CEREBRAL STROKE CLASSIFICATION USING SUPERVISED LEARNING APPROACHES

Team Members: (Batch - 13)

A.VARSHASHREE(211420104296)
P.KOMEL(211420104137)

Guided By :

Dr.K.Valarmathi

ABSTRACT

- Cerebral stroke is a life-threatening medical condition that requires immediate diagnosis and intervention to minimize its devastating effects.
- Timely and accurate classification of stroke subtypes is crucial for determining appropriate treatment strategies and improving patient outcomes.
- In this research, we explore the application of supervised learning approaches to classify cerebral stroke subtypes based on relevant medical data.
- The study involves the collection of a diverse dataset comprising anonymized patient records, including clinical indicators, medical history, risk factors, and diagnostic imaging results.
- The experimental results demonstrate the effectiveness of supervised learning approaches in accurately classifying cerebral stroke subtypes.

EXISTING SYSTEM

- **Neural-like P systems are membrane computing models** inspired by natural computing and are viewed as third-generation neural network models.
- Although real neurons have complex structures, In the biological context, neurons have intricate morphologies and a wide array of biochemical processes.
- However, in the context of Neural-like P systems, **the structures and mechanisms are simplified** for computational efficiency and ease of modeling.
- The simplification mentioned above is achieved by representing the neural like P systems using **two-dimensional graphs or tree structures**.
- The simplification of structures and communication mechanisms in classical **neural-like P systems might limit their applicability in real-world scenarios.** This limitation could arise because the simplified models may not capture all the nuances of real neural processing.

DEMERITS

- They did not implement the deployment process.
- Accuracy & performance was low.
- They only disease segmentation.

PROPOSED SYSTEM

- The study involves the collection of a diverse dataset comprising anonymized patient records, including clinical indicators, medical history, risk factors, and diagnostic imaging results.
- The data is carefully pre-processed to handle missing values, normalize numerical features, and address any potential biases.
- Next, a set of relevant features is extracted from the pre-processed data to represent the characteristics of network traffic.
- Feature selection techniques, such as **Recursive Feature Elimination (RFE)** or **Principal Component Analysis(PCA)**, are applied to reduce dimensionality and select the most informative features, thereby improving the efficiency and effectiveness of the predictive model.
- Several supervised machine learning algorithms are considered for the predictive model. Popular algorithms like Decision Trees**, Random Forests, Support Vector Machines (SVM),** Neural Networks**, and K-Nearest Neighbours (KNN)** are evaluated and compared based on their performance metrics using **cross-validation techniques.**

MERITS

- We classify the stroke.
- Accuracy & performance level improved.
- We build framework based application for deployment purposes.
- We compared more than two algorithms to get a better accuracy level.

ENVIRONMENTAL REQUIREMENTS

1. Software Requirements:

Operating System : Windows

Tools : Anaconda with jupyter Notebook

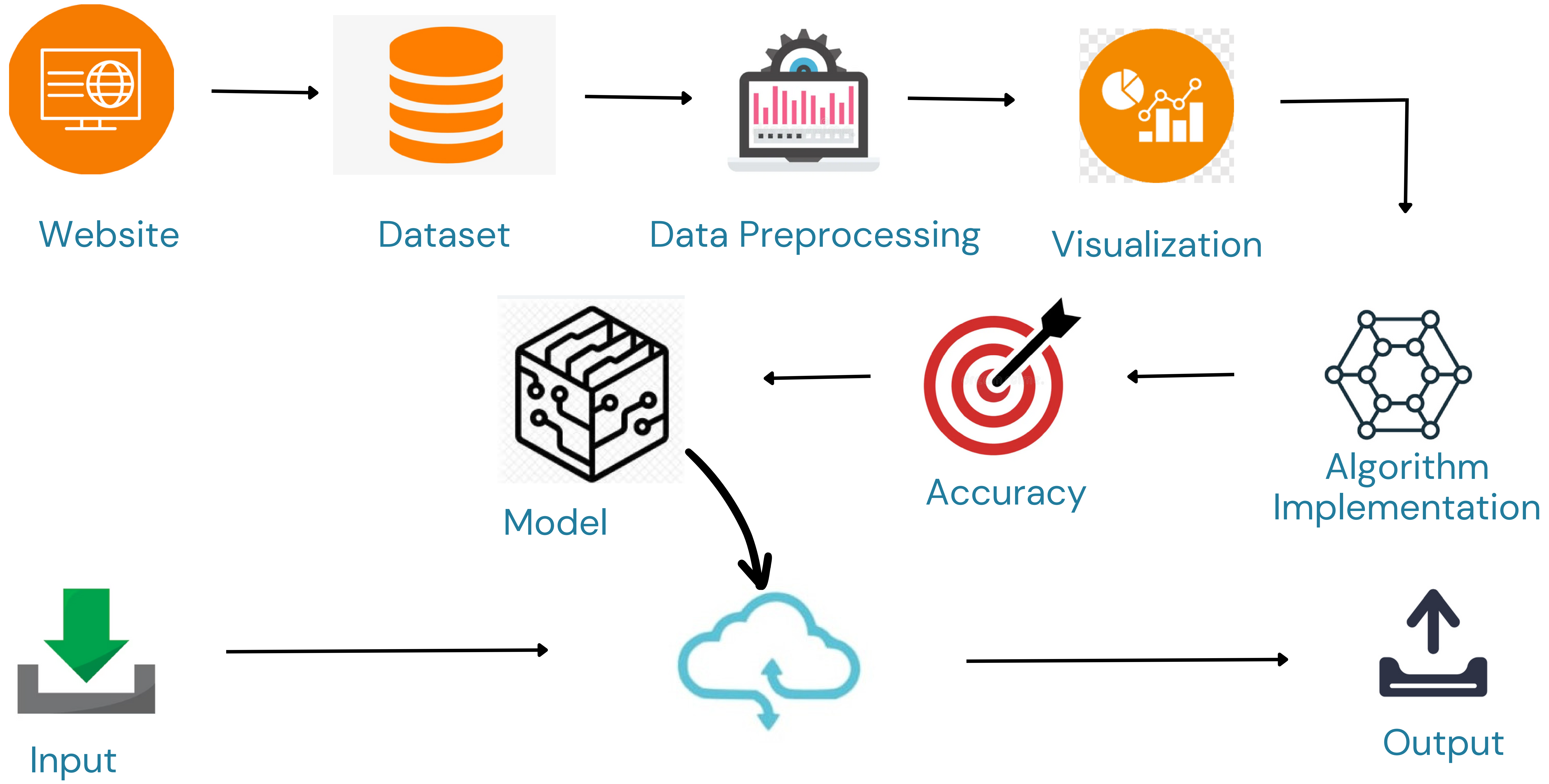
2. Hardware Requirements:

Processor : Pentium IV/III

Hard disk : minimum 100 GB

RAM : minimum 4 GB

ARCHITECTURE



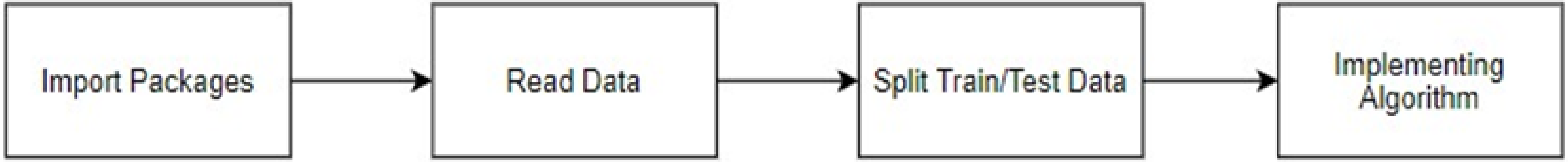
MODULES

- Module 1 : Randomforest Algorithm
- Module 2 : Adaboost classifier
- Module 3 : KNN Classifier

Module 1 : Randomforest Algorithm

- Random Forest is a popular machine learning algorithm an be used for both Classification and Regression problems.It is based on the concept of ensemble learning, which is a process of **combining multiple classifiers to solve a complex problem and to improve the performance of the model**.

- 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. **ie.** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

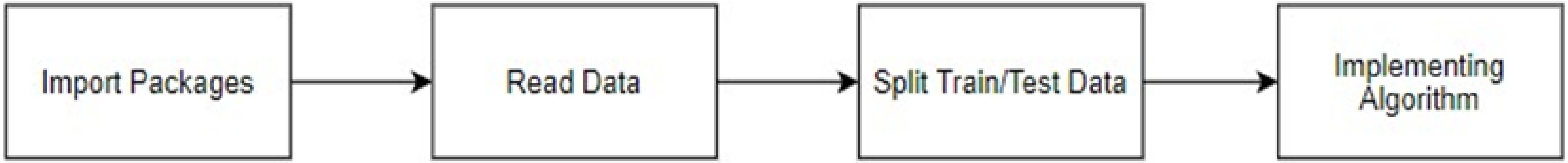


Algorithm

```
function RandomForest(S , F )
H ← ∅
for i ∈ 1, . . . , B do
S(i) ← A bootstrap sample from S
hi ← RandomizedTreeLearn(S(i), F)
H ← H ∪ {hi}
end for
return H
end function
function RandomizedTreeLearn(S , F)
At each node:
f ← very small subset of F
Split on best feature in f
return The learned tree
end function
```

Module 2 : Adaboost classifier

- This method operates iteratively, identifying misclassified data points and adjusting their weights to minimize the training error. The model continues to optimize sequentially until it yields the strongest predictor.
- AdaBoost is implemented by combining several weak learners into a single strong learner. The weak learners in AdaBoost take into account a single input feature and draw out a single split decision tree called the decision stump. Each observation is weighted equality while drawing out the first decision stump.
- The first decision stump's results are checked. if any observations are misclassified, they're given higher importance by increasing their weights. Then, another decision stump is drawn, focusing more on the previously misclassified observations. This process repeats until all observations are correctly classified.



Algorithm

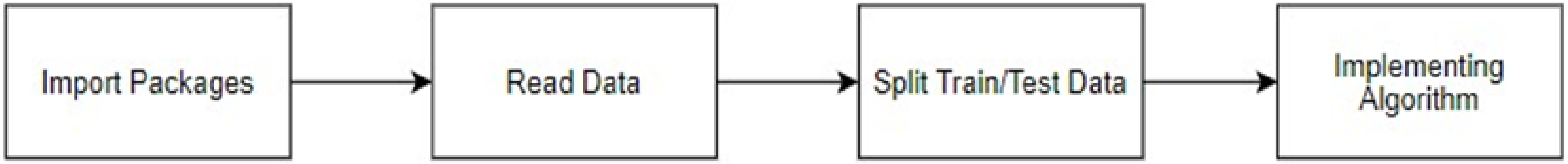
Set w[i] = 1/N, where N is the number of samples
For t = 1 to T:
a. Train a weak learner (decision stump):
b. Calculate error:
error = sum(w[i]) for misclassified samples
c. Compute learner weight:
Compute alpha[t] = 0.5 * ln((1 - error) / error)
d. Update sample weights:
For each sample i:
If the weak learner correctly classifies sample i:
w[i] *= exp(-alpha[t])
Else:
w[i] *= exp(alpha[t])
e. Normalize the weights:
Normalize w so that it sums up to 1
f(X) = sum(alpha[t] * h_t(X)) for t = 1 to T, where h_t(X) is the prediction of the t-th weak learner
Apply a threshold to the combined weak learner predictions to obtain the final classification

Module 3 : KNN classifier

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using k-NN algorithm.

- It is called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.



Algorithm

```
Function KNN_Classifier(training_data, test_instance, k):
distances = []
# Calculate distances between test_instance and each training_data point
For each training_point in training_data:
distance = Calculate_Distance(training_point, test_instance)
distances.append((training_point, distance))
# Sort distances in ascending order
distances.sort(key=lambda x: x[1])
# Get the first k neighbors
neighbors = distances[:k]
# Count the occurrences of each class among the k neighbors
class_votes = {}
For neighbor in neighbors:
class_label = neighbor[0].class_label # Assuming each training_data
point has a class label
If class_label in class_votes:
class_votes[class_label] += 1
Else:
class_votes[class_label] = 1
# Find the class label with the maximum votes
predicted_class = Max(class_votes, key=class_votes.get)
Return predicted_class
```

COMPARISON

EXISTING SYSTEM	PROPOSED SYSTEM
It focuses on capturing high-order correlations among neuron structures.	The proposed system is aimed at cerebral stroke classification using supervised learning approaches
Targeted towards medical image segmentation, specifically for tumor/organ segmentation.	Targeted towards stroke classification, aiming to enhance the accuracy and efficiency of stroke diagnosis.
Outperforms state-of-the-art methods in tumor/organ segmentation based on experimental results.	Seeks to enhance the precision and speed of stroke diagnosis, potentially assisting healthcare professionals in expedited and well-informed decision-making processes.

LITERATURE SURVEY

SI.NO	DATE	AUTHORS	TITLE	METHODOLOGY	ISSUES
1.	2023	Mr.M.THIRUNAVUKKARASU	Predicting Brain Stroke Using Supervised machine learning	Cat boost approach	With large datasets Training time may increase significantly.
2.	2023	Hong-Yu Zhou	A Unified Visual Information Preservation Framework for Self-supervised Pre-Training in Medical Image Analysis	PCRLv1	Accuracy in segmanting the tumor region is poor.
3.	2023	Valeria Mariano	Brain Stroke Classification via Machine Learning Algorithms Trained with a Linearized Scattering Operator	Distorted Born approximation	Accuracy in the output is poor.

CONCLUSION

After the literature survey, we came to know various pros and cons of different research papers and thus, proposed a system that helps to predict brain strokes in a cost effective and efficient way by taking few inputs from the user side and predicting accurate results with the help of trained Machine Learning algorithms. Thus, the Brain Stroke Prediction system has been implemented using the given Machine Learning algorithm given a Best accuracy.The system is therefore designed providing simple yet efficient User Interface design with an empathetic approach towards their users and patients.

REFERENCES

[1] Concept of stroke by healthline .[Online] .Available: https://www.cdc.gov/stroke/index.htm.

[2] Statistics of stroke by Centers for disease control and prevention .[Online] Available: https://www.cdc.gov/stroke/facts.htm.

[3] Manikandan R. Classification of stroke disease using machine learning algorithm. Neural Computing & Applications.2020;32(3):817-828.doi:10.1007/s00521-019-04041-y.

[4] 2013;Farzadfar M>T>.et al. Prediction and control of stroke by data mining. International Journal of Preventive Medicine. 2013;4(2):S245-S249.

[5] T.Pan et al.,"Cell-like spiking neural P systems with evolution rules," Soft Comput., vol,223,no.14,pp.5401-5409.

THANK YOU