append(cv2.warpAffine(X_train2d[i], M, (h, w)))
    rotate_label_clear.append(y_train[i])

    M = cv2.getRotationMatrix2D(center, 180, scale)
    rotate_clear.append(cv2.warpAffine(X_train2d[i], M, (h, w)))
    rotate_label_clear.append(y_train[i])

    M = cv2.getRotationMatrix2D(center, 270, scale)
    rotate_clear.append(cv2.warpAffine(X_train2d[i], M, (h, w)))
    rotate_label_clear.append(y_train[i])
```

Figure – 33

Figure – 33 is code snippet for creating 90, 180, 270 degrees rotation.

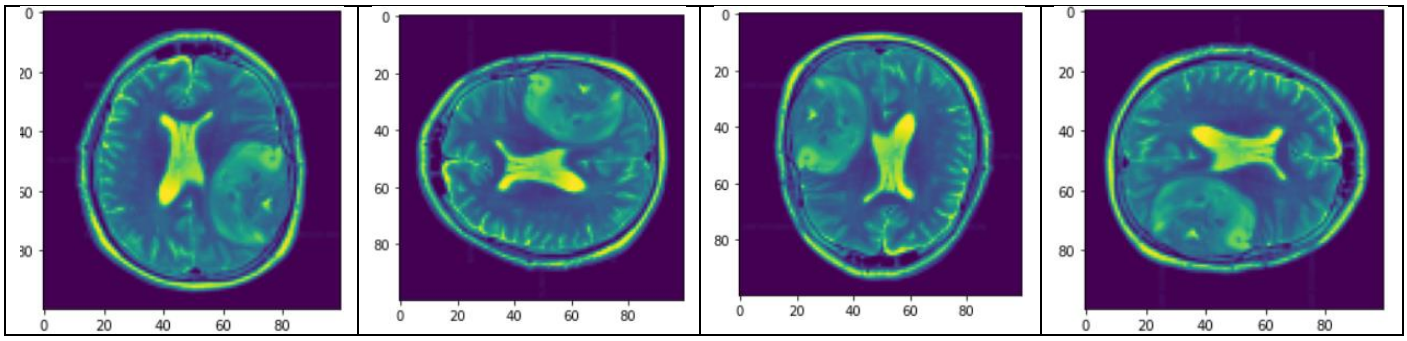


Figure – 34

Figure 34 shows, original image, 90 degrees rotation, 180 degrees rotation, 270 degrees rotation respectively.

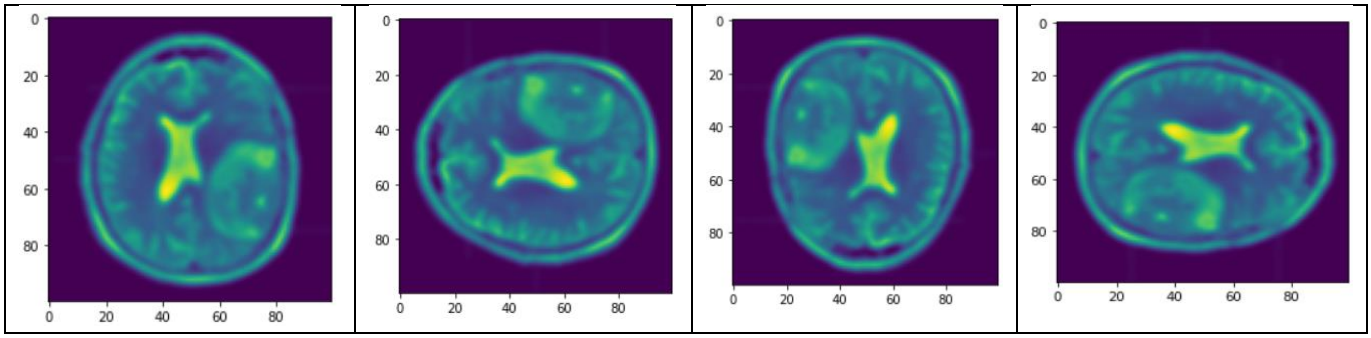


Figure – 35

Figure 35 shows, original blur image, 90 degrees rotation, 180 degrees rotation, 270 degrees rotation respectively.

The results of four algorithms are given as follows:

ALGORITHM	ACCURACY	AUC	PREC	RECALL	TP	TN	FP	FN
Logistic Regression	0.64	0.70	0.65	0.70	19	14	10	10
Random Forest	0.82	0.88	0.82	0.85	23	19	5	4
Neural Network	0.52	0.43	0.52	1.0	27	0	24	0
CNN	0.81	0.76	0.74	0.85	23	16	8	4

After rotating original images and blur images, CNN and Random forest performs better than logistic regression and Neural network. Neural network is predicting everyone has brain tumor which is equivalent to performance of base model which we discussed in modelling 1.

Indexes of the images in the X-test which gave false negative cases:

ALGORITHM	IDX	IDX	IDX	IDX	IDX	IDX	IDX	IDX
LOGISTIC REGRESSION	8	17	18	22	23	36	39	46
RANDOM FOREST	8	18	38	46				
NEURAL NETWORK	8	15	18	46	47			
CNN	8	18	21	37				

8, 18 number of indexes in X_test gives False Negative result in every algorithm.

AUC:

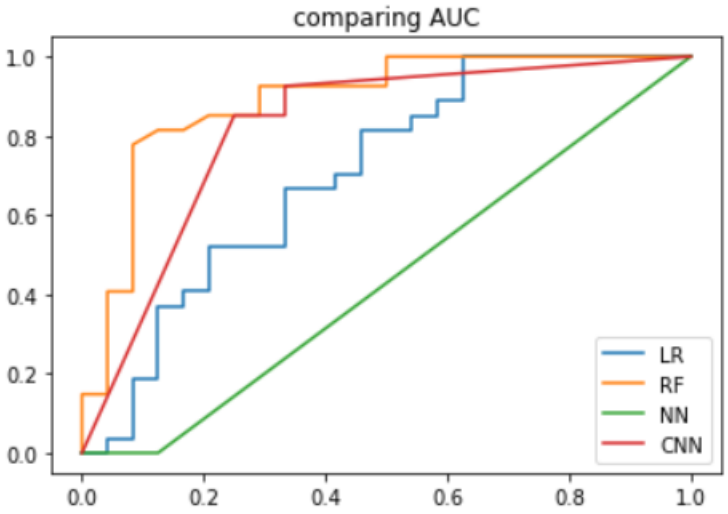


Figure – 36

AUC of random forest is highest.

Calibration curve and Probability distribution:

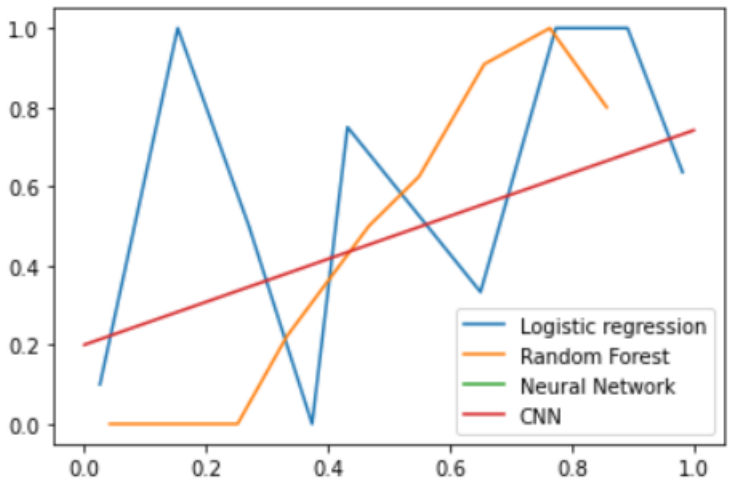


Figure – 37

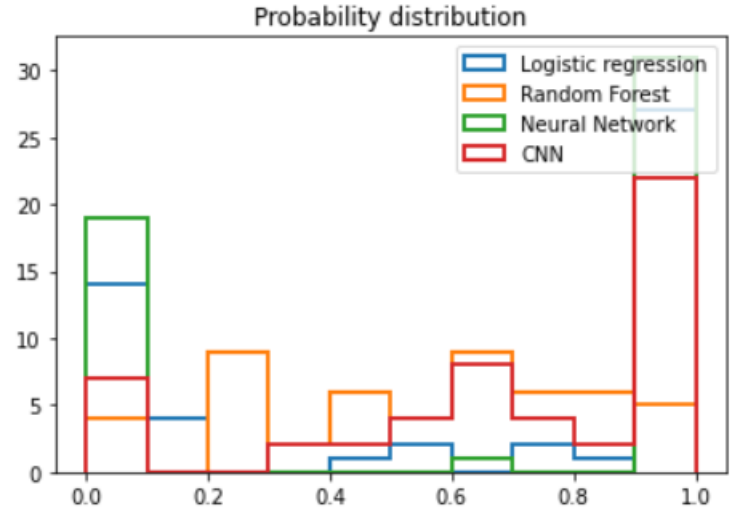


Figure – 38

Figure 37 and 38 shows the calibration curve and probability of prediction respectively. As we clearly see, CNN is most calibrated and is more confident then random forest when predicting the class of an image.

Why to add noise:

1) **To create more training example:**
   When model gets more data, it learns more and hence it reduces the chances of overfitting. Overfitting happens when model performs well in training data, but it is giving worst performance for testing data.

2) **To make model generalized:**
   In the real world, the model is not always getting nice and clear images so to make model more generalized and taking scenarios which can happen in real life like getting rotated images or blurred images we already provide that to model which makes it more robust and ready to use in real life.

Conclusion:

1) On the original training data every algorithm performs almost same.
2) As the noise is added CNN and Random Forest performs better.
3) CNN gives the most calibrated model.

References:

1) Understanding Brain Tumors. (n.d.). Retrieved from https://braintumor.org/brain-tumor-information/understanding-brain-tumors/
2) Brain tumor. (2019, April 27). Retrieved from https://www.mayoclinic.org/diseases-conditions/brain-tumor/symptoms-causes/syc-20350084
3) Quick Brain Tumor Facts. (n.d.). Retrieved from Data set https://braintumor.org/brain-tumor-information/brain-tumor-facts/

Data set:

Chakrabarty, N. (2019, April 14). Brain MRI Images for Brain Tumor Detection. Retrieved from http://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection

Libraries used

1) Numpy
2) Pandas
3) Matplotlib
4) Sklearn
5) Tesorflow & Kears
6) OpenCV2