

Deep Learning based prediction and monitoring Parkinson's Disease using voice data

line 1: 1st Given Name Surname
line 2: *dept. name of organization*
(of Affiliation)
line 3: *name of organization*
(of Affiliation)
line 4: City, Country
line 5: email address or ORCID

line 1: 2nd Given Name Surname
line 2: *dept. name of organization*
(of Affiliation)
line 3: *name of organization*
(of Affiliation)
line 4: City, Country
line 5: email address or ORCID

line 1: 3rd Given Name Surname
line 2: *dept. name of organization*
(of Affiliation)
line 3: *name of organization*
(of Affiliation)
line 4: City, Country
line 5: email address or ORCID

line 1: 4th Given Name Surname
line 2: *dept. name of organization*
(of Affiliation)
line 3: *name of organization*
(of Affiliation)
line 4: City, Country
line 5: email address or ORCID

line 1: 5th Given Name Surname
line 2: *dept. name of organization*
(of Affiliation)
line 3: *name of organization*
(of Affiliation)
line 4: City, Country
line 5: email address or ORCID

line 1: 6th Given Name Surname
line 2: *dept. name of organization*
(of Affiliation)
line 3: *name of organization*
(of Affiliation)
line 4: City, Country
line 5: email address or ORCID

Abstract—With recent progress in digitized data acquisition, machine learning and computing infrastructures, the application of AI has stretched its wings far and wide. One such field of application is medical science. Artificial intelligence is gradually changing traditional medical practices and approaches to various problems. Even when medical science is growing **with** leaps and bounds there are still some problems which are beyond the province of human expertise. The infamous name of Parkinson's disease haunts the human race even in this era of rapid scientific advancements since it has no cure to date. Thus early detection of the disease is very necessary to be better safe than sorry. PD is a progressive neurological disorder which affects the Central Nervous System (CNS) preventing the proper functioning of the same. This results in tremors, stiffness, slow movements, difficulty in balance and coordination, speech disorders and many more. In this study, we have gathered data on the speech of affected and healthy people. We have analysed the data using acoustic features such as jitter, shimmer, intensity, pitch, etc. Various machine learning models were trained and tested on the analysed and processed data which provided some insightful results. The following paper consists of a detailed comparative study of different ML models namely KNN, SVC, Decision Tree, Random Forest, Naïve Bayes, Logistic Regression, XGBoost and Gradient Boosting along with modern neural network techniques like ANN, MLP and ResNet which will help us in the early and accurate detection of this deadly disease.

Keywords—Artificial Intelligence, Parkinson's Disease, Machine Learning models, Neural Networks, Comparative analysis, Dimensionality reduction.

I. INTRODUCTION

Parkinson's disease is a chronic and progressive movement disorder that affects the human brain and central nervous system.

It is caused by the degeneration of dopamine-releasing neurons in the brain. Dopamine is a key neurotransmitter responsible for controlling our motor movements. With the decrease in dopamine the person gradually loses motor abilities resulting in tremors, inability to control movements etc.

To this day the cause of Parkinson's disease is not fully understood and hence no complete cure is also available.

The disease usually progresses slowly and hence the physical symptoms are usually not visible until the patient is already in the advanced stages of the disease.

This is where modern medical science with the help of machine learning can intervene, with early and automatic detection of Parkinson's disease, patients can undergo different supportive treatments that can reduce the symptoms and slow the progression of the disease.

In this article, we have implemented eight (08) traditional machine learning and three (03) neural network models namely, KNN Classifier, SVM Classifier, Decision Tree Classifier, Random Forest Classifier, Naive Bayes Classifier, Logistic Regression Classifier, XGBoost Classifier, Gradient Boost Classifier, Artificial Neural Network, Multi-Layer Perceptron and ResNet to classify whether a patient is suffering from Parkinson's disease or not based on their speech patterns and have done a thorough performance analysis of all the models implemented by testing each of them using a dataset.

II. RELATED WORK

III. DATA SET

IV. DATA PREPROCESSING

The majority of the datasets available are called Raw Datasets. The reason they are labelled raw datasets is that they contain many anomalies which further implies that, the dataset is not suitable to train our Machine Learning and Deep Learning Models. The dataset needs to be nurtured and structured via a series of steps commonly called the Data Preprocessing Steps, to make it suitable for the training purpose of the proposed models.

The anomalies associated with the Raw Dataset are listed as follows –

- 1) Missing Values
- 2) Skewness
- 3) Kurtosis

4) Outliers

5) Feature Selection or Data Reduction

Once the dataset is Preprocessed, it improves the proposed models' overall accuracy, precision, recall and F1-Score.

A. Missing Values

In the dataset, some values might be missing from some or all of the attributes. These missing values are represented with a 'Blank', 'null' and 'NaN' (Not a Number). The standard procedures for the effective removal/replacement of the missing values are listed below –

- 1) Drop down the entire row column containing at least one missing value
- 2) Replace the missing value(s) with the median of that attribute if applicable
- 3) Replace the missing value(s) with the mean of that attribute if applicable
- 4) Replace the missing value(s) with the Forward Fill technique
- 5) Replace the missing value(s) with the Backward Fill technique

The forward Fill technique puts in the last known value before the first missing cell of that particular attribute, and each missing cell of that particular attribute is then replaced by the same last known value. Think of it as, surfing through each row from top to bottom, identifying the last known cell value before the missing cell and then replacing all the missing cells with that value.

The Backward Fill technique puts in the last known value before the first missing cell of that particular attribute while going backwards, and each missing cell of that particular attribute is then replaced by the same last known value. Think of it as, surfing through each row from bottom to top, identifying the last known cell value before the missing cell and then replacing all the missing cells with that value.

B. Skewness

Before knowing what skewness is, the concept of Normal Distribution is important. Normal Distribution or commonly termed a Gaussian Distribution is a type of continuous probability distribution, with a bell-shaped curved. The probability density function can be defined as follows:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (1)$$

where, $f(x)$ = probability density function,
 σ = standard deviation, and
 μ = mean

A typical symmetric Gaussian Distribution looks like follows:

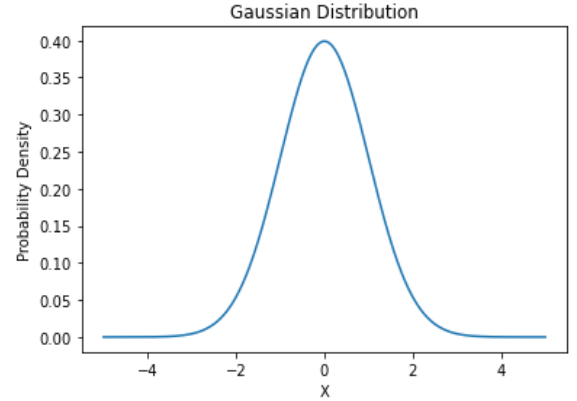


Fig. 1. Gaussian Distribution or Bell Curve

Now, skewness is a measure of how much the Gaussian Distribution is skewed or shifted from its normal position towards left or right and accordingly called Right Skewed or Left Skewed respectively. The Right Skewed value is represented using a positive value and for Left Skewed, it is represented using a negative value. The skewness can be calculated using the following formula:

$$\mu_3 = \frac{\sum_i^N (X_i - \bar{X})^3}{(N-1) \cdot \sigma^3} \quad (2)$$

Where μ_3 is the skewness, N is the number of variables in the distribution, X_i random variable, \bar{X} is the mean of the distribution and σ is the standard deviation.

The industrial accepted skewness value ranges from -1.5 to +1.5. Anything greater than +1.5 or smaller than -1.5 needs to be reduced to fit within the range. For this we can apply the following methods:

- 1) Box-Cox Transformation – It can be calculated using the following function

$$y(\lambda) = \begin{cases} \frac{(y+\lambda_2)^{\lambda_1}-1}{\lambda_1}, & \text{if } \lambda_1 \neq 0; \\ \log(y + \lambda_2), & \text{if } \lambda_1 = 0. \end{cases} \quad (3)$$

where $<*$ to be inserted*

- 2) Cube root Transformation - It can be calculated using the following function

$$f(x) = \sqrt[3]{x} \quad (4)$$

- 3) Log Transformation - It can be calculated using the following function

$$f(x) = \log(x) \quad (5)$$

A normally distributed attribute helps to increase the overall accuracy, precision, recall and F1-Score of the proposed models.

C. Kurtosis

It is a measure of how much the data clusters near the tails or the head of the curve. The Kurtosis value can be obtained using the stated equation:

$$Kurtosis = \frac{\sum(x_i - \bar{x})^4}{nS^4} \quad (6)$$

Where \bar{x} is the mean of the given data, S is the standard deviation of the data and n is the total number of observations.

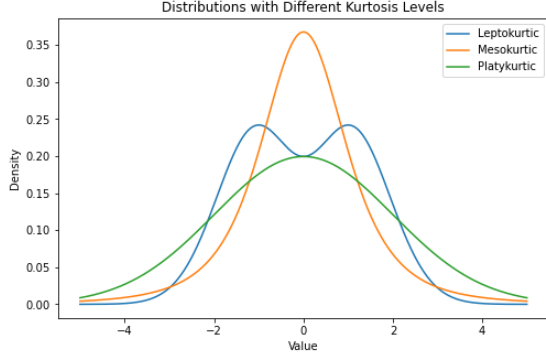


Fig. 2. Three types of Kurtosis

Based on this, there are three types of kurtosis:

- 1) *Mesokurtic* – The data here is moderately spread, with a medium peak.
- 2) *Leptokurtic* – The majority of the data is concentrated near the tails, leading to a pointed peak.
- 3) *Platykurtic* – The data is evenly spread-out, implying a flat line almost with a little bump.

The permissible and the industrial standard accepted value is 3. Anything less than 3 or greater than 3 is not permissible.

D. Outliers

To put it in simple words, Outlier means the odd one out. From Fig. 3 it is evident that the marked points have too much deviated from the rest of the pack. These points are labelled as Outliers. The best method/practice to find out the outlier of the attribute is the visual representation. From the distribution plot, the outliers can be identified easily.

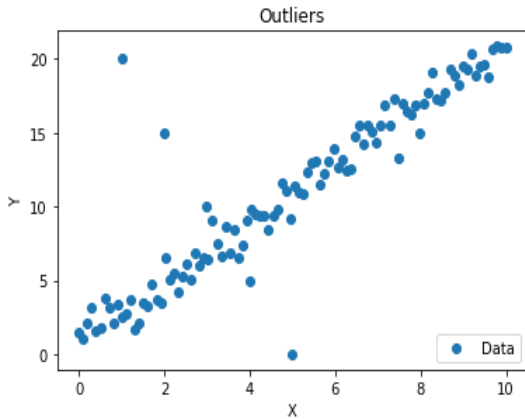


Fig. 3. Outliers present in a Linear Regression Plot

It is essential to remove or replace those outliers because they can interfere with the efficiency, accuracy, precision, recall and F1-Score of the proposed models. The industry-accepted and effective method to remove or replace those points is called Interquartile Range Function (IQR).

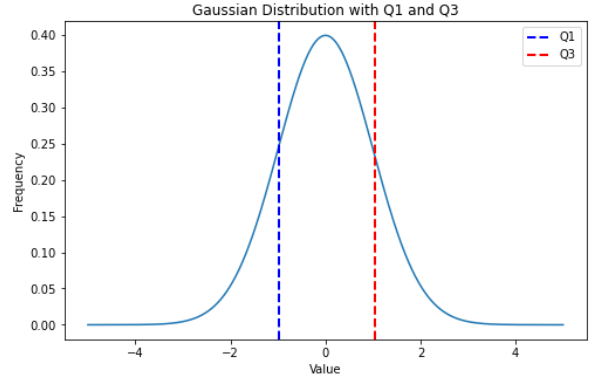


Fig. 4. Interquartile Range of a Gaussian Distribution

IQR is a statistical dispersion that represents the range from the first quartile (Q_1) to the third quartile (Q_3), implying covering 50% of the distribution area. The IQR is defined as:

$$IQR = Q_3 - Q_1 \quad (7)$$

The values lying within the IQR are considered to be in the safe zone. Any points not within the range are either dropped, if the dataset is very large, or the points are replaced by the median, which is a safer option and works well for large and relatively small datasets.

E. Feature Selection or Data Reduction

A dataset comprises several attributes which are responsible for obtaining vital information to learn and understand the pattern and relationship between the input and output values. However, not all attributes contribute the same amount of information. Dropping down those specific attributes does not harm the goal of seeking the relationship between input and output variables and also reduces the training time of our proposed ML and DL models.

There are various techniques for feature selection, including:

- 1) *Filter methods* involve selecting features based on statistical tests such as correlation, chi-square, or mutual information.
- 2) *Wrapper methods*: This involves selecting features based on the performance of a machine learning model using different subsets of features.
- 3) *Embedded methods*: This involves selecting features during the training process of a machine learning model, such as regularization techniques.
- 4) *Dimensionality reduction*: This involves transforming the original dataset into a lower-dimensional space, such as principal component analysis (PCA) or linear discriminant analysis (LDA).

When selecting features, it is important to consider the trade-off between the accuracy and complexity of the model, as well as the interpretability of the features. Additionally, it is important to use a validation set or cross-validation to ensure that the selected features generalize well to new data.

Principal Component Analysis or PCA in short is a widely used statistical technique used to reduce the dimensionality of a dataset. Dimension Reduction is a very important problem in machine learning as for smaller datasets the number of attributes can be higher than the number of instances which can lead to overfitting. PCA can be used to solve this problem by transforming the dataset into a different coordinate system where the dataset can be represented in a lower dimensional space.

PCA works by representing the original dataset in a new coordinate system where each dimension is a linear combination of the original attributes. This new coordinate system is selected in such a way that the first component captures the maximum variance in the dataset, the second component captures the maximum variance that is orthogonal to the first component and so on.

To perform PCA we first standardize the data points by subtracting the mean and divide by the standard deviation. We then compute the covariance matrix of the dataset which measures the relative variance of the attributes of the dataset.

Next we compute the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors represent the directions of the original feature space along which the variance of the data points are the highest, the eigenvalues represent how much the data points vary.

Finally we select the highest N eigenvalues and the corresponding eigenvectors and use them as the basis of the new coordinate system. We then project the data along the new coordinate system to obtain the lower dimensional representation of the dataset.

The algorithm for computing PCA can be represented as:

- 1) *Standardize the data by subtracting the mean and dividing it by the standard deviation.*

$$X_{STD} = \frac{(X - X_{Mean})}{Standard\ Deviation(X)} \quad (8)$$

Where

- 2) *Compute the covariance matrix of the standardized dataset.*
- 3) *Compute the eigenvalues and eigenvectors of the covariance matrix.*
- 4) *Sort the eigenvalues and eigenvectors according to the eigenvalues in descending order*
- 5) *Select N largest eigenvalues and corresponding eigenvectors and use them as the basis for the new coordinate system to represent the dataset in a lower dimensional feature space.*

V. MACHINE LEARNING MODELS

A. K-Nearest Neighbors Classifier

K-nearest neighbours classifier is a supervised learning technique which is non-parametric i.e. it makes no assumptions about the underlying distributions of the data. The working principle of the algorithm is the prediction of the classes based on some similarity measures (generally Euclidian distance). The new data point is classified by observing the k nearest data points around it and the most common class among the k neighbours is assigned the class to the new data point.

Euclidian distance is the measure of distance between two points in a Cartesian coordinate system. For instance, let us assume two data points M and N having coordinates (x_m, y_m) and (x_n, y_n) respectively. Then the Euclidian distance between the points can be calculated as:

$$D(M, N) = \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2} \quad (9)$$

For classification using the knn algorithm, the dataset is first split into the training and testing part. Then for each new data point, the k nearest neighbours are considered and assigned the most common class among the neighbours.

The number of neighbours i.e. K is the hyperparameter that needs to be adjusted carefully to avoid problems like overfitting or underfitting.

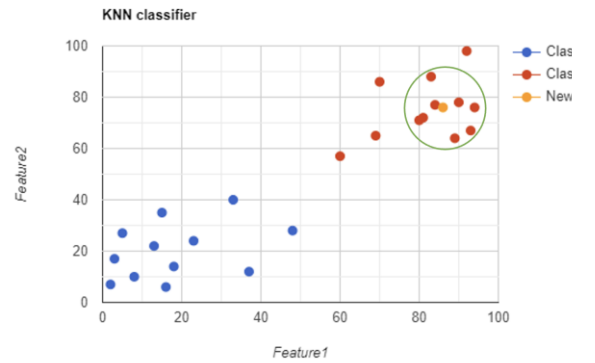


Fig. 5. Working mechanism of KNN

B. Support Vector Classifier

Support vector classifier (SVC) is a classification technique used for supervised learning problems. SVC finds out the optimal separating hyperplane also called the decision boundary that separates the data points belonging to different classes. The goal of SVC is to maximise the distance between data points close to the margin and the separating hyperplane. The data points closest to the margin are called support vectors and hence the name.

SVC has a few important parameters which need to be tuned carefully for proper results:

- 1) **c :** Regularization parameter with default value 1.0, type float.
- 2) **kernel:** Out of several choices for the kernels the appropriate one must be chosen (more on kernels later).

Popular choices for kernels:

- 1) **Linear:** It calculates the dot product of input vectors.

$$L(x, x') = x \cdot x' \quad (10)$$

- 2) **Radial Basis Function (RBF):** It uses a Gaussian function to map the input vectors to an infinite dimensional feature space.

$$R(x, x') = e^{-\gamma \|x - x'\|^2} \quad (11)$$

- 3) **Polynomial:** It is a non-linear kernel that calculates the similarity between two vectors using the following relation:

$$P(x, x') = (\Upsilon * (x \cdot x') + \text{coef}_0)^d \quad (12)$$

- 4) **Sigmoid:** It uses a tan hyperbolic function to compute the similarity between the inputs. It is a non-linear kernel.

$$S(x, x') = \tanh(\Upsilon * (x \cdot x') + \text{coef}_0) \quad (13)$$

Where x, x' are the input vectors, Υ is the hyperparameter controlling the width of the Gaussian kernel, d is the hyperparameter degree and coef_0 is the coefficient hyperparameter.

The Support Vector machine uses hinge loss as a loss function. The goal of SVM is to minimise hinge loss. The loss function is given as:

$$l = \max(0, 1 - y^i(x^i - b)) \quad (14)$$

Where y^i and x^i are the i^{th} instances in the training set and b is the bias.

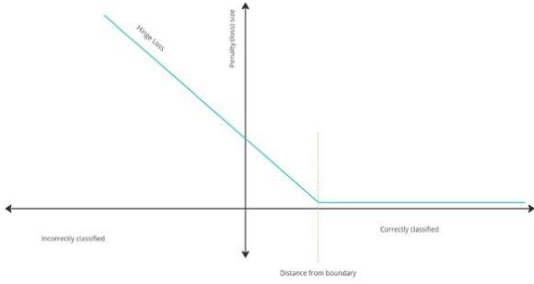


Fig. 6. Hinge Loss of SVM

C. Decision Tree Classifier

The decision tree classifier is a supervised learning technique used for the classification of input. It works by continuously separating the input space into several segments and making decisions based on certain criteria. This algorithm can be easily visualised as a tree of decisions which has consequences at each decision node.

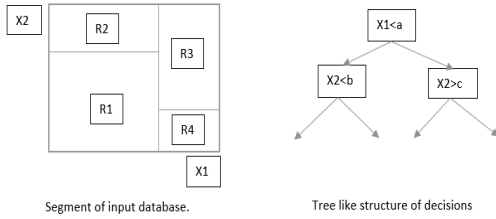


Fig. 7. Working Principle of Decision Tree

After building the tree, sometimes it tends to take a very complex form which has very high computational complexity. In order to reduce the computational load the tree is pruned which also avoids overfitting.

The decision tree algorithm uses various metrics to select the feature for the best split. Some of the commonly used metrics are:

- 1) **Information Gain:**

$$\text{Gain}(D, A) = \text{Info}(D) - \text{Info}(D|A) \quad (15)$$

$$\text{Info}(D) = -\sum_{i=1}^c (p_i * \log_2 p_i) \quad (16)$$

$$\text{Info}(D|A) = \sum_{i=1}^c \frac{|D_i|}{|D|} * \text{Info}(D_i) \quad (17)$$

- 2) **Gini Index:**

$$\text{Gini}(D) = 1 - \sum_{i=1}^c p_i^2 \quad (18)$$

Where D is the instance of the dataset, c is the number of possible classes and p_i is the probability that D_i belongs to class i .

D. Random Forest Classifier

Random forest classifier is a bit different classification algorithm. It is an ensemble learning technique which uses multiple decision trees to generate its output (a group of trees is called a forest).

Bagging of these individual trees is done to reduce the correlation between trees. This algorithm creates multiple decision trees where each tree constructed uses a different subset of features, selected with replacement also called Bootstrapping.

After all, the trees are constructed and output generated, the final output is obtained by taking the majority voting or the mean of all predictions. A random forest works similarly to the decision tree and hence is mathematically identical. However, prediction is generated as follows:

Final prediction = mode(Prediction of individual trees)

E. Naïve Bayes Classifier

Naïve Bayes classifier is a probabilistic machine learning algorithm used for classification. The backbone for this algorithm is the Bayesian network, also known as the Bayes network or Belief network.

A Bayesian network is a directed probability graph which depicts the conditional dependency of the random variables on one another. The Bayesian network has to be an acyclic graph.

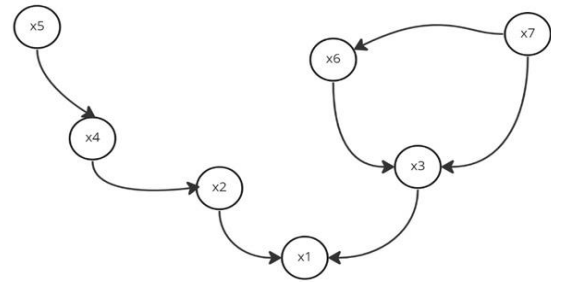


Fig. 8. Bayesian Network

Let us consider the above Bayesian network. This network shows that x_5 is an independent variable and x_4 depends on x_5 which is shown by an arrow from a random variable to the dependent variable. The joint probability distribution of the network can be factorized by using the chain rule. The factorization of the above network can be observed to be:

$$P = p(x_1|x_2, x_3).p(x_3|x_6, x_7).p(x_2|x_4).p(x_4|x_5).p(x_5).p(x_6|x_7).p(x_7) \quad (19)$$

From this graph we can also say, knowing x_2 (say) separates x_1 and x_4 i.e. x_1 and x_4 are conditionally independent if x_2 is known.

The Naïve Bayes classification algorithm uses the principle of Bayes theorem assuming that all the features are independent of each other and hence the name. The model calculates the probability of an instance belonging to a class based on the features associated with it. This is done by calculating the conditional probability of each feature given the class and then calculating the total probability using the Bayes theorem.

The instance having the highest probability of belonging to a class is classified to be a point in that class.

Conditional probability is calculated using Bayes theorem as follows:

$$P(C|X_1, X_2, X_3, \dots, X_n) = \frac{P(C)P(X_1|C)P(X_2|C)P(X_3|C)\dots P(X_n|C)}{P(X_1, X_2, X_3, \dots, X_n)} \quad (20)$$

Naive Bayes has different variations and can be used for both binary and multi-class classification.

F. Logistic Regression

Logistic regression is a classification algorithm for binary class problems using a statistical approach. This algorithm works very well where the prediction of the probability of events is involved. The logistic regression algorithm uses a logit transformation of the probability as follows:

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta x \quad (21)$$

Solving which gives us:

$$p(x) = \frac{1}{1-e^{-(\beta_0+\beta x)}} \quad (22)$$

Which is nothing but a sigmoid.

Where β_0 and β are the weights and $p(x)$ is the probability of instance x of the dataset.

Logistic regression uses the maximum likelihood estimation method to estimate the parameters of the linear combination and maximises the likelihood of the observed data given the model. An iterative algorithm such as gradient descent is used to estimate the parameters.

G. Extreme Gradient Boosting Classifier

Extreme Gradient boosting classifier, popularly known as XGBoost classifier, works using an ensemble of decision trees. This algorithm also uses multiple decision trees but in a sequential manner which is not the case in Random Forest. Here every subsequent tree tries to correct the errors of the previous tree.

The gradient-boosting algorithm is used to optimize an objective function:

$$\text{Obj} = \text{loss}(\text{actual}, \text{predicted}) + \Omega(\text{predicted}) \quad (23)$$

Where loss is the loss function of the algorithm and Ω is the regularization function used for penalization. Actual and predicted are the vectors of actual labels and predicted labels respectively.

The loss function used can be any of the popular choices such as Mean Squared Error (MSE), log loss, hinge loss etc. The mean squared error can be calculated as:

$$\text{SSE} = \sum_{i=1}^N (y_i - f(x_i))^2 \quad (24)$$

$$\text{MSE} = \frac{1}{N} * (\text{SSE}) \quad (25)$$

Where y_i is the actual label and $f(x_i)$ is the predicted label of the i^{th} instance of the dataset having N data points.

The regularization parameter can use various methods like L1 or L2 norms. Here Ω using L2 norm is represented as:

$$\Omega(\text{predicted}) = \gamma \sum w^2 \quad (26)$$

Where γ is the strength of regularization and w is the weight of the leaves.

XGBoost also uses pruning to decrease computational complexity and prevent overfitting.

H. Gradient Boosting Classifier

The Gradient Boosting algorithm uses the idea of combining various weak classifiers to make a strong classifier altogether. It iteratively keeps on adding new trees to the ensemble and updates the weights of the training examples based on the errors of the previous trees.

The algorithm first initialises the ensemble with the mean of the target variables (say). Then for each iteration i , fits a regression tree to the negative gradient of the loss function. The negative gradient can be calculated as:

$$r_{im} = -\left[\frac{\partial \text{Loss}(y_i, F_i)}{\partial F_i}\right]_{m-1} \quad (27)$$

Where $\text{Loss}()$ is the loss function used, y_n is the actual label and F_n is the predicted label for the n^{th} index of the training samples for the m^{th} iteration.

Then the optimal leaf weight is calculated for the new tree minimises the loss function.

$$\gamma_{jm} = \text{argmin}_{\gamma} \sum_i (\text{Loss}(y_i, F_i + \gamma_{jm} h_{jm}(x_i))) \quad (28)$$

Where h_{jm} is the prediction of the current tree for the i^{th} instance given that it falls in the j^{th} leaf.

The weights are then updated with the new tree and the optimal weight.

$$F_{mi} = F_{mi-1} + \eta \gamma_{jm} h_{jm}(x_i) \quad (29)$$

These steps are repeated until the desired result is achieved.

VI. NEURAL NETWORKS

A. Multilayer Perceptron Model

This is based on the concept of the Perceptron Learning Algorithm (PLA), used for binary classification with the help of a single separating hyperplane. Note that, PLA is only able to classify linearly separable data. For non-linearly separable data, it needs to be shifted to a higher dimension to make it linearly separable on that particular plane.

Multilayer Perceptron Classifier (MLP Classifier) can be thought of as a network of elementary processors arranged in a very specific manner to mimic the neural structure of a human brain, capable of learning the hidden patterns and connections between the input and output values to classify them accordingly.

The classical Perceptron (single layer) can find the separating hyperplane for linearly separable data. The first layer is called the Input Layer, comprising d number of nodes or processing units signifying d attributes present in the dataset. The final layer is called the Output Layer, comprising two nodes being a binary classifier. Each node of the Input Layer is connected to each node of the output layer, leading to a fully connected network. Refer to Fig. 9.

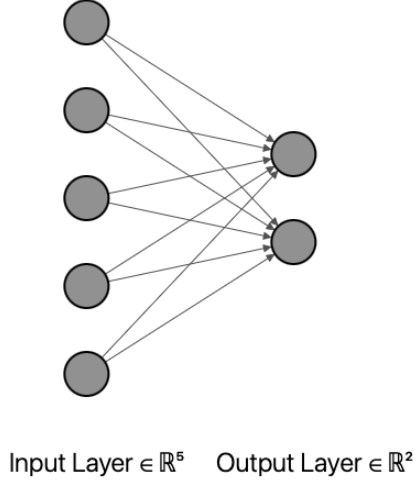


Fig. 9. Neural Network

Each Input Layer node takes in the value from each attribute. All the edges connected from the Input Layer to the Output Layer are assigned random weights. So, the total input of each node of the Output Layer is defined by

$$u_{nj} = \sum_{i=0}^d x_{ni} w_{ji} \quad (30)$$

Where $j = 1, 2, \dots, c$ implying c number of classes, u_{nj} is the total input from the input layer to the j^{th} node of the output layer, x_{ni} is the i^{th} feature of the input layer, w_{ji} is the weight associated between the j^{th} node of the output layer and i^{th} node of the input layer for the n^{th} instance of the dataset.

The value of $u_{nj} \in (-\infty, \infty)$, but the actual output has a value of 1 or 0, implying Class A and Class B respectively. Thus, a mapping is required to map the value of u_{nj} to $\{1, 0\}$ using a function. This function is called the Activation Function which maps the output value of the output node to $\{-1, 1\}$. The most effective and commonly used Activation function is the sigmoid function.

$$v_{ji} = \phi(u_{ji}) \quad (31)$$

Where, $j = 1, 2, \dots, c$, $i = 0, 1, 2, \dots, d$, v_{ji} is the output of the j^{th} node of the output layer and $\phi()$ is the activation function defined by

$$\phi(x) = \frac{1}{1 + e^{-ax}} \quad (32)$$

Where a is constant.

The learning technique associated with MLP is called the backpropagation algorithm which utilizes the concept of the Gradient Descent technique proposed by Newton. The aim is to come up with a corrected set of weights for each edge by minimizing the error using the backpropagation technique. The pointwise error is calculated as below:

$$e_n = \frac{1}{2} \sum_{j=0}^c (y_{nj} - v_{nj})^2 \quad (33)$$

Where e_n is the squared error for the n^{th} data point of the dataset, y_{nj} is the actual output of the n^{th} data point of the dataset and j^{th} node, v_{nj} is the calculated output of the n^{th} data point of the dataset and j^{th} node of the output layer and c is the total number of classes.

Hence, the total error or the overall error (E) is calculated as

$$E = \frac{1}{N} \sum_{n=1}^N e_n \quad (34)$$

$$\text{or, } E = \frac{1}{N} \sum_{n=1}^N \frac{1}{2} \sum_{j=0}^c (y_{nj} - v_{nj})^2 \quad (35)$$

$$\text{or, } E = \frac{1}{2N} \sum_{n=1}^N \sum_{j=1}^c (y_{nj} - v_{nj})^2 \quad (36)$$

Where N is the number of entries in the dataset.

The aim is to reduce the total error (E) using the backpropagation algorithm. The only variable associated with E is the weights. So, the target is to optimize the weights to minimize the error. Thus, the Error(E) can be expressed as a function of weights.

$$E = \frac{1}{2N} \sum_{n=1}^N \sum_{j=1}^c (y_{nj} - v_{nj})^2 = \psi(w_{ji}) \quad (37)$$

Where w_{ji} is the weight between the j^{th} node of the output layer and i^{th} node of the input layer.

The backpropagation algorithm is stated as

Step 1 : Randomly assign weights (w_{ji}) for $i = 0, 1, 2, \dots, d$ and $j = 1, 2, \dots, c$

Step 2 : Repeat loop starts

Step 3 : for loops starts for $n = 1$ to N

Step 4 : computer u_{nj} , v_{nj} and e_n

Step 5 : $w_{ji} = w_{ji} - \eta \frac{\partial e_n}{\partial w_{ji}}$

Step 6: for loop ends

Step 7: Repeat loop ends if the error has reduced to the permissible range or the iterations have reached their limit.

Since the error for each point is calculated and used for adjusting the weights, it is an extension of Gradient Descent known as Stochastic Gradient Descent (SGD). SGD takes care of the small local minima and ensures the algorithm is not stuck at those minima.

The MLP with a single layer can create a hyperplane to separate linearly separable data. For non-linearly separable data, it needs more than one layer called the Hidden Layer.

For MLP with one or more hidden layers, the concept remains the same. However, for each hidden layer, the weights need to be adjusted.

B. Artificial Neural Network

Artificial Neural Networks (ANNs) are a type of machine learning model or to be precise a Deep Learning Model that mimics the structure and function of the human brain. ANNs consist of a large number of interconnected processing nodes known as neurons, which are arranged in a very specific format. Each neuron receives input from all the neurons of the previous layer and processes that input before passing it on to the next layer.

The working of an ANN involves following stages

- 1) *Initialization: The network is structured by setting up the architecture which includes the total number of layers, the total number of neurons in each layer and the activation function to be used. The weights associated with each node are randomly assigned initially.*
- 2) *Forward Propagation: The data is first processed by the input layer and passed on to the first hidden layer. The neurons take in all the inputs, pass the value through an activation function and yield an output which is passed onto the next hidden layer. This process is continued till the output layer.*
- 3) *Activation Function: At each neuron, an activation function is applied to the input before it is passed on to the next layer. The activation function introduces non-linearity into the model, allowing the network to learn complex patterns in the data. There are many types of Activation Functions such as relu, softmax, sigmoid etc. among which for the intermediate layers, relu (Recti linear unit) is preferred and for the output layer sigmoid is preferred.*
- 4) *Loss Function: The output produced by the network is compared with the true output, and the difference between them is calculated. Usually, the Mean Squared Error (MSE) is used for Error Calculation. This can also be labelled as the loss function. The loss function is further used for the adjustment of the weights to make the predictions more accurate.*
- 5) *Backward Propagation: The Backpropagation method is used to consider the feedback of the loss function. It uses the concept of the Gradient Descent Algorithm proposed by Newton to update the weights of each edge. However, the implementation of Gradient Descent, has some disadvantages, like getting stuck in local minima and not being able to find the global minima. So, the Genetic Algorithm can also be used to overcome the shortages of Backpropagation.*
- 6) *Repeat: The entire process discussed above is repeated until the error is within an acceptable range or the epochs have reached the limit.*

The working of the ANN is very similar to that of the MLP Model and it is safe to conclude, MLP Model is a subclass of ANN.

C. Residual Neural Network

Residual Neural Network (ResNet) is an extension of the Convolution Neural Network (CNN). ResNet and CNN are primarily deployed for image processing as an application of Computer Vision.

The CNN was able to handle a particular number of hidden layers. For updating the weights, the Backpropagation method is used. After going back, a certain number of layers, there is a shift down of loss function. Over a significant number of layers, the gradient “vanishes” leading to the vanishing gradient problem, leading to a saturation in the performance of the CNN model. To overcome this issue, the ResNet model is deployed.

ResNet developed a unique solution, known as “skip connections”. It stacks multiple identity mappings (convolutional layers that do nothing at first), ignores those particular sets of layers and again reuses the activation functions of the previous layer. Skipping or ignoring speeds up initial training by compressing the network into compact layers.

While the network is again trained, all the layers are uncompressed and the remaining parts of the network, known as the residual parts, are allowed to elaborate more on the feature space of the input image.

The majority of the ResNet models skip two or three layers at a time with nonlinearity and batch normalization in between. More advanced ResNet models, known as HighwayNets, can learn to “skip weights”, which dynamically determine the number of layers to skip.

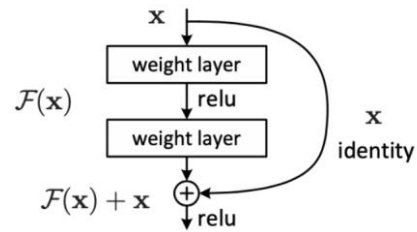


Fig. 10. Residual Block of ResNet

Fig. 10. shows the working of the Residual Block, which is responsible for skip connections in ResNet.

VII. METHODOLOGY

Data PreProcessing is a very important step in machine learning. The goal of data preprocessing is to prepare and clean the raw dataset so that efficiency and accuracy of the machine learning algorithms can be maximized.

Similarly before processing our dataset through the different machine learning algorithms we had to prepare our dataset for optimal use.

The Processes we had to go through to prepare our dataset were:

- 1) *Null Value Replacement: Real World datasets can have missing data which needs to be replaced by the mean value of the missing attribute. We checked our dataset for missing values and as the dataset did not have any missing values we did not have to do anything.*
- 2) *Skewness Reduction: Skewness is the measure of the asymmetry of the probability distribution of an attribute. Excessive skewness can lead to bias in the final model. For our dataset, we first check*

each attribute for its skewness and for any attribute which has an absolute skewness value of greater than 1 we append that attribute column to a list named *skewedCols*. Below is the list of 5 attributes each which have the highest and lowest skewness before skewness reduction:

Table

| Attribute Name | Skew Value |
|-------------------------------|---------------------|
| tqwt_TKEO_mean_dec_32 | 26.48258509147365 |
| tqwt_TKEO_std_dec_32 | 26.0620838755717 |
| tqwt_TKEO_mean_dec_33 | 24.9442663610479 |
| tqwt_TKEO_std_dec_33 | 24.283816068288253 |
| det_TKEO_mean_3_coef | 20.874872200923843 |
| det_LT_entropy_shannon_7_coef | -21.4150979727166 |
| tqwt_medianValue_dec_29 | -21.623643949849406 |
| tqwt_skewnessValue_dec_24 | -22.684339987002808 |
| tqwt_entropy_shannon_dec_33 | -25.06135227711703 |
| tqwt_entropy_shannon_dec_32 | -25.672811274750888 |

Fig ***. Attribute values with the highest and lowest skewness values before skewness removal

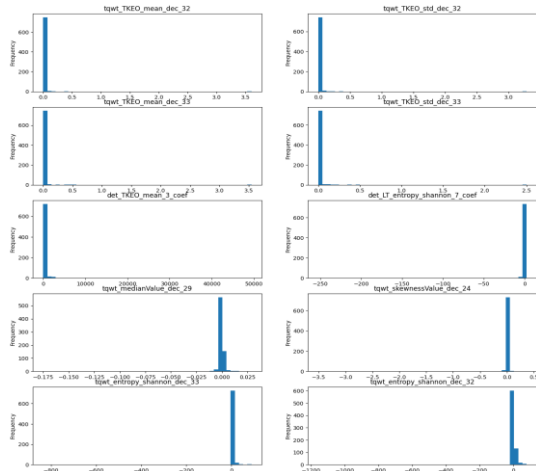


Fig. 11. Histogram of skewed attributes

Then we further classify the *skewedCols* based on whether that attribute contains any positive, zero or negative values into three separate lists named *skewedCols_PositiveVals*, *skewedCols_ZeroVals*, and *SkewedCols_NegativeVals* respectively. Then for the attributes present in the list *skewedCols_PositiveVals* we first use Box-Cox Transformation to reduce the skewness of the attributes. Box-Cox transformation works by applying a power function to the dependent variable which allows it to

be transformed into a normal distribution and reduce its skewness.

We also use cube root transformation to reduce the skewness of the attributes which are present in the lists *skewedCols_ZeroVals*, *SkewedCols_NegativeVals*. Cube Root transformation works by taking the cube root of each value of the attribute and making the attribute more closely resemble a normal distribution and reduce its skewness. Below is the final skewness of those attributes:

Table

| Attribute Name | Skew Value |
|-------------------------------|----------------------|
| tqwt_TKEO_mean_dec_32 | 0.19846778387365632 |
| tqwt_TKEO_std_dec_32 | 0.01987384966738003 |
| tqwt_TKEO_mean_dec_33 | 0.3104628329189008 |
| tqwt_TKEO_std_dec_33 | 0.07507677689667193 |
| det_TKEO_mean_3_coef | 1.4434955464066024 |
| det_LT_entropy_shannon_7_coef | -4.780452134423335 |
| tqwt_medianValue_dec_29 | -0.31596805061181005 |
| tqwt_skewnessValue_dec_24 | -1.690083153162617 |
| tqwt_entropy_shannon_dec_33 | -2.023628408215805 |
| tqwt_entropy_shannon_dec_32 | -2.120342043665047 |

Fig ***. The attributes having the highest and lowest skewness after skewness reduction

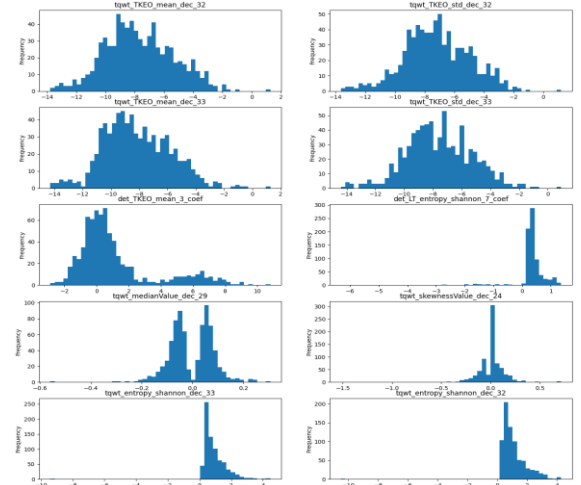


Fig. 12. Attributes with the highest skewness after skewness Reduction

- 3) **Kurtosis Reduction:** Kurtosis of an attribute is the measure of the peakedness/flatness of the probability distribution when measured relative to the normal distribution. The kurtosis of the normal distribution is considered zero. High kurtosis can lead to biases in the final model and hence we need

to reduce kurtosis before processing our data. For our dataset, we check for attributes which have a kurtosis value greater than 3. For our dataset, we did not have any such attribute and hence we did not need to do any kurtosis reduction.

- 4) **Outlier Detection:** Outliers are data points in an attribute which are significantly different from the rest of the data. Outliers are bad for machine learning as they can bias the results of the final model. For our model, We check each attribute and look for data points which are lower than the 25 percentile or higher than the 75 percentile of that attribute any data point which falls outside of this criteria is replaced by the mean of that attribute.

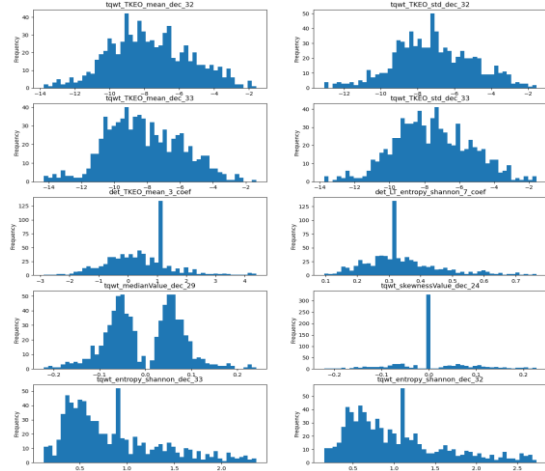


Fig. 13. Attributes with the highest skewness after outlier Reduction

- 5) **PCA:** For our dataset we loop through different number of principal components and finally got 148 principal components which was giving us the optimal results.

VIII. COMPARATIVE STUDY

After preprocessing our dataset, we trained each of our machine learning and neural network models using the dataset. We split the dataset into training and testing parts and after developing the models using the training split, we carefully tested each model using the test split. We have accumulated the following parameters to evaluate the performance of the different algorithms against our dataset.

• Accuracy

Table I.

| Name of Model | Accuracy |
|--------------------------------|----------|
| KNN Classifier | 96% |
| SVM Classifier | 88% |
| Decision Tree Classifier | 87% |
| Random Forest Classifier | 91% |
| Naive Bayes Classifier | 89% |
| Logistic Regression Classifier | 88% |
| XGBoost Classifier | 89% |
| Gradient Boost Classifier | 87% |
| ANN Classifier | 98% |
| MLP Classifier | 96% |
| Residual Neural Network | 99% |

Fig. 14. Accuracy of different models

We have plotted the accuracies of the different models for the training split in the table above. From the above table, it is evident that the Residual Neural Network and the ANN Classifier have the highest accuracy in classifying whether a patient has Parkinson's disease or not based on their attributes.

- **Error** - Error is the difference between the true result and the result predicted by the model. We have plotted the different error graphs for the different models.

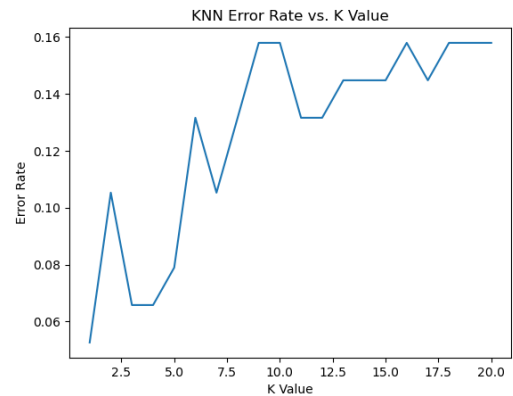


Fig. 15. Error rate vs K Value for KNN

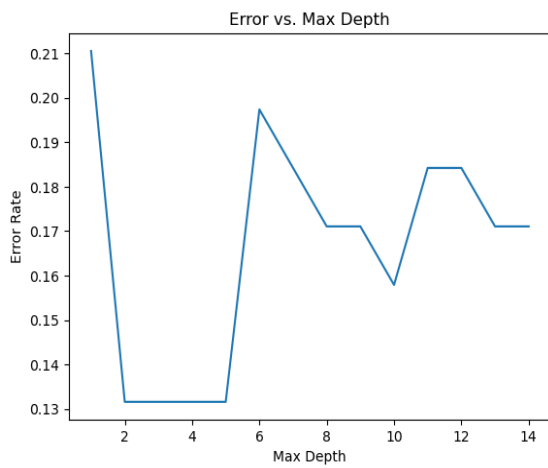


Fig. 16. Error rate vs Max depth for DTC

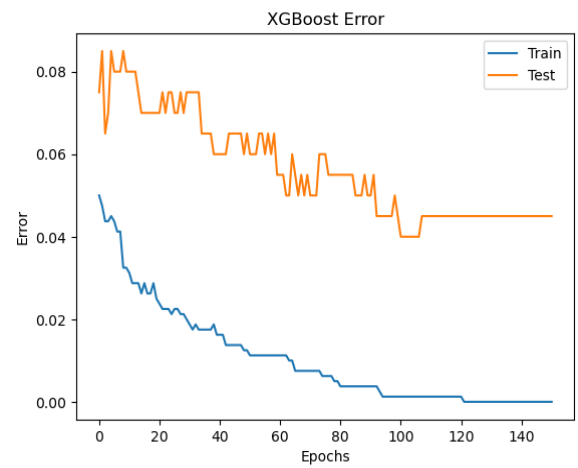


Fig. 19. Errors vs Iterations for XGBoost

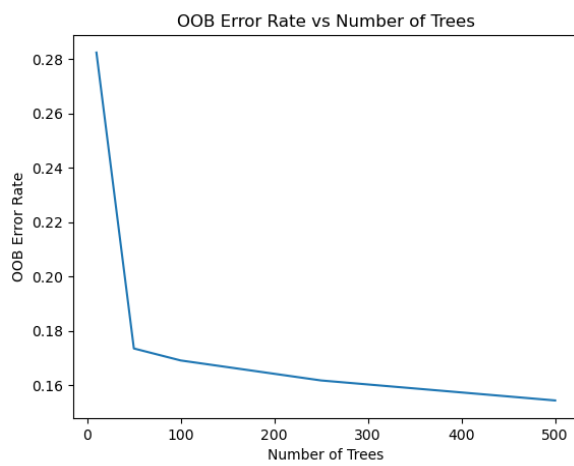


Fig. 17. OOB Error rate vs No. of trees for RFC

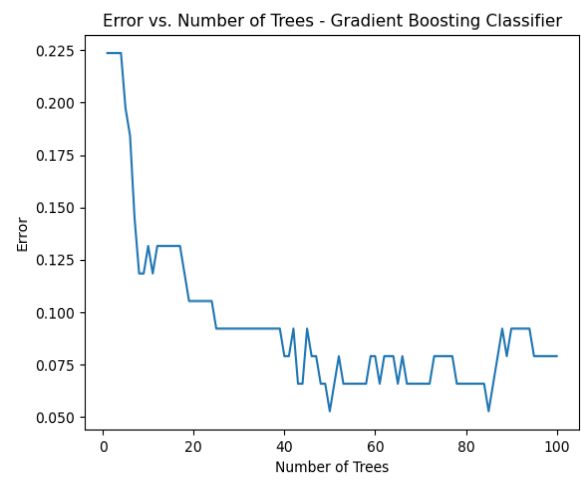


Fig. 20. Errors vs No. of trees for Gradient Boosting Classifier



Fig. 18. Error rate vs Iterations for Logistic regression

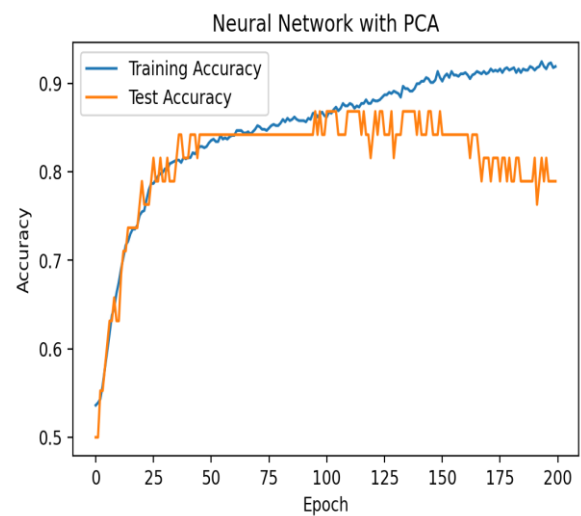


Fig. 21. Accuracy vs Iterations for ANN

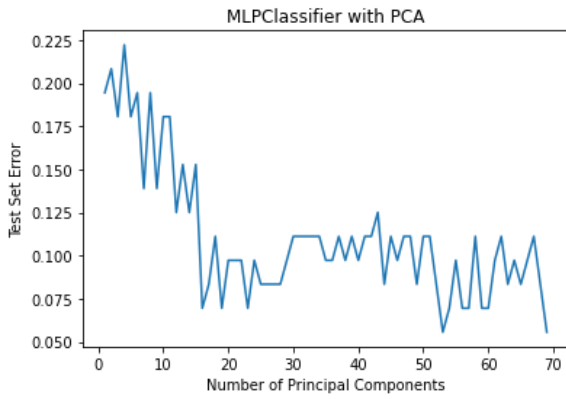


Fig. 22. Test set error vs No. of components for MLP

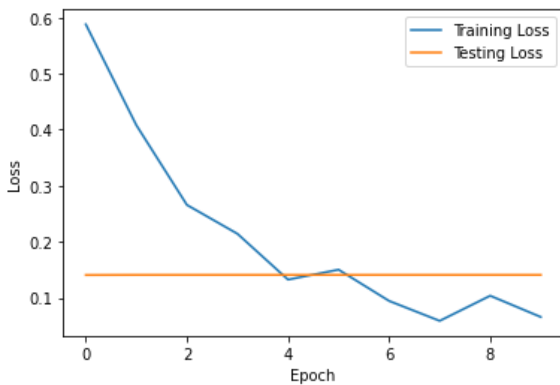


Fig. 23. Loss vs Iterations for ResNet

From the above graphs, it is evident the models are able to learn to a certain accuracy where the loss or error is minimum.

• Recall

Table II.

| Name of Model | Recall | |
|--------------------------------|--------------------|---------------------|
| | Parkinson's Absent | Parkinson's Present |
| KNN Classifier | 0.86 | 1 |
| SVM Classifier | 0.57 | 1 |
| Decision Tree Classifier | 0.71 | 0.93 |
| Random Forest Classifier | 0.67 | 1 |
| Naive Bayes Classifier | 0.62 | 1 |
| Logistic Regression Classifier | 0.57 | 1 |
| XGBoost Classifier | 0.71 | 0.96 |
| Gradient Boost Classifier | 0.67 | 0.95 |

| | | |
|-------------------------|------|------|
| ANN Classifier | 0.95 | 0.99 |
| MLP Classifier | 1 | 0.94 |
| Residual Neural Network | 0.99 | 0.99 |
| ANN Classifier | 0.95 | 0.99 |
| MLP Classifier | 1 | 0.94 |
| Residual Neural Network | 0.99 | 0.99 |

Fig. 24. Table for Recall for different Models

We have accrued the recall values of the different models for the training split in the table above. Recall measures the proportion of actual positive cases that are correctly identified by the model as positive. It is a very important metric for our use case as a high recall signifies that the model does not miss many positive cases as negative. From the above table it is evident that Residual Neural Network and the ANN Classifier are the best performing model in terms of recall.

• Precision

Table III.

| Name Of Model | Precision | |
|--------------------------------|-------------|-------------|
| | PD Negative | PD Positive |
| KNN Classifier | 1 | 0.95 |
| SVM Classifier | 1 | 0.86 |
| Decision Tree Classifier | 0.79 | 0.89 |
| Random Forest Classifier | 1 | 0.89 |
| Naive Bayes Classifier | 1 | 0.87 |
| Logistic Regression Classifier | 1 | 0.86 |
| XGBoost Classifier | 0.88 | 0.9 |
| Gradient Boost Classifier | 0.82 | 0.88 |
| ANN Classifier | 0.98 | 0.98 |
| MLP Classifier | 0.88 | 1 |
| Residual Neural Network | 0.97 | 1 |

Fig. 25. Table for Precision of different models

We have accrued the precision values of the different models for the training split in the table above. Precision is the measure the fraction of correctly predicted positive instances out of all the instances the model has predicted as positive. It is also a very important metric for our use case as a high precision value implies that the model does not

misclassify too many positive patients as negative and vice versa. From the above table, it is also evident that the Residual Neural Network is the optimal model in terms of precision.

- **F1-Score**

Table IV.

| Name of Model | F1 - Score | |
|--------------------------------|-------------|-------------|
| | PD Negative | PD Positive |
| KNN Classifier | 0.92 | 0.97 |
| SVM Classifier | 0.73 | 0.92 |
| Decision Tree Classifier | 0.75 | 0.91 |
| Random Forest Classifier | 0.8 | 0.94 |
| Naive Bayes Classifier | 0.76 | 0.93 |
| Logistic Regression Classifier | 0.73 | 0.92 |
| XGBoost Classifier | 0.79 | 0.93 |
| Gradient Boost Classifier | 0.74 | 0.91 |
| ANN Classifier | 0.96 | 0.99 |
| MLP Classifier | 0.94 | 0.97 |
| Residual Neural Network | 0.98 | 0.99 |

Fig. 26. Table for F1 - Score of different models

We have accrued the F1-scores of the different models for the training split in the table above.

F1 - Score is calculated by combining both the Precision and recall values of the model and hence is a important metric to consider for our use case as a high F1 - Score implies that the model is making accurate predictions and also minimizes false negatives reducing the risk of miss-diagnosis. From the above table it is evident that Residual Neural Network is the optimal model as it has the highest F1 - Score for both the classes.

- **AUC-ROC Curve**

The AUC-ROC curve is made by plotting the model's true positive rate against its false positive rate. A high AUC value in the ROC curve (Close to 1) implies that the model can precisely distinguish between Parkinson's Positive and Parkinson's Negative patients.

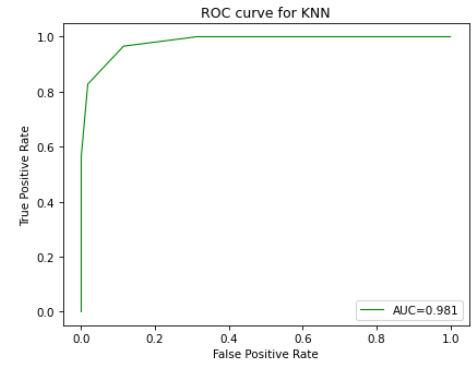


Fig. 27. AUC-ROC for KNN

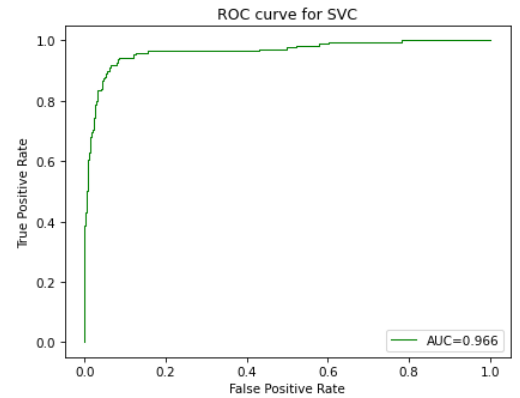


Fig. 28. AUC-ROC for SVC

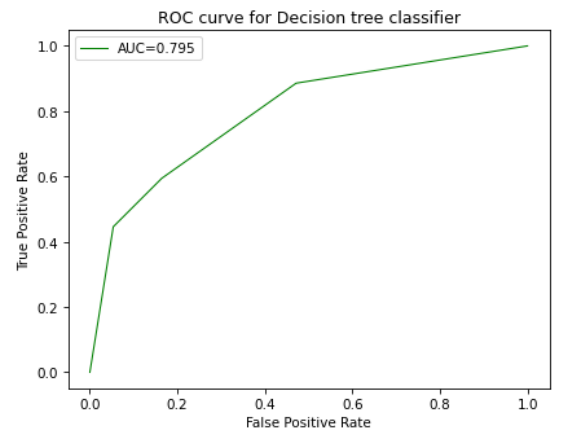


Fig. 29. AUC-ROC for DTC

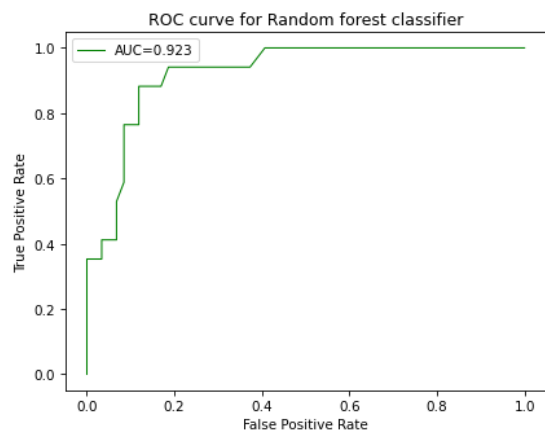


Fig. 30. AUC-ROC for RFC

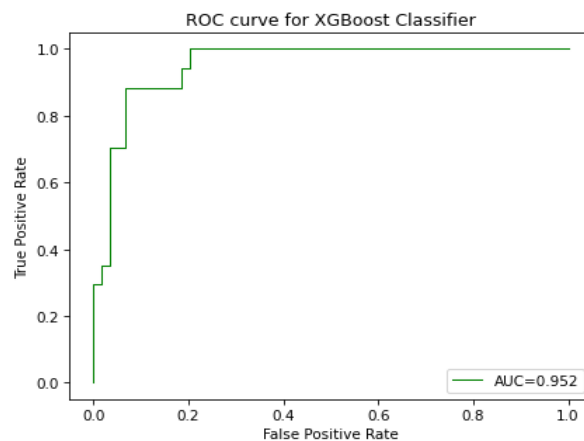


Fig. 33. AUC-ROC for XGBoost

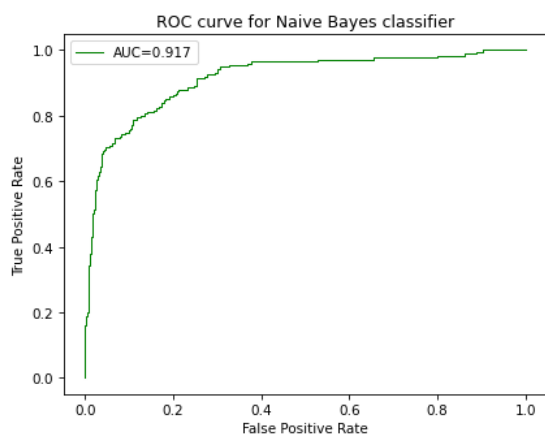


Fig. 31. AUC-ROC for Naïve Bayes

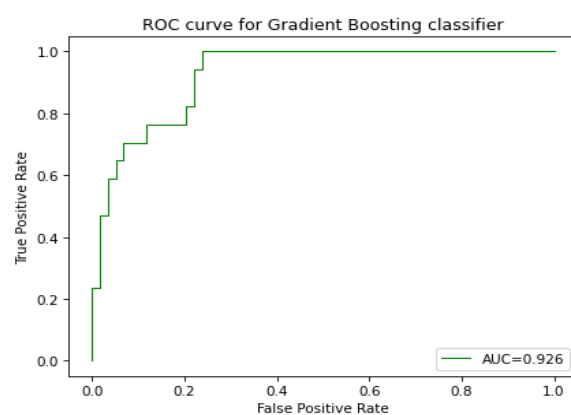


Fig. 34. AUC-ROC for Gradient Boosting classifier

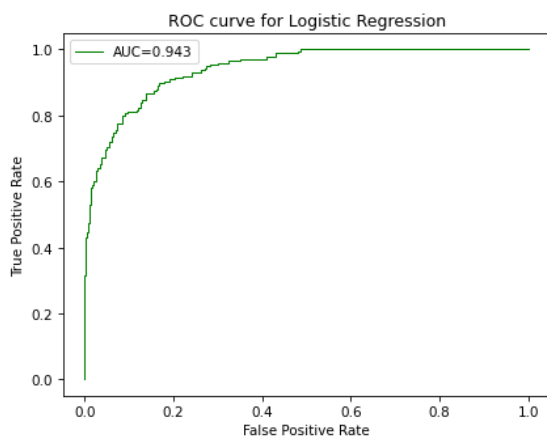


Fig. 32. AUC-ROC for Logistic Regression

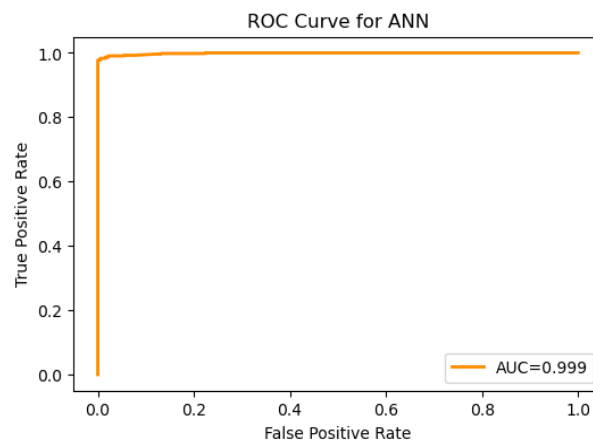


Fig. 35. AUC-ROC for ANN

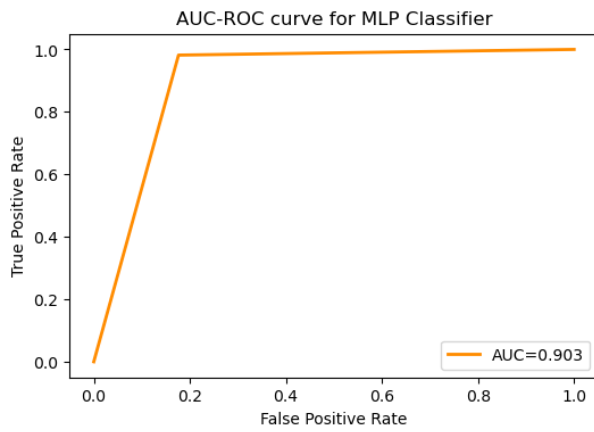


Fig. 36. AUC-ROC for MLP

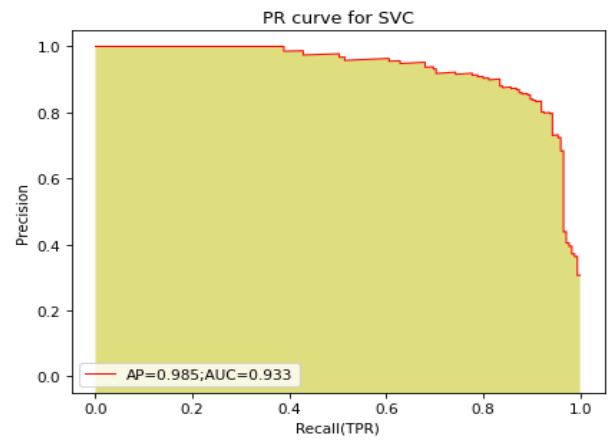


Fig. 38. AUC-PR for SVC

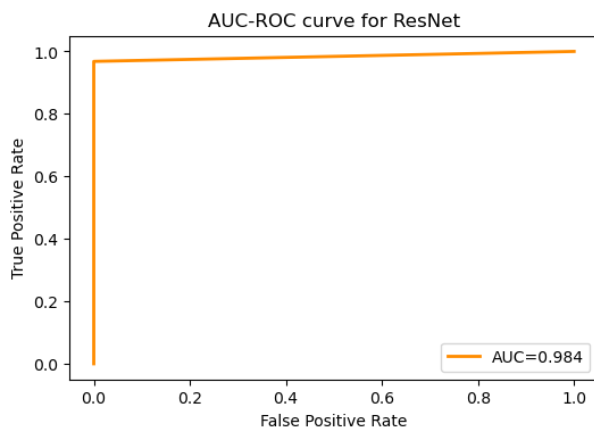


Fig. 36. AUC-ROC for ResNet

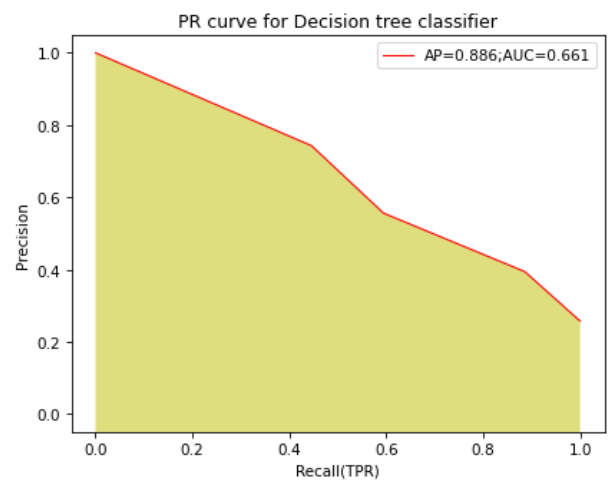


Fig. 39. AUC-PR for DTC

From the above graphs it is evident that the Neural Networks are the optimal model as they have the highest AUC-ROC.

• AUC-PR Curve

The AUC-PR curve is made by plotting the model's precision against recall of the model. In medical use cases it is a particularly important metric to consider as it can provide a more accurate evaluation of the performance model when the class distribution is imbalanced. The same is true for our dataset as well.

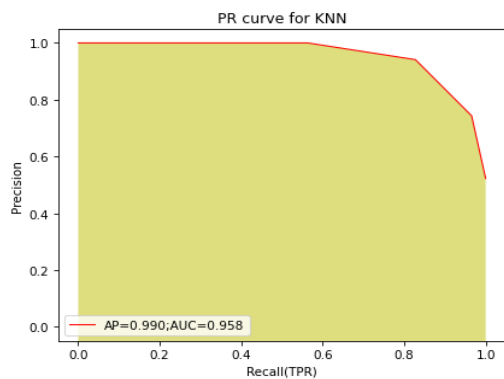


Fig. 37. AUC-PR for KNN

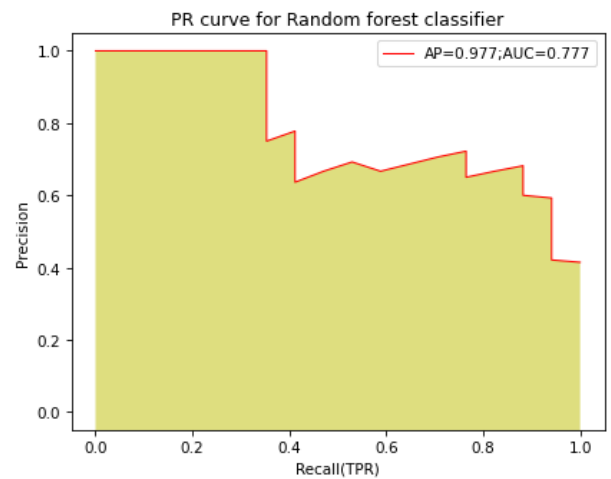


Fig. 40. AUC-PR for RFC

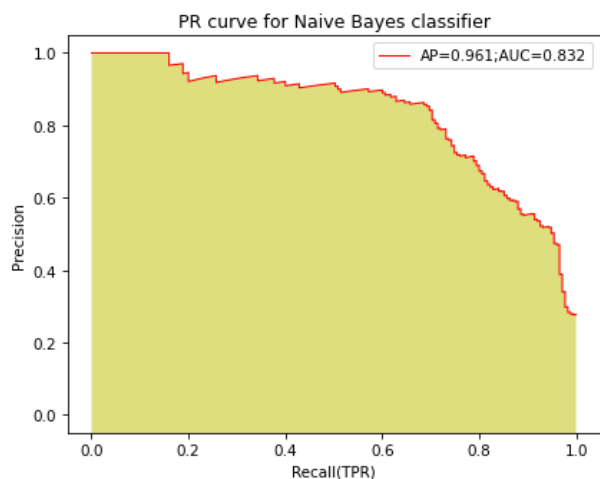


Fig. 41. AUC-PR for Naïve Bayes classifier

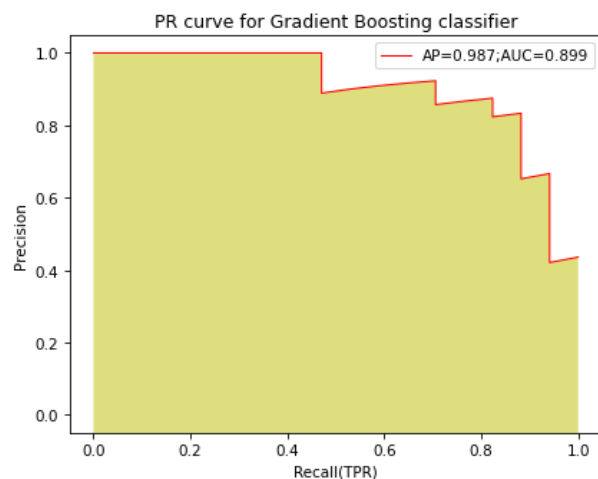


Fig. 44. AUC-PR for Gradient boosting classifier

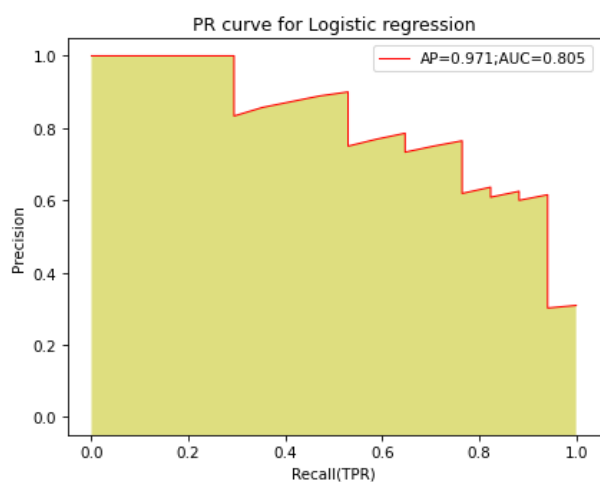


Fig. 42. AUC-PR for Logistic Regression

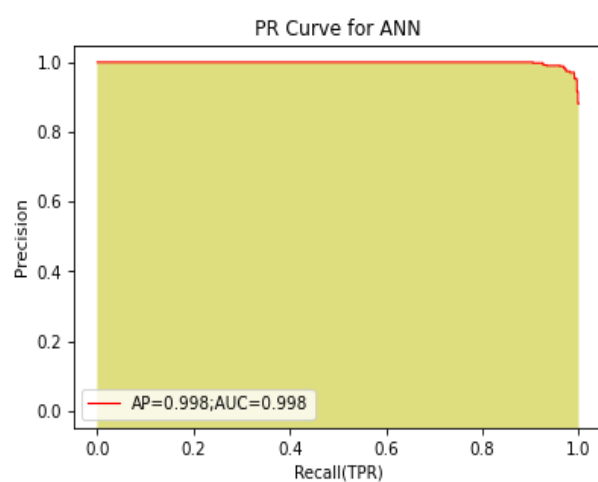


Fig. 45. AUC-PR for ANN

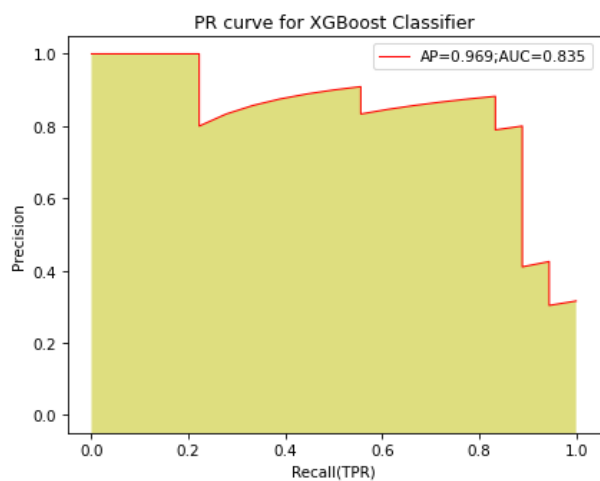


Fig. 43. AUC-PR for XGBoost

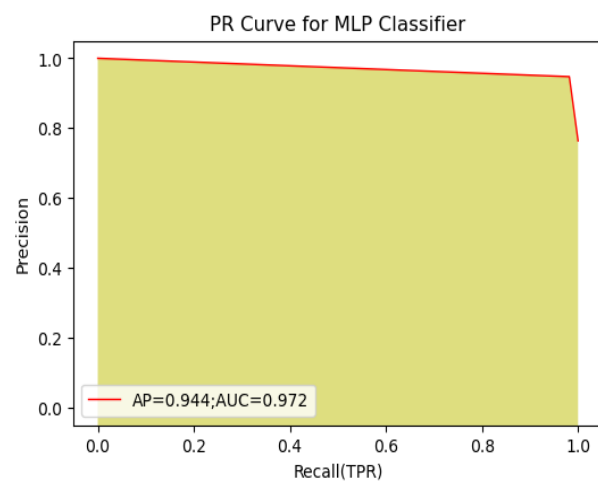


Fig. 46. AUC-PR for MLP

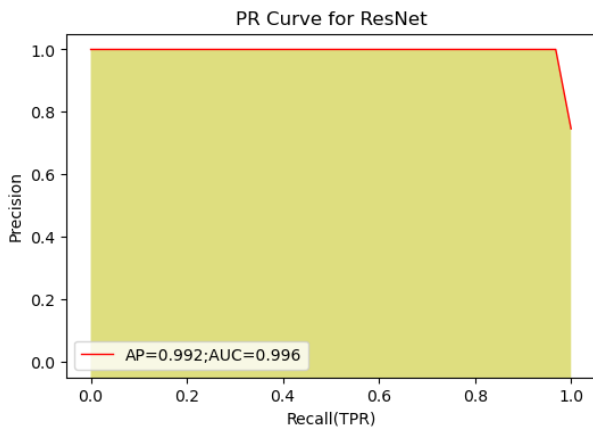


Fig. 47. AUC-PR for ResNet

recall of 0.99 and an F1-Score of 0.985. Standing next to the ResNet is ANN with an accuracy of 98%, a precision of 0.98, a recall of 0.97 and an F1-Score of 0.975. Thus, we can conclude, for our chosen dataset, the Neural Networks have performed exceptionally better in terms of the evaluation metrics. With the advent of more precise attributes in the future, the accuracy of the ML Models and Neural Networks can be boosted to a great extent using Feature Selection, Dimension Reduction and all the other effective and essential Data Pre-processing Techniques.

X. REFERENCE WORK

IX. CONCLUSION AND SCOPE FOR FUTURE WORK

The research article focuses on a comparative study between Traditional Machine Learning Algorithms and Neural Networks. From the study, it is evident that both Models have their own positive and negative aspects. The Traditional Model focuses on a simplistic approach to automate the learning process of machines and find hidden and unseen patterns. The Neural Networks, on the other hand, try to mimic the working of the human brain, by setting up processing nodes called neurons in a systematic plan and each neuron is interconnected with each others giving rise to a fully connected network.

Traditional ML models are very useful for small to medium-sized datasets, whereas, neural networks are well known for their ability to handle complicated and high-dimensional datasets. Due to the simplicity of traditional ML models, their computing time is often faster than neural networks. Thus, Neural Networks are much more resource hungry and complex to implement.

The factors responsible for choosing between implementing the traditional ML models or the Neural Networks depend upon the type and quality of the dataset, the computing resources and the budget.

As the field of Machine Learning and Artificial Intelligence continues to evolve at an exponential rate, both the Traditional Machine Learning Models and the Neural Networks will continue to upgrade themselves in terms of accuracy, computation time and resource needs. It is important to continue to explore and develop new approaches that combine the strengths of these models and address their limitations, to improve accuracy, interpretability, and scalability.

From our comparative analysis of the selected dataset, we conclude, Residual Neural Network (ResNet) has worked exceptionally better than all the traditional Machine Learning Models with an accuracy of 99%, a precision of 0.985, a