Tec Empresarial
P-ISSN: 1659-2395; E-ISSN: 1659-3359

APPROACH OF MACHINE LEARNING FOR BRAIN DISEASE DETECTION

Dr. Rajeshri Pravin Shinkar
Asst. Prof., SIES (NERUL) College of Arts, Science & Commerce (Autonomous)

Abstract
Brain is very important part of our body. Brain is a remote control of our day-to-day activity. If any problem occurs in brain, then immediate reaction we can see in our activity. The disease which disorders the brain where it causes uncontrollable or unintended movements of our body such as balancing the body or co-ordination of the activity, stiffness or shaking the body. These are the symbols of the disease name Parkinson's.
Keywords: Parkinson's disease (PD), Machine Learning, Support Vector Machine, Feature Selection.

Introduction
Parkinson's disease (PD) is a progressive neurodegenerative disorder that affects millions of people worldwide. The disease is characterized by a gradual loss of dopamine-producing neurons in the brain, leading to motor symptoms such as tremors, stiffness, and difficulty with balance and coordination. Although there is no cure for PD, early diagnosis and treatment can significantly improve the quality of life for patients. Machine learning (ML) has shown promise in predicting PD by analysing large datasets of clinical and biological information. ML models can identify patterns and relationships within data that may not be visible to human analysts, and can be used to predict the likelihood of disease onset or progression based on patient characteristics. In recent years, several studies have investigated the use of ML for PD prediction, using various types of data such as medical records, imaging studies, and genetic information. These studies have shown that ML can accurately predict PD, potentially allowing for earlier diagnosis and treatment.

Objective

Machine learning predictive models will help to classify the people who are healthy and people who are suffering from Parkinson's Disease through ML based methods/algorithms. Different AI-based techniques for the classification are reasonable for being a good support for the expert. The Machine Learning Classification technique will help to improve the accuracy & result of the model and also the dependability of diagnosis and reduce possible loopholes, hence making the PD classification more time-

Problem Statement

The main aim is to predict the prediction efficiency that would be beneficial for the patients who are suffering from Parkinson and the percentage of the disease will be reduced. Generally, in the

first stage, Parkinson's can be cured by the proper treatment. So it‘s important to identify the PD at the early stage for the betterment of the patients. The main purpose of this research work is to find the best prediction model i.e. the best machine learning technique which will distinguish the Parkinson’s patient from the healthy person. The techniques used in this problem are SVM. The experimental study is performed on the voice dataset of Parkinson’s patients which is downloaded from the Kaggle. The prediction is evaluated using evaluation metrics like confusion matrix, precision, recall. The machine learning model we have created is around 75% to 80% accurate. The disease for which there are no diagnostics methods machine learning models are able to predict whether the person has Parkinson’s disease or not. This is the power of machine learning by using which many of the real-world problems are being solved.

Technical Terminology

Support Vector Machines

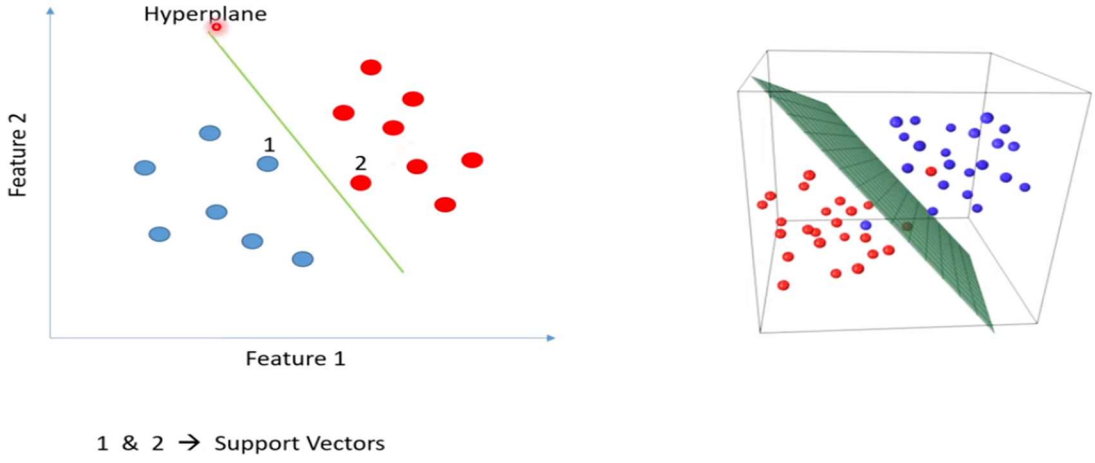
Support Vector Machines (SVM) is a powerful machine learning algorithm used for both classification and regression tasks. It works by finding an optimal hyperplane that separates different classes or predicts continuous values. SVM aims to maximize the margin, which is the distance between the hyperplane and the closest data points from each class. This approach allows SVM to generalize well and handle complex decision boundaries.

In this particular case we use Linear SVM, Linear SVM (Support Vector Machine) is a variant of the SVM algorithm that uses a linear decision boundary to separate different classes in the input data. In linear SVM, the goal is to find the best hyperplane that maximally separates the classes while maintaining a maximum margin between the decision boundary and the closest data points from each class.

The decision boundary in linear SVM is a linear combination of the input features, represented by a linear equation of the form:

$w^T * x + b = 0$

where w is the weight vector, x is the input feature vector, and b is the bias term.



The linear SVM algorithm aims to find the optimal values of the weight vector w and the bias term b by solving an optimization problem. The objective is to minimize the classification error while

maximizing the margin between the decision boundary and the support vectors, which are the data points closest to the decision boundary.

Linear SVM is particularly effective when the input data is linearly separable, meaning that there exists a hyperplane that can perfectly separate the classes. In our case, the goal is to separate the two hand positions “Open” and “Close” respectively using the Image inputs.

Program

Libraries Used

NumPy : is a Python library used for working with arrays.It also has functions for working in domain of linear algebra, fourier transform, and matrices.

Pandas : Pandas is an open-source Python library that consists of multiple modules for high-performance, easy-to-use data structures, and data analysis tools

Sklearn Model : Sklearn(or Sci kit-learn). It is a Python library that offers various features for data processing that can be used for classification, clustering, and model selection.train_test_split :train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. Sklearn train_test_split will make random partitions for the two subsets.

•X, Y the first parameter is the data set you're selecting to use.

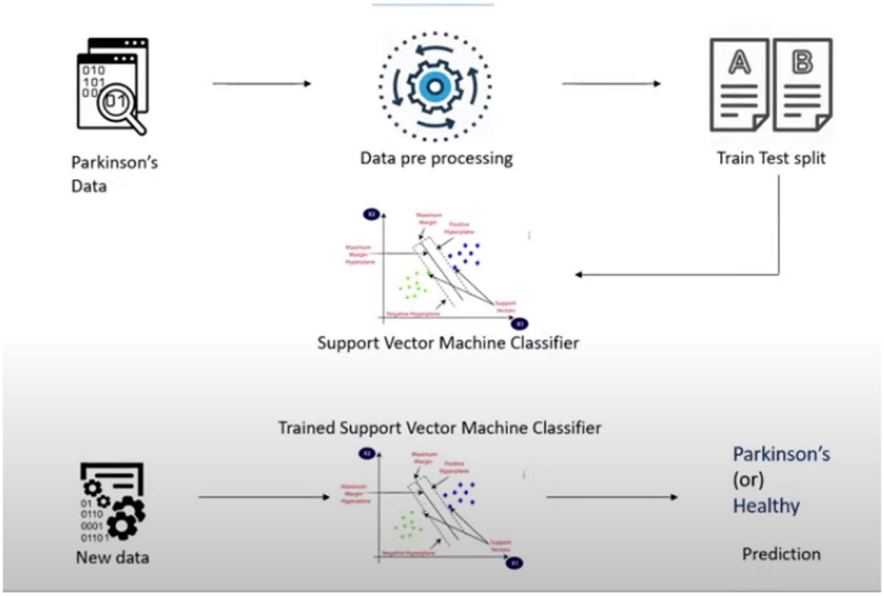
•train_size : This parameter sets the size of the training data set.

•The ideal split is said to be 80:20for training and testing. You may need to adjust it depending on the size of the data set and parameter complexity.

sklearn.metrics: The metrics module from the sci kit-learn library, which provides various evaluation metrics for machine learning models.

Working Of Program:

Training of the SVM



Data Preprocessing: The first step in developing a PD prediction model using ML is to preprocess the data. This typically involves cleaning the data, filling in missing values, and converting categorical variables into numerical variables. Data cleaning may involve removing outliers, correcting data entry errors, and dealing with inconsistencies in the data. Missing values can be filled in using techniques such as mean imputation, median imputation, or mode imputation. Categorical variables can be converted into numerical variables using techniques such as one-hot encoding or label encoding.

Feature Selection: The next step is to select the most relevant features for PD prediction. This can be done using various techniques such as correlation analysis, principal component analysis (PCA), or feature importance analysis. Correlation analysis involves identifying the features that are most strongly correlated with the target variable (PD). PCA involves reducing the dimensionality of the data by identifying the principal components that explain the most variance in the data. Feature importance analysis involves using ML models such as random forests or gradient boosting machines to identify the most important features for prediction.

Model Training: Once the relevant features have been identified, the next step is to train the ML model. This involves splitting the data into training and testing sets, fitting the model to the training data, and tuning the model hyper parameters. ML models that can be used for PD prediction include logistic regression, decision trees, random forests, support vector machines (SVMs), and neural networks.

Working Model

Importing the Dependencies

```
[ ]  1  from google.colab import drive
     2  drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[ ]  1  import numpy as np
     2  import pandas as pd
     3  from sklearn.model_selection import train_test_split
     4  from sklearn.preprocessing import StandardScaler
     5  from sklearn import svm
     6  from sklearn.metrics import accuracy_score
```

Data Collection & Analysis

```
[ ]  1  # loading the data from csv file to a Pandas DataFrame
     2  parkinsons_data = pd.read_csv('/content/parkinsons.data')
```

```
[ ]  1  # printing the first 5 rows of the dataframe
     2  parkinsons_data.head()
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHR	HNR	status	
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545	0.02211	21.033	1	0.
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403	0.01929	19.085	1	0.
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270	0.01309	20.651	1	0.
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771	0.01353	20.644	1	0.
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470	0.01767	19.649	1	0.

5 rows × 24 columns

```
1  # number of rows and columns in the dataframe
2  parkinsons_data.shape
```

```
(195, 24)
```

```
1  # getting more information about the dataset
2  parkinsons_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   name              195 non-null    object
 1   MDVP:Fo(Hz)       195 non-null    float64
 2   MDVP:Fhi(Hz)      195 non-null    float64
 3   MDVP:Flo(Hz)      195 non-null    float64
 4   MDVP:Jitter(%)    195 non-null    float64
 5   MDVP:Jitter(Abs)  195 non-null    float64
 6   MDVP:RAP          195 non-null    float64
 7   MDVP:PPQ          195 non-null    float64
 8   Jitter:DDP        195 non-null    float64
 9   MDVP:Shimmer      195 non-null    float64
 10  MDVP:Shimmer(dB)  195 non-null    float64
 11  Shimmer:APQ3      195 non-null    float64
 12  Shimmer:APQ5      195 non-null    float64
 13  MDVP:APQ          195 non-null    float64
 14  Shimmer:DDA       195 non-null    float64
 15  NHR               195 non-null    float64
 16  HNR               195 non-null    float64
 17  status            195 non-null    int64
 18  RPDE              195 non-null    float64
 19  DFA               195 non-null    float64
 20  spread1           195 non-null    float64
 21  spread2           195 non-null    float64
 22  D2                195 non-null    float64
 23  PPE               195 non-null    float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.7+ KB
```

```
# checking for missing values in each column
parkinsons_data.isnull().sum()
```

```
name              0
MDVP:Fo(Hz)       0
MDVP:Fhi(Hz)      0
MDVP:Flo(Hz)      0
MDVP:Jitter(%)    0
MDVP:Jitter(Abs)  0
MDVP:RAP          0
MDVP:PPQ          0
Jitter:DDP        0
MDVP:Shimmer      0
MDVP:Shimmer(dB)  0
Shimmer:APQ3      0
Shimmer:APQ5      0
MDVP:APQ          0
Shimmer:DDA       0
NHR               0
HNR               0
status            0
RPDE              0
DFA               0
spread1           0
spread2           0
D2                0
PPE               0
dtype: int64
```

```
# getting some statistical measures about the data
parkinsons_data.describe()
```

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer(dB)	...	Shimmer:DDA	NHR	
count	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	195.000000	...	195.000000	195.000000	195.000
mean	154.228641	197.104918	116.324631	0.006220	0.000044	0.003306	0.003446	0.009920	0.029709	0.282251	...	0.046993	0.024847	21.885
std	41.390065	91.491548	43.521413	0.004848	0.000035	0.002968	0.002759	0.008903	0.018857	0.194877	...	0.030459	0.040418	4.425
min	88.333000	102.145000	65.476000	0.001680	0.000007	0.000680	0.000920	0.002040	0.009540	0.085000	...	0.013640	0.000650	8.441
25%	117.572000	134.862500	84.291000	0.003460	0.000020	0.001660	0.001860	0.004985	0.016505	0.148500	...	0.024735	0.005925	19.198
50%	148.790000	175.829000	104.315000	0.004940	0.000030	0.002500	0.002690	0.007490	0.022970	0.221000	...	0.038360	0.011660	22.085
75%	182.769000	224.205500	140.018500	0.007365	0.000060	0.003835	0.003955	0.011505	0.037885	0.350000	...	0.060795	0.025640	25.075
max	260.105000	592.030000	239.170000	0.033160	0.000260	0.021440	0.019580	0.064330	0.119080	1.302000	...	0.169420	0.314820	33.047

8 rows × 23 columns

```
# distribution of target Variable
parkinsons_data['status'].value_counts()
```

```
# grouping the data bas3ed on the target variable
parkinsons_data.groupby('status').mean()
```

```
<ipython-input-10-fe279e55666c>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Eit
  parkinsons_data.groupby('status').mean()
```

status	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer(dB)	...	MDVP:APQ	Shimmer:DDA	NHR
0	181.937771	223.636750	145.207292	0.003866	0.000023	0.001925	0.002056	0.005776	0.017615	0.162958	...	0.013305	0.028511	0.011483
1	145.180762	188.441463	106.893558	0.006989	0.000051	0.003757	0.003900	0.011273	0.033658	0.321204	...	0.027600	0.053027	0.029211

2 rows × 22 columns

```
..          ...     ...     ...     ...     ...
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

     MDVP:Shimmer(dB)  ...  MDVP:APQ  Shimmer:DDA      NHR     HNR      RPDE  \
0               0.426  ...   0.02971      0.06545  0.02211  21.033  0.414783
1               0.626  ...   0.04368      0.09403  0.01929  19.085  0.458359
2               0.482  ...   0.03590      0.08270  0.01309  20.651  0.429895
3               0.517  ...   0.03772      0.08771  0.01353  20.644  0.434969
4               0.584  ...   0.04465      0.10470  0.01767  19.649  0.417356
..                ...  ...       ...          ...      ...     ...       ...
190             0.405  ...   0.02745      0.07008  0.02764  19.517  0.448439
191             0.263  ...   0.01879      0.04812  0.01810  19.147  0.431674
192             0.256  ...   0.01667      0.03804  0.10715  17.883  0.407567
193             0.241  ...   0.01588      0.03794  0.07223  19.020  0.451221
194             0.190  ...   0.01373      0.03078  0.04398  21.209  0.462803

          DFA   spread1   spread2        D2       PPE
0    0.815285 -4.813031  0.266482  2.301442  0.284654
1    0.819521 -4.075192  0.335590  2.486855  0.368674
2    0.825288 -4.443179  0.311173  2.342259  0.332634
3    0.819235 -4.117501  0.334147  2.405554  0.368975
4    0.823484 -3.747787  0.234513  2.332180  0.410335
..        ...       ...       ...       ...       ...
190  0.657899 -6.538586  0.121952  2.657476  0.133050
191  0.683244 -6.195325  0.129303  2.784312  0.168895
192  0.655683 -6.787197  0.158453  2.679772  0.131728
193  0.643956 -6.744577  0.207454  2.138608  0.123306
194  0.664357 -5.724056  0.190667  2.555477  0.148569

[195 rows x 22 columns]
```

```
1  print(Y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
190    0
191    0
192    0
193    0
194    0
Name: status, Length: 195, dtype: int64
```

+ Code + Text

Splitting the data to training data & Test data

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

```
[ ]  1  print(X.shape, X_train.shape, X_test.shape)
```

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

Data Standardization

```
[ ]  1  scaler = StandardScaler()
```

```
[ ]  1  scaler.fit(X_train)
```

```
StandardScaler
StandardScaler()
```

```
[ ]  1  X_train = scaler.transform(X_train)
     2
     3  X_test = scaler.transform(X_test)
```

```
1  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]]
```

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

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

```
Accuracy score of test data :  0.8717948717948718
```

ling a Predictive System

```
1  input_data = (197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.01098,0.09700,0.00563,0.00680,0.00802,0.01689,0.00339,26.77500,0.422229,0.741367,-7.348300,
2
3  # changing input data to a numpy array
4  input_data_as_numpy_array = np.asarray(input_data)
5
6  # reshape the numpy array
7  input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
8
9  # standardize the data
10 std_data = scaler.transform(input_data_reshaped)
11
12 prediction = model.predict(std_data)
13 print(prediction)
14
15
16 if (prediction[0] == 0):
17   print("The Person does not have Parkinsons Disease")
18
19 else:
20   print("The Person has Parkinsons")
```

Results

Here in this project, we are going to collect the patient details from datasets. These details will be of various forms of parameters such as age, height, weight, walking gait, etc. Speech dataset: The Multi-Dimensional Voice Program (MDVP) is a well-established software program used for quantitative acoustic signal assessment of voice quality. The MDVP calculates a number of acoustic parameters including shimmer, short-term perturbations of the amplitude, and jitter, short-term perturbations of the frequency. Here we have used data of 195 patients. MDVP: The Multi-Dimensional Voice Program (MDVP) is a computer program that can calculate as many as 33 acoustic parameters from a voice sample. It is standard software for acoustic assessment which is widely used by many researchers in the voice field for being very comprehensive. The MDVP appears to have potential for rapid quantitative assessments of voice in both research and clinical applications it diagnosis of pediatric vocal cord dysfunction. Dysphonia is a phonation disorder with the difficulty in the voice production. Dysphonia can be observed with hoarse, harsh, or breathy vowel sounds, as a result of impaired ability of the vocal folds to properly vibrate during exhalation. Here we have used data of 77 patients. These datasets will have all patients' details according to our need. We need to mine the data from the set of given data. Here the concept of machine learning is been used. Here the input data are being pre-processed according to our need, Parkinson's affected people are been tabulated and using machine learning algorithm we are predicting how patients are being affected. The below graph Fig 2 is the graph that represents the accuracy rate using different algorithms and speech datasets for detection of Parkinson's disease. Fig 3 shows the accuracy rate using different algorithms and tremor datasets for detection of Parkinson’s disease.

Conclusion

Parkinson’s disease 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 algorithms such as SVM and Random Forest. We checked our Parkinson disease data and find out SVM is the best Algorithm to predict the onset of the disease which will enable early treatment and save a life. In this process we can predict the Parkinson’s disease in patient’s body using machine learning technology and this method makes the process easy to our user. Our analysis provides very accurate performance in detecting Parkinson's disease using SVM algorithm.

References

1. D. Calne, ‘‘Is idiopathic parkinsonism the consequence of an event or a process?’’ Neurology, vol. 44, no. 1, pp. 5–10, Jan. 1994.
2. Z. Fan, F. Xu, X. Qi, C. Li, and L. Yao, ‘‘Classification of Alzheimer’s disease based on brain MRI and machine learning,’’ Neural Comput. Appl., vol. 32, no. 7, pp. 1927–1936, Apr. 2020

3. R. Jain, N. Jain, A. Aggarwal, and D. J. Hemanth, ‘‘Convolutional neural network based Alzheimer’s disease classification from magnetic resonance brain images,’’ Cognit. Syst. Res., vol. 57, pp. 147–159, Oct. 2019.
4. C. S. Eke, E. Jammeh, X. Li, C. Carroll, S. Pearson, and E. Ifeachor, ‘‘Early detection of Alzheimer’s disease with blood plasma proteins using support vector machines,’’ IEEE J. Biomed. Health Informat., vol. 25, no. 1, pp. 218–226, Jan. 2021.