Parkinson's Disease Detection using Machine Learning

A

Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

K. Greta

1602-18-737-069

Under the guidance of

Mr. K. Srinivasa Chakravarthy

Assistant Professor

Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2022

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

I, **K. Greta** bearing hall ticket number, **1602-18-737-069**, hereby declare that the project report entitled **Parkinson's Disease Detection using Machine Learning** under the guidance of **Mr. K. Srinivasa Chakravarthy,** Assistant Professor, Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

Date:

K. Greta
1602-18-737-069

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

BONAFIDE CERTIFICATE

This is to certify that the project entitled **Parkinson’s Disease Detection using Machine Learning** being submitted by **K. Greta** bearing **1602-18-737-069** in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by her under my guidance.

Mr. K. Srinivasa Chakravarthy	**Dr. K. Ram Mohan Rao**
Assistant Professor	**Professor and Head**
Internal Guide	**Department of Information Technology**

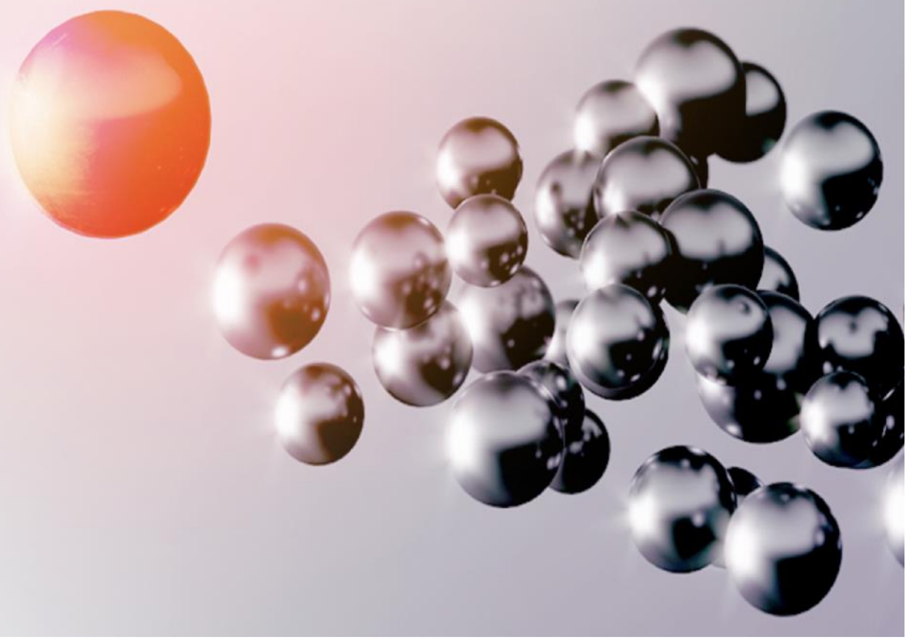
External Examiner

ACCOLITE DIGITAL

Transforming The Future, Now

Offer of Employment

Great Place To Work® Certified SEP 2021–SEP 2022 INDIA™



ABOUT US

Accolite Digital is an innovative, best-in-class digital transformation services provider, successfully delivering design driven complex digital transformation initiatives to leading Fortune 500 clients. We do so by through our differentiated offerings, best-in-class delivery, deep and long-standing customer relationships, and top-tier engineering talent comprising a team of bright and passionate technologists.

Our differentiated offering spans digital product engineering, cloud and DevOps, data and AI, customer experience, cyber security, and design services. We provide these services across banking and financial services, insurance, technology media and telecom, healthcare, and logistics industries.

Founded in 2007, today with a headcount of over 2500, Accolite has a presence across the United States, Canada, and Europe and has digital labs in Bangalore, Hyderabad, Gurugram, and Chennai.

COMPANY OVERVIEW

Founded	2007
Headquarters	Addison, Texas, USA
Digital Labs	Bangalore |Hyderabad | Gurgaon | Mumbai | Chennai
Other Locations	London |New York | San Francisco | Tampa | Nashville
Strategic Investor	New Mountain Capital, $ 33B+ AUM Private Equity

DIGITAL SERVICES

- Digital Product Engineering
- Cloud Engineering & DevOps
- CX & Design
- Cyber Security

KEY VERTICALS

- Banking Financial Services, Insurance
- Technology, Media & Telecom
- Healthcare
- Logistics

Dear **Greta Koutika,**

Congratulations on your selection at Accolite Digital, a best-in-class digital transformation services provider. I am delighted to extend this offer of employment to you.

At Accolite Digital, our mission is to solve our client's most complex digital challenges by engaging the brightest of technical minds, such as you. We believe in creating a work environment that enables our people to pursue their careers and balance their personal and professional goals so that we can achieve extraordinary results by winning together.

As a workplace, our culture is about openness, inclusion, and the willingness to take on the toughest challenges while doing the right thing every time. Being certified as a 'Great Place to Work' recognizes our robust value systems and reinforces our focus on creating a happy, healthy, and safe workplace.

With a team comprising the best & brightest technical minds, we believe in having a proactive and client-centric approach to deliver positive business outcomes.

We count on your abilities and believe that you will be a valuable addition to our team. I look forward to having you onboard soon and together achieve our vision to scale to a $1B+ organization.

Wishing you a long and rewarding career at Accolite!

Best regards,
Leela Kaza
Founder & CEO

CONFIDENTIAL

8th December 2021

Greta Koutika

Subject: Offer of Internship

Dear **Greta Koutika,**

Based on our recent discussions, we are pleased to offer you an Internship with Accolite Digital India Pvt. Ltd. The internship is a significant experience in the course of your developing into a qualified professional. Therefore, we do hope you will use this opportunity to add value mutually to and from the organization.

The details of your internship extension with us are as follows: -

1. Date of Joining: **14th February 2022**
2. Internship Duration: **14th February 2022 – 12th August 2022**
3. Location: **Hyderabad**
4. Stipend: **INR 20,000 per month**

Probation:

You shall initially be under probation for a 30-day period from the date of joining our service. The Company reserves the right to terminate your internship at any time during your probation. You will be required to give 15 days' notice in writing to Accolite in case you wish to resign / leave the service.

Separation at the instance of the employee:

1) You agree that if you resign from Accolite before completion of your internship program or decline a full-time offer from
Accolite, in the event that Accolite extends the same to you; or resign from full-time position within one (1) year of joining
Accolite thereafter, you are liable to pay as below:

a) You agree that you shall return the stipend amount owed to you during the internship period duration.

b) The expenditure incurred by Accolite on account of your Accolite University program, training, and development, capped at **INR 1,50,000**.

Signature: K. Greta

ACKNOWLEDGEMENT

This is an acknowledgement of the intensive drive and the technical competence of many individuals who have contributed to the success of the project. The satisfaction that accompanies the successful completion of the project would not have been possible without the kind support and help of many individuals. I would like to extend our sincere thanks to all of them.

I am obliged and grateful to Dr. S.V. RAMANA, Principal of Vasavi College of Engineering and Dr. K. RAM MOHAN RAO, Professor and Head of the Department of Information Technology, Vasavi College of Engineering, Hyderabad for their valuable suggestions and support in all respects during the course of B.E.

I would like to thank Mr. K. SRINIVASA CHAKRAVARTHY, Assistant Professor for his guidance and constructive suggestions throughout the project. His guidance and encouragement have helped me immensely in the preparation of this project. His constant support made me understand this project and its manifestations in great depth and helped me to complete the assigned tasks.

I am also grateful to Dr. S.K. CHAYA DEVI, Associate Professor for her guidance and consistent encouragement. I would like to express my sincere thanks to all faculty members and staff of the Department of Information Technology for their generous help in various ways for the completion of this project.

ABSTRACT

Parkinson Disease is a brain neurological disorder. It leads to shaking of the body, hands and provides stiffness to the body. No proper cure or treatment is available yet at the advanced stage. Treatment is possible only when done at the early or onset of the disease. These will not only reduce the cost of the disease but will also possibly save a life.

Most Parkinson Disease (PD) patients suffer from vocal cord disorders. Speech impairment is an early indicator of PD. This project focuses on developing of Parkinson's disease detection system using auditorial features such as various speech signal processing algorithms including Time Frequency Features, Mel Frequency Cepstral Coefficients (MFCCs), Wavelet Transform based Features, Vocal Fold Features and TWQT features have been applied to the speech recordings of Parkinson's Disease (PD) patients to extract clinically useful information for PD assessment, several measures of variation in amplitude.

These extracted features can be modelled by various machine learning models like Classification Trees, Support Vector Machine, KNN, Logistic Regression, Linear Discriminant analysis, Random Forest classifier, Ada Boost and a few deep learning models i.e., Feed forward neural networks with two hidden layers and an ensemble of the built deep learning models. New data will be given to this model then it should be able to detect whether a person has a Parkinson's or not. The training set data extracted from the voice recordings of 188 individuals which are classified as PD and 64 healthy controls. The proposed deep learning models show the highest accuracy of 90%.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Overview:

Parkinson’s Disease causes several motor and non-motor symptoms, which gradually become prominent after different disease progression stages. One of the motor symptoms that people with PD may experience is the change in their speech quality or difficulty speaking in worse cases. However, everyone with PD does not experience the same symptoms. Similarly, all the patients do not develop changes in their speech at the same stage of the disease either. For those who are affected, the voice may get softer, breathier, or hoarse over time. The voice tone may become monotone, lacking the usual ups and downs along with some other problems. As a result of this, the patient gradually finds verbal communication very difficult, and also, people who listen to them need to ask them to repeat the sentences quiet often. In medical terms, these problems are known as dysarthria, hypophonia, tachyphemia, etc.

Another significant indicator of this disease is the tremor, which is commonly misdiagnosed as the essential tremor (ET). While both ET and PD are neurological disorders, ET is commonly seen in the middle-aged population; of course, the onset could be during the life span. On the other hand, people after 55–60 years of age are in general affected by PD. Like voice degradation, this tremor also makes the patient’s life miserable as the limbs’ involuntary movements hinder them from performing the day-to-day activities comfortably. Unfortunately, there is no cure for this disease yet but medication to keep the symptoms under control. Besides, early detection of PD is essential as the treatments such as levodopa/carbidopa are more effective if administered in the early stages of the disease. Besides, nonpharmacologic treatments, such as increased exercise, are also easier to perform in the early stages of PD, which may help slow down disease progression as found by studies.

1.2 Problem Statement

Parkinson’s Disease(PD) affects patients’ quality of life, makes social interaction more difficult for them, and worsens their financial condition with extravagant medical expenses. PD causes several motor and non-motor symptoms, which gradually become prominent after different disease progression stages. One of the secondary motor symptoms that people with PD may experience is the change in their speech quality or difficulty speaking in worse cases. However, everyone with PD does not experience the same symptoms. Similarly, all the patients do not develop changes in their speech at the same stage of the disease either. For those who are affected, the voice may get softer, breathier, or hoarse over time. The voice tone may become monotone, lacking the usual ups and downs along with some other problems. As a result of this, the patient gradually finds verbal communication very difficult, and also, people who listen to them need to ask them to repeat the sentences quiet oft. In medical terms, these problems are known as dysarthria, hypophonia, tachyphemia, etc.

Various speech signal processing algorithms including Time Frequency Features, Mel Frequency Cepstral Coefficients (MFCCs), Wavelet Transform based Features, Vocal Fold Features and TWQT features have been applied to the speech recordings of Parkinson's Disease (PD) patients to extract clinically useful information for PD assessment. Another significant indicator of this disease is the tremor , which is commonly misdiagnosed as the essential tremor (ET). While both ET and PD are neurological disorders, ET is commonly seen in the middle-aged population; of course, the onset could be during the life span. On the other hand, people after 55–60 years of age are in general affected by PD. Like voice degradation, this tremor also makes the patient’s life miserable as the limbs’ involuntary movements hinder them from performing the day to-day activities comfortably.

1.3 Motivation

With the growing number of the aged population, the number of Parkinson’s disease (PD) affected people is also mounting. Unfortunately, due to insufficient resources and awareness in underdeveloped countries, proper and timely PD detection is highly challenged. Besides, all PD patients’ symptoms are neither the same nor they all become pronounced at the same stage of the illness. Therefore, by combining more than one symptom by collecting data and will be fed to the machining learning model and deep learning models. No proper cure or treatment is available yet at the advanced stage. Treatment is possible only when done at the early or onset of the disease. These will not only reduce the cost of the disease but will also possibly save a life. Most methods available can detect Parkinson in an advanced stage; which means loss of approx.. 60% dopamine in basal ganglia and is responsible for controlling the movement of the body with a small amount of dopamine. Most Parkinson Disease (PD) patients suffer from vocal cord disorders. Speech impairment is an early indicator of PD.

1.4 Aim of the problem

The proposed models are machine learning models which identify whether a person has Parkinson’s Disease or not. Speech disability is an early indicator of Parkinson’s Disease, so this model identifies the disease by using auditorial features such as speech. These features are modelled by Various machine learning models like Classification Trees, Support Vector Machine, KNN, Logistic Regression, Linear Discriminant analysis, Random Forest classifier, Ada Boost and a few deep learning models i.e., Feed forward neural networks with two hidden layers. The network will be formed as shown in the Fig 1. New data will be given to this model then it should be able to detect whether a person has a Parkinson’s or not.

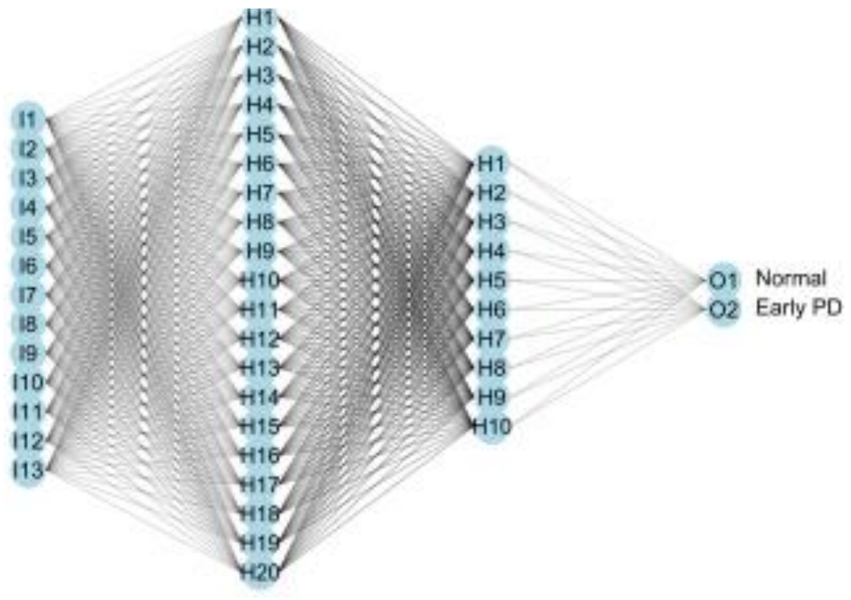


Fig 1 FNN

Three deep learning models with different structures are trained to show robustness of the results with respect to different hyperparameters. And ensemble them using averaging. The first network, abbreviated as DEEP1, has 40 and 20 neurons in the first and second hidden layers, respectively. The second network (DEEP2) has has 20 and 20 neurons in the first and second hidden layers, respectively. The third network (DEEP3) has has 20 and 10 neurons in the first and second hidden layers, respectively.

Lastly, we combine the three deep learning models to form an ensemble (DEEP_EN) by averaging the outputs. The deep learning models are trained with 100 epochs of data.

2. LITERATURE SURVEY

Accurate and early detection of PD is vital due to its ability to provide crucial information to slow down the progression of PD Various methods have proposed to help detection PD based on different kinds of measurements including speech data gait patterns force tracking, data smell identification data and spontaneous cardiovascular oscillations.

H. Gunduz[1]:

Author proposed a deep learning-based 5arkinson's disease classification using vocal feature sets. In an early detection algorithm of PD based on vocal features is designed. It has been illustrated that the use of Wrappers subset selection is suitable because of the low dimensionality of the selected feature and improved PD detection capability.

Imanne El Maachi, Guillaume-Alexandre Bilodeau, Wassim Bouachir[2]:

Authors have proposed a PD detection system is introduced using a 1D convolutional neural network based on the gait signals. However, the performance of PD detection based on both speech and gait analyses is generally limited by the sensitivity to background noise in speech recording, causing a high number of false alarms and missed detection.

U. Gupta, H. Bansal, and D. Joshi[3]:

They proposed a model based on Evaluation of hand-writing kinematics and pressure for differential diagnosis of Parkinson's disease. Other approaches employed hand writing measurement for Parkinson's diagnosis In the PD diagnosis approach has been proposed based on handwriting measurements gathered from patients with PD. It has been shown that improved PD diagnosis is obtained when taking into consideration the age and sex information in the decision process.

T. Arroyo-Gallego, M. J. Ledesma-Carbayo, A. Sánchez-Ferro[4]:

In this approach to detect motor impairment in PD based on mobile touchscreen typing is introduced. Essentially, the proposed algorithm uncovers signs of PD motor by analysing touchscreen typing features that include descriptive statistics (covariance, skewness, and kurtosis) and time information.

S. Ashour, A. El-Attar, N. Dey, H. A. El-Kader, and M. M. A. El-Naby[5]:

Deep learning-based techniques have gained special attention in PD diagnosis due to their capacity to handle big data and achieving high accuracy with free-assumption on data distribution. Authors applied a Long short term memory algorithm to detect the Freezing of Gait (FOG), which a good indicator of PD patients that may cause falling. It has been shown that LSTM outperforms the SVM in detecting FOG.

R. Das[6]:

Author has proposed a comparison of multiple classification methods for diagnosis of 6arkinson disease. The performance of four classifiers, Decision Trees, Regression, Dmneural, and Neural Networks (NN), has been compared in detecting PD, and the best accuracy of 92.9% is obtained using NN algorithm.

M. Little, P. McSharry, E. Hunter, J. Spielman, and L. Ramig[7]:

Authors have proposed a model based on suitability of dysphonia measurements for telemonitoring of 6arkinson's disease. The support vector machine (SVM) is applied to only four dysphonic features for PD classification due to its ability to extract nonlinearity by using nonlinear kernels.

A. Wagner, N. Fixler, and Y. S. Resheff,[8]:

Authors have proposed a wavelet-based approach to monitoring 6arkinson's disease symptoms which suggest a method based on wavelet to analyze data collected from smartwatches worn by nineteen patients affected by PD. This method showed good ability in detecting symptoms of tremor, bradykinesia, and dyskinesia

V. Illner, P. Sovka, and J. Rusz,[9]:

Authors have proposed a method based on validation of freely-available pitch detection algorithms across various noise levels in assessing speech captured by smartphone in 7arkinson’s disease and also based on an approach using the sawtooth inspired pitch estimator (SWIPE) scheme is used to assess speech disorders recorded via smartphone caused by Parkinson’s disease. Acceptable results have been achieved.

3. EXISTING METHOD

Parkinson’s Disease is a progressive nervous system disorder that affects movement leading to shaking, stiffness and difficulty with walking, balance, and coordination. Parkinson’s symptoms usually begin gradually and get worse over time. It is mostly seen in elderly persons. It is one of the critical diseases in health care. Traditional diagnosis is not reliable as they rely on evaluation of movements that are sometimes subtle to human eye and difficult to classify, sometimes these symptoms are often overlooked making diagnosis of PD at an early stage challenging.

PD detection systems are focused on recognizing the severity of symptoms using several types of instruments. One of the most common symptoms is the vocal problem, and most patients faces vocal defections in the early stages of the disease. Therefore, health systems based on vocal disorders have leading position on recent PD detection studies. Several speech signal processing techniques were used to obtain clinically relevant features, and extracted features were fed into various artificial learning methods to obtain reliable decisions in PD classification. While Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are the common algorithms in PD classification, Random Forest (RF), and K-Nearest Neighbors (KNN) are also properly used due to their simplicity and ease of understanding.

The success of mentioned algorithms is directly related to the quality of the features selected from the data. Although it is difficult to manually select the relevant features that represent the intrinsic properties of the speech (audio) data, the latent properties of the data can be learned automatically via deep learning approach. Hierarchical layers in Deep Neural Networks (DNN) can create deep abstract representations that are used as input features in ML tasks.

3. SYSTEM REQUIREMENTS SEPCIFICATION

A System Requirements Specification is a structured collection of information that embodies the requirements of a project. This section explains about both Software and Hardware requirements which are required for the completion of the project.

4.1 Software requirements

Software requirements establish the agreement between your team and the customer on what the application is supposed to do. Without a description of what features will be included and details on how the features will work, the users of the software can't determine if the software will meet their needs**.** The key software requirements required for the project are:

- Google colab
- Python
- Tensorflow
- Keras
- Scikit learn

4.1.1 Google colab

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs and TPUs . Colab notebooks are stored in Google Drive, or can be loaded from GitHub. Colab notebooks can be shared just as you would with Google Docs or Sheets. Simply click the Share button at the top right of any Colab notebook, or follow these Google Drive file sharing instructions.

If you are familiar with the **Jupyter Notebook**, you can think of Google Colab as a supercharged version of the Jupyter Notebook, hosted on Google's cloud servers, with multiple convenient features. And if you are not familiar with it, do not worry, because this tutorial does not require any prior knowledge on Jupyter Notebook.
There are several reasons to opt to use google colab instead of a plain Jupyter Notebook instance, like:

1. Pre-Installed libraries
2. Saved on the cloud
3. Collaboration

1. Pre-Installed Libraries

Anaconda distribution of Jupyter Notebook shipped with several pre-installed data libraries, such as Pandas, NumPy, Matplotlib, which is awesome. Google Colab, on the other hand, provides even more pre-installed machine learning libraries such as Keras, TensorFlow, and PyTorch.

2. Saved on the Cloud

When you opt to use a plain Jupyter notebook as your development environment, everything is saved in your local machine. If you are cautious about privacy, this may be a preferred feature for you. However, if you want your notebooks to be accessible to you from any device with a simple Google log-in, then Google Colab is the way to go.

3. Collaboration

Another great feature that Google Colab offers is the collaboration feature. If you are working with multiple developers on a project, it is great to use Google Colab notebook. Just like collaborating on a Google Docs document, you can co-code with multiple developers using a Google

Colab notebook. Besides, you can also share your completed work with other developers.

4.1.2 Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python’s simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

4.1.3 Tensorflow:

The standard name for Machine Learning in the Data Science industry is Tensorflow. TensorFlow may be a free and open-source software library for machine learning. It is often used across a variety of tasks but features a particular specialize in training and inference of deep neural networks. Tensorflow may be a symbolic math library supported dataflow and differentiable programming. It facilitates building of both statistical machine learning solutions as well as deep learning through its extensive interface of CUDA GPUs.

4.1.4 Keras:

Keras is an open-source neural network library that provides support for Python. It is popular for its modularity, speed, and ease of use. Therefore, it can be used for fast experimentation as well as rapid prototyping. It provides support for the implementation of Convolutional Neural Networks, Recurrent Neural Networks as well as both. It is capable of running seamlessly on the **CPU** and **GPU**. Compared to more widely popular libraries like Tensorflow and Pytorch, Keras provides user-friendliness that allows the users to readily implement neural networks without dwelling over the technical jargon.

4.1.5 Scikit-learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Components of scikit-learn:

Scikit-learn comes loaded with a lot of features. Here are a few of them to help you understand the spread:

- **Supervised learning algorithms:** Think of any supervised machine learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of machine learning algorithms is one of the big reasons for the high usage of scikit-learn
- **Cross-validation:** There are various methods to check the accuracy of supervised models on unseen data using sklearn.

4.2 Hardware Requirements

As we are using Deep learning we need high processing is required. For this we should use GPU for fast processing. Although a minimum of 8GB RAM can do the job, 16GB RAM and above is recommended for most of the deep learning tasks. When it comes to CPU, a minimum of 7th generation (Intel Core i5 processor) is recommended.

4. PROPOSED METHOD AND RESULTS

Different models have been created by various machine learning models like Classification Trees, Support Vector Machine, KNN, Logistic Regression, Linear Discriminant analysis, Random Forest classifier, Ada Boost and a few deep learning models i.e., Feed forward neural networks with two hidden layers and an ensemble of the built deep learning models. New data will be given to this model then it should be able to detect whether a person has a Parkinson's or not. The algorithm of each model have the steps as shown in Fig 5. This section describes about each step present in the algorithm.

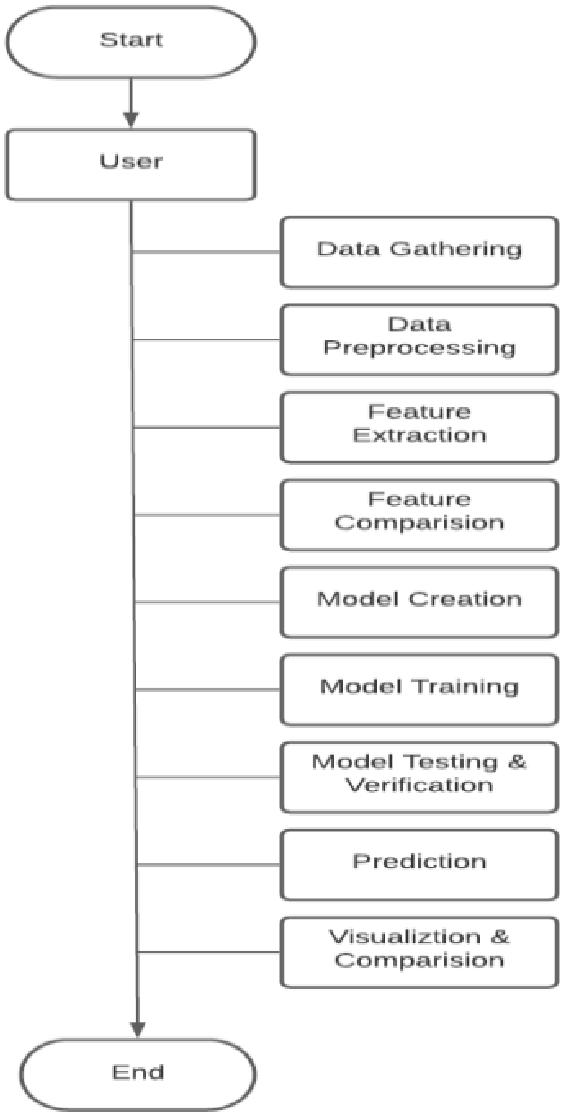


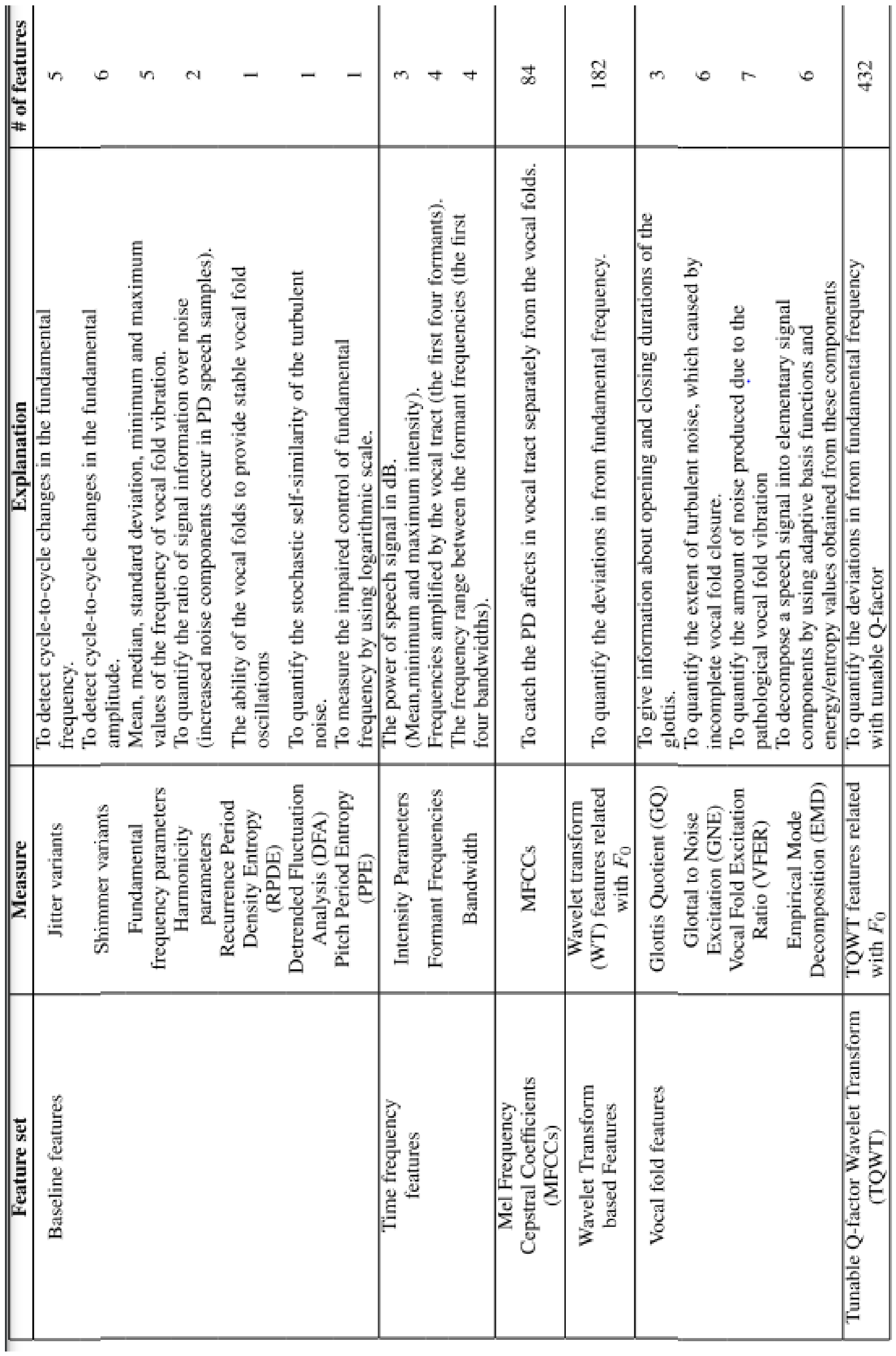
Fig 5 Algorithm flow

5.1 Data Gathering:

A machine learning model is built by learning and generalizing from training data, then applying that acquired knowledge to new data it has never seen before to make predictions and fulfil its purpose. Lack of data will prevent you from building the model, and access to data isn’t enough. Useful data needs to be clean and in a good shape.Identify your data needs and determine whether the data is in proper shape for the machine learning project. The focus should be on data identification, initial collection, requirements, quality identification, insights and potentially interesting aspects that are worth further investigation.

As shown in Fig5.1, the data used in this project were gathered from 188 patients with PD (107 men and 81 women) with ages ranging from 33 to 87 (65.1Â±10.9) at the Department of Neurology in CerrahpaÅŸa Faculty of Medicine, Istanbul University. The control group consists of 64 healthy individuals (23 men and 41 women) with ages varying between 41 and 82 (61.1Â±8.9). During the data collection process, the microphone is set to 44.1 KHz and following the physicianâs examination, the sustained phonation of the vowel /a/ was collected from each subject with three repetitions.

Various speech signal processing algorithms including Time Frequency Features, Mel Frequency Cepstral Coefficients (MFCCs), Wavelet Transform based Features, Vocal Fold Features and TWQT features have been applied to the speech recordings of Parkinson’s Disease (PD) patients to extract clinically useful information for PD assessment. Different required parameters have been extracted from the voice recordings with the help of all these tools.



Feature set	Measure	Explanation	# of features
Baseline features	Jitter variants	To detect cycle-to-cycle changes in the fundamental frequency.	5
	Shimmer variants	To detect cycle-to-cycle changes in the fundamental amplitude.	6
	Fundamental frequency parameters	Mean, median, standard deviation, minimum and maximum values of the frequency of vocal fold vibration.	5
	Harmonicity parameters	To quantify the ratio of signal information over noise (increased noise components occur in PD speech samples).	2
	Recurrence Period Density Entropy (RPDE)	The ability of the vocal folds to provide stable vocal fold oscillations	1
	Detrended Fluctuation Analysis (DFA)	To quantify the stochastic self-similarity of the turbulent noise.	1
	Pitch Period Entropy (PPE)	To measure the impaired control of fundamental frequency by using logarithmic scale.	1
Time frequency features	Intensity Parameters	The power of speech signal in dB. (Mean,minimum and maximum intensity).	3
	Formant Frequencies	Frequencies amplified by the vocal tract (the first four formants).	4
	Bandwidth	The frequency range between the formant frequencies (the first four bandwidths).	4
Mel Frequency Cepstral Coefficients (MFCCs)	MFCCs	To catch the PD affects in vocal tract separately from the vocal folds.	84
Wavelet Transform based Features	Wavelet transform (WT) features related with F_0	To quantify the deviations in from fundamental frequency.	182
Vocal fold features	Glottis Quotient (GQ)	To give information about opening and closing durations of the glottis.	3
	Glottal to Noise Excitation (GNE)	To quantify the extent of turbulent noise, which caused by incomplete vocal fold closure.	6
	Vocal Fold Excitation Ratio (VFER)	To quantify the amount of noise produced due to the pathological vocal fold vibration	7
	Empirical Mode Decomposition (EMD)	To decompose a speech signal into elementary signal components by using adaptive basis functions and energy/entropy values obtained from these components	6
Tunable Q-factor Wavelet Transform (TQWT)	TQWT features related with F_0	To quantify the deviations in from fundamental frequency with tunable Q-factor	432

Fig 5.1 Dataset attributes

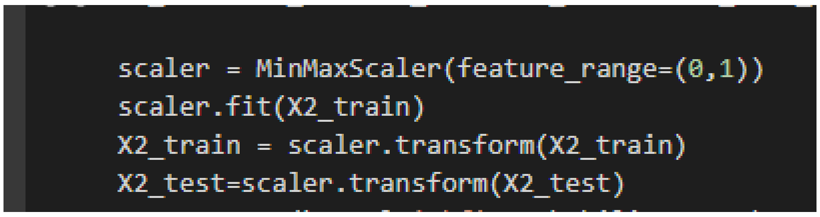
5.2 Data Pre-processing:

Procedures during the data preparation, collection and cleansing process include the following:

- Standardize formats across different data sources.
- Replace incorrect data.
- Enhance and augment data.
- Remove extraneous information and deduplication.
- Reduce noise reduction and remove ambiguity.
- Normalize or standardize data to get it into formatted ranges.

  The **MinMaxScaler** is to rescale variables into the range [0,1] as shown in Fig 5.2.1, although a preferred scale can be specified via the "feature_range" argument and specify a tuple, including the min and the max for all variables.

  - **Fit the scaler using available training data**. For normalization, this means the training data will be used to estimate the minimum and maximum observable values. This is done by calling the *fit()* function.
  - **Apply the scale to training data**. This means you can use the normalized data to train your model. This is done by calling the *transform()* function.
  - **Apply the scale to data going forward**. This means you can prepare new data in the future on which you want to make predictions.

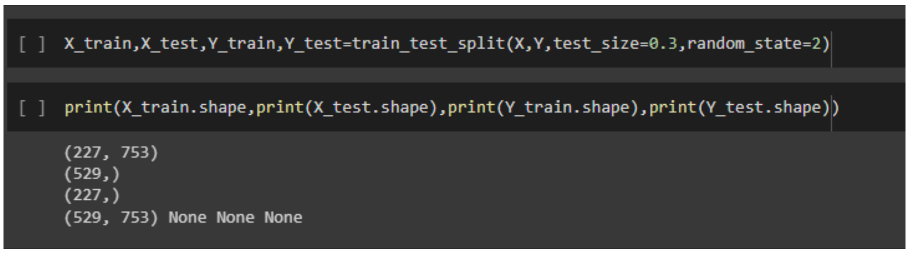


```
scaler = MinMaxScaler(feature_range=(0,1))
scaler.fit(X2_train)
X2_train = scaler.transform(X2_train)
X2_test=scaler.transform(X2_test)
```

Fig 5.2.1 Standardization of data

- Sample data from large data sets.
- Select features that identify the most important dimensions and, if necessary, reduce dimensions using a variety of techniques.

- Split data into training, test and validation sets as shown in Fig 5.2.2. The dataset is splitted in the ratio of **70:30** where 70% of the data is used to train the model and 30 % of the data is used to test the model.



```
[ ]  X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=2)

[ ]  print(X_train.shape,print(X_test.shape),print(Y_train.shape),print(Y_test.shape))

     (227, 753)
     (529,)
     (227,)
     (529, 753) None None None
```

Fig 5.2.2 train-test split

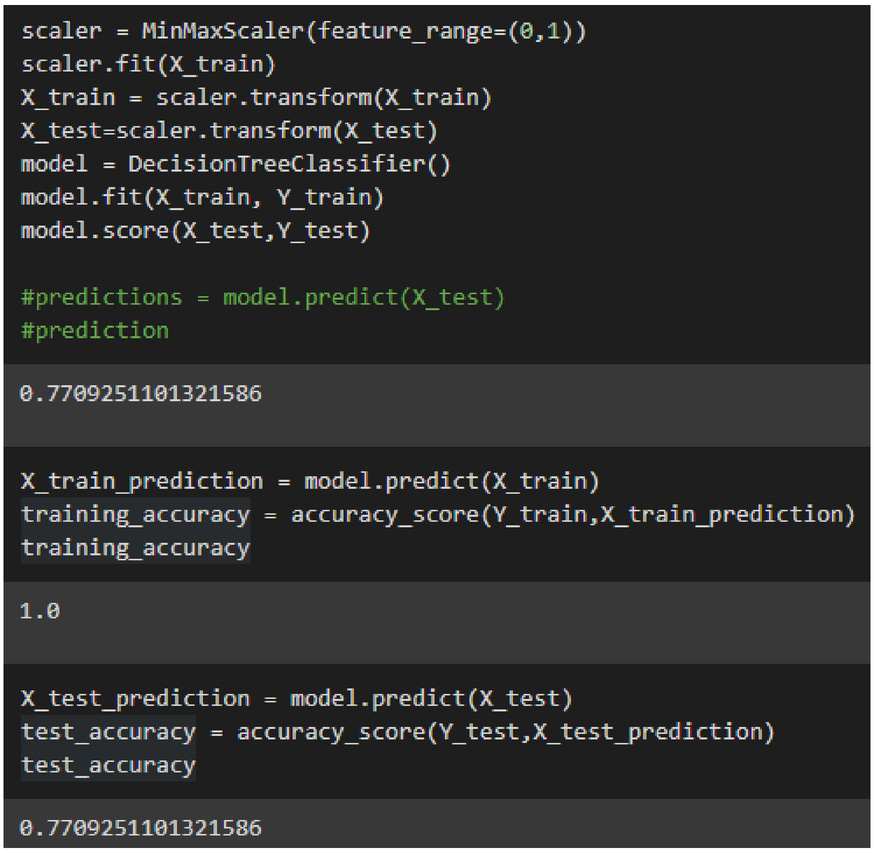
5.3 Model Creation:

This phase requires model technique selection and application, model training, model hyperparameter setting and adjustment, model validation, ensemble model development and testing, algorithm selection, and model optimization.

The following machine learning models are used:

5.3.1 Classification Trees:

Classification tree recursively splits the feature space into sub-sets such that the Gini impurity is minimized at each step. The classification tree algorithm can be built as shown in the Fig 5.3.1. It first grows a tree to the maximum depth such that each leaf node is pure, then prunes upwards to balance the classification error.



```
scaler = MinMaxScaler(feature_range=(0,1))
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test=scaler.transform(X_test)
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
model.score(X_test,Y_test)

#predictions = model.predict(X_test)
#prediction
```

```
0.7709251101321586
```

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

```
1.0
```

```
X_test_prediction = model.predict(X_test)
test_accuracy = accuracy_score(Y_test,X_test_prediction)
test_accuracy
```

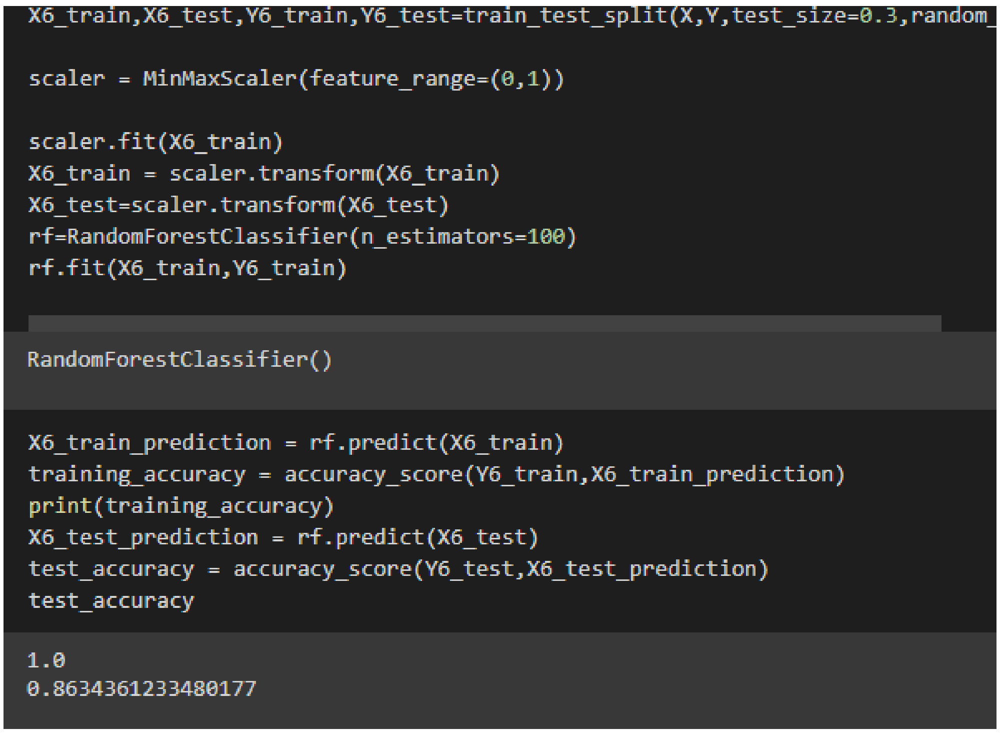
```
0.7709251101321586
```

Fig 5.3.1 Classification trees

5.3.2 Random Forest:

Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The results after applying the Random forest algorithm are shown in the Fig 5.6.6.

The train and test accuracy are shown in the Fig 5.3.2. The number of trees used as estimators is 100.



```
X6_train,X6_test,Y6_train,Y6_test=train_test_split(X,Y,test_size=0.3,random_

scaler = MinMaxScaler(feature_range=(0,1))

scaler.fit(X6_train)
X6_train = scaler.transform(X6_train)
X6_test=scaler.transform(X6_test)
rf=RandomForestClassifier(n_estimators=100)
rf.fit(X6_train,Y6_train)
```

```
RandomForestClassifier()
```

```
X6_train_prediction = rf.predict(X6_train)
training_accuracy = accuracy_score(Y6_train,X6_train_prediction)
print(training_accuracy)
X6_test_prediction = rf.predict(X6_test)
test_accuracy = accuracy_score(Y6_test,X6_test_prediction)
test_accuracy
```

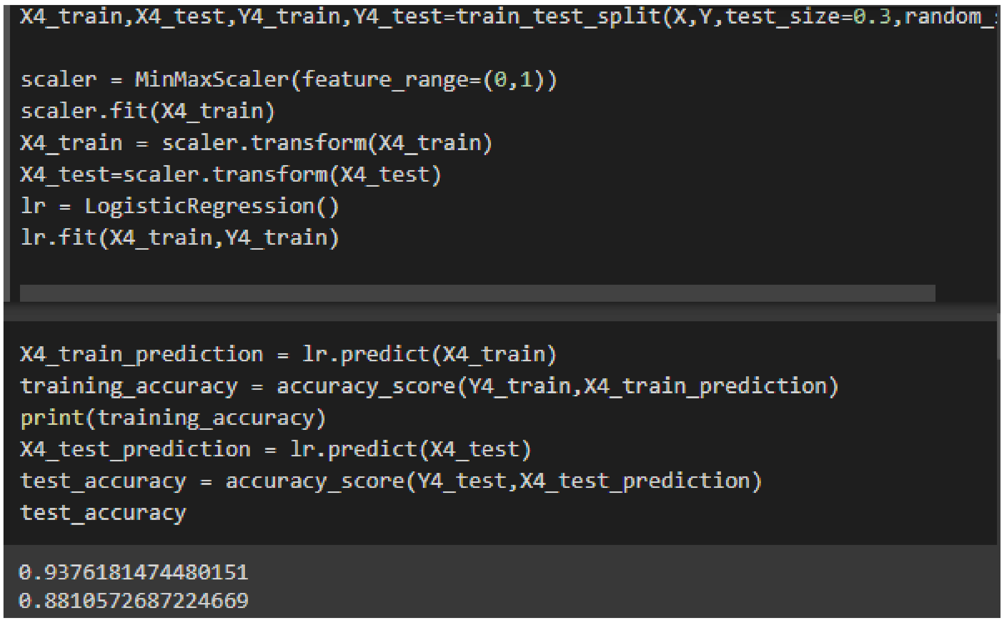
```
1.0
0.8634361233480177
```

Fig 5.3.2 Random Forest

5.3.3 Logistic Regression:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. Here we perform Binary Logistic regression where a dependent variable will have only two possible types either 1 and 0. Here the dependent variable is the status of the person whether the person is a PD patient or a healthy control.

The results after applying the Logistic regression algorithm are shown in the Fig 5.6.4. The train and test accuracy are shown in the Fig 5.3.3.



```
X4_train,X4_test,Y4_train,Y4_test=train_test_split(X,Y,test_size=0.3,random_

scaler = MinMaxScaler(feature_range=(0,1))
scaler.fit(X4_train)
X4_train = scaler.transform(X4_train)
X4_test=scaler.transform(X4_test)
lr = LogisticRegression()
lr.fit(X4_train,Y4_train)
```

```
X4_train_prediction = lr.predict(X4_train)
training_accuracy = accuracy_score(Y4_train,X4_train_prediction)
print(training_accuracy)
X4_test_prediction = lr.predict(X4_test)
test_accuracy = accuracy_score(Y4_test,X4_test_prediction)
test_accuracy
```

```
0.9376181474480151
0.8810572687224669
```

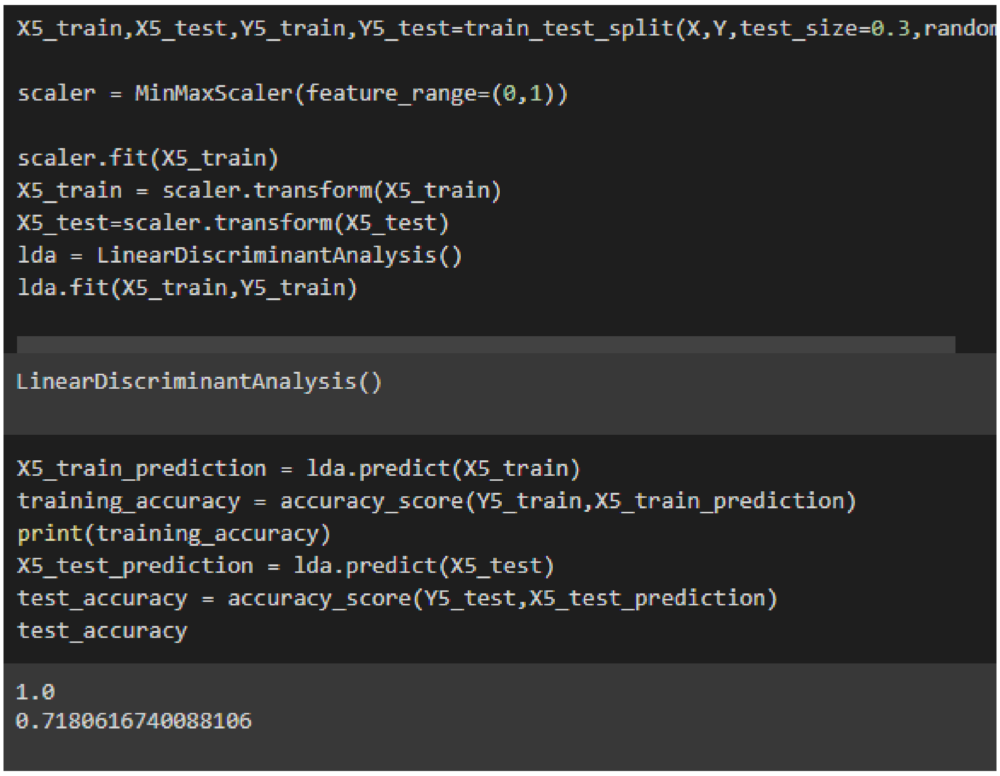
Fig 5.3.3 Logistic Regression

5.3.4 Linear Discriminant Analysis:

Linear Discriminant Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modelling differences in groups i.e. separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space. A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.

The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix. The fitted model can also be used to reduce the dimensionality of the input by projecting it to the most discriminative directions, using the transform method.

The results after applying the Linear Discriminant Analysis algorithm are shown in the Fig 5.6.5. The train and test accuracy are shown in the Fig 5.3.4



```
X5_train,X5_test,Y5_train,Y5_test=train_test_split(X,Y,test_size=0.3,randon

scaler = MinMaxScaler(feature_range=(0,1))

scaler.fit(X5_train)
X5_train = scaler.transform(X5_train)
X5_test=scaler.transform(X5_test)
lda = LinearDiscriminantAnalysis()
lda.fit(X5_train,Y5_train)
```

```
LinearDiscriminantAnalysis()
```

```
X5_train_prediction = lda.predict(X5_train)
training_accuracy = accuracy_score(Y5_train,X5_train_prediction)
print(training_accuracy)
X5_test_prediction = lda.predict(X5_test)
test_accuracy = accuracy_score(Y5_test,X5_test_prediction)
test_accuracy
```

```
1.0
0.7180616740088106
```

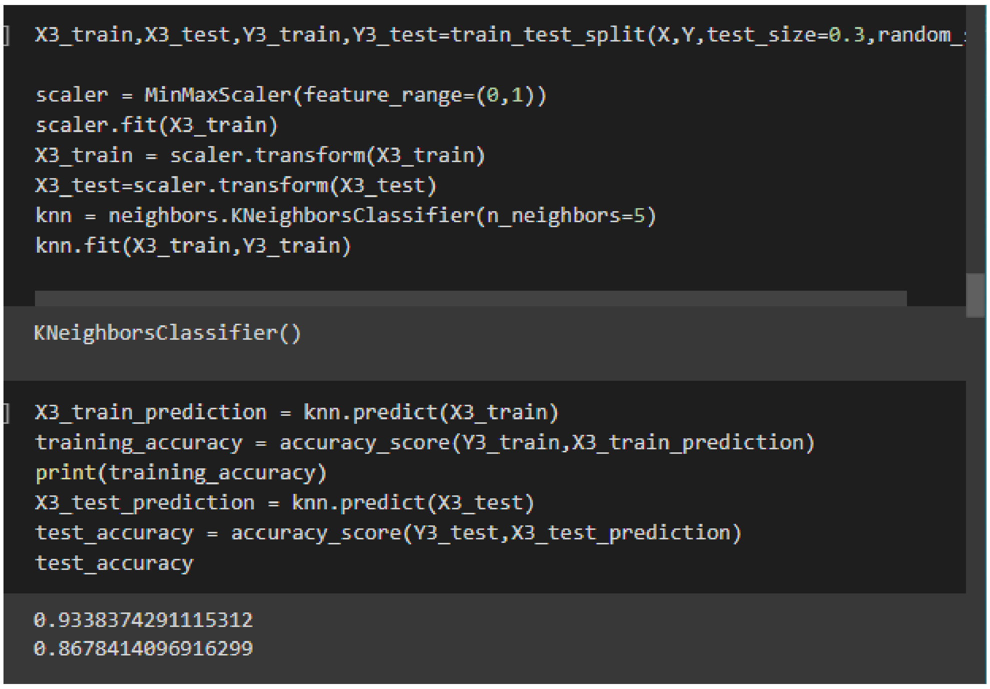
Fig 5.3.4 Linear Discriminant Analysis

5.3.5 K-Nearest Neighbours(KNN):

KNN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most like the available categories. It stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using it.

The number of nearest neighbours is considered as 5.

The results after applying the K-Nearest Neighbours algorithm are shown in the Fig 5.6.3. The train and test accuracy are shown in the Fig 5.3.5.



```
X3_train,X3_test,Y3_train,Y3_test=train_test_split(X,Y,test_size=0.3,random_

scaler = MinMaxScaler(feature_range=(0,1))
scaler.fit(X3_train)
X3_train = scaler.transform(X3_train)
X3_test=scaler.transform(X3_test)
knn = neighbors.KNeighborsClassifier(n_neighbors=5)
knn.fit(X3_train,Y3_train)
```

```
KNeighborsClassifier()
```

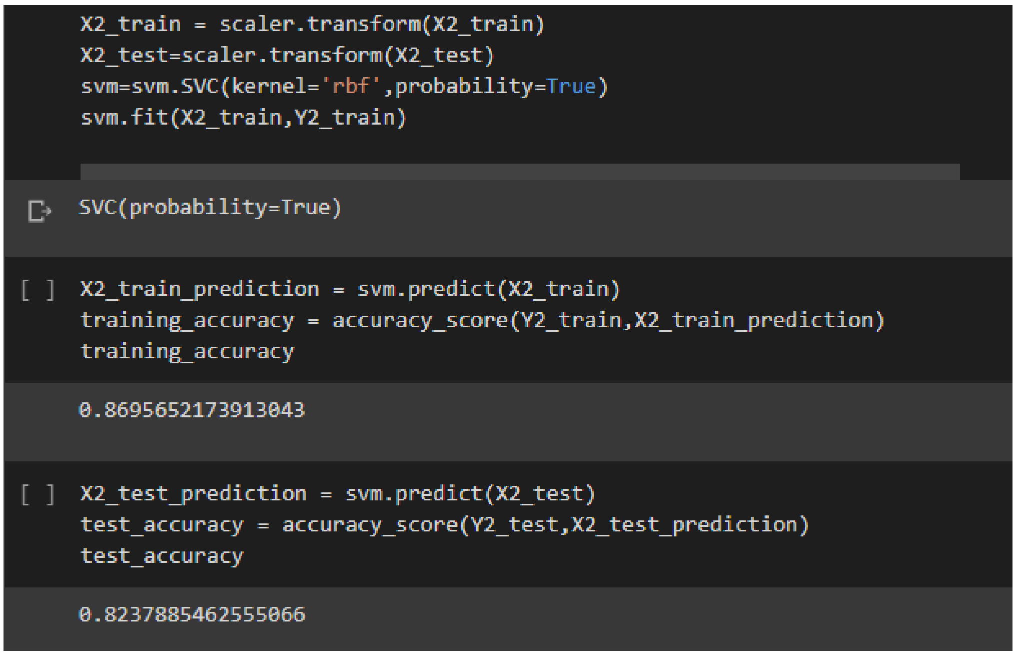
```
X3_train_prediction = knn.predict(X3_train)
training_accuracy = accuracy_score(Y3_train,X3_train_prediction)
print(training_accuracy)
X3_test_prediction = knn.predict(X3_test)
test_accuracy = accuracy_score(Y3_test,X3_test_prediction)
test_accuracy
```

```
0.9338374291115312
0.8678414096916299
```

Fig 5.3.5 KNN

5.3.6 Support Vector Machine(SVM):

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. The kernel used is **'rbf'.** The results after applying the Support Vector Machine algorithm are shown in the Fig 5.6.2. The train and test accuracy are shown in the Fig 5.3.6.



```
X2_train = scaler.transform(X2_train)
X2_test=scaler.transform(X2_test)
svm=svm.SVC(kernel='rbf',probability=True)
svm.fit(X2_train,Y2_train)
```

```
SVC(probability=True)
```

```
[ ] X2_train_prediction = svm.predict(X2_train)
    training_accuracy = accuracy_score(Y2_train,X2_train_prediction)
    training_accuracy
```

```
0.8695652173913043
```

```
[ ] X2_test_prediction = svm.predict(X2_test)
    test_accuracy = accuracy_score(Y2_test,X2_test_prediction)
    test_accuracy
```

```
0.8237885462555066
```

Fig 5.3.6 Support Vector Machine

5.3.7 AdaBoost:

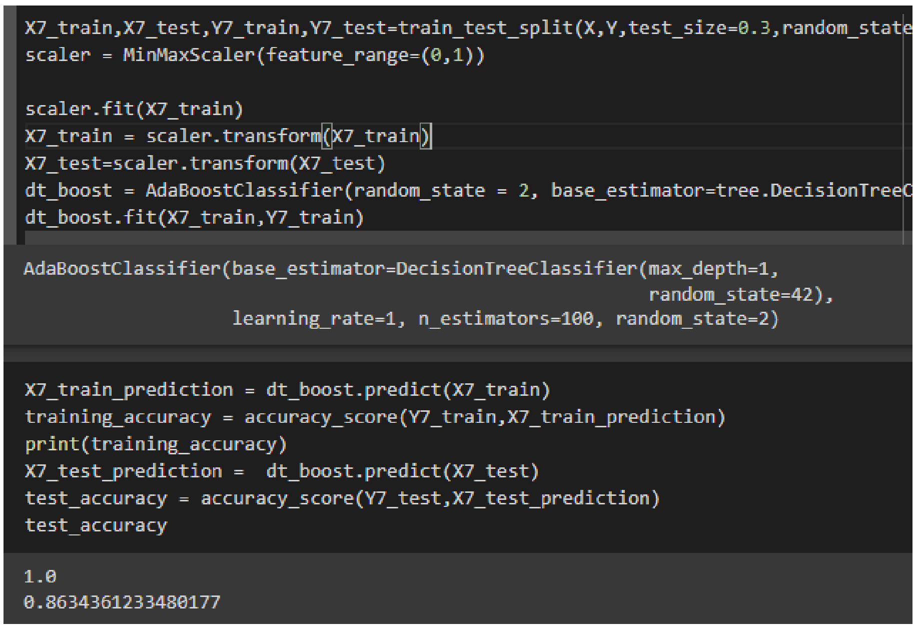
AdaBoost is an ensemble learning method (also known as "meta-learning") which was initially created to increase the efficiency of binary classifiers. AdaBoost uses an iterative approach to learn from the mistakes of weak classifiers, and turn them into strong ones. A single classifier may not be able to accurately predict the class of an object, but when we group multiple weak classifiers with each one progressively learning from the others' wrongly classified objects, we can build one such strong model. The classifier mentioned here could be any of your basic classifiers, from Decision Trees (often the default) to Logistic Regression, etc.

base_estimator *object, default=None*

- The base estimator from which the boosted ensemble is built. Support for sample weighting is required, as well as proper classes_ and n_classes_ attributes. If None, then the base estimator is DecisionTreeClassifier initialized with max_depth=1.

- n_estimators *int, 100*
- The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. learning_rate*float, default=1.0*
- Weight applied to each classifier at each boosting iteration. A higher learning rate increases the contribution of each classifier. There is a trade-off between
  the learning_rate and n_estimators parameters.

The results after applying the AdaBoost algorithm are shown in the Fig 5.6.7. The train and test accuracy are shown in the Fig 5.3.7.



```
X7_train,X7_test,Y7_train,Y7_test=train_test_split(X,Y,test_size=0.3,random_state
scaler = MinMaxScaler(feature_range=(0,1))

scaler.fit(X7_train)
X7_train = scaler.transform(X7_train)
X7_test=scaler.transform(X7_test)
dt_boost = AdaBoostClassifier(random_state = 2, base_estimator=tree.DecisionTreeC
dt_boost.fit(X7_train,Y7_train)
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1,
                                                         random_state=42),
                   learning_rate=1, n_estimators=100, random_state=2)
```

```
X7_train_prediction = dt_boost.predict(X7_train)
training_accuracy = accuracy_score(Y7_train,X7_train_prediction)
print(training_accuracy)
X7_test_prediction =  dt_boost.predict(X7_test)
test_accuracy = accuracy_score(Y7_test,X7_test_prediction)
test_accuracy
```

```
1.0
0.8634361233480177
```

Fig 5.3.7 Ada Boost

5.3.8 Deep Learning Models

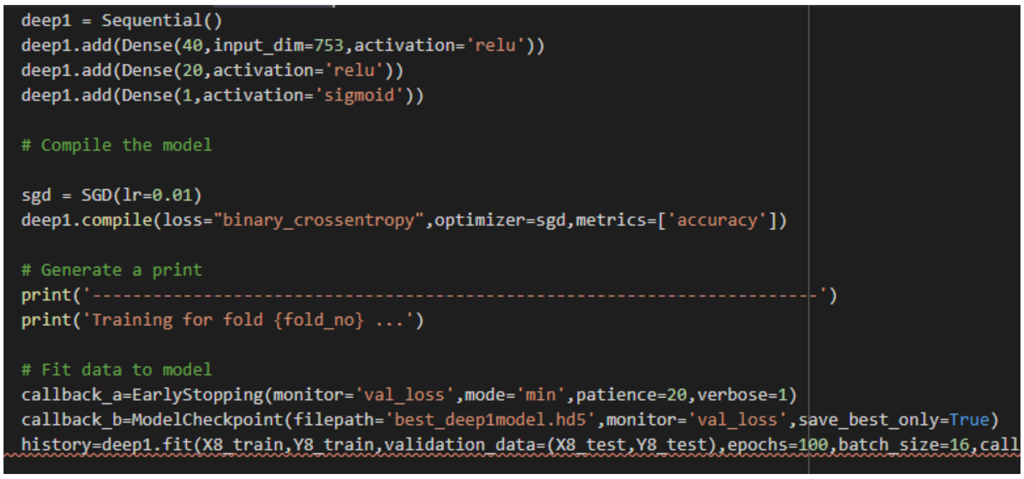
The neurons are information processing modules, which essentially are simple nonlinear transformations of inputs. For the supervised feed-forward neural network (FNN) considered in this paper, when raw-data are fed into the network, the deep learning algorithm can automatically extract hierarchical representations of the data which are best suited for the underlying learning task. The model contains an FNN which has two hidden layers. The first hidden layer has twenty neurons, and the second hidden layer has ten neurons.

Keras package is used to build the network architecture.

Deep 1:

1. Import the necessary packages to create a simple feedforward neural network with Keras.
2. The Sequential class indicates that our network will be feedforward and layers will be added to the class sequentially, one on top of the other.
3. The Dense classis the implementation of our fully connected layers. For our network to actually learn, we need to apply SGD to optimize the parameters of the network
4. For the first layer we pass the inputs as our data. Thus the shape of the input will be the count of the dimensions i.e., 753 and then learn 40 weights in this layer and apply relu activation function.
5. For the next layer it learns 20 weights and uses relu as a activation function.
6. For the final layer which is used to classify the data we use sigmoid activation function which learns only one weight in this binary classification.
7. We initialize the SGD optimizer with a learning rate of 0.01. We'll use the category cross-entropy loss function as our loss metric .
8. We'll allow our network to train for a total of 100 epochs using a batch size of 16 data points at a time.

Two Call back functions have been used as shown in Fig 5.3.8.1 while applying the algorithm. The first Call back function is Early Stopping which is used to stop the algorithm if the accuracy of the algorithm will not increase after some epochs. Second Call back function is Model Checkpoint which is used to save the best model obtained while performing the algorithm. The results after applying the Deep Learning and the Accuracy, loss obtained after each fold is plotted in the form of a graph as shown in the Fig 5.6.8.1.



```
deep1 = Sequential()
deep1.add(Dense(40,input_dim=753,activation='relu'))
deep1.add(Dense(20,activation='relu'))
deep1.add(Dense(1,activation='sigmoid'))

# Compile the model

sgd = SGD(lr=0.01)
deep1.compile(loss="binary_crossentropy",optimizer=sgd,metrics=['accuracy'])

# Generate a print
print('------------------------------------------------------------------------')
print('Training for fold {fold_no} ...')

# Fit data to model
callback_a=EarlyStopping(monitor='val_loss',mode='min',patience=20,verbose=1)
callback_b=ModelCheckpoint(filepath='best_deep1model.hd5',monitor='val_loss',save_best_only=True)
history=deep1.fit(X8_train,Y8_train,validation_data=(X8_test,Y8_test),epochs=100,batch_size=16,call
```

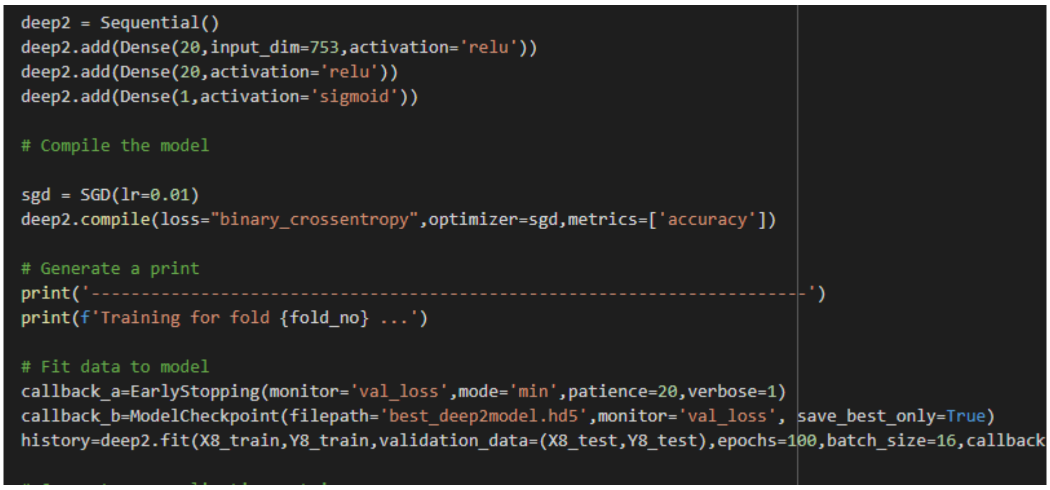
Fig 5.3.8.1 Deep1

Deep 2:

1. Import the necessary packages to create a simple feedforward neural network with Keras.
2. The Sequential class indicates that our network will be feedforward and layers will be added to the class sequentially, one on top of the other.
3. The Dense classis the implementation of our fully connected layers. For our network to actually learn, we need to apply SGD to optimize the parameters of the network
4. For the first layer we pass the inputs as our data. Thus the shape of the input will be the count of the dimensions i.e., 753 and then learn 20 weights in this layer and apply relu activation function.
5. For the next layer it learns 20 weights and uses relu as a activation function.

6. For the final layer which is used to classify the data we use sigmoid activation function which learns only one weight in this binary classification.
7. We initialize the SGD optimizer with a learning rate of 0.01. We'll use the category cross-entropy loss function as our loss metric.
8. We'll allow our network to train for a total of 100 epochs using a batch size of 16 data points at a time.

Two Call back functions have been used as shown in Fig 5.3.8.2 while applying the algorithm. The first Call back function is Early Stopping which is used to stop the algorithm if the accuracy of the algorithm will not increase after some epochs. Second Call back function is Model Checkpoint which is used to save the best model obtained while performing the algorithm. The results after applying the Deep learning algorithm and the Accuracy, loss obtained after each fold is plotted in the form of a graph as shown in the Fig 5.6.8.2.



```
deep2 = Sequential()
deep2.add(Dense(20,input_dim=753,activation='relu'))
deep2.add(Dense(20,activation='relu'))
deep2.add(Dense(1,activation='sigmoid'))

# Compile the model

sgd = SGD(lr=0.01)
deep2.compile(loss="binary_crossentropy",optimizer=sgd,metrics=['accuracy'])

# Generate a print
print('------------------------------------------------------------------------')
print(f'Training for fold {fold_no} ...')

# Fit data to model
callback_a=EarlyStopping(monitor='val_loss',mode='min',patience=20,verbose=1)
callback_b=ModelCheckpoint(filepath='best_deep2model.hd5',monitor='val_loss', save_best_only=True)
history=deep2.fit(X8_train,Y8_train,validation_data=(X8_test,Y8_test),epochs=100,batch_size=16,callback
```

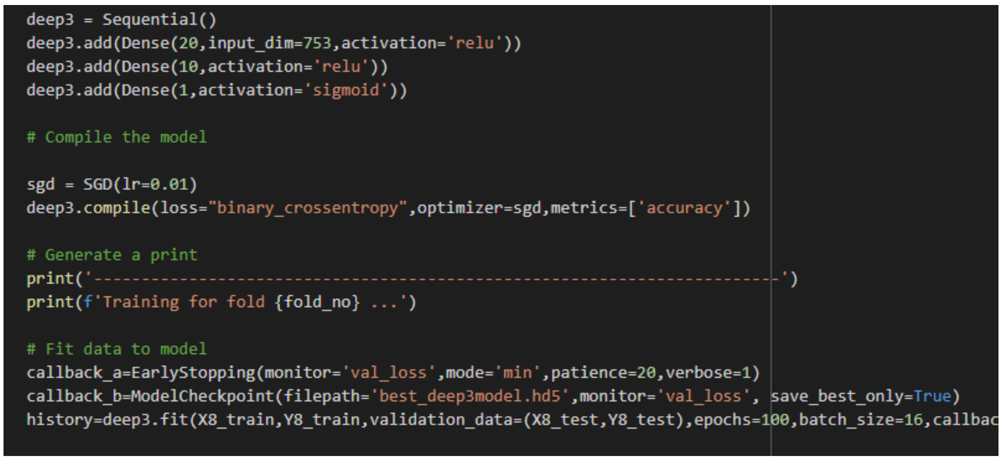
Fig 5.3.8.2 Deep2

Deep 3:

1. Import the necessary packages to create a simple feedforward neural network with Keras.

2. The Sequential class indicates that our network will be feedforward and layers will be added to the class sequentially, one on top of the other.
3. The Dense classis the implementation of our fully connected layers. For our network to actually learn, we need to apply SGD to optimize the parameters of the network
4. For the first layer we pass the inputs as our data. Thus the shape of the input will be the count of the dimensions i.e., 753 and then learn 20 weights in this layer and apply relu activation function.
5. For the next layer it learns 10 weights and uses relu as a activation function.
6. For the final layer which is used to classify the data we use sigmoid activation function which learns only one weight in this binary classification.
7. We initialize the SGD optimizer with a learning rate of 0.01. We'll use the category cross-entropy loss function as our loss metric .
8. We'll allow our network to train for a total of 100 epochs using a batch size of 16 data points at a time.

Two Call back functions have been used as shown in Fig 5.3.8.3 while applying the algorithm. The first Call back function is Early Stopping which is used to stop the algorithm if the accuracy of the algorithm will not increase after some epochs. Second Call back function is Model Checkpoint which is used to save the best model obtained while performing the algorithm. The results after applying the Deep learning algorithm and the accuracy, loss obtained after each fold is plotted in the form of a graph as shown in the Fig 5.6.8.3.



```
deep3 = Sequential()
deep3.add(Dense(20,input_dim=753,activation='relu'))
deep3.add(Dense(10,activation='relu'))
deep3.add(Dense(1,activation='sigmoid'))

# Compile the model

sgd = SGD(lr=0.01)
deep3.compile(loss="binary_crossentropy",optimizer=sgd,metrics=['accuracy'])

# Generate a print
print('------------------------------------------------------------------------')
print(f'Training for fold {fold_no} ...')

# Fit data to model
callback_a=EarlyStopping(monitor='val_loss',mode='min',patience=20,verbose=1)
callback_b=ModelCheckpoint(filepath='best_deep3model.hd5',monitor='val_loss', save_best_only=True)
history=deep3.fit(X8_train,Y8_train,validation_data=(X8_test,Y8_test),epochs=100,batch_size=16,callbac
```

Fig 5.3.8.3 Deep3

Three deep learning models with different structures to show robustness of the results with respect to the hyperparameters. The three trained models are all feed-forward neural networks with two hidden layers.

The first network, abbreviated as DEEP1, has 40 and 20 neurons in the first and second hidden layers, respectively. The second network (DEEP2) has 20 and 20 neurons in the first and second hidden layers, respectively. The third network (DEEP3) has 20 and 10 neurons in the first and second hidden layers, respectively.

5.3.9 Ensemble Deep Learning:

Lastly, we combine the three deep learning models to form an ensemble (DEEP_EN) by averaging the outputs. The results are shown in fig 5.6.9

5.4 Validation and Testing:

All the built models are tested with the help of K-fold Cross validation.

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=5 becoming 15-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    0. Take the group as a hold out or test data set
    1. Take the remaining groups as a training data set
    2. Fit a model on the training set and evaluate it on the test set
    3. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores.

5.5 Metrics used to Validate:

Confusion Matrix: Confusion Matrix as the name suggests gives us a matrix as output and describes the complete performance of the model. There are 4 important terms :

- **True Positives** : The cases in which we predicted YES and the actual output was also YES.
- **True Negatives** : The cases in which we predicted NO and the actual output was NO.
- **False Positives** : The cases in which we predicted YES and the actual output was NO.

- **False Negatives** : The cases in which we predicted NO and the actual output was YES.

5.5.1 Accuracy:

Accuracy for the matrix can be calculated by taking average of the values lying across the **“main diagonal”** i.e

$$Accuracy = \frac{TruePositive + TrueNegative}{TotalSample}$$

5.5.2 Sensitivity:

True Positive Rate (Sensitivity) : True Positive Rate is defined as *TP/ (FN+TP)*. True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points.

$$TruePositiveRate = \frac{TruePositive}{FalseNegative + TruePositive}$$

5.5.3 Specificity:

True Negative Rate (Specificity) : True Negative Rate is defined as *TN / (FP+TN)*. False Positive Rate corresponds to the proportion of negative data points that are correctly considered as negative, with respect to all negative data points.

5.5.4 Precision:

It is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

5.5.5 Recall:

It is the number of correct positive results divided by the number of ***all*** relevant samples (all samples that should have been identified as positive).

$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives}$$

5.5.6 F1- Score:

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances).

Mathematically, it can be expressed as :

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

5.5.7 Area Under Curve

It is one of the most widely used metrics for evaluation. It is used for binary classification problem. *AUC* of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example.

AUC is the area under the curve of plot *False Positive Rate vs True Positive Rate* at different points in **[0, 1]**.

5.5.8 Classification Report:

A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model.

The reported averages include macro average (averaging the unweighted mean per label), weighted average (averaging the support-weighted mean per label), and sample average (only for multilabel classification). Micro average (averaging the total true positives, false negatives and false positives) is only shown for multi-label or multi-class with a subset of classes, because it corresponds to accuracy otherwise and would be the same for all metrics.

5.5.9 Logarithmic loss:

It works by penalising the false classifications. It works well for multi-class classification. When working with Log Loss, the classifier must assign probability to each class for all the samples. Suppose, there are N samples belonging to M classes, then the Log Loss is calculated as below :

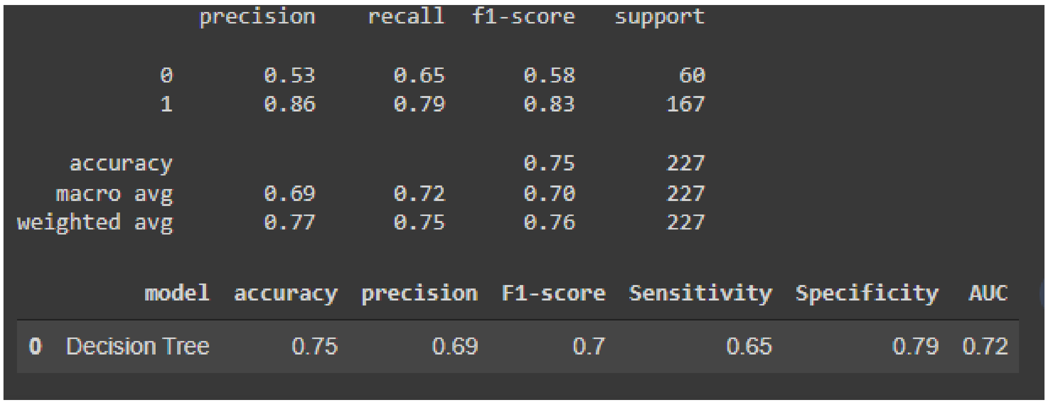
$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log(p_{ij})$$

where, y_ij, indicates whether sample i belongs to class j or not, p_ij, indicates the probability of sample i belonging to class j. Log Loss has no upper bound and it exists on the range [0, ∞). Log Loss nearer to 0 indicates higher accuracy, whereas if the Log Loss is away from 0 then it indicates lower accuracy.

5.6 RESULTS

The data frame contains the results of all the evaluation metrics which are mentioned in the previous section. These results are finally plotted in the form of graph as shown in Fig 5.7 in order to find the best model among all the models.

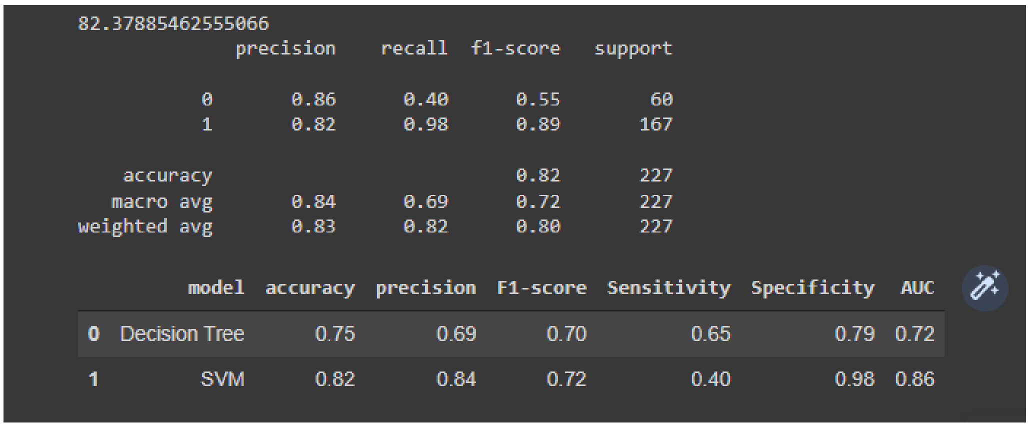
After applying the decision trees algorithm, the results are captured from the classification report. All the required parameters are retrieved and stored in a data frame. The report and data frame is shown in Fig **5.6.1.**



```
              precision    recall  f1-score   support

           0       0.53      0.65      0.58        60
           1       0.86      0.79      0.83       167

    accuracy                           0.75       227
   macro avg       0.69      0.72      0.70       227
weighted avg       0.77      0.75      0.76       227
```

	model	accuracy	precision	F1-score	Sensitivity	Specificity	AUC
0	Decision Tree	0.75	0.69	0.7	0.65	0.79	0.72

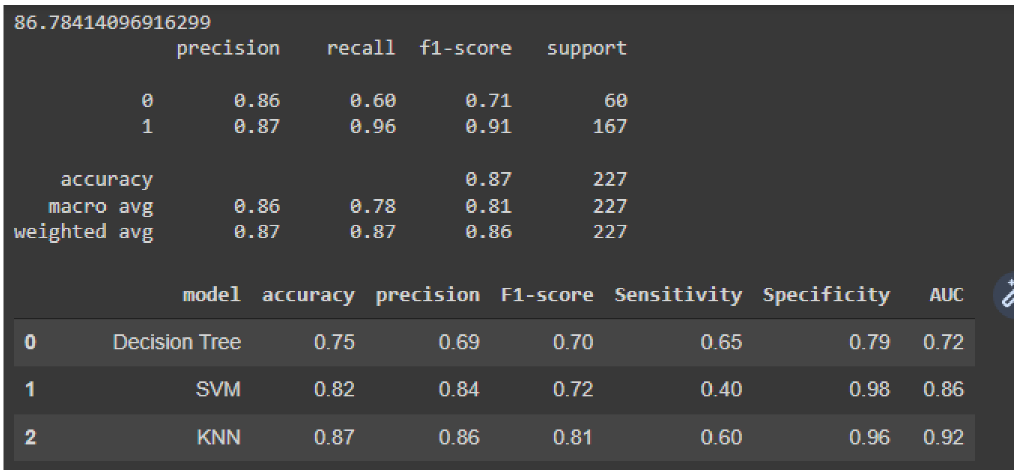
Fig 5.6.1 Decision trees



```
82.37885462555066
              precision    recall  f1-score   support

           0       0.86      0.40      0.55        60
           1       0.82      0.98      0.89       167

    accuracy                           0.82       227
   macro avg       0.84      0.69      0.72       227
weighted avg       0.83      0.82      0.80       227
```

	model	accuracy	precision	F1-score	Sensitivity	Specificity	AUC
0	Decision Tree	0.75	0.69	0.70	0.65	0.79	0.72
1	SVM	0.82	0.84	0.72	0.40	0.98	0.86

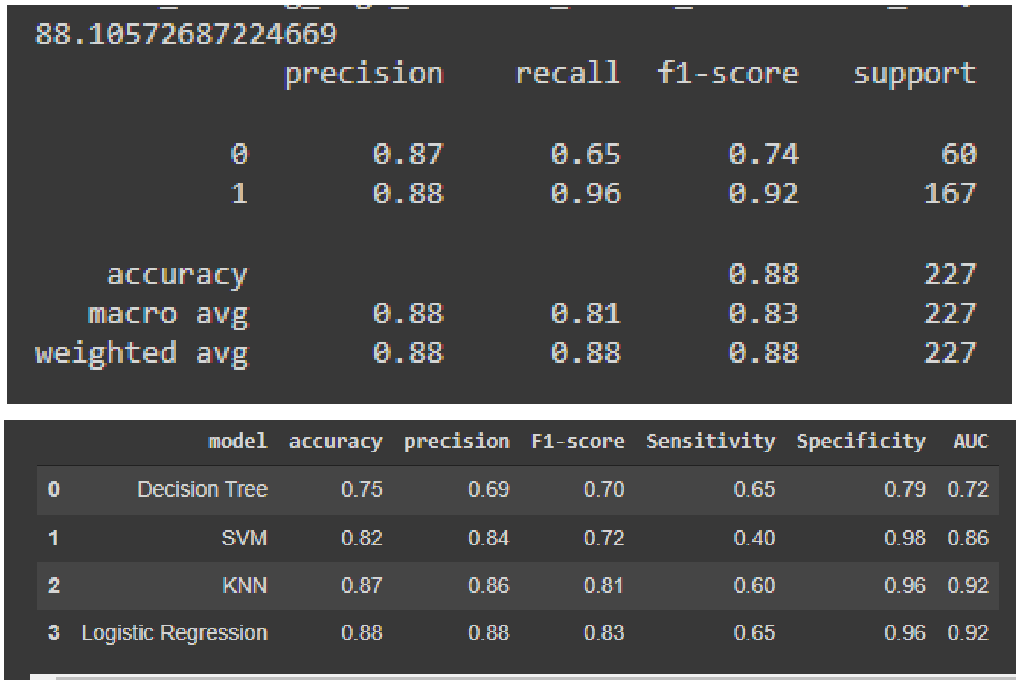
Fig 5.6.2 Support vector Machines(SVM)



```
86.78414096916299
              precision    recall  f1-score   support

           0       0.86      0.60      0.71        60
           1       0.87      0.96      0.91       167

    accuracy                           0.87       227
   macro avg       0.86      0.78      0.81       227
weighted avg       0.87      0.87      0.86       227
```

	model	accuracy	precision	F1-score	Sensitivity	Specificity	AUC
0	Decision Tree	0.75	0.69	0.70	0.65	0.79	0.72
1	SVM	0.82	0.84	0.72	0.40	0.98	0.86
2	KNN	0.87	0.86	0.81	0.60	0.96	0.92

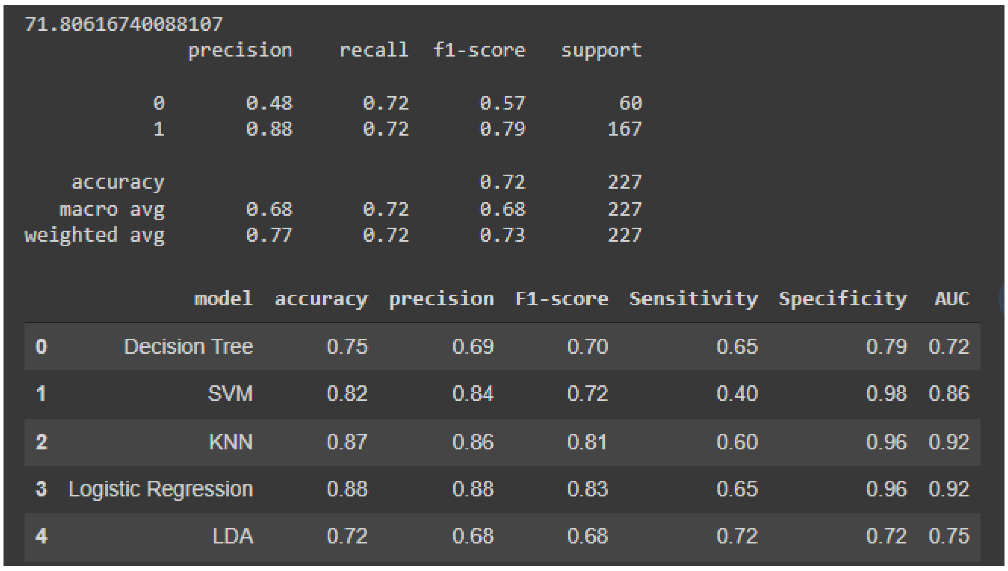
Fig 5.6.3 K nearest neighbours(KNN)



```
88.10572687224669
              precision    recall  f1-score   support

           0       0.87      0.65      0.74        60
           1       0.88      0.96      0.92       167

    accuracy                           0.88       227
   macro avg       0.88      0.81      0.83       227
weighted avg       0.88      0.88      0.88       227
```

	model	accuracy	precision	F1-score	Sensitivity	Specificity	AUC
0	Decision Tree	0.75	0.69	0.70	0.65	0.79	0.72
1	SVM	0.82	0.84	0.72	0.40	0.98	0.86
2	KNN	0.87	0.86	0.81	0.60	0.96	0.92
3	Logistic Regression	0.88	0.88	0.83	0.65	0.96	0.92

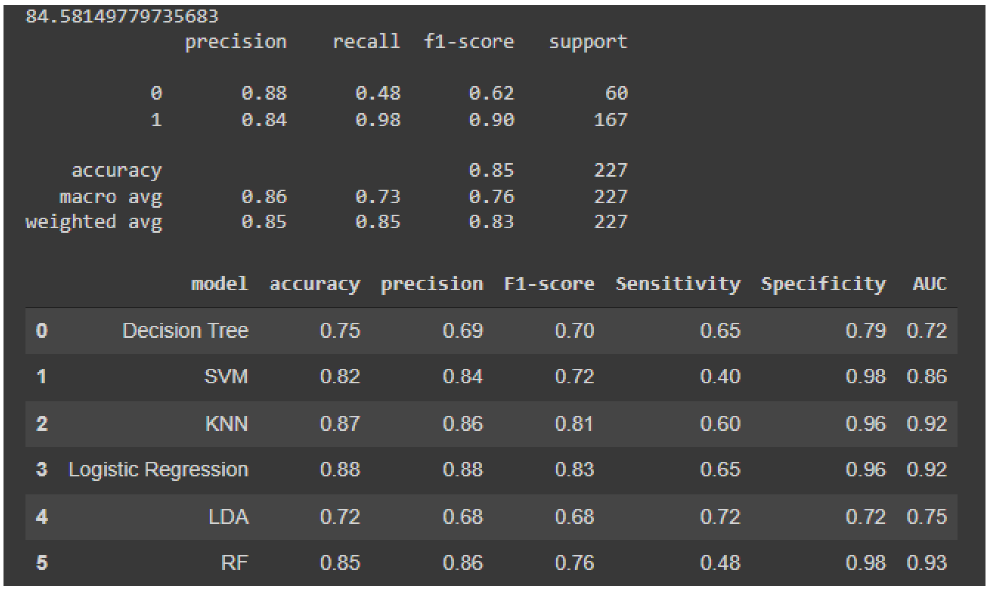
Fig 5.6.4 Logistic Regression



```
71.80616740088107
              precision    recall  f1-score   support

           0       0.48      0.72      0.57        60
           1       0.88      0.72      0.79       167

    accuracy                           0.72       227
   macro avg       0.68      0.72      0.68       227
weighted avg       0.77      0.72      0.73       227
```

	model	accuracy	precision	F1-score	Sensitivity	Specificity	AUC
0	Decision Tree	0.75	0.69	0.70	0.65	0.79	0.72
1	SVM	0.82	0.84	0.72	0.40	0.98	0.86
2	KNN	0.87	0.86	0.81	0.60	0.96	0.92
3	Logistic Regression	0.88	0.88	0.83	0.65	0.96	0.92
4	LDA	0.72	0.68	0.68	0.72	0.72	0.75

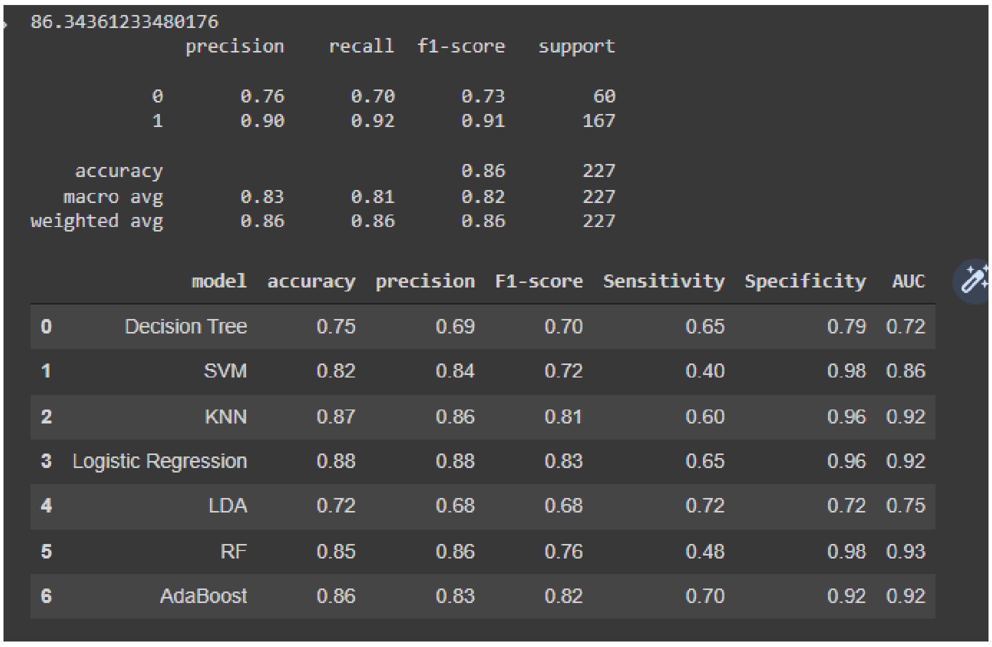
Fig 5.6.5 Linear Discriminant Analysis(LDA)



```
84.58149779735683
              precision    recall  f1-score   support

           0       0.88      0.48      0.62        60
           1       0.84      0.98      0.90       167

    accuracy                           0.85       227
   macro avg       0.86      0.73      0.76       227
weighted avg       0.85      0.85      0.83       227
```

	model	accuracy	precision	F1-score	Sensitivity	Specificity	AUC
0	Decision Tree	0.75	0.69	0.70	0.65	0.79	0.72
1	SVM	0.82	0.84	0.72	0.40	0.98	0.86
2	KNN	0.87	0.86	0.81	0.60	0.96	0.92
3	Logistic Regression	0.88	0.88	0.83	0.65	0.96	0.92
4	LDA	0.72	0.68	0.68	0.72	0.72	0.75
5	RF	0.85	0.86	0.76	0.48	0.98	0.93

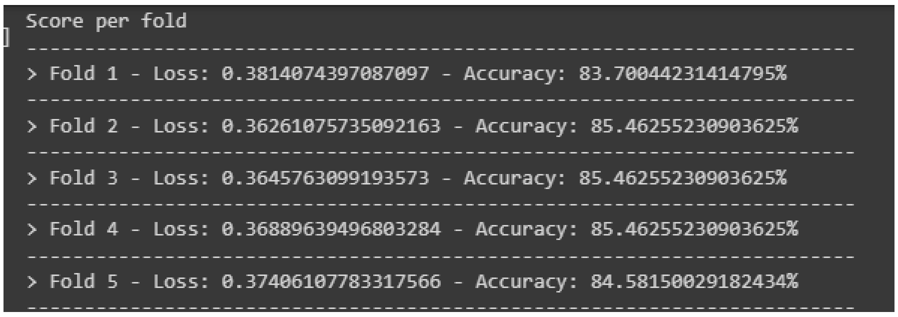
Fig 5.6.6 Random Forest Classifier(RF)



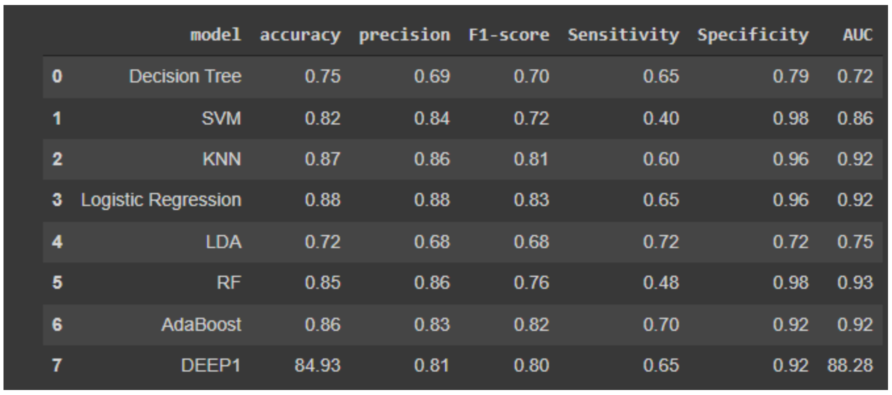
```
86.34361233480176
              precision    recall  f1-score   support

           0       0.76      0.70      0.73        60
           1       0.90      0.92      0.91       167

    accuracy                           0.86       227
   macro avg       0.83      0.81      0.82       227
weighted avg       0.86      0.86      0.86       227
```

	model	accuracy	precision	F1-score	Sensitivity	Specificity	AUC
0	Decision Tree	0.75	0.69	0.70	0.65	0.79	0.72
1	SVM	0.82	0.84	0.72	0.40	0.98	0.86
2	KNN	0.87	0.86	0.81	0.60	0.96	0.92
3	Logistic Regression	0.88	0.88	0.83	0.65	0.96	0.92
4	LDA	0.72	0.68	0.68	0.72	0.72	0.75
5	RF	0.85	0.86	0.76	0.48	0.98	0.93
6	AdaBoost	0.86	0.83	0.82	0.70	0.92	0.92

Fig 5.6.7 Ada Boost



```
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.3814074397087097 - Accuracy: 83.70044231414795%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.36261075735092163 - Accuracy: 85.46255230903625%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.3645763099193573 - Accuracy: 85.46255230903625%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.36889639496803284 - Accuracy: 85.46255230903625%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.37406107783317566 - Accuracy: 84.58150029182434%
------------------------------------------------------------------------
```



	model	accuracy	precision	F1-score	Sensitivity	Specificity	AUC
0	Decision Tree	0.75	0.69	0.70	0.65	0.79	0.72
1	SVM	0.82	0.84	0.72	0.40	0.98	0.86
2	KNN	0.87	0.86	0.81	0.60	0.96	0.92
3	Logistic Regression	0.88	0.88	0.83	0.65	0.96	0.92
4	LDA	0.72	0.68	0.68	0.72	0.72	0.75
5	RF	0.85	0.86	0.76	0.48	0.98	0.93
6	AdaBoost	0.86	0.83	0.82	0.70	0.92	0.92
7	DEEP1	84.93	0.81	0.80	0.65	0.92	88.28

Fig 5.6.8.1 Deep1

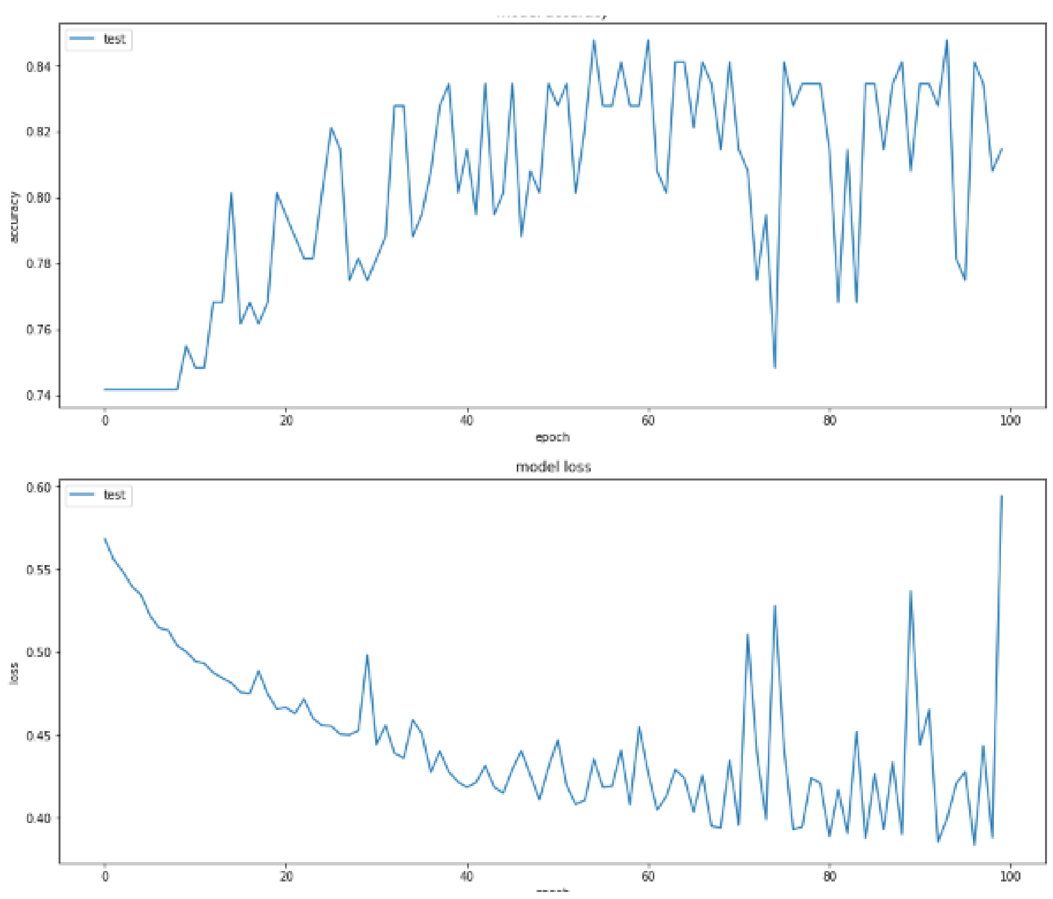
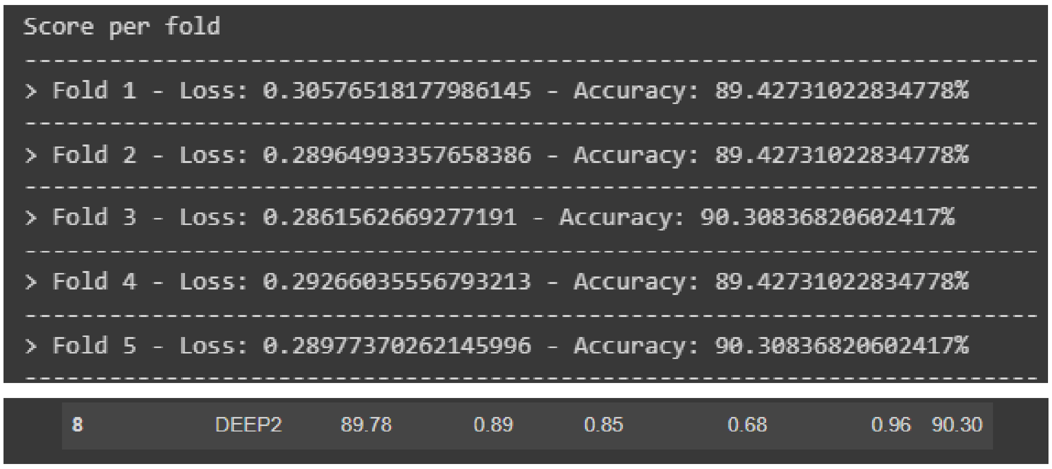


Fig 5.6.8.1 Deep-1 Accracy-loss curves



```
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.30576518177986145 - Accuracy: 89.42731022834778%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.28964993357658386 - Accuracy: 89.42731022834778%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.2861562669277191 - Accuracy: 90.30836820602417%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.29266035556793213 - Accuracy: 89.42731022834778%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.28977370262145996 - Accuracy: 90.30836820602417%
------------------------------------------------------------------------
```

```
8          DEEP2      89.78      0.89      0.85      0.68      0.96   90.30
```

Fig 5.6.8.2 Deep2

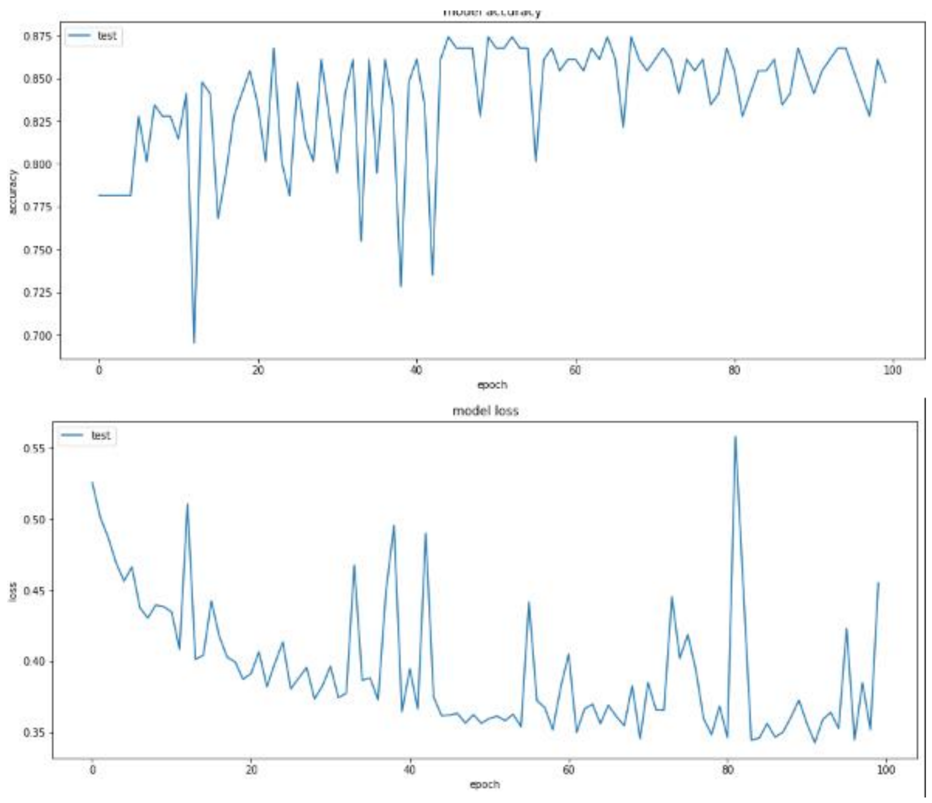
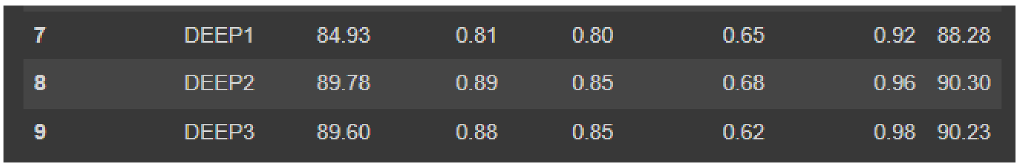
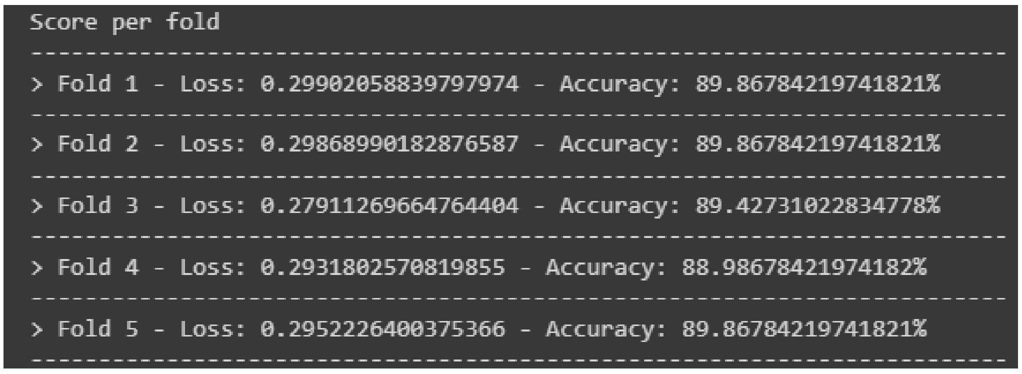


Fig 5.6.8.2 Deep-2 Accracy-loss curves



7	DEEP1	84.93	0.81	0.80	0.65	0.92	88.28
8	DEEP2	89.78	0.89	0.85	0.68	0.96	90.30
9	DEEP3	89.60	0.88	0.85	0.62	0.98	90.23



```
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.29902058839797974 - Accuracy: 89.86784219741821%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.29868990182876587 - Accuracy: 89.86784219741821%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.27911269664764404 - Accuracy: 89.42731022834778%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.2931802570819855 - Accuracy: 88.98678421974182%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.2952226400375366 - Accuracy: 89.86784219741821%
------------------------------------------------------------------------
```

Fig 5.6.8.3 Deep3

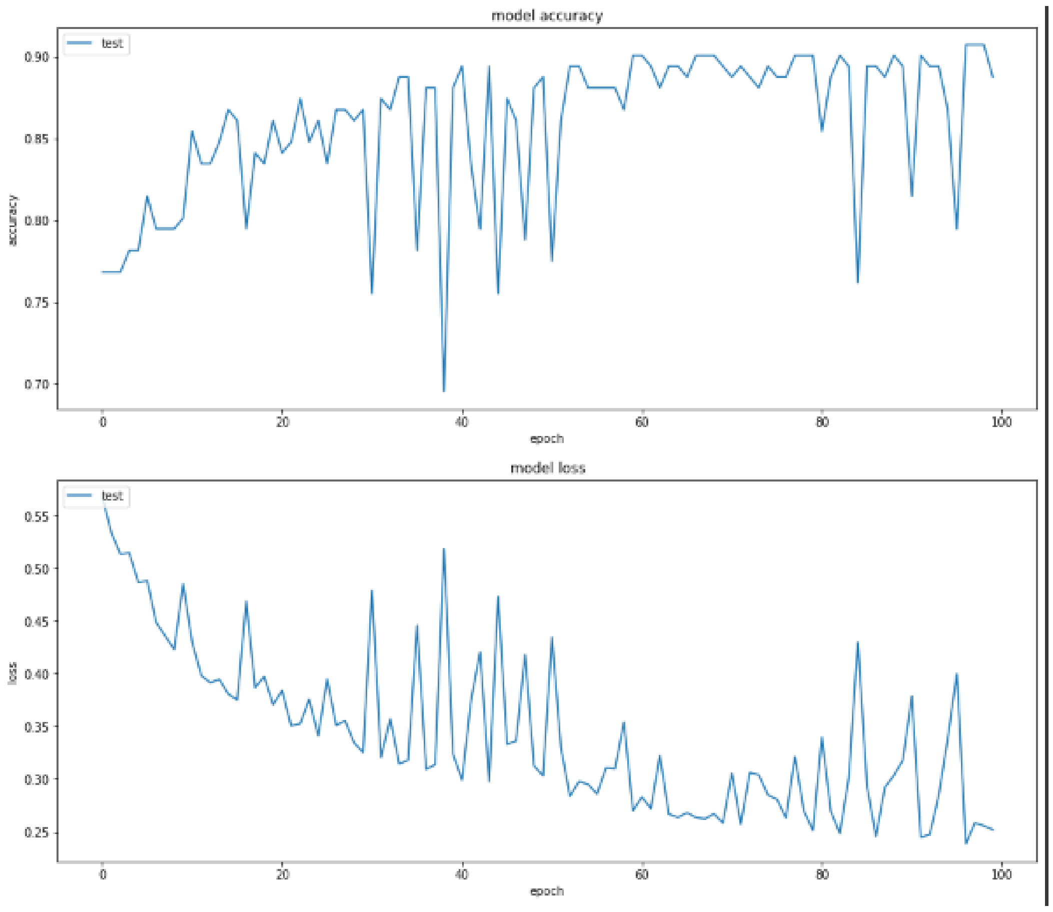
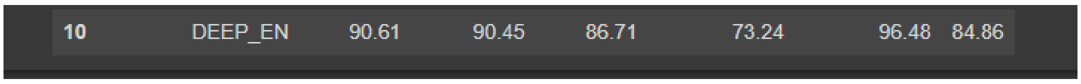
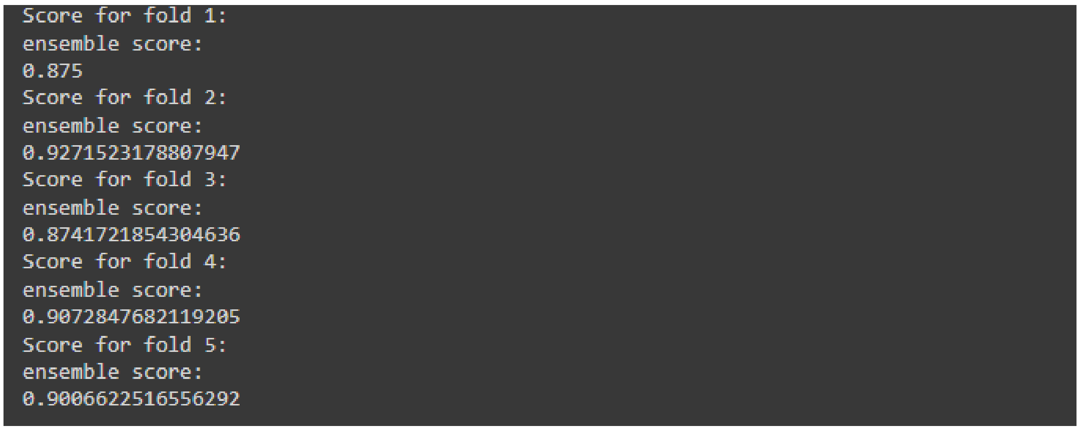


Fig 5.6.8.3 Deep-3 Accracy-loss curves



10	DEEP_EN	90.61	90.45	86.71	73.24	96.48	84.86



```
Score for fold 1:
ensemble score:
0.875
Score for fold 2:
ensemble score:
0.9271523178807947
Score for fold 3:
ensemble score:
0.8741721854304636
Score for fold 4:
ensemble score:
0.9072847682119205
Score for fold 5:
ensemble score:
0.9006622516556292
```

Fig 5.6.9 Ensemble Learning

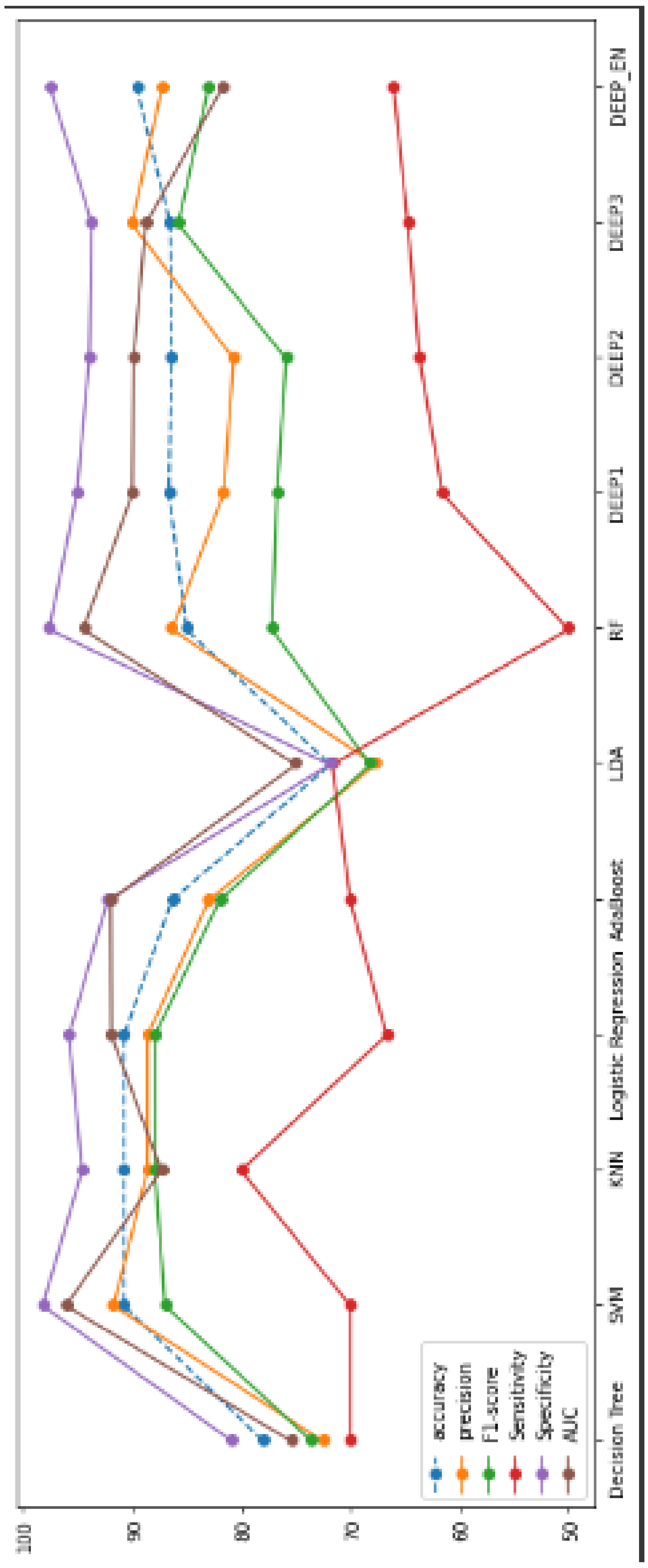


Fig 5.7 Comparision

5. CONCLUSION AND FUTURE SCOPE

Conclusion:

Parkinson's disease is one of the most common neurodegenerative diseases affecting the aging population. Awareness of the disease manifestations, the treatments, and the progressive long-term course of the disease is necessary for the optimal management of the cases. Tremendous progress has been made in understanding the neuropathology of PD and its progression throughout the nervous system. A deep learning model to automatically discriminate normal individuals and patients affected by PD based on vocal feature sets. The proposed deep learning model showed good detection capacity by reaching an accuracy of 90 %. This is mainly due to the desirable characteristics of the deep learning model in learning linear and nonlinear features from PD data without the need for hand-crafted features extraction. Results showed that the designed deep learning offers superior detection performance compared to the considered machine learning models.

Future Work:

New surgical approaches and gene therapies for Parkinson's are also currently being tested. Unfortunately, it takes a very long time for new therapies to go through the testing and approval process, and not all novel treatments prove successful.

The future work of this project is to gather the data related to premotor features To detect the early PD, we consider thirteen features based on information from previous similar studies as provided below. • RBDSQ score. The REM sleep Behavior Disorder Screening Questionnaire (RBDSQ) is a specific questionnaire for rapid eye movement behavior disorder (RBD) and create and evaluate the models. This helps in early detection of parkinsons disease.

REFERENCES

[1] H. Gunduz, “Deep learning-based parkinson’s disease classification using vocal feature sets,” IEEE Access, vol. 7, pp. 115 540–115 551, 2019.

[2] A. Tsanas, M. A. Little, P. E. McSharry, and L. O. Ramig, “Accurate telemonitoring of parkinson’s disease progression by noninvasive speech tests,” IEEE transactions on Biomedical Engineering, vol. 57, no. 4, pp. 884–893, 2009.

[3] D. Braga, A. M. Madureira, L. Coelho, and R. Ajith, “Automatic detection of parkinson’s disease based on acoustic analysis of speech,” Engineering Applications of Artificial Intelligence, vol. 77, pp. 148–158, 2019.

[4] I. El Maachi, G.-A. Bilodeau, and W. Bouachir, “Deep 1d-convnet for accurate parkinson disease detection and severity prediction from gait,” Expert Systems with Applications, vol. 143, p. 113075, 2020.

[5] U. Gupta, H. Bansal, and D. Joshi, “An improved sex-specific and age dependent classification model for parkinson’s diagnosis using handwriting measurement,” Computer Methods and Programs in Biomedicine, vol. 189, p. 105305, 2020.

[6] A. Wagner, N. Fixler, and Y. S. Resheff, “A wavelet-based approach to monitoring parkinson’s disease symptoms,” in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017, pp. 5980–5984.

[7] T. Arroyo-Gallego, M. J. Ledesma-Carbayo, A. Sánchez-Ferro, I. Butterworth, C. S. Mendoza, M. Matarazzo, P. Montero, R. López-Blanco, V. Puertas-Martin, R. Trincado et al., “Detection of motor impairment in parkinson’s disease via mobile touchscreen typing,” IEEE Transactions on Biomedical Engineering, vol. 64, no. 9, pp. 1994–2002, 2017.

[8] M. Little, P. McSharry, E. Hunter, J. Spielman, and L. Ramig, “Suitability of dysphonia measurements for telemonitoring of parkinson’s disease,” Nature Precedings, pp. 1–1, 2008

APPENDIX

Code:

```
from keras.models import load_model
n_members = 3
models = list()
for i in range(n_members):
  #best_deep3model.hd5
  filename = 'best_deep' + str(i + 1) + 'model.hd5'
  model = load_model(filename)
  models.append(model)
def ensemble_predictions(models, X8_test):
  outcomes = [model.predict(X8_test) for model in models]
  outcomes = np.array(outcomes)
  ensemble_prediction = np.round(np.average(outcomes, axis=0),0)
  return ensemble_prediction
def evaluate_n_models(models, n_models, X8_test, Y8_test):

  subset = models[:n_models]
  print(subset)
  outcome = ensemble_predictions(subset, X8_test)
  return accuracy_score(Y8_test, outcome)

single_scores, ensemble_scores = list(), list()

for i in range(1, len(models)+1):
  ensemble_score = evaluate_n_models(models, i, X8_test, Y8_test)
  single_score = models[i-1].evaluate(X8_test, Y8_test)
  ensemble_scores.append(ensemble_score)
  single_scores.append(single_score[1])

print("Individual scores")
print(single_scores)
print("ensemble scores")
print(ensemble_scores)

x_axis = [i for i in range(1, len(models)+1)]
pyplot.plot(x_axis, single_scores, marker='o', linestyle='None')
pyplot.plot(x_axis, ensemble_scores, marker='o')
pyplot.show()
```