International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 01 | January - 2023 Impact Factor: 7.185 ISSN: 2582-3930

# PREDICTION OF NEUROLOGICAL DISEASE USING MACHINE LEARNING

E. Durga Prasad[1], U. Harsha VArdhan[2], M. Dhiraj Yadav[3], Dr. Y. Srinivasulu[4]

[1,2,3]Student, [4]Professor

[1,2,3,4]Department of Electronics and Communication Engineering

[S1,2,3,4]Sreenidhi Institute of Science and Technology, Ghatkesar, Hyderabad, India

**ABSTRACT** - In the modern era, neurological diseases are a growing concern. Finding neurological disease in its early stages is a challenging task for a doctor. However, accurate disease diagnosis at an early stage is crucial because treatments are more likely to enhance patients' quality of life at that point. We suggest a statistical approach using the most prevalent symptoms of neurological diseases, which are gait, tremors, and micrographic, since there is no established test to identify the symptoms. In order to find the classification algorithm that provides the highest accuracy in diagnosing the patients, this involves analyzing the correlation between both the symptoms and classifying the obtained data using various classification algorithms. Greater medical care and higher standards of care may result from earlier detection**.**

***Keywords :*** Neurological Disease, micrographic, diagnosis.

## INTRODUCTION

A group of diseases affecting the brain and spinal cord known as neurological disorders are characterized by a progressive decline in the structure and/or function of neuronal cells. In addition to having a variety of symptoms, neurological disorders can also be brought on by a wide range of unidentified causes and factors. Due to their variable symptoms and course of progression, many neurological diseases go undiscovered for a long time. More sensitive diagnostic methods are required for the diagnosis because the symptoms worsen as the disease progresses. In general, this project's primary goal is to identify a disease as soon as possible. Early disease detection increases the likelihood that treatments will enhance the patients' and their families' quality of life

## 2. LITERATURE VIEW

### 2.1 PROBLEM STATEMENT

The current system requires the patient to follow several tests like PET scan, CT scan and other scans. But these tests tend to take up a lot of time to gather results and a lot of waiting has to be done, the patient would need to wait for the results and their condition might worsen. Thus, there is a need for a rapid check so that diagnosis can be done earlier which would provide a much-needed quality of life improvement for the diseased patient**.** Our project aims at detecting at neurological disease as early as possible.

### 2.2 EXISITING SYSTEM

The existing system of detecting the neurological diseases are dependent on a lot of Scanning which would take up a lot of time by which the condition of the patient of might worsen.

Some existing systems are :

- MRI/CT Scan
- PET Scan
- SPECT Scan

These tests are very costly to be carried out and have an error rate of 25% and the results might take some time to arrive which would deteriorate the patient’s life.

### 2.3 PROPOSED SYSTEM :

The issues can be resolved with a low error rate by utilizing machine learning techniques. In order to increase accuracy by determining the correlation between these symptoms, the majority of neurological diseases use gait, tremors, and handwriting samples as their dataset. The goal of our system is to develop a model that can accurately predict whether a patient has a disease or not by examining a variety of symptoms. Since every symptom analysis individually has some drawbacks, using breath samples has been shown to fall short of producing clinically useful results. For instance, handwriting is a complex activity in which other factors can influence motor movement. In speech recognition, additional steps like noise removal and speech segmentation are also necessary. Consequently, to prevent the aforementioned issues

## 3. COMPONENTS

### 3.1 SOFTWARE REQUIREMENTS

- Any Operating System which supports Python
- Python libraries: Numpy, Pandas.

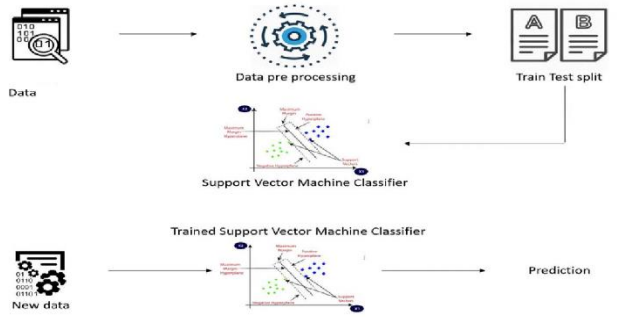
### 3.2 HARDWARE COMPONENTS

- Processor – intel i5 8th gen
- Memory – 8GB RAM
- Hardisk – 10GB

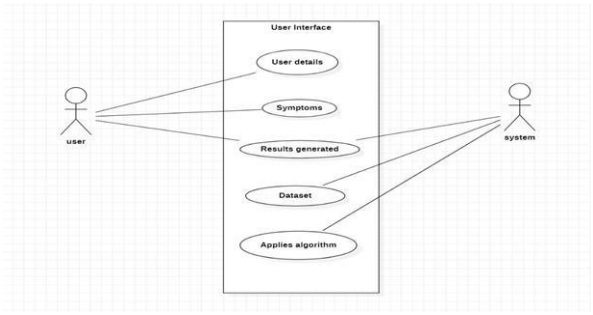
## 4. SYSTEM DESIGN

The process of defining a system's architecture, parts, modules, interfaces, and data in order to meet predetermined requirements is known as systems design. It may be considered the application of systems theory to the process of product development. The most popular techniques for designing computer systems are increasingly those that focus on object-oriented analysis and design.

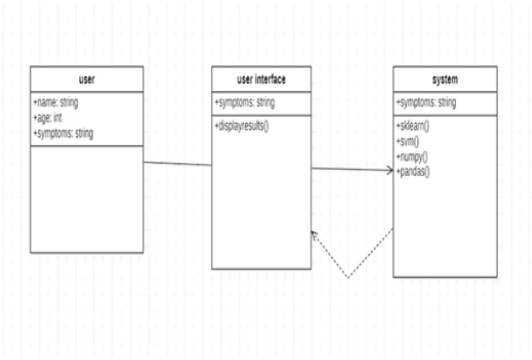
### 4.1 SYSTEM ARCHITECTURE



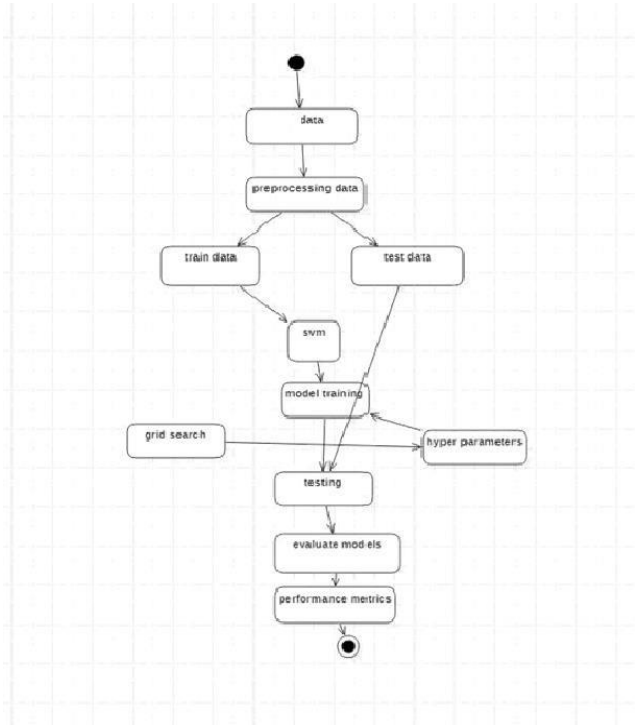

### 4.2 ULM DIAGRAMS



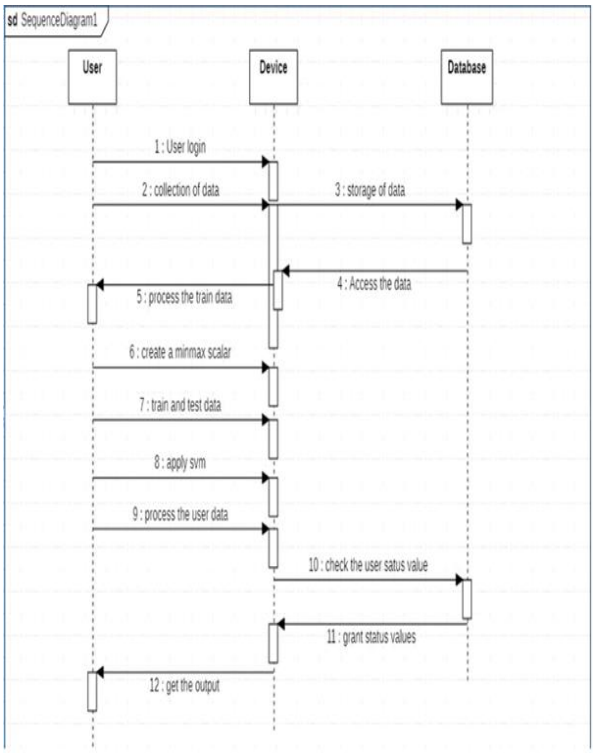

**Use-Case Diagram**




**Class Diagram**




**Activity Diagram**




**Sequence Diagram**

## 5. IMPLEMENTATION

### 5.1 LANGUAGE IR YECHNOLOGY USED

Python : Python is the language used for developing the prediction system. Python supports many libraries which have been used for this ML project. The libraries used for this project are: z NUMPY
PANDAS SCIKIT LAERN

### 5.2 ALGORITHM

The main algorithm used for this project is SVM.

Support Vector Machine (SVM):
A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text. Compared to newer algorithms like neural networks, they have two main advantages: higher speed and better performance with a limited numberof samples (in the thousands). This makes the algorithm very suitable for text classification problems, where it's common to have access to a dataset of at most a couple of thousands of tagged samples.

### 5.3CODES

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
```

**Libraries**

```
# loading the data from csv file to a Pandas DataFrame
data = pd.read_csv('data.csv')

# printing the first 5 rows of the dataframe
data.head()
```

| | name | MDVP:Fo(Hz) | MDVP:Fhi(Hz) | MDVP:Flo(Hz) | MDVP:Jitter(%) | MDVP:Jitter(Abs) | MDVP:RAP | MDVP:PPQ | Jitter:D |
|---|---|---|---|---|---|---|---|---|---|
| 0 | phon_R01_S01_1 | 119.992 | 157.302 | 74.997 | 0.00784 | 0.00007 | 0.00370 | 0.00554 | 0.01 |
| 1 | phon_R01_S01_2 | 122.400 | 148.650 | 113.819 | 0.00968 | 0.00008 | 0.00465 | 0.00696 | 0.01 |
| 2 | phon_R01_S01_3 | 116.682 | 131.111 | 111.555 | 0.01050 | 0.00009 | 0.00544 | 0.00781 | 0.01 |
| 3 | phon_R01_S01_4 | 116.676 | 137.871 | 111.366 | 0.00997 | 0.00009 | 0.00502 | 0.00698 | 0.01 |
| 4 | phon_R01_S01_5 | 116.014 | 141.781 | 110.655 | 0.01284 | 0.00011 | 0.00655 | 0.00908 | 0.01 |

5 rows x 24 columns

**Loading Dataset**

```
X = data.drop(columns=['name','status'], axis=1)
Y = data['status']

print(X)

     MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0        119.992       157.302        74.997         0.00784
1        122.400       148.650       113.819         0.00968
2        116.682       131.111       111.555         0.01050
3        116.676       137.871       111.366         0.00997
4        116.014       141.781       110.655         0.01284
..           ...           ...           ...             ...
190      174.188       230.978        94.261         0.00459
191      209.516       253.017        89.488         0.00564
192      174.688       240.005        74.287         0.01360
193      198.764       396.961        74.904         0.00740
194      214.289       260.277        77.973         0.00567

     MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  \
0             0.00007   0.00370   0.00554     0.01109       0.04374
1             0.00008   0.00465   0.00696     0.01394       0.06134
2             0.00009   0.00544   0.00781     0.01633       0.05233
3             0.00009   0.00502   0.00698     0.01505       0.05492
4             0.00011   0.00655   0.00908     0.01966       0.06425
..                ...       ...       ...         ...           ...
190           0.00003   0.00263   0.00259     0.00790       0.04087
191           0.00003   0.00331   0.00292     0.00994       0.02751
192           0.00008   0.00624   0.00564     0.01873       0.02308
193           0.00004   0.00370   0.00390     0.01109       0.02296
194           0.00003   0.00295   0.00317     0.00885       0.01884
```

**Preparing the dataset**

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)

print(X.shape, X_train.shape, X_test.shape)

(195, 22) (156, 22) (39, 22)
```

**Separating the train and test data**

```
scaler = StandardScaler()

scaler.fit(X_train)

StandardScaler()

X_train = scaler.transform(X_train)

X_test = scaler.transform(X_test)

print(X_train)

[[ 0.63239631 -0.02731081 -0.87985049 ... -0.97586547 -0.55160318
   0.07769494]
 [-1.05512719 -0.83337041 -0.9284778  ...  0.3981808  -0.61014073
   0.39291782]
 [ 0.02996187 -0.29531068 -1.12211107 ... -0.43937044 -0.62849605
  -0.50948408]
 ...
 [-0.9096785  -0.6637302  -0.160638   ...  1.22001022 -0.47404629
  -0.2159482 ]
 [-0.35977689  0.19731822 -0.79063679 ... -0.17896029 -0.47272835
   0.28181221]
 [ 1.01957066  0.19922317 -0.61914972 ... -0.716232    1.23632066
  -0.05829386]]
```

**Data Standardization**

```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)

print('Accuracy score of training data : ', training_data_accuracy)

Accuracy score of training data :  0.8846153846153846

# accuracy score on training data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)

print('Accuracy score of test data : ', test_data_accuracy)

Accuracy score of test data :  0.8717948717948718
```

**Calculating the Accuracy**

## 6.RESULTS

```
input_data = (197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.01098,0.09700,0.00563,0.006

# changing input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the data
std_data = scaler.transform(input_data_reshaped)

prediction = model.predict(std_data)
print(prediction)


if (prediction[0] == 0):
  print("The Person does not have the Disease")

else:
  print("The Person has the disease")

[0]
The Person does not have the Disease
```

**Prediction**

## 7.CONCLUSIONN AND FUTURE SSCOPE

### CONCLUSION

Most Neurological diseases affects the CNS of the brain and has yet no treatment unless it's detected early. Late detection leads to no treatment and loss of life. Thus, its early detection is significant. For early detection of the disease, we utilized machine learning algorithm Support Vector Machine. We found out that SVM is the best Algorithm which gives best accuracy compared to other algorithms, to predict the onset of the disease which will enable early treatment and save the lives.

## FUTURE SCOPE

In future work, we can focus on different techniques to predict the disease using different datasets. In this research, we using binary attribute (1- diseased patients, 0-non-diseased patients) for patient's classification. In the future we will use different types of attributes for the classification of patients and also identify the different stages for a disease.

## REFERENCES


- M. Chen, Y. Hao, K. Hwang, L. Wang, and L. Wang, Disease prediction by machine learning over big data from healthcare communities" IEEE Access, vol. 5, no.1, pp.
- X. Bi, S. Li, B. Xiao, Y. Li, G. Wang, and X. Ma, ''Computer aided Alzheimer's disease diagnosis by an unsupervised deep learning technology,'' Neurocomputing, vol. 392, pp. 296–304,Jun. 2020.
- X. Bi, X. Zhao, H. Huang, D. Chen, and Y. Ma, ''Functional brain network classification for Alzheimer's disease detection with deep features and extreme learning machine, Cognit. Compute, vol. 12, no. 3, pp. 513–527, May 2020.
- S. Sharma, R. K. Dudeja , G. S. Aujla, R. S. Bali, and N. Kumar, ''Detras: Deep learning-based healthcare framework for IoT-based assistance of alzheimer patients,'' Neural Computing. Appl., pp. 1–13,Sep. 2020.