Abstract

This research paper carries a detailed comparative analysis of six machine learning algorithms, which are random forest, K-nearest neighbour (KNN), support vector machine (SVM), XG boost, multiple layer perceptron (MLP), and deep neural network (DNN), with the primary objective of predicting PD. There is presently no treatment for the neurological condition Parkinson's disease., making early detection crucial for managing symptoms and improving patient outcomes. However, the absence of definitive medical or blood tests for diagnosis has necessitated the exploration of machine learning-based prediction models.

The UCI Machine Learning Repository has the dataset that was utilised in this work, which was acquired from the University of Oxford. It comprises 195 instances from 31 individuals, providing a robust foundation for training and testing the algorithms. To enhance the performance of these models, we employed various techniques such as dimension reduction and feature ranking. These techniques helped in refining the dataset and focusing on the most relevant features for disease prediction.

Several metrics, including Accuracy, Precision, Recall, F1-score, ROC AUC score, and confusion matrix, were used to assess each algorithm's performance. These measures provided a comprehensive understanding of each model’s strengths and weaknesses.

The K-NN classifier, when applied to the original dataset, emerged as the most effective model with an impressive accuracy and recall score of 97%. This indicates that the K-NN classifier correctly identified 97% of the cases, demonstrating its potential as a reliable tool can aid in the early diagnosis of Parkinson's illness.

The XG Boost algorithm also showed promising results with an accuracy rate of 95% when used with the original dataset. This high level of accuracy highlights the potential of machine learning algorithms for illness prediction and diagnosis.

This study highlights the potential of machine learning algorithms for Parkinson's disease prediction. While challenges remain, our findings provide a strong foundation for future research in this critical area of healthcare.

CONTENTS

CHAPTER 1 INTRODUCTION

1.1 Parkinson's Disease – An Introduction

After Alzheimer's disease, the most prevalent form of common neurological disorder in the world. It is caused mainly by the breakdown of cells in the nervous system. It was called after James Parkinson, sometimes known as PD, who labelled it paralysis agitans and provided his last name. These neurological conditions are brought on by the gradual destruction and death of neurons inside the nervous system. The brain's functioning components are called neurons. PD causes progressive degeneration of neurons in a certain area of the human brain called the substantia nigra. This part is responsible for producing dopamine. It serves a crucial function in movement regulation. The loss of dopamine causes symptoms like tremors, rigidity, etc. The degeneration of neurons significantly reduces the production of dopamine, and this leads to disruption of communication between brain regions and symptoms of PD. It has been found that the accumulation of proteins like alpha-synuclein causes the degeneration of neurons.

The number of PD patients is increasing at an alarming rate. (Ali H. Rajput, 2007) report that every 20 people in one lakh are detected with PD. According (Campenhausen, 2005), it is estimated that there has been a rate of prevalence of around 250 per million and a rate of incidence of eleven to nineteen per million in Europe. PD patients are estimated to be numbering above ten lakhs in North America. According to the study, titled "Incidence of Parkinson's Disease in North America," there has been an increase of 50% in PD patients over the years. It was reported at around 60,000, and now it has risen to 90,000 a year. In China, it is estimated that PD will reach fifty per cent of its population. And alarmingly, according to Schrag et al. (2002), twenty per cent of PD cases go undiagnosed. India had around more than half a million PD cases in 2016 and this is expected to rise at a higher rate.

PD is mainly diagnosed in older people. The likelihood that an aged person may get Parkinson's disease is very high. In every study, it is observed that a person's age plays a crucial role as a factor for PD, with the risks increasing at a steep rate when age crosses 50, as per Elbaz et al., 2000. Since this disorder is progressive, the symptoms of the disease will worsen as age progresses. Additionally, the research revealed that men are more vulnerable to PD prevalence than women. (Baldereschi et al., 2000; Haaxma et al., 2007).

A lot of factors are present that influence the chance of contracting PD. One of the important factors is genetics. The primary molecular and physiological cause of neurodegenerative illnesses is thought to be a genetic factor. The measurements and their impacts on various genes and their actions determine the condition or seriousness of neurodegenerative illness, which gradually worsens over time. Pharmacodynamic and pharmacokinetic genetic variables are mostly what cause neurodegenerative diseases. Another important factor is the environment. The surroundings, or the location where a person resides, are the environment. Therefore, the environment is a significant aspect that has an impact not only on the brain of humans but also on other life forms. Numerous studies and statistics indicate that the environment has a substantial impact on the onset of neurodegenerative diseases, including Alzheimer's and Parkinson's. The following environmental elements are having a rapid impact on neurodegenerative disorders:

- Contact with insecticides and heavy metals (such as lead and aluminium)
- Air quality: respiratory ailments are brought on by pollution.
- Water quality: Pollution of the water is caused by biotic and abiotic substances.
- Unhealthy lifestyle: This breeds sedentary behaviour and obesity.

Major symptoms of Parkinson's disease include the following:

- Tremors affect the body parts, mainly the hands, arms, legs, jaw, or head.
- Muscle stiffness caused by muscle contraction for an extended period.
- Slower movements
- affected balance and coordination of the body, which may cause falling.

Along with these symptoms, patients may also experience emotional problems such as depression and others. Also, they might have trouble swallowing, chewing, or speaking. There are also symptoms related to urinary and constipation issues. They are also prone to skin problems.

Parkinson's disease symptoms and pace of development vary from person to person. At first, the signs will show slowly and minutely. For example, at an earlier stage, a person with PD may experience light tremors and difficulty standing from sitting. They also experience that

their sound is too low or that their handwriting is small or not neat enough. It is his or her family or friends who are the first to notice the change in the patient. They notice that his or her face may appear without expression, there is no emotion going around in the face, or their movement with a hand or leg is different from theirs.

Parkinsonian gait, which includes characteristics of a propensity to lean forward, having little or huge footfall, and reducing the swinging of their arms, is a gait that persons with Parkinson's disease gradually develop. They will struggle to start or regulate movement. At initially, only one side of the body or perhaps a single limb is affected by the symptoms. Then it spreads to other parts of the body, and gradually its intensity also increases. There will be an imbalance of symptoms between the sides, even after spreading to both sides. There are also reports that people with PD also experience trouble with their sleep, constipation, Odor lessness, and tiredness of the legs, even before visible symptoms like trembling or stiffness. Symptoms also increase with age.

1.2 Diagnosis and Treatments

Diagnosis

For the diagnosis or detection of Parkinson’s disease, currently, no blood or laboratory test is available to diagnose. Medical professionals presently conduct a neurological exam and review a patient's medical history to determine whether they have Parkinson's disease (PD).Also, we can detect the presence of PD if the symptoms are improving after taking medicines.

Another challenge in detecting PD is the similarity in symptoms with other diseases. Sometimes people may experience symptoms like PD for diseases like system atrophy and dementia with Lewy bodies. These diseases may also be misdiagnosed as Parkinson’s but certain tests and responses to medicine can evaluate it better. So, it is very important to find an accurate diagnosis test to Fast-track the treatment.

Treatments

Another main and scary issue with PD is that there is no cure for the disease, even though certain medicines, surgeries, or other methods of treatment can reduce or improve the symptoms.

1. Medicines

Medicines can help relieve the symptoms.

- Improve the dopamine level in the brain.

- making an impact on other brain chemicals such as neurotransmitters, which help to share information between cells of the brain.
- Nonmotor symptoms can be improved.

The most used medicine for PD treatment is levodopa. Neural cells, with the help of this medicine, make dopamine to revive the brain's diminishing supply of messages since the depletion of dopamine is the major cause of PD. Along with levodopa, patients also use a medicine named carbidopa. which helps to diminish the side effects caused by levodopa medicine, which include low blood pressure, uneasiness, a tendency to vomit, sickness, etc. Also, carbidopa helps reduce the intake of levodopa to produce dopamine. Levodopa also comes with other serious side effects, such as sudden stopping of intake, which may cause a condition of not being able to move or breathing difficulty. Medical practitioners also prescribe dopamine agonists for increased dopamine production. Along with these medicines, they also prescribe enzyme inhibitors for increasing dopamine levels by reducing enzyme breaking time for breaking dopamine in the brain, amantadine for decreasing involuntary movements, and anticholinergic drugs for decreasing rigidity of muscles and tremors.

2. Stimulation of the deep brain

Some people may not respond well to medication. For those patients, medical practitioners recommend stimulation of the deep brain. In this surgical process, they install electrodes inside the brain and connect them to an electrical appliance installed on the chest. These devices and electrodes stimulate certain areas of the brain where movement is controlled painlessly to stop symptoms related to movements such as slow movement, rigidity, and tremors.

3. Other treatments

- Therapies such as physical, occupational, and speech therapy help to fight against disorders in voice and movement issues and reduce decreases in mental functions.
- Maintaining a good diet for overall fitness
- Physical exercises for strengthening muscles and improving body balance and coordination.
- For reducing tension, massage therapy is used.
- To increase stretching and flexibility, therapies like yoga are used.

1.3 Motivation of the work

The number of 65+ individuals or older with neurodegenerative disease is alarmingly high, and it is growing each year. And the scarier thing about neurodegenerative diseases is that there has

been no permanent cure for them so far. But if we can detect the disease at an earlier stage, we may have the chance to control it. Out of these numbers, nearly thirty per cent of people have a disease in an incurable stage. Only treatments are available to those who have minor symptoms. The buildup of protein molecules in the cell, which results in neuronal degeneration, is what causes Parkinson's disease.

Until now, researchers have identified the root cause and symptoms of PD but have been unable to find a permanent cure. We are now providing treatments to improve the symptoms. The sooner the PD is diagnosed, the better the chance of giving proper treatment. The challenging part of diagnosing PD is that there has been no medical or blood test available to identify the disease, and only the doctor’s experience in the matter has been the way to diagnose the disease until now. There will also be a chance of mixing up the diagnosis with another disease due to the similarities in the symptoms. All these issues indicate the urgent need for novel detection of PD for better treatment. In this situation, the diagnosis of PD with machine learning comes in handy.

1.4 Problem Statement

This study's primary objective is to examine different machine learning methods for identifying PD patients from healthy individuals. Our aim to predict the efficiency of prediction would be immensely helpful for people with symptoms to diagnose their disease at an early stage. It would enable them to get proper treatment for this disease.

It is crucial to determine PD at earlier stage for better treatment of patients. This research paper intends to find the best prediction model based on their efficiency and other performance measures for distinguishing PD patients from healthy individuals. In this study, we evaluate the performance of several machine learning methods, including K-nearest neighbour (KNN), support vector machine (SVM), random forest, XG boost, multiple layer perceptron, and deep neural network (DNN). The National Centre for Voice, founded by Little et al., assisted in obtaining the dataset from the University of Oxford (UO) repository for use in the study. Performance metrics including precision, recall accuracy, and f1-score are used to analyse the prediction. The author also employs feature selection, where the significance of each feature is considered along with its value for prediction. In this study, we also use feature reduction techniques like PCA to check how they will perform and try to plot the data into a 2D graph with the help of t-SNE.

1.5 Organization of Thesis

This study is partitioned into five distinct sections. The first section covers an introduction to Parkinson's disease, its symptoms, and current diagnosis techniques. Within this section, we will also discuss our project topic and related concepts. The second section provides an overview of previous studies and research on this topic in a Literary Survey. The third section introduces the system architecture, datasets, various algorithms, and other methodologies. In the fourth section, we'll carry out an experimental examination of our system, quantify its performance, and compare it to other models. Finally, the concluding section of this thesis offers a conclusion about the work and prospects.

CHAPTER 2 LITERATURE REVIEW

The dataset utilized in the research is assumed to have an accuracy of 90% to distinguish a person from whether he has PD or not. It is very important to detect or identify the disease at the very earliest so that the rate of progression of PD can slow down. There have been a good number of studies conducted on this or similar dataset which all use different algorithms to better predict the disease. One of the main areas that affects PD patients is the speech or vocal area. There are mainly two kinds of speech problems, and they are hypophonia and dysarthria. Too soft and weak voice from the patient indicates that it is hypophonia and slow speech or voice which is very difficult to interpret to the listener indicates dysarthria and at it is considered more serious than hypophonia because it may cause difficult to central neural system. Therefore, medical practitioners who examines PD check dysarthria and take necessary steps to improve intensity of voice. There has been a lot of study on this topic along with that many researchers have tried various pre-processing and feature selection technique to better predict the disease.

Ghaida Alharbi, Ebtisam Alharbi, Ibrahim Almubark, and Raya Alshammri proposed a paper titled "Machine learning approaches to identify Parkinson's disease using voice signal features (2023)" which compares various machine learning models such as KNN, SVM, Decision Tree, Random Forest and Multi-layer perceptron models. In this paper, the authors use feature selection technique like SelectKBest and then uses SMOTE (Synthetic Minority Over-Sampling Technique) to reduce the imbalance between Healthy samples and patient sample as their ratio is about 1:3. Then after they used Hyper parameter tuning like GridSearchCV for finding the optimum hyper parameter values. Between these machine learning models Multilayer perceptron model with SMOTE and GridSearchCV came as the best model with accuracy of 98% followed by SVM with 95%. It is understood that feature selection techniques are not helping to achieve higher accuracies. Therefore, all the features are used to build the model.

Hariharan, Kemal Polat, R. Sindhu proposed the paper titled "A new hybrid intelligent system for accurate detection of Parkinson's disease (2014)" which compares three different classification models; Least Probabilistic neural network (PNN), general regression neural network (GRNN), and SVM. Rather than the machine learning models, it is the preprocessing

technique Gaussian Mixture Model (GMM) is highlighted in this paper. It is a preprocessing technique for feature weighting which allows to discriminate class more easily. Later that they use PCA and LDA for feature reduction and sequential backward selection and sequential forward selection for feature selection. It is found that 1-nearest neighbor leave-one-out classification performs better. Here author also uses 10-fold cross validation technique and compares them. The performance accuracy ranges from 95% to up to even 100% in the study.

Jianping Li, Asad Malik, Tanvir Ahmad, Amjad Ali, Shah Nazir, Ijaz Ahad, Mohammad Shahid, Muhammad Hammad Memon, Jalaluddin Khan, and Amin Ul Haq proposed the paper titled "Feature Selection Based on L1-Norm Support Vector Machine and Effective Recognition System for Parkinson's disease Using Voice Recordings" and this study compares SVM model with and without L1 Norm SVM classification for feature selection. In this paper, the authors first preprocess the data with StandardScaler and Min Max Scaler then uses L1 norm SVM classification for feature selection. Later, it applies 10-fold cross validation to SVM for machine learning modelling. It is found that L1 norm SVM for feature selection performs better than using all 22 features and the execution time is very less compared to later. L1 norm SVM classifier can achieve 99% accuracy.

"Comparative analysis of the classification performance of machine learning classifiers and deep neural network classifier for prediction of Parkinson's disease" is a paper proposed by Salah Ud Din, Ijaz Ahad, Ruinan Sun, Zhilong Lai, Jianping Li, Muhammad Hammad Memon, Jalaluddin Khan, Amin Ul Haq, and Salah Ud Din, Ijaz Ahad, Ruinan Sun, Zhilong Lai, Jianping Li, Muhammad Hammad Memon, Jalaluddin Khan, Amin Ul Haq, and. In this paper, the authors compare three machine learning models, SVM, K nearest neighbour and logistic regression along with deep neural network (DNN). In preprocessing stage, the dataset is applied with StandardScaler and Min Max Scaler. Later dataset is split into 70-30 ratio and applied the machine learning models. In this study K-NN came out to be the best classifier with accuracy of 94% among machine learning algorithms and overall DNN came out to be the best with an accuracy of 98%.

Akshaya Dinesh, Jennifer He proposed the paper "Using Machine Learning to Diagnose Parkinson's Disease from Voice Recordings" in which the authors used different classification methods such as Boosted Decision Tree, Decision Forests, Decision Jungle, Locally Deep SVM, Logistic Regression, Neural Networks, and SVM are some examples of machine

learning algorithms. They used Azure Machine Learning Platform for analysis. Out of these models, boosted decision tree model has the most accuracy with 91%. For feature selection the author used Pearson correlation scoring method for identifying top 10 features.

"An Enhanced Chaos-based Firefly model for Parkinson's disease Diagnosis and Classification" is the paper proposed by Sujata Dash, Ruppa Thulasiram and Parimala Thulasiraman. Through the integration of the Enhanced Chaos Firefly Algorithm, SVM, and RBF kernel, this work suggests an unique prediction model. This study also compares the above model ECFA SVM with GA-SVM (Genetic Algorithm – SVM) and CFA-SVM. First, the dataset is normalised. For feature selection we use GA and ECFA. By reducing unnecessary and redundant data from the dataset, this approach uses the ECFA model to discover viable feature combinations. In the initial step of ECFA deployment, GA is utilised to produce potential candidate solutions for the population, and subsequently logistic map-based CFA is used to update the population. The learning accuracy of the SVM defines the fitness value of the firefly, and the ECFA finds the optimum combination of features by adaptively scanning the feature space. ECFA-SVM came out be the best classification with an accuracy of 97.95%.

Zahari Abu Bakar, Dzufi Iszura Ispawi, Nur Farahiah Ibrahim and Nooritawati Md Tahir proposed the paper "Classification of Parkinson's disease Based on Multilayer Perceptrons (MLPs) Neural Network and ANOVA as a Feature Extraction". This study compares the two MLP neural network algorithms Levenberg-Marquardt (LM) and Scaled Conjugate Gradient (SCG) with and without the use of ANOVA. The challenge of minimising non-linear functions is addressed by the LM method. Compared to LM, SCG performs similarly but uses modest memory power. ANOVA has been able reduce the features into 6 that too with a good F score which is more than 1000. It is found that LM performs better than SCG in almost every hidden layer with peak accuracy of 97.86 which is before ANOVA whereas SCG's max. score is 79% for the same. With ANOVA, LM has able to achieve an accuracy of 90.81 compared to 83.76 of SCG's.

Anuj Anand, Md Amaan, Nithya, John Sahaya Rani Alex proposed the paper "Evaluation of Machine learning and Deep learning algorithms combined with dimensionality reduction techniques for classification of Parkinson's Disease". In this study the authors compare 5 different machine learning models (Logistic Regression, KNN, Naïve bayes, Random Forest, SVM, Decision tree) and deep neural network along with and without 2-dimension reduction

techniques which are PCA and Kernal PCA (KPCA). Along with this, grid search is also done for optimal parameters for each algorithm and cross validation is also done. This study also tries to find how much Time Complexity is taken for each algorithm. In this study, KNN classifier after the PCA dimensity reduction technique came out to be the best classification with an accuracy oof 95.52%.

"Diagnosis of The Parkinson Disease Using Enhanced Fuzzy Min-Max Neural Network and OneR Attribute Evaluation Method" is a paper proposed by Mohammad Falah Mohammad and Osama Nayel Al Sayaydeha. The authors of this paper developed an improved fuzzy min-max neural network classifier for Parkinson's disease identification using the OneR attribute assessment approach (EFMM-OneR). Here, OneR is a feature selection method based on error. OneR proposes candidate feature subsets and choose features with least errors after data normalisation. EFMM is relatively new classification which is an improved version of FMM. FMM overcomes the problem of catastrophic forgetting problem of neural networks. Later, various machine learning and neural network models are compared to this one.

Ali H. Al-Fatlawi, Mohammed H. Jabardi, Sai Ho Ling proposed the paper "Efficient Diagnosis System for Parkinson's Disease Using Deep Belief Network". This study proposes a deep belief network (DBN) for Parkinson’s predictionDBN, which consists of two stacked Restricted Boltzmann Machines (RBMs) and one output layer, is used to categorise Parkinson's disease. The parameters of the networks must be optimised using two phases of learning. Unsupervised learning is the first step, which employs RBMs to solve the issue that might arise from the initial weights' unpredictable beginning value. Second, supervised learning using the backpropagation technique is employed for the finishing touches. This model has achieved an accuracy of 94%.

Rahib H. Abiyev and Sanan Abizade propose the paper "Diagnosing Parkinson’s Diseases Using Fuzzy Neural System". This paper proposes fuzzy neural system for PD diagnosis. It has been identified that FNS improves the recognition rate. This model has been able to achieve an accuracy of 100%

Name of the study	Authors	Best Classifier	Feature selection/reduction technique	Accuracy
Machine Learning Approaches to Identify Parkinson's Disease Using Voice Signal Features	Raya Alshammri, Ghaida Alharbi, Ebtisam Alharbi and Ibrahim Almubark 2023	Multilayer Perceptron (MLP)	Gridsearchcv Along With SMOTE	98%
A New Hybrid Intelligent System for Accurate Detection of Parkinson's Disease	M. Hariharan, Kemal Polat, R. Sindhu 2014	General Regression Neural Network (GRNN)	Gaussian Mixture Model (GMM)	100%
Comparative Analysis of The Classification Performance of Machine Learning Classifiers and Deep Neural Network Classifier for Prediction of Parkinson Disease	Amin Ul Haq, Jianping Li, Muhammad Hammad Memon, Jalaluddin Khan, Salah Ud Din, Ijaz Ahad, Ruinan Sun, Zhilong Lai	Deep Neural Network	Standscaler And Minmax Scaler	98%
Using Machine Learning To Diagnose Parkinson's Disease From Voice Recordings	Akshaya Dinesh, Jennifer He	Boosted Decision Tree	Pearson Correlation Scoring Method	91%
An Enhanced Chaos-Based Firefly Model for Parkinson's Disease Diagnosis and Classification	Sujata Dash, Ruppa Thulasiram, Pa rimala Thulasiraman	Enhanced Chaos Firefly Algorithm and SVM With RBF Kernel (ECFA SVM)	CFA & Genetic Algorithm (GA)	97.95%
Classification Of Parkinson's Disease Based on Multilayer Perceptrons (Mlps) Neural Network and ANOVA As a Feature Extraction	Zahari Abu Bakar, Dzufi Iszura Ispawi, Nur Farahiah Ibrahim, Nooritawati Md Tahir	Levenberg-Marquardt (LM) On MLP	Nil	97.85%

Diagnosis Of the Parkinson Disease Using Enhanced Fuzzy Min-Max Neural Network and Oner Attribute Evaluation Method	Anuj Anand, Md Amaan, Nithya, John Sahaya Rani Alex	KNN	PCA	95.51%
Efficient Diagnosis System for Parkinson's Disease Using Deep Belief Network	Osama Nayel Al Sayaydeha, Mohammad Falah Mohammad	Fuzzy Min-Max Neural Network and Oner Attribute Evaluation Method (EFMM Oner)	Oner	94.21%
Diagnosing Parkinson's Diseases Using Fuzzy Neural System	Ali H. Al-Fatlawi, Mohammed H. Jabardi, Sai Ho Ling	Deep Belief Network	Nil	94%
	Rahib H. Abiyev and Sanan Abizade	Fuzzy Neural System (Fns)	Nil	100%

Table 2.1 Comparison of best algorithms from various studies

Chapter 3 Methodology

3.1 Proposed System

3.1.1 System Architecture

Machine learning is a technique which the system can learn its own without the need of programming explicitly. In this research, K-nearest neighbour (KNN), support vector machine (SVM), random forest, XG boost, multiple layer perceptron, and deep neural network are the six machine learning techniques that the author compared (DNN). To get the overview of main components of system and its important work relationships architecture diagram is highly helpful. Architecture diagram shows the execution flow, and it consists of following steps:

- Voice dataset obtained from UCI Machine Learning repository will be the first block of flow chart for the architecture diagram and this dataset is used for distinguishing PD.
- The following step in the process is to pre-process the dataset so that the data can be formatted to readable data.
- Moving forward in the diagram we split the voice dataset into 2 components which are test data and training data. We typically divide the dataset in an 80:20 ratio. The next step is to use this training dataset.
- The training data is used to a variety of machine learning techniques in the following stage, including K-nearest neighbour (KNN), support vector machine (SVM), random forest, XG boost, multiple layer perceptron, and deep neural network (DNN). Along with this we also measure various performance measures of the models.
- In the following block of the architecture diagram these models are tested with the test data.
- At last, the various performance measures of machine learning models are compared.

These are the major steps involved in the system architecture. This process is carried out in order. In addition to these stages, we also look at the idea of using feature reduction techniques like PCA and feature selection techniques like SelectKBest to determine the dataset's most crucial features. Graphical chart of system architecture is drawn below.

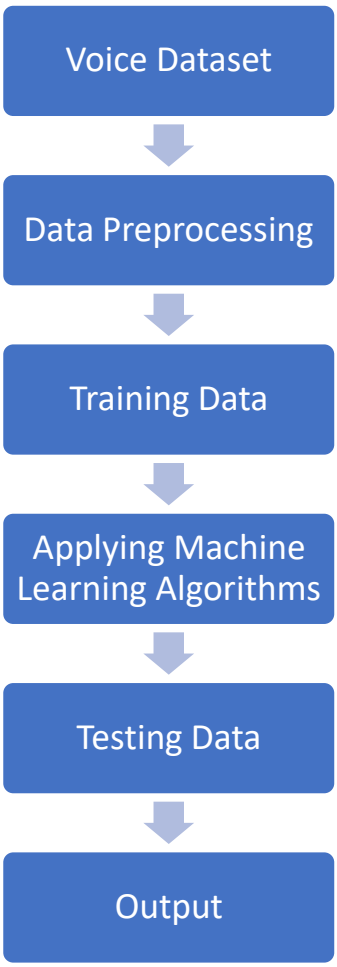

Figure 3.1 System Architecture

3.2 Voice Dataset

The major goal of this module is to find and collect and address all issues regarding data. It is very important to find the accurate data along with its authenticity and make sure that the taken would be beneficial for the study. The University of California, Irvine (UCI) machine learning repository provided the voice dataset, which was composed of 195 voice recordings of exams performed on 31 patients.

Link to the Voice Dataset

The dataset utilised in the study is hosted in the UCI Machine Learning Repository and was obtained from the University of Oxford (UO) repository with help from the National Centre for Voice, which was established by Little et al (2007, 2009). In the original paper, feature extraction techniques for voice problems in general were described. 31 people's voice recordings were used in the study, comprising 23 Parkinson's disease (PD) patients (16 men

and 7 women), and 8 healthy controls (3 men and 5 women). The dataset includes 195 records, 24 columns, and several biological voice measures, as shown in Table 1. The “name” column of table below is separated into rows that represent individual vocal recordings and columns that represent each voice measurement. Each patient received an average of six recordings; nine individuals had seven recordings, while 22 patients received six. Patients ranged in age from 46 to 85 (mean 65.8, standard deviation 9.8), and between 0 and 28 years had passed since diagnosis. One voice recording lasting 36 seconds corresponds to each row. The voice was captured using a microphone that was calibrated by Little et al. and positioned 8 cm from the lips in a sound-treated booth provided by an industrial acoustic business (2009). The "status" column in the dataset is set to 0 for healthy subjects (HC) and 1 for those with Parkinson's disease (PD) to distinguish between those with and without the condition.

Name	ASCII subject name and recording number
MDVP: Fo (Hz)	Average vocal fundamental frequency
MDVP: Fhi (Hz)	Maximum vocal fundamental frequency
MDVP: Flo (Hz)	Minimum vocal fundamental frequency
MDVP: Jitter (%) MDVP: Jitter (Abs) MDVP: RAP MDVP: PPQ Jitter: DDP	Several measures of variation in fundamental frequency
MDVP: Shimmer MDVP: Shimmer(dB) Shimmer: APQ3 Shimmer: APQ5 MDVP: APQ Shimmer: DDA	Several measures of variation in amplitude
NHR, HNR	Two measures of ratio of noise to tonal components in the voice
status	Health status of the subject (one) – Parkinson’s, (zero) – healthy
RPDE D2	Two nonlinear dynamical complexity measures
DFA	Signal fractal scaling exponent
spread1, spread2, PPE	Three nonlinear measures of fundamental frequency variation.

Table 3.2 Features in the voice dataset

```
In [4]:
# loading the data from csv file to a Pandas DataFrame
parkinsons_data = pd.read_csv('Parkinson disease.csv')
```

```
In [13]:
# printing the first 10 rows of the dataframe
parkinsons_data.head(10)
pd.options.display.max_columns = 15
parkinsons_data.head(10)
```

Out[13]:

name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	...	status	RPDE	DFA	spread1	spread2
phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	...	1	0.414783	0.815285	-4.813031	0.266482
phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	...	1	0.458359	0.819521	-4.075192	0.335590
phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	...	1	0.429895	0.825288	-4.443179	0.311173
phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	...	1	0.434969	0.819235	-4.117501	0.334147
phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	...	1	0.417356	0.823484	-3.747787	0.234513
phon_R01_S01_6	120.552	131.162	113.787	0.00968	0.00008	0.00463	...	1	0.415564	0.825069	-4.242867	0.299111
phon_R01_S02_1	120.267	137.244	114.820	0.00333	0.00003	0.00155	...	1	0.596040	0.764112	-5.634322	0.257682
phon_R01_S02_2	107.332	113.840	104.315	0.00290	0.00003	0.00144	...	1	0.637420	0.763262	-6.167603	0.183721
phon_R01_S02_3	95.730	132.068	91.754	0.00551	0.00006	0.00293	...	1	0.615551	0.773587	-5.498678	0.327769
phon_R01_S02_4	95.056	120.103	91.226	0.00532	0.00006	0.00268	...	1	0.547037	0.798463	-5.011879	0.325996

rows × 24 columns

Figure 3.2 Sample Dataset

3.3 Data Pre-processing

Pre-processing the data is one of the very important stages in machine learning and other data related problems. Data pre-processing include a variety of steps like cleaning of data, transform the data and joining raw data into making it apt for analysis. Main of this process is to increase the data quality and data mining would be more suitable after this process. Efficiency and accuracy of ML model can be increased with data pre-processing.

At first, we will find the information about the data, what are types of value for each feature, what is the shape of the data etc…. In the next section we will find any null values are present in the data followed by calculating different statistical measures about the data. Further on, we will check the correlation between the features to identify any relationship between them. Then PCA is used to investigate the viability of feature ranking and dimensity reduction techniques for selecting key features in the dataset. In the next sections, we will explore these in detail.

- **Understanding the data**

  In this section, dataset is being explored. The shape of a dataset is a term used in data analysis to describe the dimensions of the dataset. In Python, Pandas and NumPy are

the libraries which helps to get the shape of the data. Shape refers to the rows and columns in the dataset. In Voice dataset, the shape is (195,24).

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

Figure 3.3.1 Algorithm for shape of data

The dataset has no missing values.

To find more information about the dataset regarding type of data and count of each feature we use “info ()” command in Python. In our dataset all the columns except “names” and “status” are having a type of float64 and there is no missing value in the dataset.

```
parkinsons_data = pd.read_csv('Parkinson disease.csv')
parkinsons_data.info()
```

Figure 3.3.2 Algorithm for information about data

- **Checking null value**

  Checking null value is a crucial step in data preprocessing. A Null value means that a value is missing for a particular feature in dataset. Why finding null values is important in machine learning is that it can make an impact of performance in the model. If the data has a null point, we may ignore the entire row or we make a central tendency measure to replace the null value on that column. In the voice dataset it is found that the null values are absent.

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

Figure 3.3.3 Algorithm for checking null values.

- **Statistical Measures**

  Statistical measures are quantities that capture various characteristics of a set of data. It contains numerous measures of dispersion, such as standard deviation and variance, as well as measurements of central tendencies, such as mean, median, and mode. It is very important to compute statistical measures because it provides a summary of data, gives

the idea of outliers if any present and is helpful for data normalization and handling missing values.

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

Figure 3.3.4 Algorithm for statistical measures

- **Correlations**

  Correlation is a statistical measure measure which computes how much two variables are related to each other. In voice dataset, variables are the features which are used to predict the target variable. Correlations helps to know the degreee of association between two features. Value of correlations varies from -1 to 1. Correlations are of 3 types. Out of which positive correlation have a value greater than zero and as one feature increases other feature also increases in this type of correlation. In negative correlations its value is reanging from zero to negative one and as one feature features increases other will decrease in this correlation. No correlation have a value of zero which implies that one feature's increase or decrease does not have the impact on the other.

  We can predit one variable from other if they are closely correlated. It helps to understand the data better. It is very key to identify correalations among features as they can impact a lot on ML models. We uses correlation matrix to plot the correlation of each features into one graph. From this garaph we can identify the highly correlated features which of are extreme colours in the heatmap of correaltion matrix and change the data accordingly so that we can reduce the damage caused by it.

```
# plotting imports
import seaborn as sns
import matplotlib.pyplot as plt
# figure size
plt.figure(figsize=(20, 20))

# correlation matrix
dataplot = sns.heatmap(parkinsons_data.corr(), annot=True, fmt='.2f', cmap="RdYlGn")
```

Figure 3.3.5 Algorithm for correlation matrix

- **Dimensity Reduction Tecniques**

  Intending to reduce the number of variables in a dataset while preserving as much crucial data as feasible, dimensionality reduction is a statistical approach.It entails

transferring higher-dimensional data into a low-dimensional setting while preserving the fundamental elements of the starting data.. This method is frequently applied in machine learning to solve classification and regression issues while producing a more accurate prediction model.The two basic methods for dimensionality reduction are as follows:

- Feature Extraction: This includes generating new features by merging or otherwise altering the original features.
- Feature Selection: This involves selecting a portion of the unique qualities that are most relevant to the current problem.

The following are some advantages of using dimensionality reduction on a particular dataset:

- lowering the amount of storage space needed for the dataset.
- Reduced feature dimensions result in shorter calculation training times.
- aiding in fast data visualisation.
- removing unnecessary features (if any) while taking multicollinearity into account.

Principal Component Analysis is one well-liked method for dimensionality reduction (PCA). The statistical technique known as PCA uses an orthogonal transformation to convert a set of correlated variables into a set of uncorrelated variables. It was first presented in 1901 by mathematician Karl Pearson. The fundamental goal of PCA, without any pre understanding of the target values, is to reduce the dimensionality of a dataset while keeping the strongest correlations or patterns between the variables.

Because a large attribute variance indicates a strong divide between classes and hence decreases dimensionality, PCA works by taking each attribute's variance into account. Principal Components are the transformed new features or output of PCA. These components are orthogonal, which means they are not connected. As the number of components increases, their relevance diminishes, with the first principal component having the highest importance and the nth principal component having the least.

PCA is useful in machine learning in several ways, including:

- Data compression: High-dimensional datasets may be made more manageable and easier to store by using PCA to decrease their dimensionality.

- PCA may be used to extract the most crucial characteristics from a dataset, which can enhance model performance.
- By keeping just the core components that account for the majority of the variance in the data, PCA can help in the elimination of noise and outliers.
- Visualization: High-dimensional data may be seen by reducing its dimensionality to two or three major components.

```
from sklearn.decomposition import PCA

# Create a PCA object
pca = PCA()

# Fit the PCA on the data
pca.fit(X)

# Get the explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_

# Print the percentage of variance explained by each principal component
for i, ratio in enumerate(explained_variance_ratio):
    print(f"Principal Component {i+1}: {ratio*100:.2f}%")
```

Figure 3.3.6 Algorithm for PCA

- **Feature Ranking**

  Feature Ranking is a technique in machine learning where features are examined and ranked according to their relevance. This is a critical step in many machine learning algorithms and is commonly conducted as a pre-processing step before training a model. Feature ranking can assist enhance the performance of your models by lowering the amount of features that need to be evaluated, which can in turn reduce processing time and memory needs.

  One popular technique to feature ranking is to employ a scoring function that allocates a score to each feature based on its relevance. The scores can be used to rank the features from most important to least important. Once you have ran
  ked the features, you can select a subset of them for inclusion in your machine learning model.

SelectKBest is one such technique used for feature ranking. It's a form of filter-based feature selection strategy in machine learning[7]. In filter-based feature selection approaches, the feature selection process is done independently of any specific machine learning algorithm. Instead, it depends on statistical measurements to evaluate and rank the features.

SelectKBest works by maintaining the first k aspects of X with the greatest ratings. The scoring function accepts two arrays, X and y, and outputs either a single array with scores or a set of arrays (scores, pvalues).The number of top features to select (k) can be chosen by the user.

In machine learning, SelectKBest can be advantageous in various ways:
- Reduction in Overfitting: SelectKBest can assist prevent overfitting, which happens when a model learns the detail and noise in the training data to the point that it adversely affects the model's performance on new data, by selecting just the most pertinent features.
- Improvement in Accuracy: SelectKBest can potentially improve model accuracy by deleting irrelevant characteristics from the dataset.
- Faster Training Time: SelectKBest can reduce the computational complexity of the model by restricting the amount of features, leading to shorter training times.

```
import pandas as pd
from sklearn.feature_selection import SelectKBest, mutual_info_classif

# Select all features with mutual information
selector = SelectKBest(mutual_info_classif, k='all')
selector.fit(X_df, Y)

# Get the scores and create a DataFrame with feature names and their corresponding scores
feature_scores = pd.DataFrame({'Feature': X_df.columns, 'Score': selector.scores_})

# Sort the features based on their scores in descending order
sorted_feature_scores = feature_scores.sort_values(by='Score', ascending=False)

print(sorted_feature_scores)
```

Figure 3.3.7 Algorithm for Feature Selection

3.4 Training Data

Making the training and test sets from the dataset:

Our dataset must be divided into a separate training set and test set for machine learning. One of the most important parts of knowledge preprocessing is typically this since it helps our machine learning model function better.

Imagine that we trained our machine learning model using one dataset, and then we tested it using a completely other dataset. The knowledge of the relationships between the models will thus be challenging for our model.

If we correctly train our model and it has very high training accuracy, but we also provide a different dataset there, the performance will suffer. To get good performance with both the training set and the test dataset, we always try to build a machine learning model.

```
X = parkinsons_data.drop(columns=['status','names'], axis=1)
Y = parkinsons_data['status']
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_pca, Y, test_size=0.2, random_state=0)
```

Figure 3.4 Algorithm for splitting dataset

3.5 Machine Learning Algorithms

A subset of artificial intelligence called "machine learning" is largely focused with the development of algorithms which enable a computer to learn from data and prior experiences on their own1. system learning allows a system to automatically learn from data, enhance performance from experiences, and anticipate things without being explicitly programmed.

Machine learning is an emerging technology which allows computers to learn autonomously from prior data. It employs numerous techniques for developing mathematical models and generating predictions using historical data or information. It is highly useful in situations like recognition of image, speech. Filtering of spams, detecting diseases etc...

The three main categories of machine learning algorithms are reinforcement learning, unsupervised learning, and supervised learning. Supervised Learning algorithms are taught utilising labelled data, i.e., input data coupled with accurate output. The algorithm learns a mapping from inputs to outputs and utilises it to predict the outcome for fresh inputs. Examples include Support Vector Machines, Decision Trees, and Linear Regression.

Unsupervised Learning techniques are employed when the information regarding the output is not known or not supplied. These algorithms find the underlying structure in the data, such grouping or clustering of data points. K-Means clustering and hierarchical clustering are two examples. With reinforcement learning, a computer learns how to behave in a given environment by taking actions and observing the rewards or results that arise from those actions.

We will examine six algorithms in particular in this study: K-nearest neighbour (KNN), support vector machine (SVM), random forest, XG boost, multiple layer perceptron, and deep neural network (DNN).We will discuss each algorithm in detail in the coming sections.

1. K-nearest neighbour (KNN)

A supervised learning approach known as K-Nearest Neighbors (KNN) is utilised for both classification and regression. It is a supervised learning classifier that is non-parametric and uses proximity to provide classifications or predictions about how each data point will be grouped. The K-Nearest Neighbors (KNN) algorithm is a generalization of the closest neighbor (NN) technique.

• In order for KNN to work, the data must be in a feature space. The data points are really in metric space, which is more precise. The data might be multidimensional vectors or simply scalars. The fact that the points are in feature space gives them a feeling of distance, which need not be Euclidean, though it is frequently utilised.

Each set of training data is made up of a group of vectors and a class label for each vector. The simplest case is when it is either + or (for positive or negative classes). KNN, however, can function just as well with any number of classes.

• In addition, we receive the single integer "k". This figure indicates how many neighbors—whose existence is defined by the distance between two points—have an effect on classification. If there are two courses, this is often an odd number. If k=1, then the approach is simply termed the nearest neighbour algorithm.

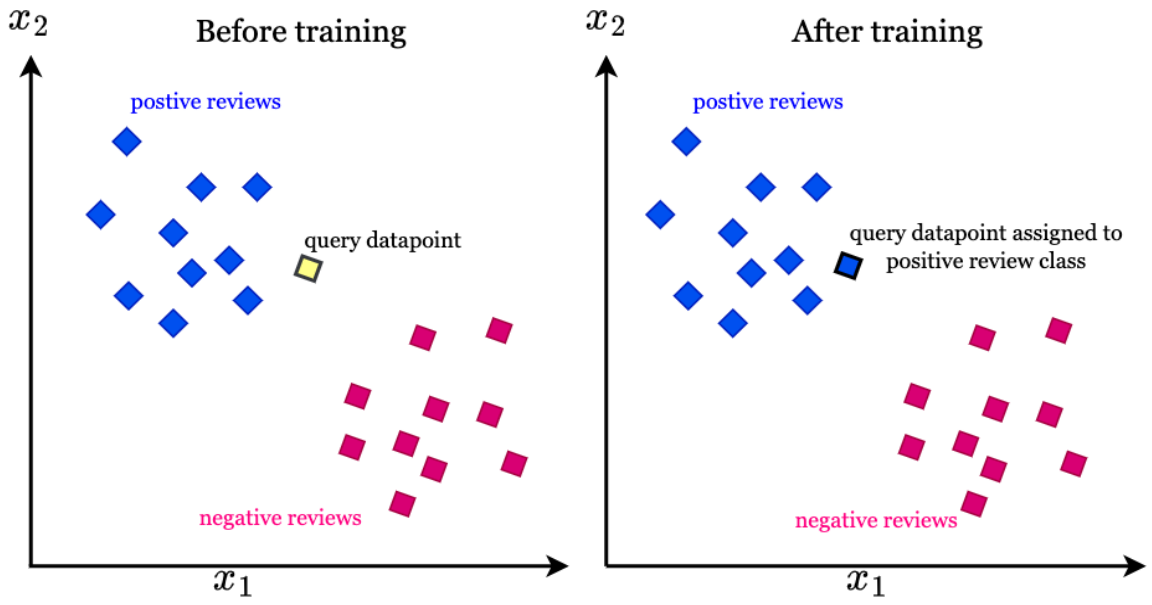

Figure 3.5.1.1 K-NN example

In the above figure, before training the new datapoint lies in between two reviews. After training new datapoint is added to the positive reviews class because the distance between the datapoint and chosen class is lesser than the other class.

In our algorithm, at first, we import “KneighborsClassifier” module from scikit learn python library. We then normalize the data by applying StandardScaler and MinMaxScaler for converting all the values into 0 to 1. We then apply various parameters to find the best.
Performance measures for the algorithm. It is done using the help of GridSearchCV which helps to find the optimal parameters. Data is split into 80:20 ratio. We compare the performance of this KNN classifier by applying two datasets, which are one with normal dataset and other with PCA dataset. For analysing the performance of the method, we employ a variety of performance metrics, such as accuracy, precision, f-score, and ROC-AUC values, and we compare them with both normal and PCA datasets.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Apply StandardScaler and MinMaxScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
scaler = MinMaxScaler()
X = scaler.fit_transform(X)


# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

# Train a KNN model by varying its parameters
parameters = {'n_neighbors':[1,3, 5, 7]}
knn = KNeighborsClassifier()
clf = GridSearchCV(knn, parameters)
clf.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = clf.predict(X_test)
y_pred_proba = clf.predict_proba(X_test)[:,1]

# Calculate the prediction scores
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Print the prediction scores
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
print(f'ROC AUC: {roc_auc:.2f}')
```

Figure 3.5.1.2 Algorithm for K-NN

2. Support Vector Machine (SVM) Classifier

A Support Vector Machine (SVM) is a classifier that uses a hyper-plane to distinguish between the various classes of input. SVM creates the hyper-plane in the test data after modelling it using the training data. The SVM model creates a hyper-plane in an effort to identify the region of the data matrix where various classes of data may be widely separated. The number of features determines the hyperplane's size. When there are just two input characteristics, the hyperplane is basically a line.

The hyperplane turns into a 2-D plane when there are three input characteristics. When there

are more than three features, it becomes challenging to imagine. SVM works when the data is small but complex. Also, this algorithm is resilient to outliers.

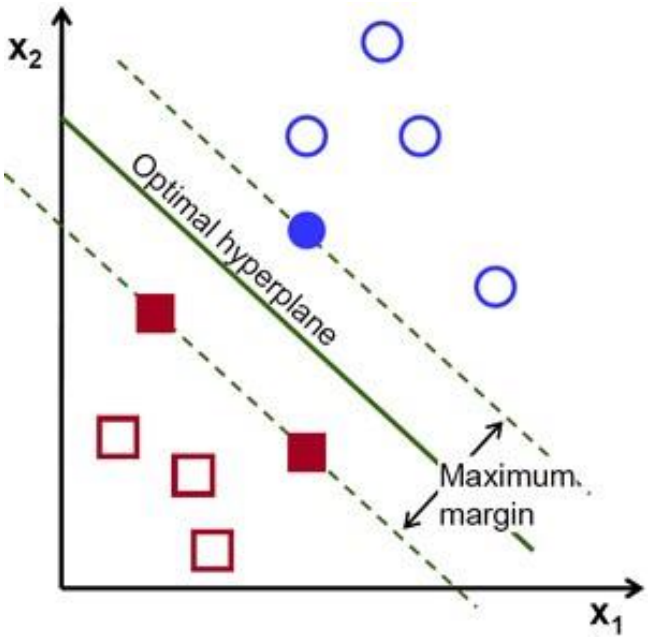

Figure 3.5.2.1 Support vector machine

In the above figure, the marked training data points fall into the red and blue categories. A hyper-plane can be created to categorise them linearly, but the question is: Which method is best given that there are other ways to do so? The margin between the classes is optimised by choosing the optimum hyper-plane. The hyper-plane need not be linear. In SVM, a hyper-plane may also act as a non-linear classifier by using a technique called the kernel-trick.

In our algorithm, at first, we import the “SVC” module from the scikit learn python library. We then normalize the data by applying StandardScaler and MinMaxScaler to convert all the values into 0 to 1. We then apply various parameters like kernels and C values to find the best performance measures for the algorithm. It is done using the help of GridSearchCV which helps to find the optimal parameters. Data is split into 80:20 ratio. We compare the performance of this SVM classifier by applying two datasets, which are one with a normal dataset and the other with a PCA dataset. For analysing the performance of the method, we employ a variety of performance metrics, such as accuracy, precision, f-score, and ROC-AUC values, and we compare them with both normal and PCA datasets.

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Apply StandardScaler and MinMaxScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

# Train an SVM model by varying its parameters
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
svc = SVC(probability=True)
clf = GridSearchCV(svc, parameters)
clf.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = clf.predict(X_test)
y_pred_proba = clf.predict_proba(X_test)[:,1]

# Calculate the prediction scores
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Print the prediction scores
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
print(f'ROC AUC: {roc_auc:.2f}')
```

Figure 3.5.2.2 Algorithm for SVM Classifier

3. Random Forest Classifier

A supervised learning method called a random forest classifier uses decision trees to generate predictions. It is a meta-estimator, meaning that it constructs a model by integrating many decision trees. The random forest method is notable for its excellent accuracy and its ability to handle huge datasets with multiple characteristics.

A random subset of the training data is initially built using the random forest approach. Then a decision tree is trained using this subset. The procedure is performed numerous times,

forming several decision trees. The forecasts from the several decision trees are combined to get the final forecast.
The random forest method can increase the accuracy of forecasts by lowering the variance of individual decision trees. This is so that each decision tree is less likely to be overfitted to the training set of data as each tree is trained on a different subset of the data.

The random forest method can also handle huge datasets with a variety of properties. This is because each decision tree only analyses a subset of the characteristics. This helps to lower the computational complexity of the method.

The random forest algorithm is a great tool for generating predictions. It is noted for its great accuracy and its ability to handle huge datasets with multiple characteristics. However, it is crucial to remember that the random forest approach may be computationally costly, and thus may not be ideal for all workloads.

The following are some advantages of using a random forest classifier:

- High accuracy:

Random forest classifiers are recognised for their excellent accuracy, particularly when dealing with huge datasets with multiple characteristics.

- Robustness:

Random forest classifiers are very resilient to overfitting, meaning that they can still produce good predictions even when the training data is not fully representative of the test data.

- Interpretability:

Random forest classifiers are reasonably straightforward to read, making them a suitable option for jobs where it is crucial to understand why a certain prediction was produced.

The following are some drawbacks of using a random forest classifier:

- Computational complexity:

Random forest classifiers may be computationally costly to train, particularly when dealing with huge datasets.

- Sensitivity to hyperparameters:

The selection of important parameters, such as the number of trees in the forest and the size of the random subsets used to train each tree, may affect how well random forest classifiers perform.

- Not always the greatest choice:

Random forest classifiers are not always the best solution for all jobs. For example, they may not be ideal for situations where speed is crucial or when the data is not well-represented by a decision tree model.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Apply StandardScaler and MinMaxScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

# Train a Random Forest Classifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = rfc.predict(X_test)
y_pred_proba = rfc.predict_proba(X_test)[:,1]

# Calculate the prediction scores
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Print the prediction scores
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
print(f'ROC AUC: {roc_auc:.2f}')
```

Figure 3.5.3 Algorithm for RF Classifier

In our algorithm, at first, we import the “RandomForestClassifier” module from the scikit learn python library. We then normalize the data by applying StandardScaler and MinMaxScaler to convert all the values into 0 to 1. Data is split into 80:20 ratio. We compare the performance of this Random Forest classifier by applying two datasets, which are one with a normal dataset and the other with a PCA dataset. For analysing the performance of the method, we employ a variety of performance metrics, such as accuracy, precision, f-score, and ROC-AUC values, and we compare them with both normal and PCA datasets.

4. Gradient Boost Classification

The Gradient Boosting Classifier is a form of ensemble machine learning technique that combines the predictions of numerous independent models to generate a final prediction. It belongs to the class of boosting algorithms, which repeatedly trains weak learners and combines them to form a strong learner.

In the context of classification, the Gradient Boosting Classifier works by building a chain of decision trees, where each successive tree attempts to fix the errors produced by the prior trees. It constructs the model in a forward stage-wise fashion, and each new tree fits on the residual errors (the difference between the actual labels and the predictions provided by the prior trees). We employ a Gradient Boosting Regressor when the target column is continuous, and a Gradient Boosting Classifier when the problem is one of classification. Between the two, the "Loss function" is the sole distinction. Gradient descent will be used to add weak learners in order to lower this loss function. Since it is based on a loss function, we will have unique loss functions for regression problems, such as Mean squared error (MSE), and for classification problems, such as log-likelihood. Here are some major aspects of the Gradient Boosting Classifier:

• Gradient Descent Optimization: The method minimizes a loss function by applying gradient descent optimization. In order to reduce the loss at each iteration, it modifies the model's parameters.

• Weak Learners: The individual models utilised in gradient boosting are often weak learners, such as decision trees with a limited depth or shallow trees. These weak learners are educated to be marginally better than random guessing.

• Combining Predictions: The predictions of many weak learners are merged via a weighted total to generate the final prediction. The weights given to each student depend on their success throughout training.

• Regularization: Gradient boosting methods commonly incorporate regularization techniques to avoid overfitting. Regularization aids in managing the complexity of the model and increases generalization to unknown data.

• Gradient Boosting Variants: There are different variants of gradient boosting, such as Gradient Boosting Machines (GBM), XG Boost, and Light GBM, which introduce additional enhancements to the basic gradient boosting framework, such as regularization techniques, parallel processing, and efficient tree construction algorithms.

The Gradient Boosting Classifier is notable for its ability to handle complicated datasets, capture non-linear correlations, and achieve high predicted accuracy. However, it may need careful adjustment of hyperparameters and might be computationally costly compared to simpler models. Here we utilise the XG Boost Classifier for classification.

In our algorithm, at first, we import the "XGBClassifier" module from the boost python library. We then normalize the data by applying StandardScaler and MinMaxScaler to convert all the values into 0 to 1. Data is split into 80:20 ratio. We compare the performance of this XGB classifier by applying two datasets, which are one with a normal dataset and the other with a PCA dataset. For analysing the performance of the method, we employ a variety of performance metrics, such as accuracy, precision, f-score, and ROC-AUC values, and we compare them with both normal and PCA datasets.

```
from xgboost import XGBClassifier
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Apply StandardScaler and MinMaxScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

# Train a boosted decision tree model
xgb = XGBClassifier()
xgb.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = xgb.predict(X_test)
y_pred_proba = xgb.predict_proba(X_test)[:,1]

# Calculate the prediction scores
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Print the prediction scores
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
print(f'ROC AUC: {roc_auc:.2f}')
```

Figure 3.5.4 Algorithm for XG Boost Classifier

5. Multi-layer Perceptron Classification

MLP is the abbreviation for multi-layer perception. Thick layers, which convert any input dimension to the needed dimension, are fully related to it. A neural network with several layers is called a multi-layer perception. We connect neurons such that their outputs become the inputs of other neurons to create a neural network.

A multi-layer perceptron contains one input layer with one neuron (or node) for each input, one output layer with one node for each output, as well as any number of hidden layers with any number of nodes each.

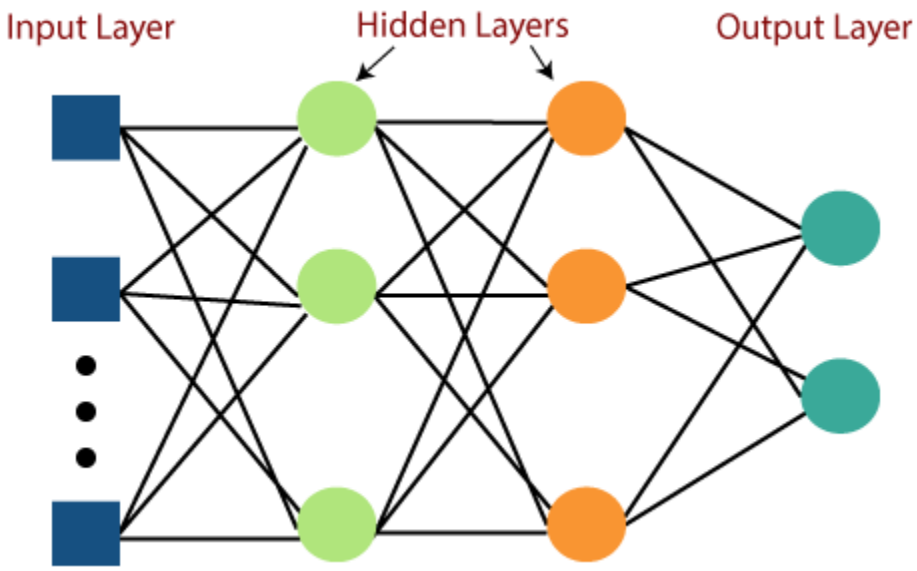

Figure 3.5.5.1 MLP

We can see that the multi-layer perceptron diagram above includes three inputs, which results in three input nodes, and three nodes for the hidden layer. There are two output nodes since the output layer offers two outputs. In the diagram above, the nodes in the input layer take data and forward it for further processing. In a similar way, the nodes in the hidden layer process the information before sending it to the output layer.

The multi-layer perception uses sigmoid activation functions at each node. The input real values are transformed into integers between 0 and 1 by the sigmoid activation function.

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Apply StandardScaler and MinMaxScaler
scaler = StandardScaler()
X_pca = scaler.fit_transform(X_pca)
scaler = MinMaxScaler()
X_pca = scaler.fit_transform(X_pca)


# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_pca, Y, test_size=0.2, random_state=0)

# Train an MLP model by varying its parameters
parameters = {'hidden_layer_sizes': [(50,), (100,), (50, 50)], 'activation': ['logistic', 'tanh', 'relu']}
mlp = MLPClassifier(max_iter=10000)
clf = GridSearchCV(mlp, parameters)
clf.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = clf.predict(X_test)
y_pred_proba = clf.predict_proba(X_test)[:,1]

# Calculate the prediction scores
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Print the prediction scores
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
print(f'ROC AUC: {roc_auc:.2f}')
```

Figure 3.5.5.2 Algorithm for MLP Classifier

In our algorithm, at first, we import the “MLPClassifier” module from the sci-kit learn python library. We then normalize the data by applying StandardScaler and MinMaxScaler to convert all the values into 0 to 1. Data is split into 80:20 ratio. We vary parameters like the number of hidden layers, and activation function with the help of GridSearchCV to find the optimum model. We compare the performance of this MLP classifier by applying two datasets, which are one with a normal dataset and the other with a PCA dataset. For analysing the performance of the method, we employ a variety of performance metrics, such as accuracy, precision, f-score, and ROC-AUC values, and we compare them with both normal and PCA datasets.

6. Deep Neural Network

An artificial neural network with several layers of nodes is known as a deep neural network (DNN). These layers are fully connected, meaning each node in one layer connects with a given weight to every node in the subsequent layer.
The depth of Deep Neural Networks is determined by the number of layers that input must pass through in a multi-stage pattern recognition process. DNNs are highly complex neural networks, typically consisting of more than two layers.

A DNN has three layers: an input layer that receives raw data, an output layer that generates final values, and a final layer. Between them are numerous hidden layers where the real processing is done via a series of weighted connections. The nodes in these layers execute non-linear changes to their inputs.

The fundamental advantage of DNNs is their capacity to do automatic feature extraction from raw data, commonly known as feature learning. This makes it possible to develop sophisticated systems that can extract complex patterns and representations from massive volumes of data. Still, Training deep neural networks (DNNs) requires significant computing power and abundant labelled data. Overfitting is a common issue, especially when training with insufficient data or inadequate regularization.

In our algorithm, at first, we import the ‘Sequential’ and ‘Dense’ modules from the Keras python library. We then normalize the data by applying StandardScaler and MinMaxScaler to convert all the values into 0 to 1. Data is split into 80:20 ratio. We compare the performance of this MLP classifier by applying two datasets, which are one with a normal dataset and the other with a PCA dataset. For analysing the performance of the method, we employ a variety of performance metrics, such as accuracy, precision, f-score, and ROC-AUC values, and we compare them with both normal and PCA datasets.

```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Apply StandardScaler and MinMaxScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

# Create a deep neural network model
model = Sequential()
model.add(Dense(32, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=10)

# Make predictions on the testing set
y_pred_proba = model.predict(X_test)
y_pred = (y_pred_proba > 0.5).astype(int)


# Calculate the prediction scores
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Print the prediction scores
print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
print(f'ROC AUC: {roc_auc:.2f}')
```

Figure 3.5.6 Algorithm for DNN Classifier

3.6 Testing Data

The pre-processed data were used to train the Parkinson's disease prediction model, it is evaluated using various data points. During this testing process, a test dataset is provided to the

model to check for accuracy and correctness. All training methods are reviewed to determine the best model to use. The model is used to forecast values for the test dataset after being fitted using the training data.. These projected values are then compared to the testing data to accurately calculate and compare model performance.

3.7 Performance Measures

Performance metrics are crucial in machine learning to assess a categorization model's efficacy. These measures help compare different models, monitor a model's progress over time, and identify areas for improvement. There are several performance measures available to use for classification purposes.

3.7.1 Accuracy

One of the most common performance measures for categorization is accuracy, This is obtained by dividing the total number of cases by the number of occurrences that were correctly categorised.

$$\text{Accuracy} = \frac{\text{Number of correctly classified instances}}{\text{Total instances}}$$

When using a certain metric to evaluate data with varying target variable classes, it is important to consider the balance of those classes. For instance, if a fruit image dataset contains 60% Apples and 40% Mangos, it would be appropriate to use the Accuracy metric to estimate whether an image is of an Apple or Mango, resulting in a 97% accuracy forecast. However, if the target variable predominantly belongs to one class, such as in a disease prediction model where only 5 out of 100 people have the disease, it is not recommended to use the Accuracy metric. In this scenario, if the model predicts every person as not having the disease, the resulting accuracy of 95% would be inaccurate.

3.7.2 Confusion Matrix

A table that displays the expected results of a binary classifier is known as a confusion matrix. When the real values are known, it is used to assess how well a classification model performs

on a set of test data. True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are the four categories that make up the table (FN). When both the expected result and the actual result are true, TP takes place. TN occurs when the outcome is both predicted and actual. When the outcome is false, but the prediction was correct, FP happens. FN happens when the outcome is true even though the prognosis was incorrect.

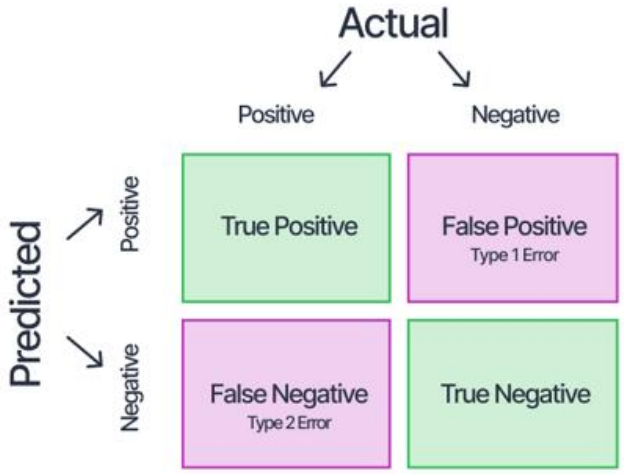

Figure 3.7.2 Confusion Matrix

3.7.3 Precision

The precision ratio is the proportion of accurately anticipated positive instances to all positively predicted cases.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

3.7.4 Recall

By dividing the total number of true positives by the sum of true positives and false negatives, recall is computed.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

The recall and precision of a classifier are two measures that determine how well it performs in terms of false negatives and false positives. A high recall value, close to 100%, limits false negatives, while a high precision value, also close to 100%, minimizes false positives. In plainer language, increasing accuracy lowers the likelihood of false positives, while increasing recall lowers the likelihood of false negatives.

3.7.5 F1-score

The F1 Score, also known as the F-score, is a statistical tool used to evaluate binary classification models based on predictions for the positive class. It is computed using Precision and Recall. By combining these two metrics, the F1 Score gives a single score that reflects their overall performance. To determine the F1 Score, the harmonic mean of Precision and Recall is applied., with equal importance given to each variable.

$$F1 - score = \frac{2 * precision * recall}{precision + recall}$$

The F1 Score is particularly useful when both Precision and Recall are essential for evaluation, but one of them is more important than the other. For example, if false positives are more critical than false negatives, or vice versa, the F1 Score should be used. As it considers both accuracy and recall, the F1 Score provides a comprehensive assessment of the model's performance.

3.8 t-distributed Stochastic Neighbor Embedding (t-SNE)

By assigning a location to each data point on a two- or three-dimensional map, the T-distributed stochastic neighbour embedding (t-SNE) statistical technique makes it possible to visualise high-dimensional data. This method is based on Stochastic Neighbor Embedding, which Sam Roweis and Geoffrey Hinton first developed. The t-distributed form was later suggested by Laurens van der Maaten. For visual purposes, t-SNE is a nonlinear dimensionality reduction technique that transforms high-dimensional data into a low-dimensional space with two or three dimensions. The t-SNE algorithm finds patterns in the data by examining the similarities between data points that share particular traits. The conditional chance that point A would

select point B as its neighbour is used to gauge how similar two points are. In order to create a complete representation of data points in lower-dimensional space, the technique then seeks to minimise the difference between these conditional probabilities (or similarities) in higher-dimensional and lower-dimensional space.

```
from sklearn.manifold import TSNE
import pandas as pd
import matplotlib.pyplot as plt



# Apply t-SNE to the data
tsne = TSNE(n_components=2, random_state=0)
X_tsne = tsne.fit_transform(X)

# Create a scatter plot of the t-SNE transformed data
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=Y)
plt.show()
```

Figure 3.8 Algorithm for t-SNE

CHAPTER 4 EXPERIMENTAL ANALYSIS AND RESULTS

4.1 Exploratory Analysis

1. **Comparing means of features between patients with Parkinson's disease and healthy persons**

```
In [80]:
1 # grouping the data based on the target variable
2 data.groupby('status').mean()
3
```

Out[80]:

status	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	RPDE	DFA	spread2	D2
0	181.937771	223.636750	145.207292	0.442552	0.695716	0.160292	2.154491
1	145.180762	188.441463	106.893558	0.516816	0.725408	0.248133	2.456058

Fig 4.1.1 Comparing Means

It is visible from the data that the mean of each feature is significantly different between PD patients and healthy persons.

2. **Checking the spread of data of features between PD patients and healthy persons**

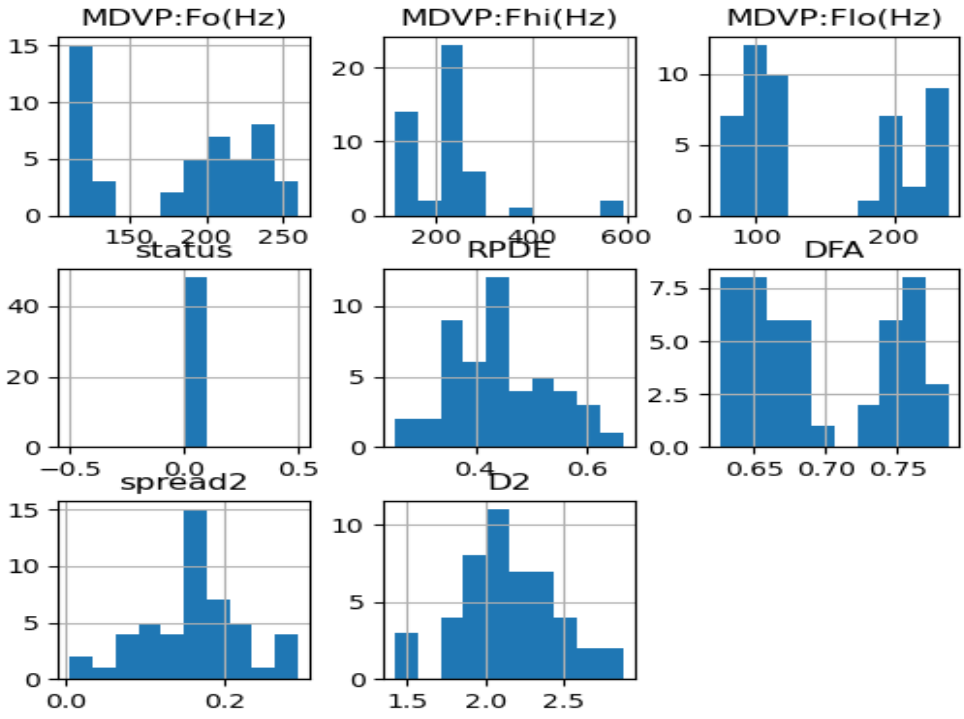

Fig 4.1.2(a) Histogram of the spread of data of normal people

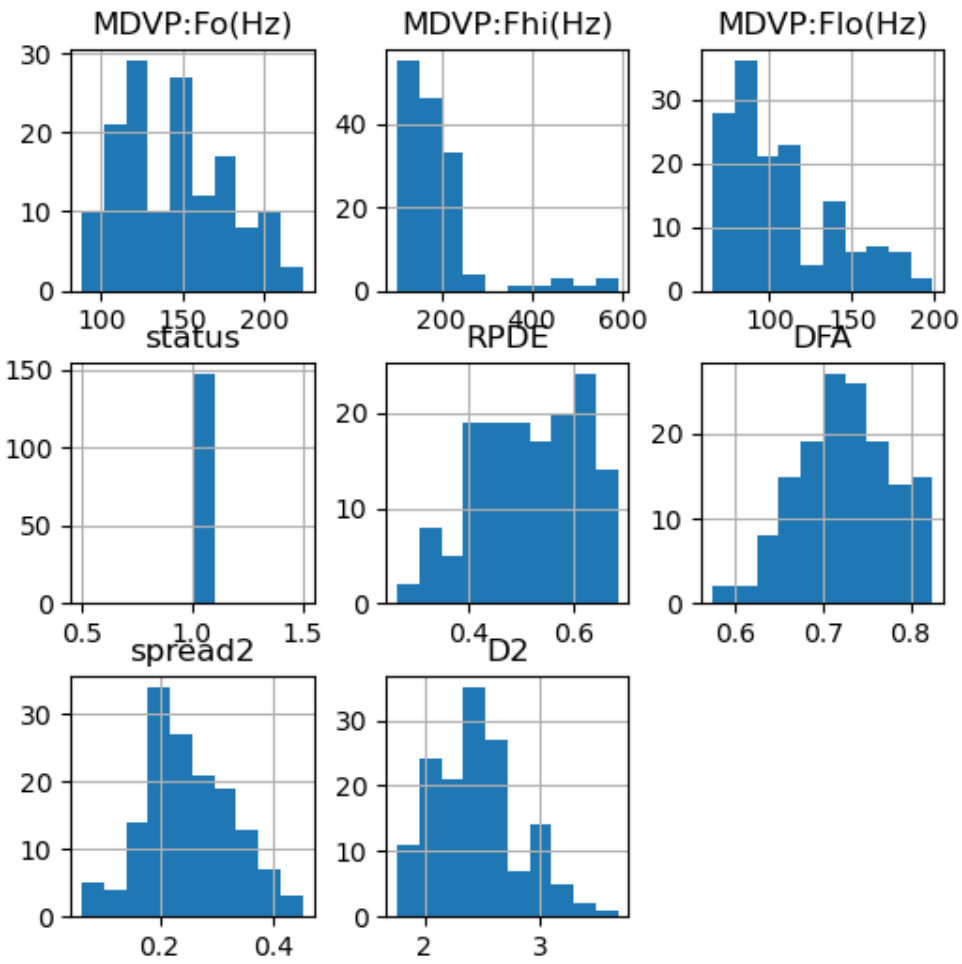

Fig 4.1.2(b) Histogram of the spread of data of PD patients.

There is a significant change in the spread of features between the two classes.

3. Correlation between features

According to the figure 4.1.3, it is identified that there are some features which are correlated themselves. This may hinder the performance of the model. Correlations between the features that are darker in colour are said to higher correlation between themselves.

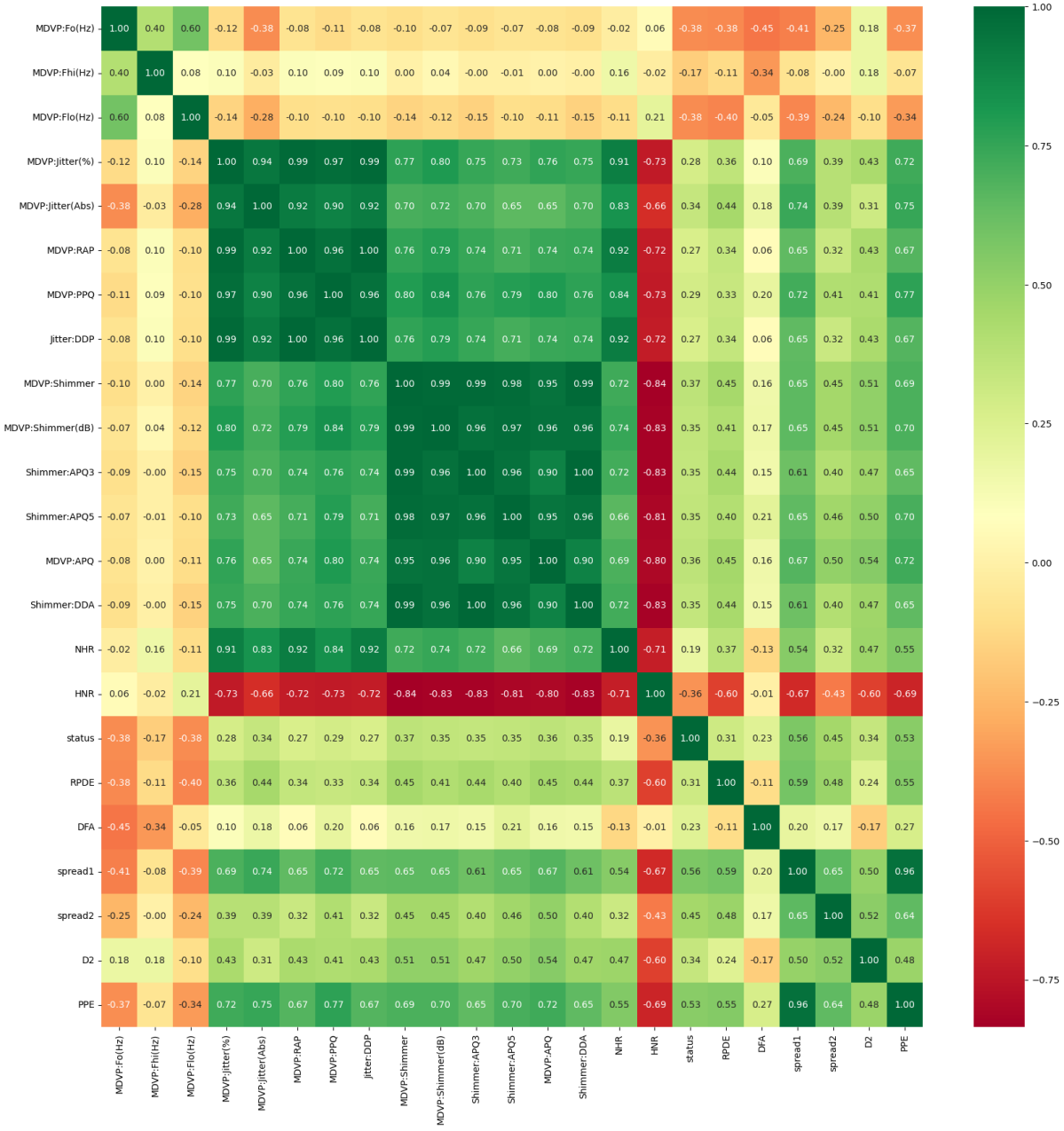

Figure 4.3.1 Correlation Matrix

4. Principal Component Analysis

While conducting PCA for dimension reduction, it is found that the first three principal components can explain the variability with a combined sum of 99.83%. It means that the first three principal components contain the most information, and we can remove other principal components without losing any significant information about the data. It also makes the data simpler and more efficient.

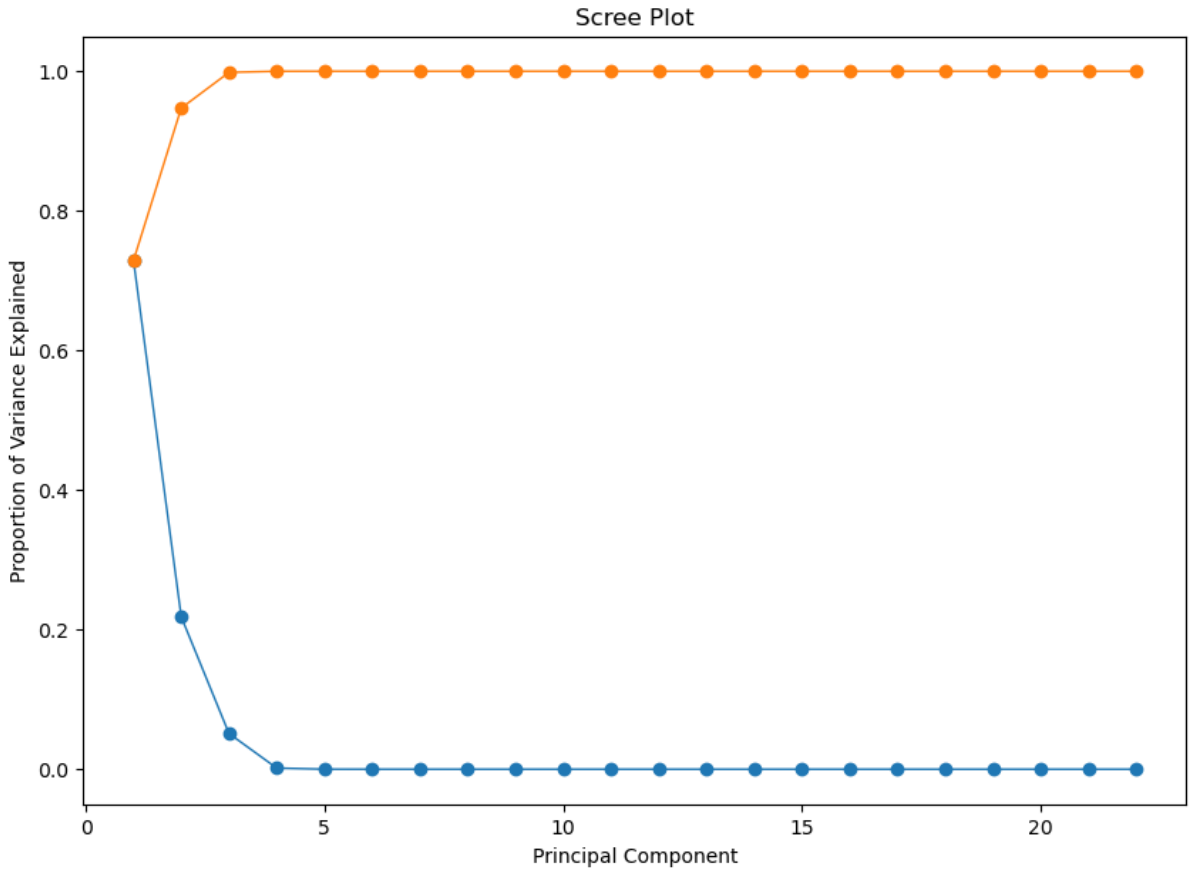

Figure 4.1.4 Scree Plot

5. Feature Ranking

SelectKBest is the technique used for feature ranking in this research. Feature ranking helps to identify the most important features in the dataset. This helps to make the dataset simpler and more efficient by only using the most important features. By removing redundant features, we can improve the performance of the models. The first five features in Figure 4.1.5 are the five most important features which have combined scores of more than 75 percentile.

	Feature	Score
21	PPE	0.253982
18	spread1	0.221368
0	MDVP:Fo(Hz)	0.206593
19	spread2	0.192002
12	MDVP:APQ	0.179986
2	MDVP:Flo(Hz)	0.174088
4	MDVP:Jitter(Abs)	0.166592
1	MDVP:Fhi(Hz)	0.140713
15	HNR	0.132462
14	NHR	0.120201
3	MDVP:Jitter(%)	0.112645
11	Shimmer:APQ5	0.111933
10	Shimmer:APQ3	0.108849
13	Shimmer:DDA	0.108726
8	MDVP:Shimmer	0.103000
9	MDVP:Shimmer(dB)	0.100967
6	MDVP:PPQ	0.097706
5	MDVP:RAP	0.097245
17	DFA	0.095116
7	Jitter:DDP	0.089369
16	RPDE	0.037003
20	D2	0.017913

Figure 4.1.5 Feature Ranking with Scores

6. t-distributed Stochastic Neighbor Embedding (t-SNE)

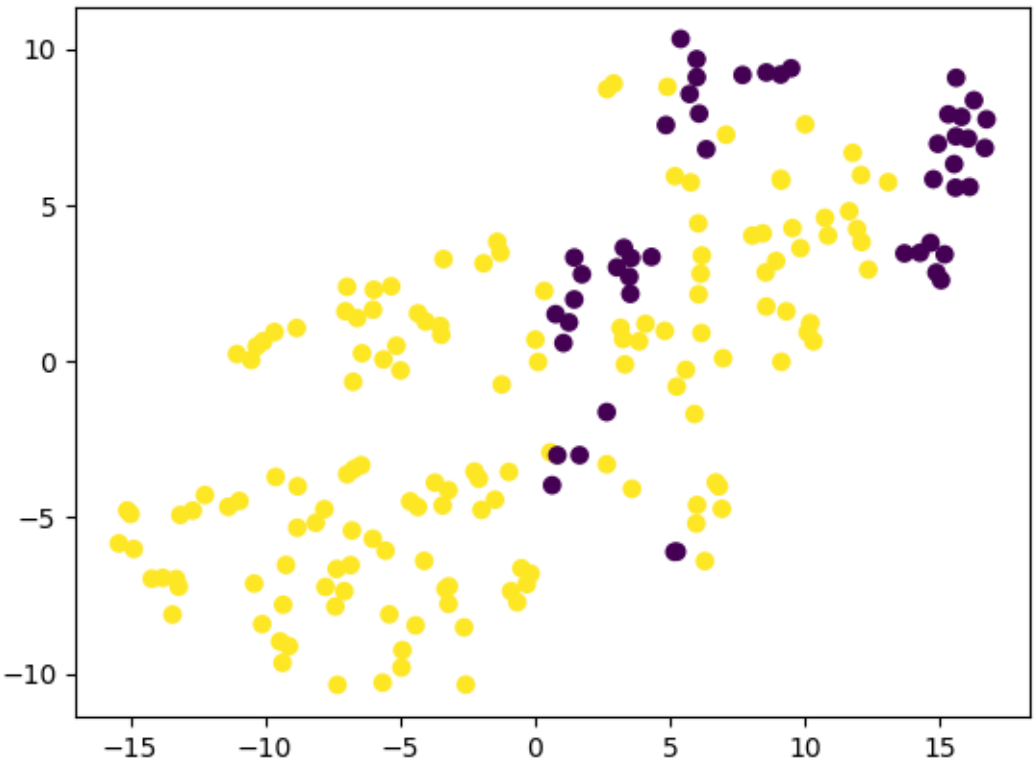

Figure 4.1.6 t-SNE plot

t-SNE is a dimension reduction technique that helps to plot the dataset into low dimensions. In figure 4.1.6, the yellow ones are the healthy people, and the purple ones are the PD patients. It is very helpful to identify where the clusters of each class are located and helps to know how many are there.

4.2 Performance Measures of Algorithms

1. Support Vector Machine (SVM) classification

The performance measures for the SVM classifier are given in the Table 4.2.1.

Support Vector Machine

Dataset	Accuracy	Precision	Recall	F-Score	ROC AUC
PCA	0.9	0.9	0.97	0.93	0.94
Normal	0.87	0.9	0.93	0.92	0.96

Table 4.2.1 Performance measures for SVM classifier.

From the above table, it is found that PCA applied datasets perform slightly better than original datasets. This leads to a conclusion that PCA can noise reduction, data redundancy is reduced, and it improved the classification of SVM. For our datasets, which consists of 147 healthy persons and 48 patients makes it an unbalanced class. For a dataset like Parkinson's disease prediction which is off a health-related issue, the most important performance measure is the recall. A high recall means that your model can accurately detect many true positives, which is crucial in medical settings to avoid false negatives and missing patients with the condition. Here PCA applied dataset achieves a recall .97 and original dataset have a recall with 0.93 which is both a good score. Although recall is important, precision is essential in medical diagnosis. A model with low precision can lead to unnecessary stress and medical procedures for patients who are incorrectly diagnosed with the disease. Here, both datasets can achieve a precision score of 0.90. Accuracy as well as f-score for PCA dataset is slightly larger than original dataset. The confusion matrix for both models is given below in the figure 4.2.1

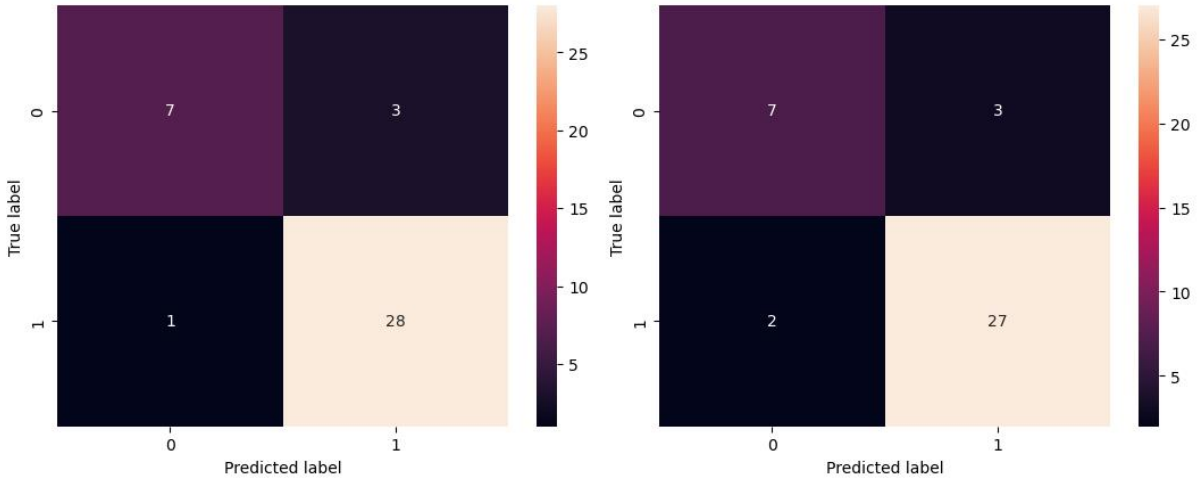

Figure 4.2.1 (a) Confusion Matrix for PCA dataset (b) Confusion Matrix for original dataset

2. K- Nearest Neighbor Classifier (K-NN)

The performance measures for the KNN classifier are given in the Table 4.2.2.

K-NN

Dataset	Accuracy	Precision	Recall	F-Score	ROC AUC
PCA	0.9	0.9	0.97	0.93	0.83
Normal	0.97	1	0.97	0.98	0.98

Table 4.2.2 Performance measures for KNN classifier

K-NN classifier is the one of the best models for this dataset. Unlike to SVM classifier, in KNN classifier original dataset performs better than the PCA applied dataset. It might be due to distance distortion since K-NN classifier is a distance-based technique and dimension reduction techniques have a chance to distort the distance between the class when performing it which leads to impact on performance. Also, PCA also is a reason for information loss while performing or if our datasets have a nonlinear relationship, dimension reduction using PCA might not be a good idea. Original datasets have an accuracy of 97% compared to 90% of PCA dataset. For precision and F-score, original have slight advantage over the other whereas the recall for both is same. Significant difference is observed in the ROC-AUC value, and this might be due to the reasons as mentioned above. The confusion matrix for both models is given below in the figure 4.2.2. As we can see, K-NN with normal dataset have zero false positive cases and only one false negative case for both datasets.

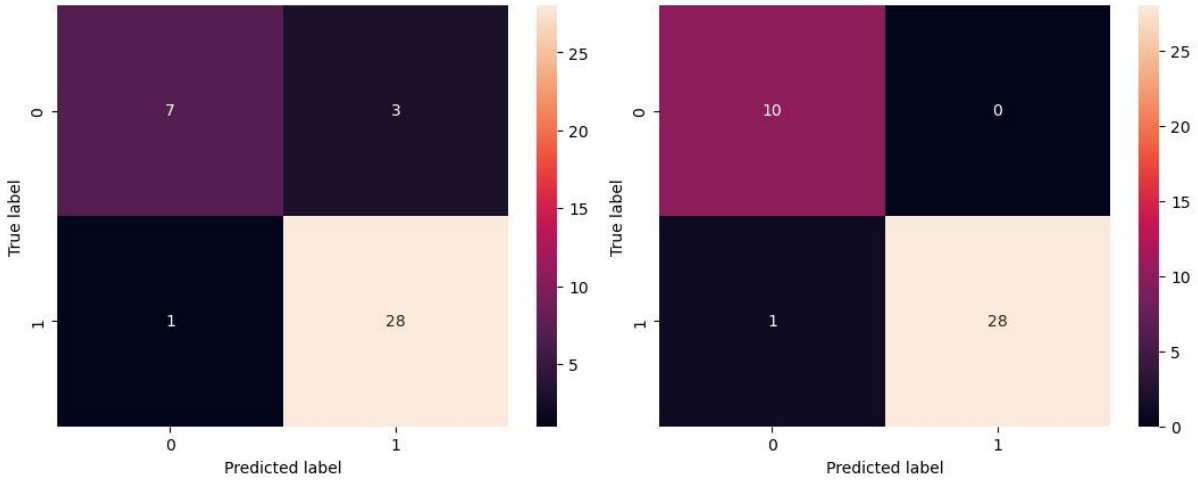

Figure 4.2.2 (a) Confusion Matrix for PCA dataset (b) Confusion Matrix for original dataset

3. Random Forest Classifier (RF Classifier)

The performance measures for the RF classifier are given in the Table 4.2.3.

Random Forest

Dataset	Accuracy	Precision	Recall	F-Score	ROC AUC
PCA	0.87	0.88	0.97	0.92	0.88
Normal	0.92	0.96	0.93	0.95	0.99

Table 4.2.3 Performance measures for RF classifier

As like K-NN, RF classifier is slightly performed better in original dataset than PCA applied dataset. This might be due to strength of RF which is ability to capture complex interaction and patterns between the variables and PCA might reduce or remove these complexities while performing or it might it be due to information loss from PCA. Original dataset achieves an accuracy of 92% compared to 87% of other. Precision, F-score, ROC AUC values are also higher for original dataset whereas Recall is higher for PCA dataset. The confusion matrix for both models is given below in the figure 4.2.3.

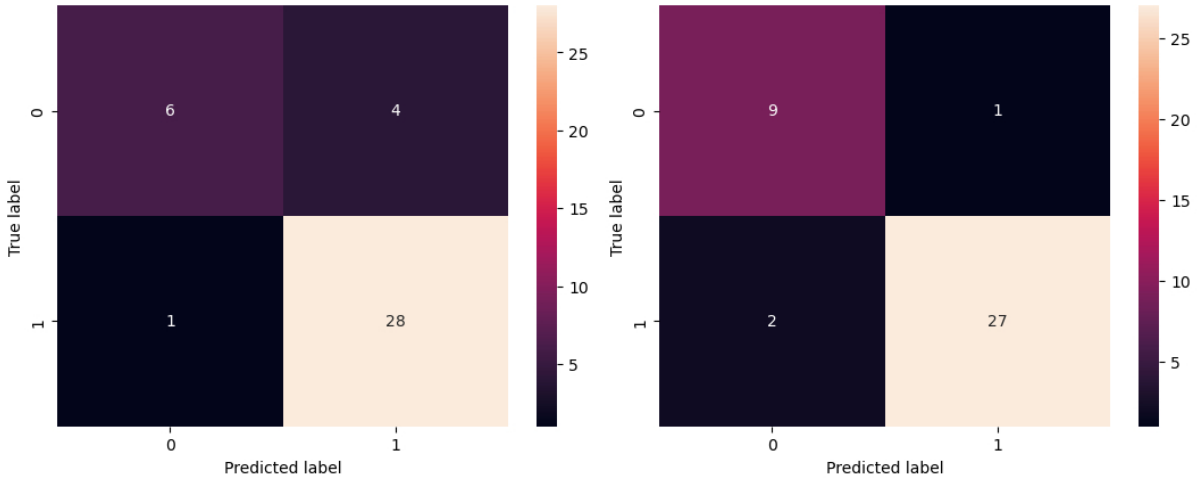

Figure 4.2.3 (a) Confusion Matrix for PCA dataset (b) Confusion Matrix for original dataset

4. XG Boost Classifier

The performance measures for the XG Boost classifier are given in the Table 4.2.4.

XG Boost

Dataset	Accuracy	Precision	Recall	F-Score	ROC AUC
PCA	0.79	0.86	0.86	0.86	0.9
Normal	0.95	1	0.93	0.96	0.99

Table 4.2.4 Performance measures for XG Boost classifier.

There is big margin of difference in performance measures between original dataset and PCA applied dataset. PCA applied dataset is the worst amongst the model whereas the model with original dataset is one among the best model. It is mainly due to the classifier itself since XG boost is a great classifier for handling complex models and higher dimension datasets while the PCA dilutes the complexities. Loss of information and nonlinear relationship among the variables are also probable cause of poor performance. PCA applied dataset is only able to achieve an accuracy of 79% and a score of 0.86 for precision, recall, F-score. On the contrary, original dataset applied model has been able to achieve an accuracy of 95% and with higher other performance measures. The confusion matrix for both models is given below in the figure 4.2.4.

Being one of the best models, original dataset applied model have zero false positive cases and only a single false negative case. Whereas it is four for both performance measures in PCA-applied dataset.

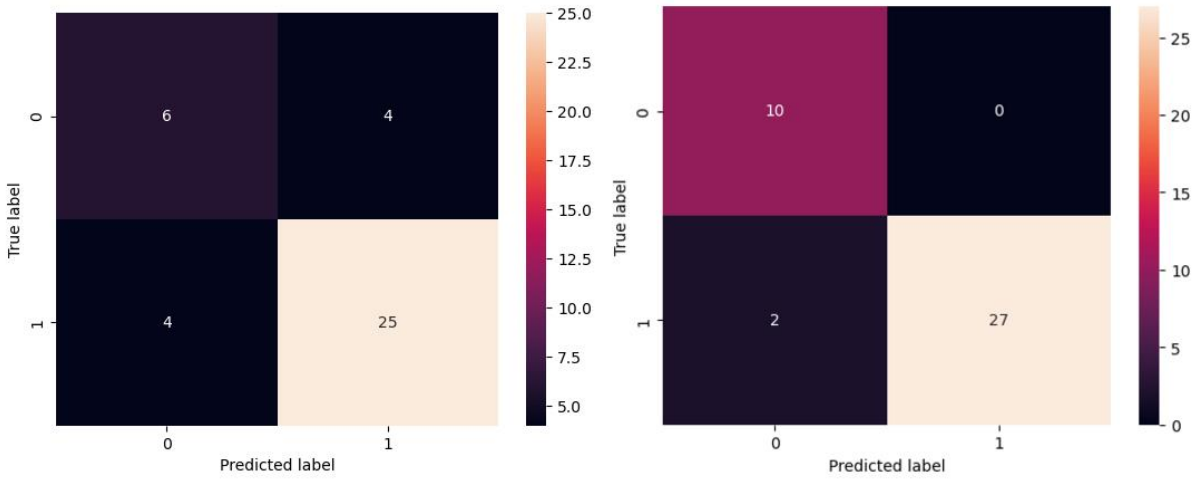

Figure 4.2.4 (a) Confusion Matrix for PCA dataset (b) Confusion Matrix for original dataset

5. Multiple Layer Perceptron Classifier (MLP Classifier)

The performance measures for the MLP classifier are given in the Table 4.2.5.

Multiple Layer Perceptron

Dataset	Accuracy	Precision	Recall	F-Score	ROC AUC
PCA	0.87	0.85	1	0.92	0.78
Normal	0.87	0.9	0.93	0.92	0.88

Table 4.2.5 Performance measures for MLP classifier.

Overall, MLP model with original dataset performs better than PCA applied dataset even though the recall for PCA applied dataset is 1. It is caused to due to the imbalance in the classes and higher recall means that MLP classifier predicting most instance as healthy persons. Loss of information and nonlinear relationships also contribute to the poor performance of PCA applied datasets. Both models have been able to achieve an accuracy of 87%. Precision is higher for original dataset (0.9) against PCA (0.85). ROC AUC value is also significantly less for PCA applied dataset. The confusion matrix for both models is given below in the figure 4.2.5.

As stated before, even though the false negative rate is zero the false positive rate is higher due to the lower precision rate in PCA applied dataset.

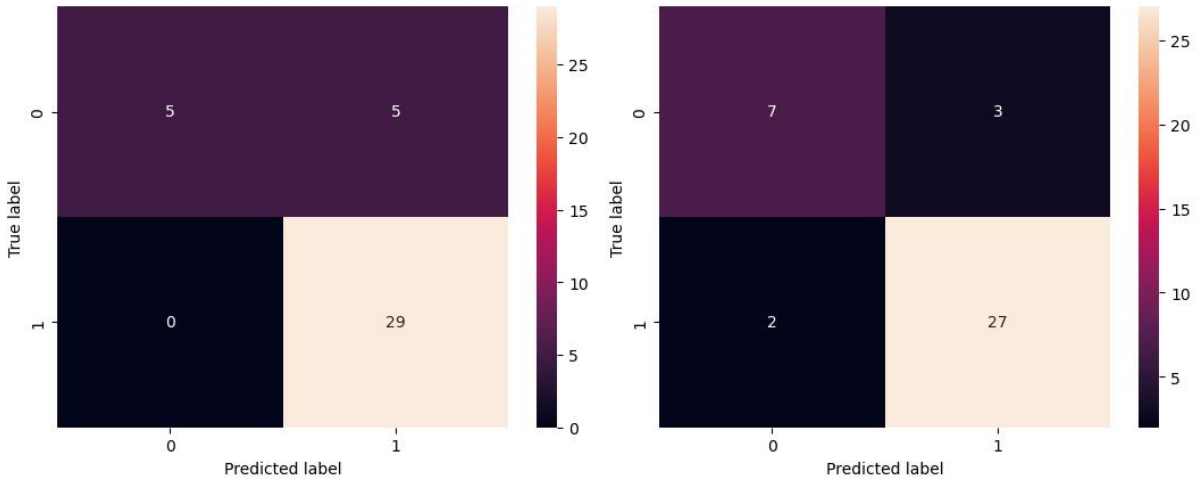

Figure 4.2.5 (a) Confusion Matrix for PCA dataset (b) Confusion Matrix for original dataset

6. Deep Neural Network Classifier (DNN Classifier)

The performance measures for the DNN classifier are given in the Table 4.2.6.

Deep Neural Network

Dataset	Accuracy	Precision	Recall	F-Score	ROC AUC
PCA	0.85	0.85	0.97	0.9	0.83
Normal	0.82	0.89	0.86	0.88	0.92

Table 4.2.6 Performance measures for DNN classifier.

In DNN classifier, it is evident that PCA applied dataset performs slightly better than original datasets. It is due to dimension reduction by PCA which makes network to train easier and PCA helps to reduce the noise in the dataset which will further help in training the network. Also, overfitting of the data is also compromised while using PCA. Recall for PCA applied dataset have been able achieve a score of 0.97 compared to 0.86 for original dataset. Regarding accuracy, the scores hover around 85% for PCA and 82% for original. ROC AUC value is slightly higher for original dataset than PCA dataset. The confusion matrix for both models is given below in the figure 4.2.6.

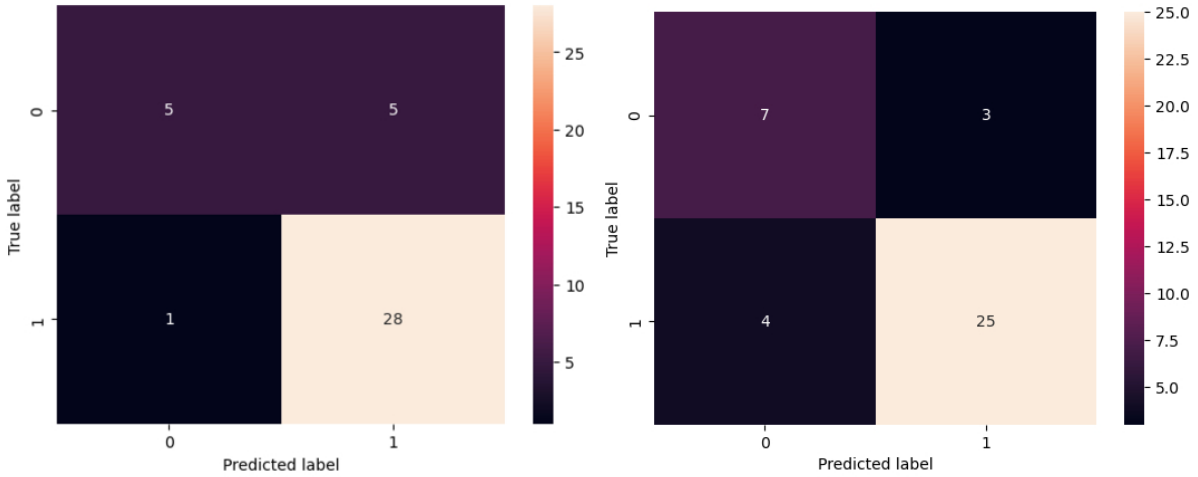

Figure 4.2.6 (a) Confusion Matrix for PCA dataset (b) Confusion Matrix for original dataset

4.3 Comparison of Algorithms

1. Comparison of Recall for Algorithms

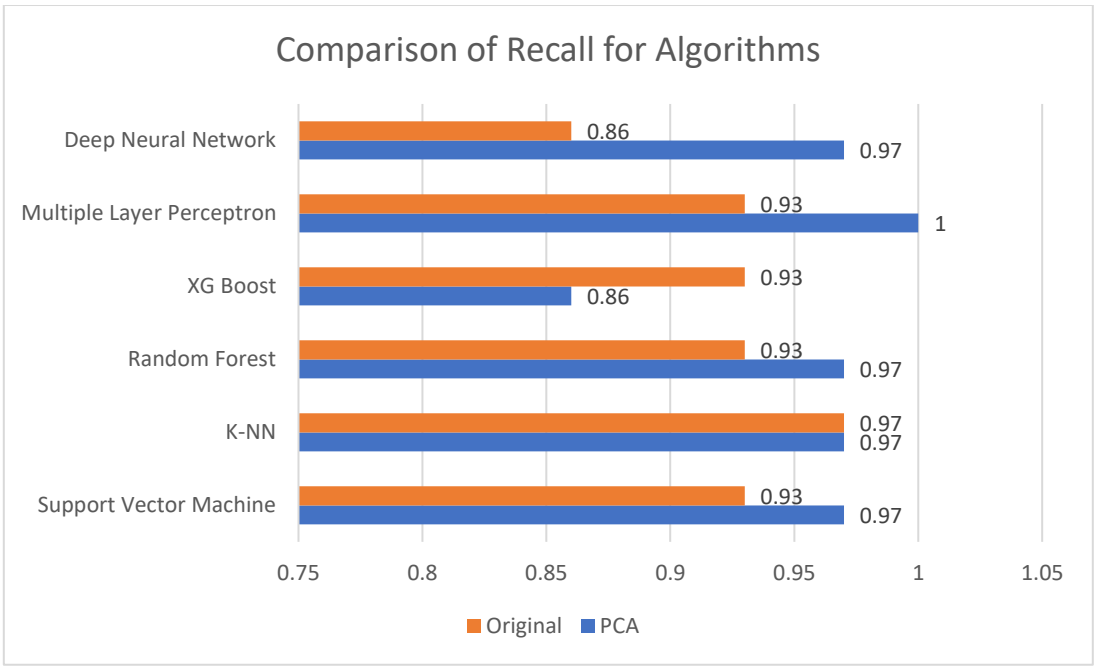

Figure 4.3.1 Comparison of Recall

Being the dataset are having imbalanced classes and the data is for Parkinson's disease prediction, recall is said to be an important performance measure for models. Out of these models MLP classifier came out with highest score of 1 with PCA applied dataset. Even though MLP classifier achieved the best recall score, we cannot say it is the best, since we have also

to acknowledge other performance measures. Followed by MLP, both K-NN classifier and PCA applied DNN and SVM classifier shares a recall score oof 0.97. PCA applied XG Boost classifier came out to be worst among recall scores with a scores with 0.86 along DNN on original dataset.

2. Comparison of accuracy by algorithms

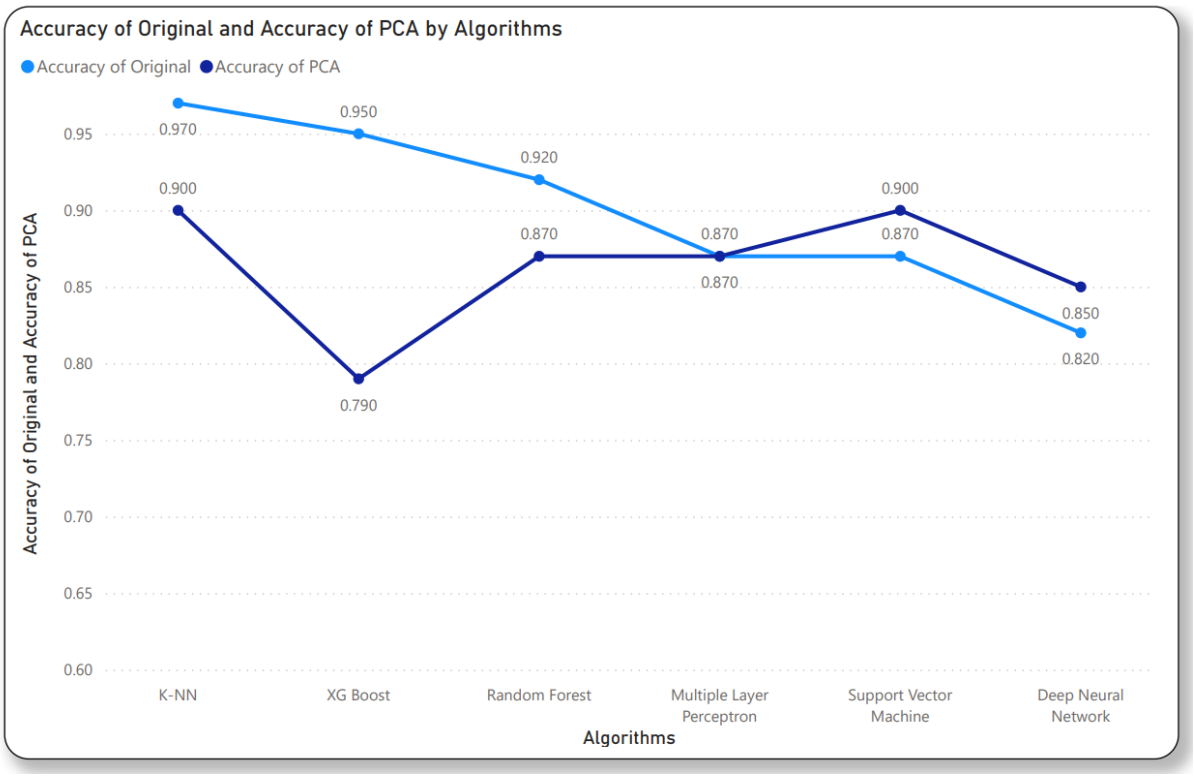

Figure 4.3.2 Comparison of Accuracy by Algorithms

K-NN classifier on original dataset came out to be the best classifier with 97% accuracy. Followed by XG Boost classifier with 95% accuracy. In SVM and DNN classifiers PCA applied datasets came out to be the better classifier than the other. MLP classifier has achieved an accuracy of 87% for both of its dataset.

3. Comparison of Other Performance Measures

Performance measures such as precision, F-score, ROC AUC scores are compared in the figure 4.3.3.

	Precision		F-Score		ROC AUC	
Algorithm	PCA	Original	PCA	Original	PCA	Original
Support Vector Machine	0.9	0.9	0.93	0.92	0.94	0.96
K-NN	0.9	1	0.93	0.98	0.83	0.98
Random Forest	0.88	0.96	0.92	0.95	0.88	0.99
XG Boost	0.86	1	0.86	0.96	0.9	0.99
Multiple Layer Perceptron	0.85	0.9	0.92	0.92	0.78	0.88
Deep Neural Network	0.85	0.89	0.9	0.88	0.83	0.92

Figure 4.3.2 Comparison of other performance measures

Overall, precision scores for original dataset performs better than PCA applied datasets. K-NN and XG Boost Classifiers has been able to achieve a precision score of 1. This means that there will be no false positive cases for both the classifiers. Being the harmonic mean of performance measures recall and precision, F-score proves to be a valuable performance measure and out these scores K-NN came out to be the best classifier with score 0.98 and MLP classifier came out to be worst with 0.78 score. Regarding ROC AUC value K-NN, RF, XG Boost have achieved higher score greater than 0.98.

CHAPTER 5 CONCLUSION

Parkinson's disease is one of the most serious neurodegenerative conditions. The worst aspect of this illness is that there hasn't been a remedy discovered yet. Only solution for this disease is to diagnose the disease as soon as possible so that we can improve the symptoms. So early detection of this disease is much needed. Making things worser, there is no medical or blood tests are available to diagnose this condition. We are currently relying only on the expertise of the doctors. So, machine learning based disease prediction is much needed in this hour.

In this study, we investigated six algorithms—K-nearest neighbour (KNN), support vector machine (SVM), random forest, XG boost, multiple layer perceptron (MLP), and deep neural network—for the prediction of Parkinson's disease (DNN). Our aim is to train the dataset using these algorithms and compare them so that to find the best model. With support from the National Centre for Voice, which was established by Little et al., our dataset was hosted at the UCI Machine Learning Repository and was acquired from the University of Oxford (UO) repository (2007, 2009). We have an instance of 195 with 31 people. We have compared various techniques to find the optimal performance such as dimension reduction techniques, feature ranking techniques etc… We have utilised a range of performance metrics, including the confusion matrix, F1-score, ROC AUC score, and accuracy, precision, and recall.

From the results, K-NN classifier with original dataset came out to be the outstanding classifier with performing well in every performance measures. It records an accuracy of 97% as well as recall score of 0.97. It is then followed by XG Boost algorithm also with the original dataset having accuracy of 95%. Most of the models have achieved a great score and some of them even crosses 95% of performance measures. Therefore, we can say that these models can be used directly to predict the Parkinson’s disease.

Even though we have achieved a great performance in the model, there is always a room for improvement. A new dataset with much more instances could shed the light to better understanding of the disease. Also, we have faced some challenges while performing due to imbalanced classes. Techniques like Synthetic Minority Over-sampling Technique (SMOTE) might be able to overcome the issues.

References

1. Rajput, A., & Rajput, A. H. (2007). Old age and Parkinson's disease. In Handbook of clinical neurology (Vol. 84, pp. 427-444). Elsevier.
2. Von Campenhausen, S., Bornschein, B., Wick, R., Bötzel, K., Sampaio, C., Poewe, W., ... & Dodel, R. (2005). Prevalence and incidence of Parkinson's disease in Europe. *European neuropsychopharmacology*, *15*(4), 473-490.
3. De Rijk, M. D., Launer, L. J., Berger, K., Breteler, M. M., Dartigues, J. F., Baldereschi, M., ... & Hofman, A. (2000). Prevalence of Parkinson's disease in Europe: A collaborative study of population-based cohorts. Neurologic Diseases in the Elderly Research Group. *Neurology*, *54*(11 Suppl 5), S21-3.
4. Haaxma, C. A., Bloem, B. R., Borm, G. F., Oyen, W. J., Leenders, K. L., Eshuis, S., ... & Horstink, M. W. (2007). Gender differences in Parkinson's disease. *Journal of Neurology, Neurosurgery & Psychiatry*, *78*(8), 819-824.
5. Alshammri, R., Alharbi, G., Alharbi, E., & Almubark, I. (2023). Machine learning approaches to identify Parkinson's disease using voice signal features. *Frontiers in Artificial Intelligence*, *6*, 1084001.
6. Hariharan, M., Polat, K., & Sindhu, R. (2014). A new hybrid intelligent system for accurate detection of Parkinson's disease. *Computer methods and programs in biomedicine*, *113*(3), 904-913.
7. Haq, A. U., Li, J. P., Memon, M. H., Malik, A., Ahmad, T., Ali, A., ... & Shahid, M. (2019). Feature selection based on L1-norm support vector machine and effective recognition system for Parkinson's disease using voice recordings. *IEEE access*, *7*, 37718-37734.
8. Haq, A. U., Li, J., Memon, M. H., Khan, J., Din, S. U., Ahad, I., ... & Lai, Z. (2018, December). Comparative analysis of the classification performance of machine learning classifiers and deep neural network classifier for prediction of Parkinson disease. In *2018 15th international computer conference on wavelet active media technology and information processing (ICCWAMTIP)* (pp. 101-106). IEEE.
9. Dinesh, A., & He, J. (2017, November). Using machine learning to diagnose Parkinson's disease from voice recordings. In *2017 IEEE MIT Undergraduate Research Technology Conference (URTC)* (pp. 1-4). IEEE.

10. Dash, S., Thulasiram, R., & Thulasiraman, P. (2017, December). An enhanced chaos-based firefly model for Parkinson's disease diagnosis and classification. In *2017 International Conference on Information Technology (ICIT)* (pp. 159-164). IEEE.
11. Bakar, Z. A., Ispawi, D. I., Ibrahim, N. F., & Tahir, N. M. (2012, March). Classification of Parkinson's disease based on Multilayer Perceptrons (MLPs) Neural Network and ANOVA as a feature extraction. In *2012 IEEE 8th International Colloquium on Signal Processing and its Applications* (pp. 63-67). IEEE.
12. Anand, A., Haque, M. A., Alex, J. S. R., & Venkatesan, N. (2018, December). Evaluation of Machine learning and Deep learning algorithms combined with dimentionality reduction techniques for classification of Parkinson's Disease. In *2018 IEEE international symposium on signal processing and information technology (ISSPIT)* (pp. 342-347). IEEE.
13. Al Sayaydeha, O. N., & Mohammad, M. F. (2019, April). Diagnosis of the Parkinson disease using enhanced fuzzy min-max neural network and OneR attribute evaluation method. In *2019 International Conference on Advanced Science and Engineering (ICOASE)* (pp. 64-69). IEEE.
14. Al-Fatlawi, A. H., Jabardi, M. H., & Ling, S. H. (2016, July). Efficient diagnosis system for Parkinson's disease using deep belief network. In *2016 IEEE Congress on evolutionary computation (CEC)* (pp. 1324-1330). IEEE.
15. Abiyev, R. H., & Abizade, S. (2016). Diagnosing Parkinson's diseases using fuzzy neural system. *Computational and mathematical methods in medicine, 2016*.