PARKINSON’S DISEASE PREDICTION USING CATBOOST, RANDOMFOREST AND XGBOOST

A

MINOR PROJECTREPORT

Submitted by

UMANG TIWARI	**UDIT JAIN**	**HIMANI SHEORAN**
Enrollment No: 10314802718	**Enrollment No: 10214802718**	**Enrollment No: 04514802718**

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ANDENGINEERING

Under the Guidance of

Mrs. ZAMEER FATIMA

Assistant Professor

Department of Computer Science and Engineering

Maharaja Agrasen Institute ofTechnology,

PSP area, Sector – 22, Rohini, New Delhi – 110085

(Affiliated to Guru Gobind Singh Indraprastha, New Delhi)

(DEC 2021)

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

CERTIFICATE

This is to Certified that this MINOR project report " PARKINSON'S DISEASE PREDICTION USING CATBOOST, RANDOMFOREST AND XGBOOST " is submitted by " UMANG TIWARI ENROLLMENT NO:10314802718, UDIT JAIN ENROLLMENT NO: 10214802718, HIMANI SHEORAN ENROLLMENT NO: 04514802718 " who carried out the project work under my supervision.

I approve this MINOR project for submission.

Prof. Namita Gupta	Mrs. Zameer Fatima
Designation (HoD, CSE)	Assistant Professor

ACKNOWLEDGEMENT

It gives me immense pleasure to express my deepest sense of gratitude and sincere thanks to my respected guide Mrs. Zameer Fatima, Assistant Professor, Computer Science And Engineering MAIT Delhi, for their valuable guidance, encouragement and help for completing this work. Their useful suggestions for this whole work and co-operative behavior are sincerely acknowledged.

I am also grateful to my teacher Mrs. Zameer Fatima for her constant support and guidance. (If Required)

I also wish to express my indebtedness to my parents as well as my family member whose blessings and support always helped me to face the challenges ahead.

Place: Delhi

Student Name with Roll no.:

Umang tiwari, 10314802718

Udit Jain, 10214802718

Himani Sheoran, 04154802718

Date:

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1 : INTRODUCTION

INTRODUCTION

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

WHY IS MACHINE LEARNING?

Resurging interest in machine learning is due to the same factors that have made data mining and Bayesian analysis more popular than ever. Things like growing volumes and varieties of available data, computational processing that is cheaper and more powerful, and affordable data storage.

All of these things mean it's possible to quickly and automatically produce models that can analyse bigger, more complex data and deliver faster, more accurate results – even on a very large scale. And by building precise models, an organization has a better chance of identifying profitable opportunities – or avoiding unknown risks.

The nearly limitless quantity of available data, affordable data storage, and the growth of less expensive and more powerful processing has propelled the growth of ML. Now many industries are developing more robust models capable of analyzing bigger and more complex data while delivering faster, more accurate results on vast scales. ML tools enable organizations to more quickly identify profitable opportunities and potential risks.

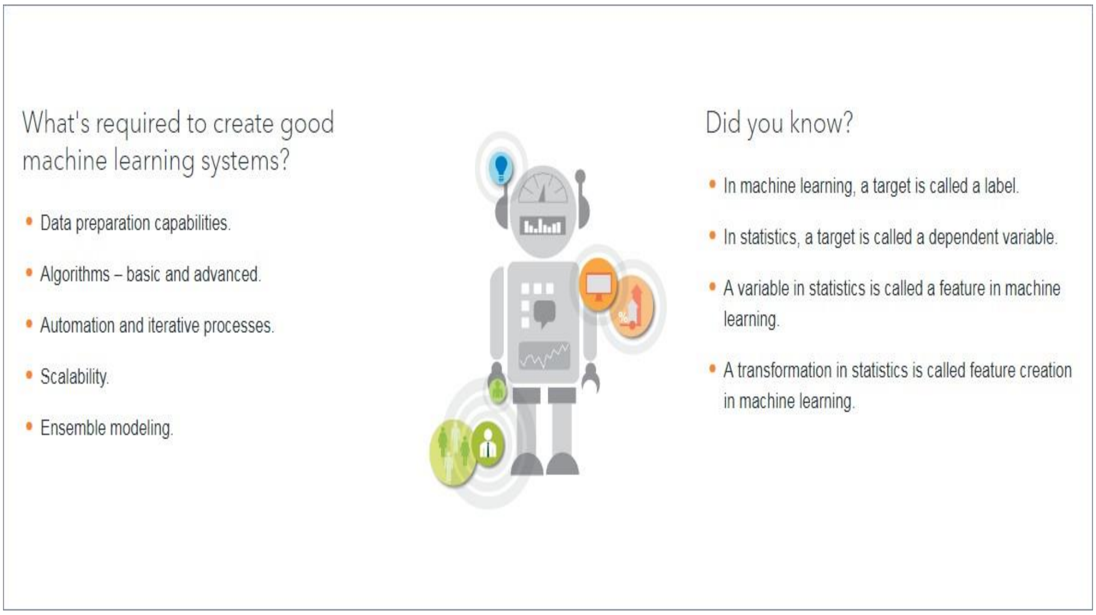


Figure 1.1 Machine Leaning Introduction

EVOLUTION OF MACHINE LEARNING

Because of new computing technologies, machine learning today is not like machine learning of the past. It was born from pattern recognition and the theory that computers can learn without being programmed to perform specific tasks; researchers interested in artificial intelligence wanted to see if computers could learn from data. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to independently adapt. They learn from previous computations to produce reliable, repeatable decisions and results. It's a science that's not new – but one that has gained fresh momentum.

While many machine learning algorithms have been around for a long time, the ability to automatically apply complex mathematical calculations to big data – over and over, faster and faster – is a recent development.

APPLICATION OF MACHINE LEARNING

The value of machine learning technology has been recognized by companies across several industries that deal with huge volumes of data. By leveraging insights obtained from this data, companies are able work in an efficient manner to control costs as well as get an edge over their

competitors. This is how some sectors / domains are implementing machine learning –

FINANCIAL SERVICES

Companies in the financial sector are able to identify key insights in financial data as well as prevent any occurrences of financial fraud, with the help of machine learning technology. The technology is also used to identify opportunities for investments and trade. Usage of cyber surveillance helps in identifying those individuals or institutions which are prone to financial risk, and take necessary actions in time to prevent fraud.

MARKETING AND SALES

Companies are using machine learning technology to analyse the purchase history of their customers and make personalized product recommendations for their next purchase. This ability to capture, analyse, and use customer data to provide a personalized shopping experience is the future of sales and marketing.

HEALTHCARE

With the advent of wearable sensors and devices that use data to access health of a patient in real time, ML is becoming a fast- growing trend in healthcare. Sensors in wearable provide real-time patient information, such as overall health condition, heartbeat, blood pressure and other vital parameters. Doctors and medical experts can use this information to analyse the health condition of an individual, draw a pattern from the patient history, and predict the occurrence of any ailments in the future.

TRANSPORTATION

Efficiency and accuracy are key to profitability within this sector; so is the ability to predict and mitigate potential problems. ML's data analysis and modeling functions dovetail perfectly with businesses within the delivery, public transportation, and freight transport sectors. ML uses algorithms to find factors that positively and negatively impact a supply chain's success, making machine learning a critical component within supply chain management.

GOVERNMENT

Systems that use machine learning enable government officials to use data to predict potential future scenarios and adapt to rapidly changing situations. ML can help to improve cybersecurity and cyber intelligence, support counterterrorism efforts, optimize operational preparedness, logistics management, and predictive maintenance, and reduce failure rates. This recent article highlights 10 more applications for machine learning within the healthcare industry.

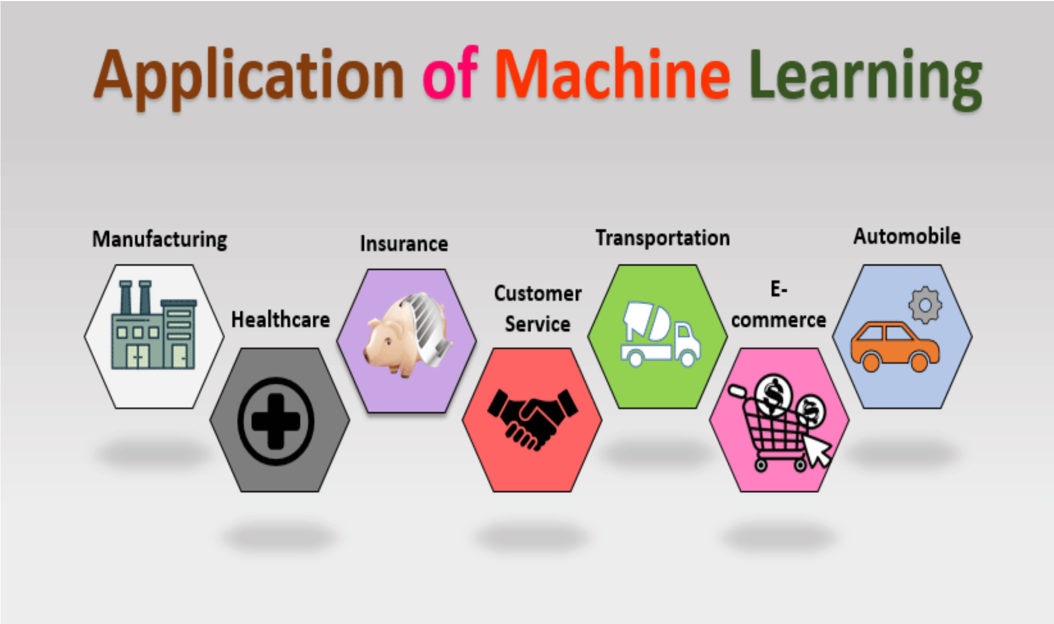


Figure 1.2 Application Of Machine Learning

METHODS OF MACHINE LEARNING

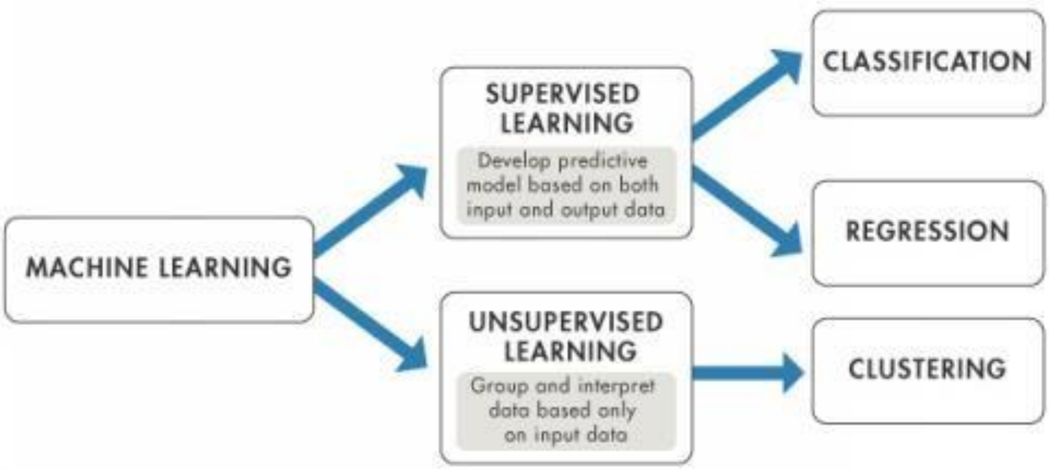


Figure 1.3 Machine Learning Classification

SUPERVISED LEARNING

These algorithms are trained using labeled examples, in different scenarios, as an input where the desired outcome is already known. An equipment, for instance, could have data points such as "F" and "R" where "F" represents "failed" and "R" represents "runs". A learning algorithm will receive a set of input instructions along with the corresponding accurate outcomes. The learning algorithm will then compare the actual outcome with the accurate outcome and flag an error, if there is any discrepancy. Using different methods, such as regression, classification, gradient boosting, and prediction, supervised learning uses different patterns to proactively predict the values of a label on extra unlabeled data. This method is commonly used in areas where historical data is used to predict events that are likely to occur in the future. For instance, anticipate when a credit card transaction is likely to be fraudulent or predict which insurance customers are likely to file their claims.

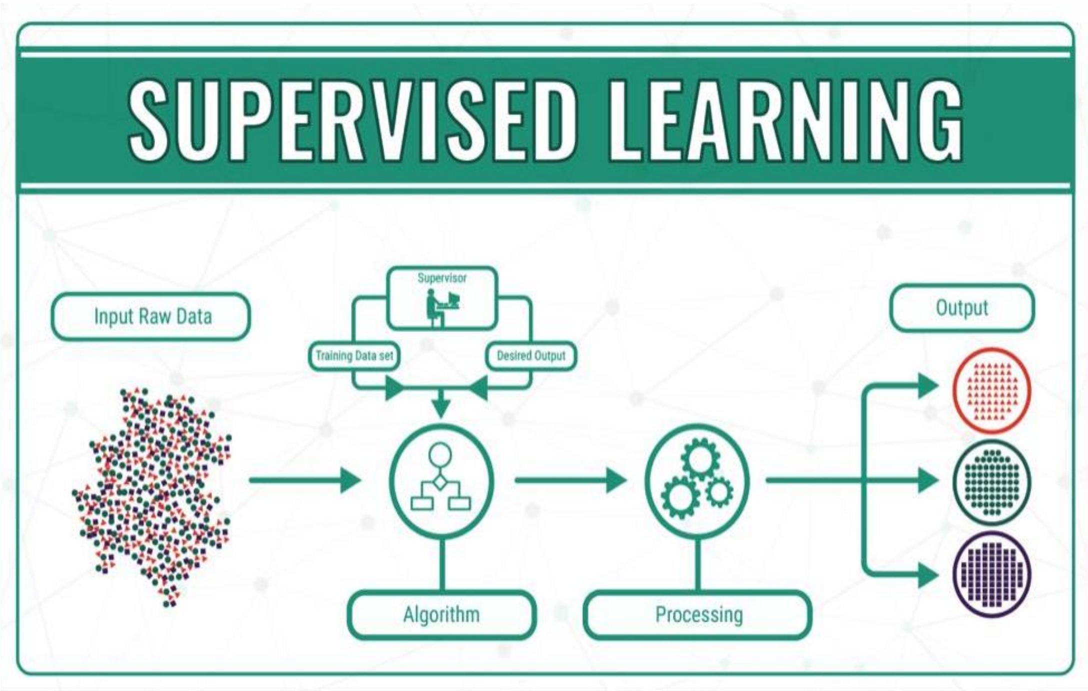


Figure 1.4 Supervised Learning

UNSUPERVISED LEARNING

This method of ML finds its application in areas were data has no historical labels. Here, the system will not be provided with the "right answer" and the algorithm should identify what is being shown. The main aim here is to analyse the data and identify a pattern and structure within the available data set. Transactional data serves as a good source of data set for unsupervised learning. For instance, this type of learning identifies customer segments with similar attributes and then lets the business to treat them similarly in marketing campaigns. Similarly, it can also identify attributes that differentiate customer segments from one another. Either ways, it is about identifying a similar structure in the available data set. Besides, these algorithms can also identify outliers in the available data sets.

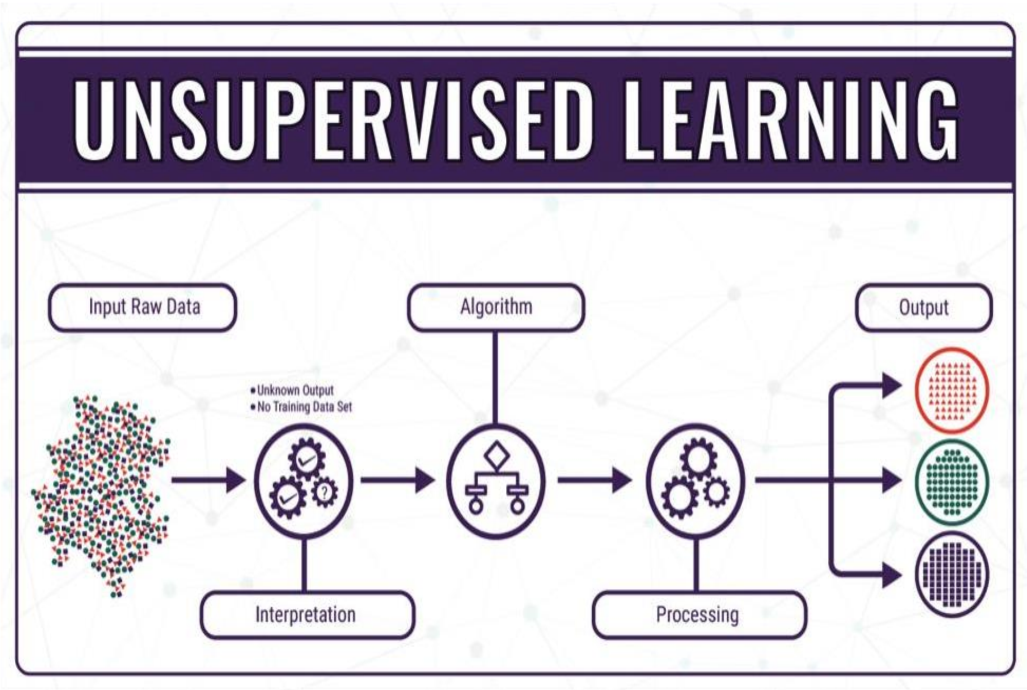


Figure 1.5 Unsupervised Learning

PROBLEM STATEMENT

Parkinson's disease occurs when nerve cells, or neurons, in an area of the brain that controls movement become impaired and/or die. Normally, these neurons produce an important brain chemical known as dopamine. When the neurons die or become impaired, they produce less dopamine, which causes the movement problems of Parkinson's.Besides motor symptoms, the person may see, hear, or experience things that are not real (hallucinations), or believe things that are not true (delusions).

PROJECT OBJECTIVE

Prediction of Parkinson’s Disease with the help of voice Dataset which helps to treat the people in early stages. In worst cases, Patients have great difficulty walking or standing. They are not able to live alone and require a wheelchair to move around. Assistance is needed in all daily activities.

CHAPTER 2 : SUPERVISED AND UNSUPERVISED LEARNING

SUPERVISED LEARNING

Supervised learning is the Data mining task of inferring a function from labeled training data.The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value. A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a “reasonable” way.

List of Common Algorithms-

- K-Nearest Neighbour
- Linear Regression
- Decision Trees
- Naïve Bayes
- Support Vector Machines (SVM)

K-NEAREST NEIGHBOUR

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute. As an example, consider the following table of data points containing two features:

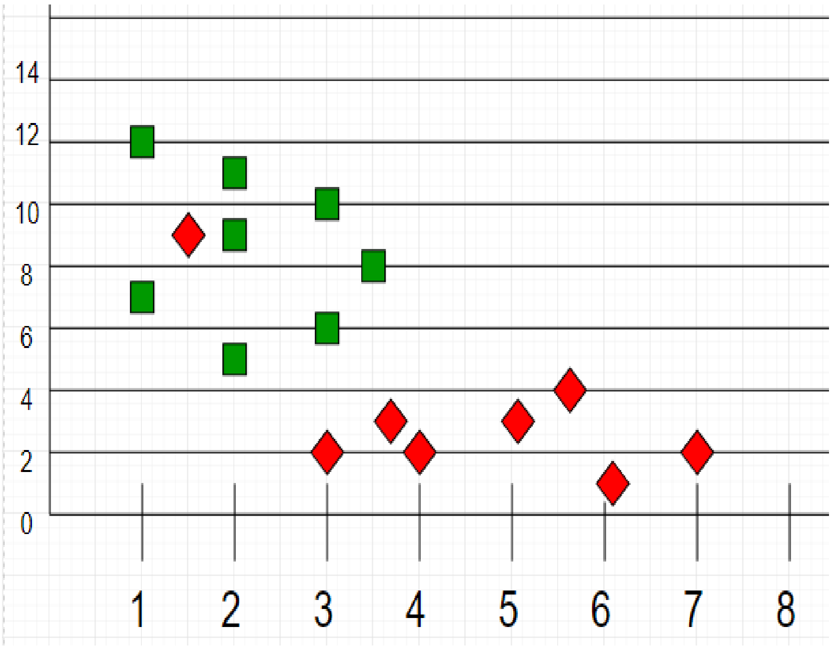


Figure 2.1 K-Nearest Neighbour

LINEAR REGRESSION

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

Hypothesis function for Linear Regression :

$$y = a + bx$$

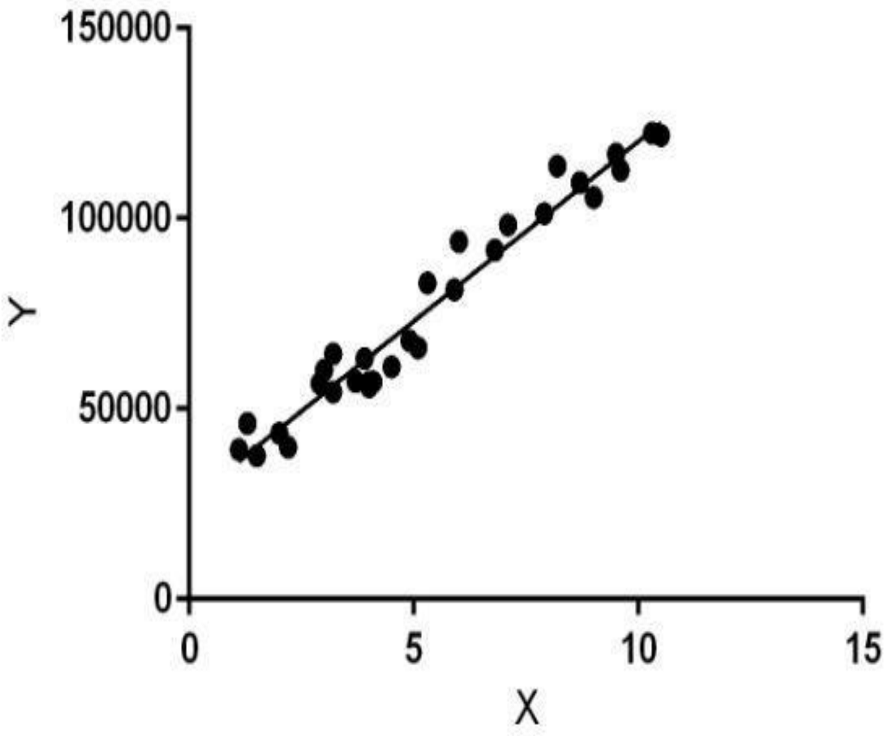


Figure 2.2 Linear Regression

DECISION TREE

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Decision Tree Representation :

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.

An instance is classified by starting at the root node of the treetesting the attribute specified by this nodethen moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.

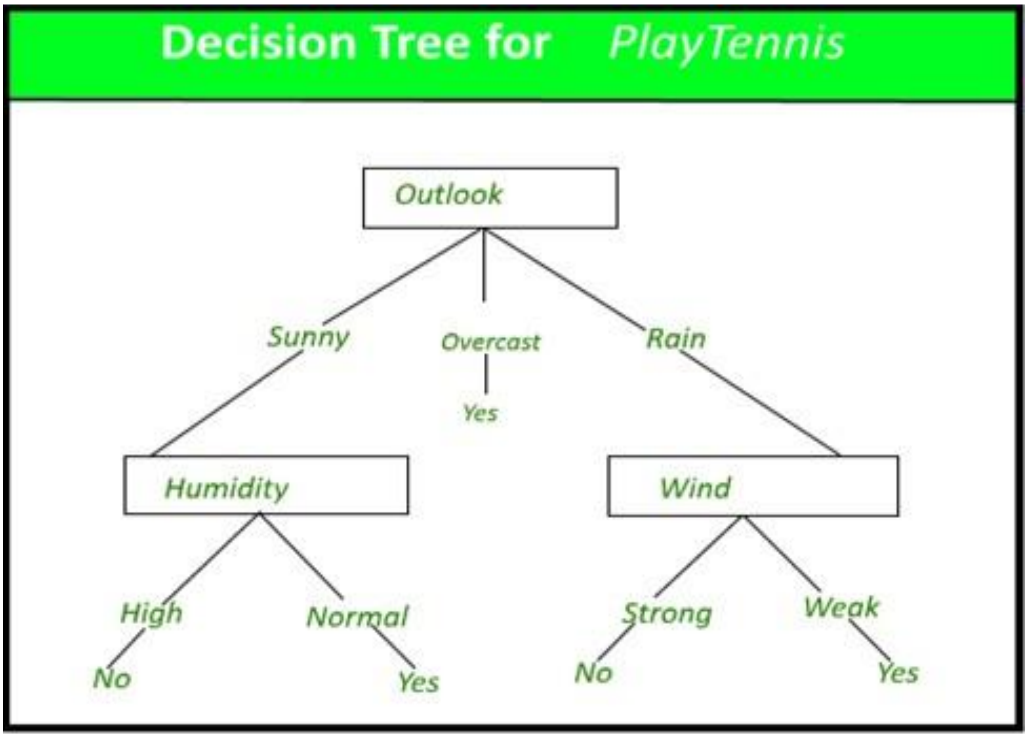


Figure 2.3 Decision Tree

NAIVE BAYES

Naive Bayesian model (NBN) is easy to build and very useful for large datasets. This method is composed of direct acyclic graphs with one parent and several children. It assumes independence among child nodes separated from their parent.

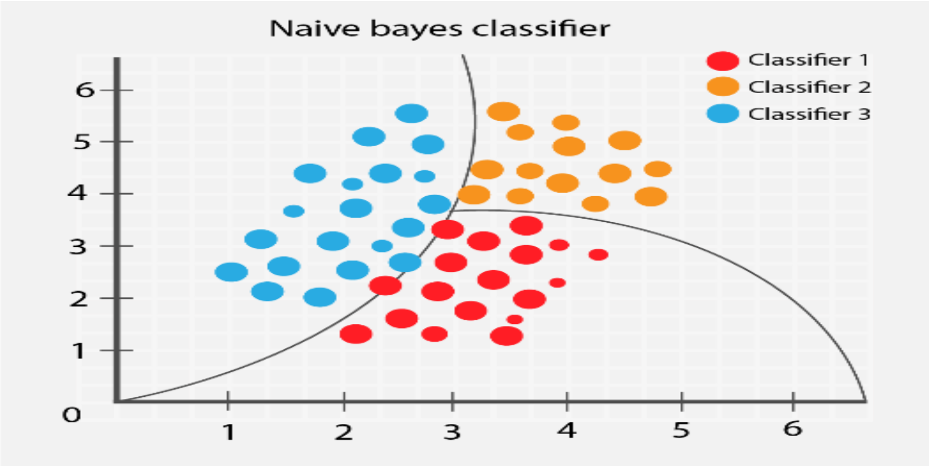


Figure 2.4. Naïve Bayes

SUPPORT VECTOR MACHINE (SVM)

Support vector machine (SVM) is a type of learning algorithm developed in 1990. This method is based on results from statistical learning theory introduced by Vap Nik.SVM machines are also closely connected to kernel functions which is a central concept for most of the learning tasks. The kernel framework and SVM are used in a variety of fields. It includes multimedia information retrieval, bioinformatics, and pattern recognition.

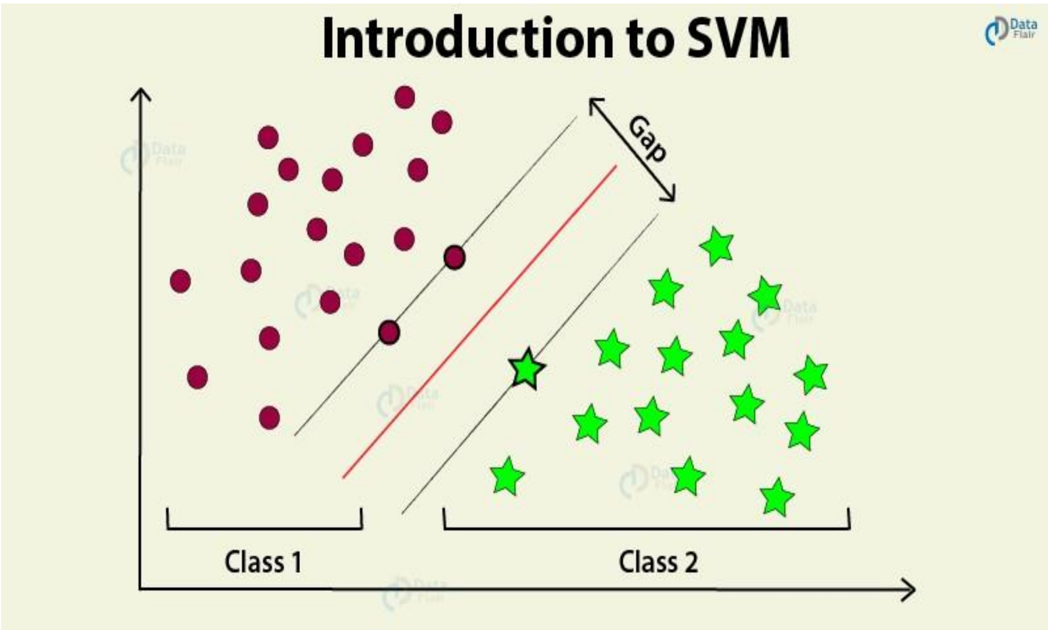


Figure 2.5 Introduction to SVM

UNSUPERVISED LEARNING

Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore machine is restricted to find the hidden structure in unlabeled data by our-self.

K-Means Clustering

K-means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.

3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
4. Compute the sum of the squared distance between data points and all centroids.
5. Assign each data point to the closest cluster (centroid).
6. Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

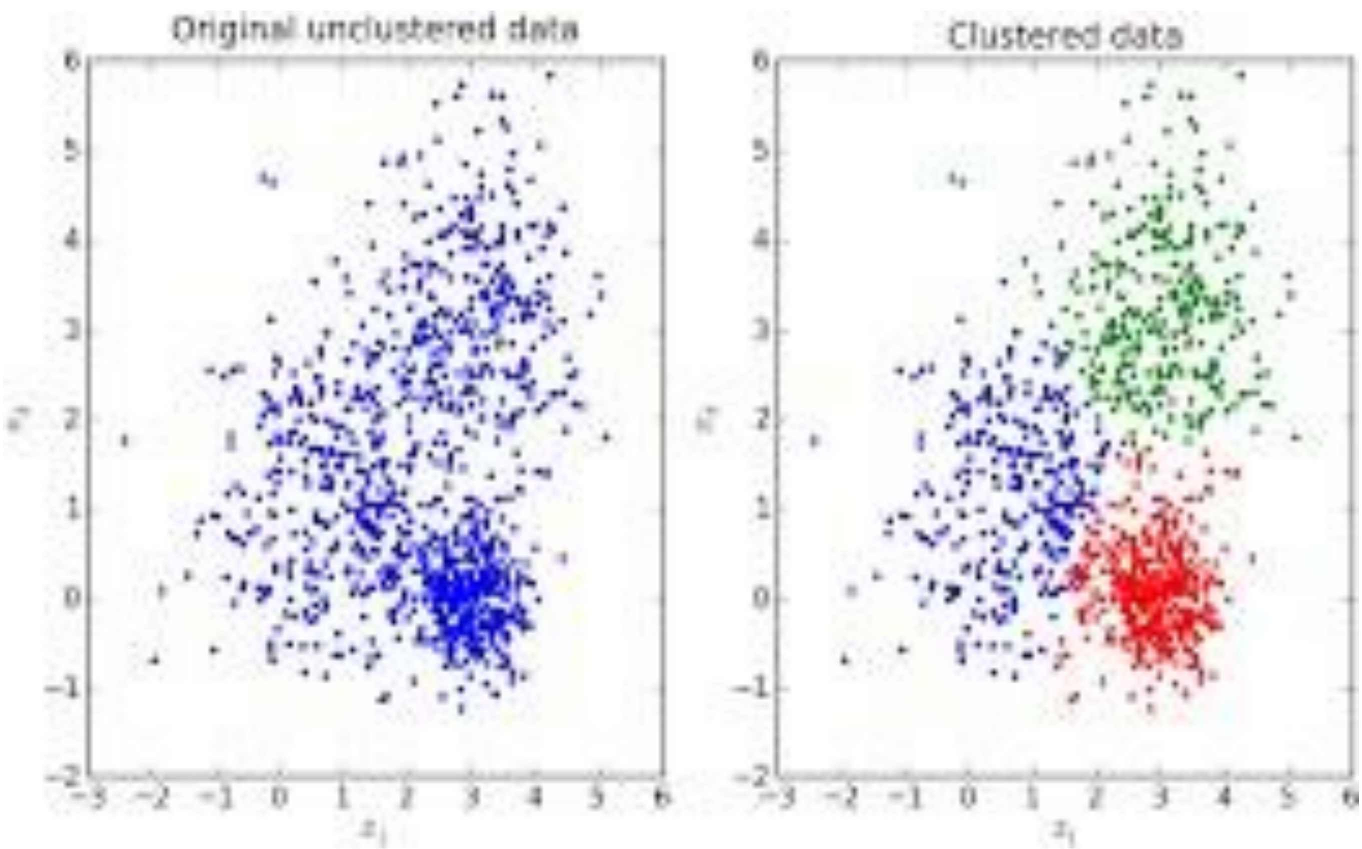


Figure 2.6 K-Means clustering

HierarchicalClustering

It's also known as hierarchical cluster analysis, is an algorithm that groups similar objects into groups called clusters. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other.

Hierarchical clustering starts by treating each observation as a separate cluster. Then, it repeatedly executes the following two steps:
(1) identify the two clusters that are closest together, and (2) merge the two most similar clusters. This iterative process continues until all the clusters are merged together. This is illustrated in the diagrams below.

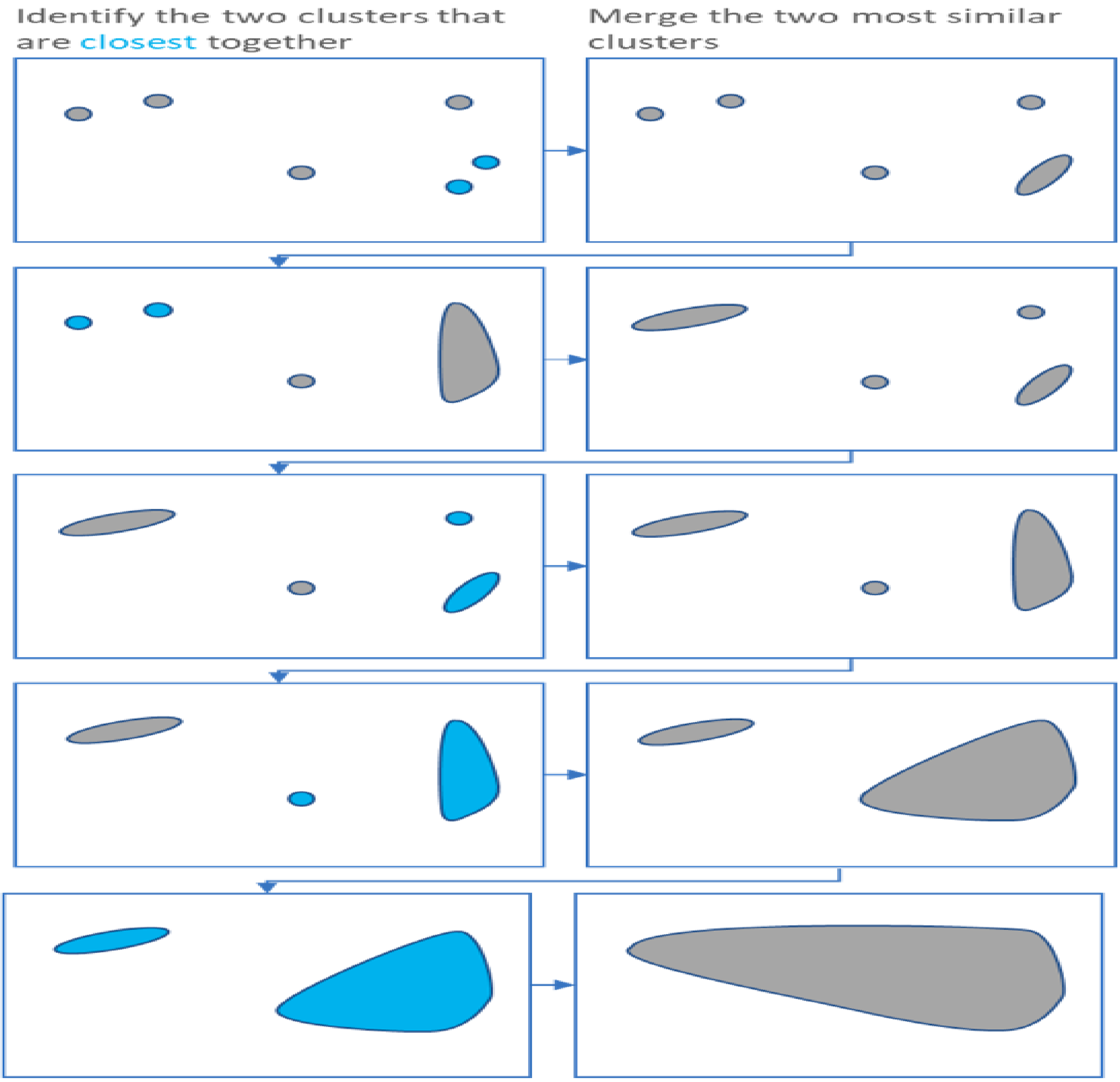


Figure 2.7 Hierarchical Clustering

CHAPTER 3 : DATA PREPARATION AND MODELING

IMPORTING LIBRARIES

Importing libraries like numpy,pandas, scikit-learn, seaborn, matplotlib for mathematical calculation, store data in dataframes, data visulaization and for model preparation.

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split as tts
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,matthews_corrcoef as mat,confusion_matrix,roc_auc_score,classification_report
from sklearn.model_selection import GridSearchCV
from catboost import CatBoostClassifier
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
```

LOADING DATASET

The Data is loaded using the pandas library of Python. We create a dataframe using thelibrary for the dataset.

```
df=pd.read_csv("parkinsons.csv")
```

```
df.head()
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shir
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	

5 rows × 24 columns

CHECKING NULL VALUES

Check null values in the dataset. If there is any null values replace that with a integer like -1 or 999.

```
df.isnull().sum()
```

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

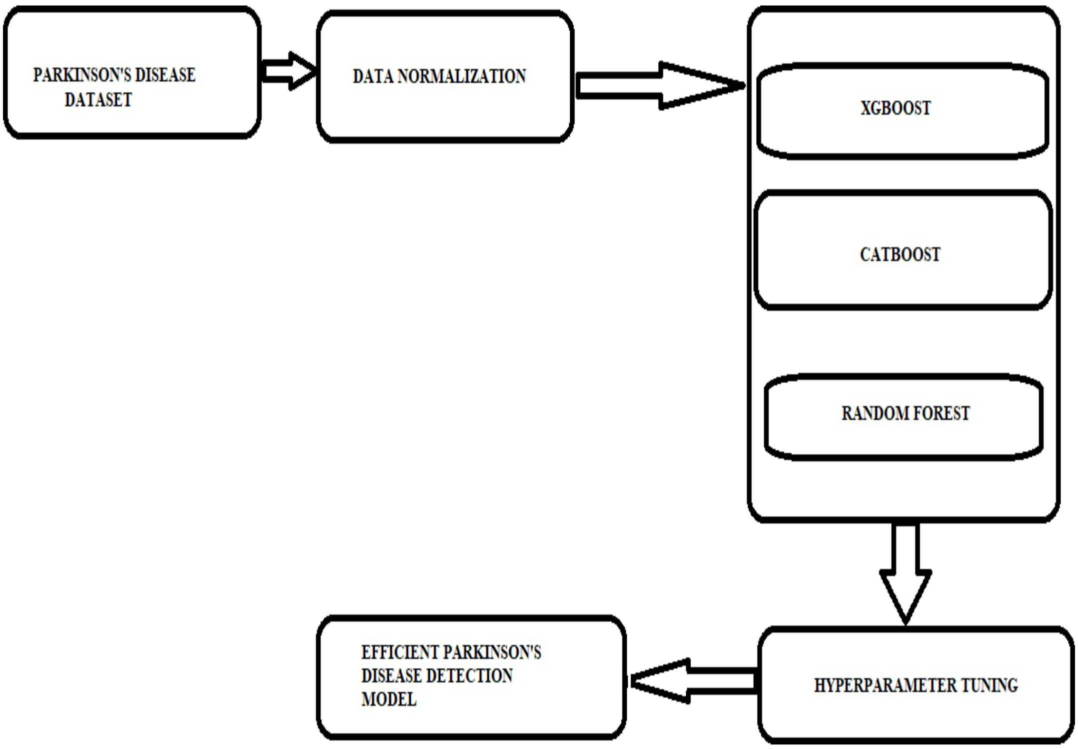


Figure 3.1. Model Preparation

NORMALIZE DATASET

Normalize the data show the value of independent column lies on a column scale which reduce the error in prediction and adjust the weight accordingly. After normalization all the value lies between 0 and 1.

```
x=df.drop(['status','name'],axis=1)
norm = MinMaxScaler().fit(x)
X= norm.transform(x)
```

SPLITTING DATASET

Split the dataset in test and train: 80% for training purpose and 20% for testing purpose to check how our model is doing prediction accurately.

```
x_train,x_test,y_train,y_test=tts(X,y,test_size=0.3,random_state=10)
```

BALANCING DATASET

Balancing of the dataset is required when there is a class in minority which made the dataset made the dataset biased towards the other class. We balanced our data using SMOTE (Synthetic Minority Oversampling Technique) with the help of imbalance library.

```
smk = SMOTE(random_state=5)
X_res,y_res=smk.fit_sample(x_train,y_train)
sns.countplot(x=y_res)
```

MODEL PREPARATION

Train the different model (like Random Forest, XGBoost and CatBoost) with train dataset and calculate the accuracy and MCC (Mathew Correlation Coefficient) by checking the error in the predicted value and select the best model to calculate the house price.

HYPERPARAMETER TUNING

The aim of hyperparameter tuning is to get the best possible parameter for our model. We did Hyperparameter tuning with the help of GridSearchCV

```
Grid_CBC = GridSearchCV(estimator=CBC, param_grid = parameters, cv = 3, n_jobs=-1, scoring='roc_auc')
Grid_CBC.fit(X_res, y_res)
```

cause it searches for best set of hyperparameters from a grid of hyperparameters values.

CHAPTER 4 : PROJECT CODE

4.1. CODE SNAPSHOT

In [14]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split as tts
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,matthews_corrcoef as mat,confusion_matrix,roc_auc_score,classification_report
from sklearn.model_selection import GridSearchCV
from catboost import CatBoostClassifier
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
```

In [ ]:

In [2]:
```
df=pd.read_csv("parkinsons.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shir
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	

5 rows × 24 columns

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
name                0
MDVP:Fo(Hz)         0
MDVP:Fhi(Hz)        0
MDVP:Flo(Hz)        0
MDVP:Jitter(%)      0
MDVP:Jitter(Abs)    0
MDVP:RAP            0
MDVP:PPQ            0
```

```
MDVP:PPQ            0
Jitter:DDP          0
MDVP:Shimmer        0
MDVP:Shimmer(dB)    0
Shimmer:APQ3        0
Shimmer:APQ5        0
MDVP:APQ            0
Shimmer:DDA         0
NHR                 0
HNR                 0
status              0
RPDE                0
DFA                 0
spread1             0
spread2             0
D2                  0
PPE                 0
dtype: int64
```

In [8]:
```
df.count()
```

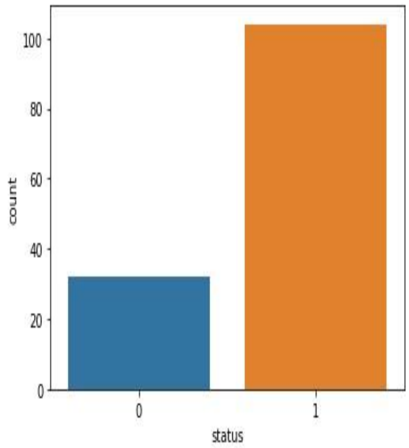
Out[8]:
```
name                195
MDVP:Fo(Hz)         195
MDVP:Fhi(Hz)        195
MDVP:Flo(Hz)        195
MDVP:Jitter(%)      195
MDVP:Jitter(Abs)    195
MDVP:RAP            195
MDVP:PPQ            195
Jitter:DDP          195
MDVP:Shimmer        195
MDVP:Shimmer(dB)    195
Shimmer:APQ3        195
Shimmer:APQ5        195
MDVP:APQ            195
Shimmer:DDA         195
NHR                 195
HNR                 195
status              195
RPDE                195
DFA                 195
spread1             195
spread2             195
D2                  195
PPE                 195
dtype: int64
```

In [21]:
```
x=df.drop(['status','name'],axis=1)
norm = MinMaxScaler().fit(x)
X= norm.transform(x)
```

```
y=df['status']
x_train,x_test,y_train,y_test=tts(X,y,test_size=0.3,random_state=10)
```

In [22]:
```
sns.countplot(x=y_train)
```

Out[22]: <AxesSubplot:xlabel='status', ylabel='count'>



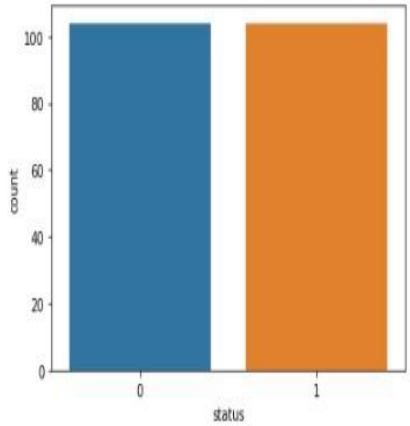
In [11]:
```
bestfeatures = SelectKBest(score_func=chi2, k=12)
fit = bestfeatures.fit(x_train,y_train)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']
featureScores['Specs'].values
print(featureScores.nlargest(12,'Score'))
```

```
               Specs     Score
2       MDVP:Flo(Hz)  3.355342
18           spread1  3.308019
21               PPE  3.150308
10      Shimmer:APQ3  2.656304
13       Shimmer:DDA  2.655040
8       MDVP:Shimmer  2.622205
11      Shimmer:APQ5  2.417668
9   MDVP:Shimmer(dB)  2.339823
12          MDVP:APQ  1.996787
19           spread2  1.455480
16              RPDE  1.425923
6           MDVP:PPQ  1.389665
```

In [12]:
```
smk = SMOTE(random_state=5)
X_res,y_res=smk.fit_sample(x_train,y_train)
```

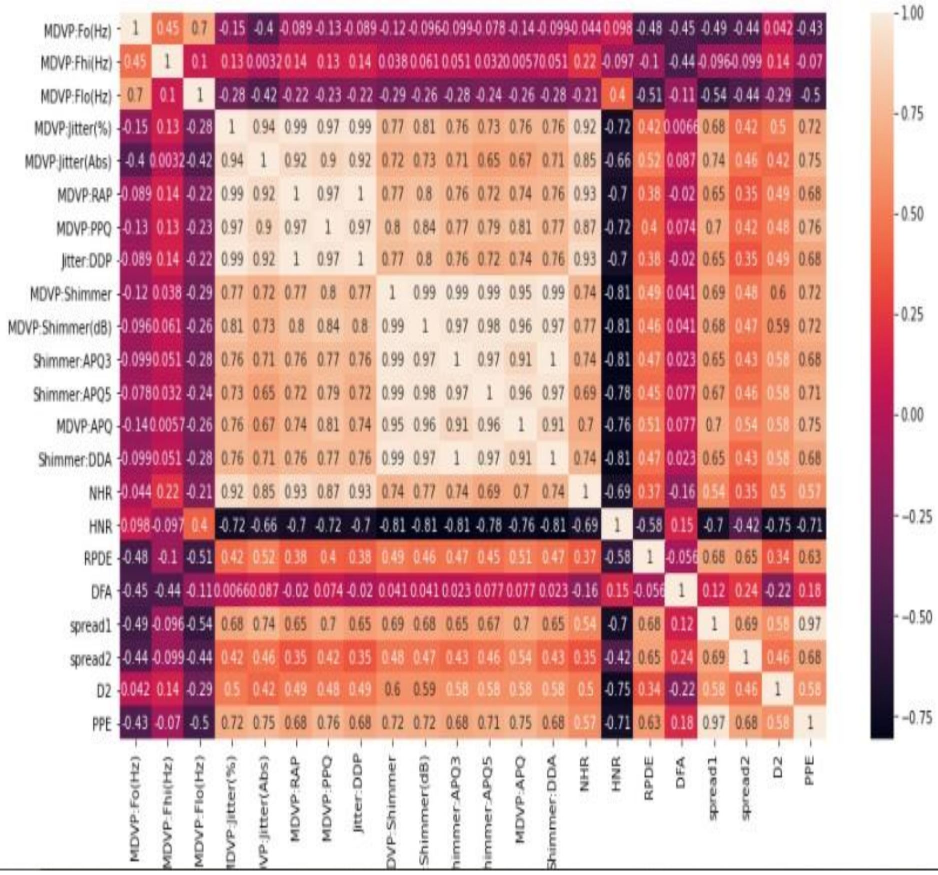
```
sns.countplot(x=y_res)
```

Out[12]: <AxesSubplot:xlabel='status', ylabel='count'>



In [24]:
```
corr=pd.DataFrame(X_res,columns=x.columns).corr()
plt.figure(figsize=(15,8))
sns.heatmap(corr,annot=True)
```

Out[24]: <AxesSubplot:>



```
In [113]: CBC = CatBoostClassifier()
parameters = {'depth'         : [4,5,6,7,8,9, 10],
              'learning_rate' : [0.01,0.02,0.03,0.04],
               'iterations'    : [10, 20,30,40,50,60,70,80,90, 100]
              }
Grid_CBC = GridSearchCV(estimator=CBC, param_grid = parameters, cv = 3, n_jobs=-1, scoring='roc_auc')
Grid_CBC.fit(X_res, y_res)
```

```
0:	learn: 0.6504315	total: 73.5ms	remaining: 3.6s
1:	learn: 0.6111848	total: 143ms	remaining: 3.44s
2:	learn: 0.5724511	total: 214ms	remaining: 3.36s
3:	learn: 0.5398092	total: 281ms	remaining: 3.23s
4:	learn: 0.5035354	total: 351ms	remaining: 3.15s
5:	learn: 0.4712113	total: 420ms	remaining: 3.08s
6:	learn: 0.4415604	total: 491ms	remaining: 3.01s
7:	learn: 0.4145727	total: 557ms	remaining: 2.92s
8:	learn: 0.3908228	total: 623ms	remaining: 2.84s
9:	learn: 0.3653889	total: 690ms	remaining: 2.76s
10:	learn: 0.3463940	total: 755ms	remaining: 2.68s
11:	learn: 0.3264386	total: 827ms	remaining: 2.62s
12:	learn: 0.3098187	total: 841ms	remaining: 2.39s
13:	learn: 0.2955969	total: 909ms	remaining: 2.34s
14:	learn: 0.2827522	total: 977ms	remaining: 2.28s
15:	learn: 0.2708706	total: 1.05s	remaining: 2.23s
16:	learn: 0.2617134	total: 1.13s	remaining: 2.19s
17:	learn: 0.2471553	total: 1.2s	remaining: 2.13s
18:	learn: 0.2368828	total: 1.27s	remaining: 2.08s
19:	learn: 0.2242652	total: 1.35s	remaining: 2.03s
20:	learn: 0.2153650	total: 1.43s	remaining: 1.97s
21:	learn: 0.2062277	total: 1.5s	remaining: 1.91s
22:	learn: 0.1988887	total: 1.58s	remaining: 1.86s
23:	learn: 0.1913115	total: 1.66s	remaining: 1.8s
24:	learn: 0.1843860	total: 1.73s	remaining: 1.73s
25:	learn: 0.1766381	total: 1.8s	remaining: 1.66s
26:	learn: 0.1691058	total: 1.86s	remaining: 1.59s
27:	learn: 0.1634434	total: 1.93s	remaining: 1.52s
28:	learn: 0.1565739	total: 2s	remaining: 1.45s
29:	learn: 0.1499341	total: 2.06s	remaining: 1.38s
30:	learn: 0.1439289	total: 2.13s	remaining: 1.3s
31:	learn: 0.1391029	total: 2.19s	remaining: 1.24s
32:	learn: 0.1332442	total: 2.27s	remaining: 1.17s
33:	learn: 0.1297694	total: 2.34s	remaining: 1.1s
34:	learn: 0.1265283	total: 2.42s	remaining: 1.04s
35:	learn: 0.1228433	total: 2.49s	remaining: 968ms
36:	learn: 0.1182102	total: 2.57s	remaining: 902ms
37:	learn: 0.1144968	total: 2.64s	remaining: 834ms
38:	learn: 0.1108843	total: 2.71s	remaining: 765ms
39:	learn: 0.1077750	total: 2.79s	remaining: 698ms
40:	learn: 0.1046697	total: 2.86s	remaining: 628ms
41:	learn: 0.1012985	total: 2.93s	remaining: 558ms
42:	learn: 0.0987516	total: 2.98s	remaining: 486ms
43:	learn: 0.0961321	total: 3.05s	remaining: 416ms
44:	learn: 0.0933564	total: 3.12s	remaining: 346ms
45:	learn: 0.0908700	total: 3.17s	remaining: 276ms
46:	learn: 0.0890591	total: 3.23s	remaining: 207ms
47:	learn: 0.0867856	total: 3.3s	remaining: 137ms
48:	learn: 0.0848308	total: 3.37s	remaining: 68.7ms
```

Out[113]:
```
GridSearchCV(cv=3,
             estimator=<catboost.core.CatBoostClassifier object at 0x000001747B6A51C8>,
             n_jobs=-1,
             param_grid={'depth': [4, 5, 6, 7, 8, 9, 10],
                         'iterations': [10, 20, 30, 40, 50, 60, 70, 80, 90,
                                        100],
                         'learning_rate': [0.01, 0.02, 0.03, 0.04]},
             scoring='roc_auc')
```

In [114]:
```
print(" Results from Grid Search " )
print("\n The best estimator across ALL searched params:\n",Grid_CBC.best_estimator_)
print("\n The best score across ALL searched params:\n",Grid_CBC.best_score_)
print("\n The best parameters across ALL searched params:\n",Grid_CBC.best_params_)
```

```
 Results from Grid Search 

 The best estimator across ALL searched params:
 <catboost.core.CatBoostClassifier object at 0x000001747B6A5B48>

 The best score across ALL searched params:
 0.9978071228491396

 The best parameters across ALL searched params:
 {'depth': 10, 'iterations': 50, 'learning_rate': 0.04}
```

In [11]:
```
new_model=CatBoostClassifier(depth=10, iterations=50, learning_rate= 0.04)
new_model.fit(X_res, y_res)
```

```
0:	learn: 0.6504315	total: 103ms	remaining: 5.06s
1:	learn: 0.6111848	total: 137ms	remaining: 3.3s
2:	learn: 0.5724511	total: 172ms	remaining: 2.69s
3:	learn: 0.5398092	total: 206ms	remaining: 2.37s
4:	learn: 0.5035354	total: 239ms	remaining: 2.15s
5:	learn: 0.4712113	total: 274ms	remaining: 2.01s
6:	learn: 0.4415604	total: 308ms	remaining: 1.89s
7:	learn: 0.4145727	total: 345ms	remaining: 1.81s
8:	learn: 0.3908228	total: 380ms	remaining: 1.73s
9:	learn: 0.3653889	total: 414ms	remaining: 1.66s
10:	learn: 0.3463940	total: 447ms	remaining: 1.58s
11:	learn: 0.3264386	total: 480ms	remaining: 1.52s
12:	learn: 0.3098187	total: 487ms	remaining: 1.39s
13:	learn: 0.2955969	total: 522ms	remaining: 1.34s
14:	learn: 0.2827522	total: 559ms	remaining: 1.3s
15:	learn: 0.2708706	total: 593ms	remaining: 1.26s
16:	learn: 0.2617134	total: 626ms	remaining: 1.21s
17:	learn: 0.2471553	total: 659ms	remaining: 1.17s
18:	learn: 0.2368828	total: 694ms	remaining: 1.13s
19:	learn: 0.2242652	total: 730ms	remaining: 1.09s
20:	learn: 0.2153650	total: 764ms	remaining: 1.05s
21:	learn: 0.2062277	total: 797ms	remaining: 1.01s
22:	learn: 0.1988887	total: 831ms	remaining: 976ms
23:	learn: 0.1913115	total: 867ms	remaining: 940ms
24:	learn: 0.1843860	total: 901ms	remaining: 901ms
25:	learn: 0.1766381	total: 937ms	remaining: 865ms
26:	learn: 0.1691058	total: 970ms	remaining: 826ms
27:	learn: 0.1634434	total: 1.01s	remaining: 790ms
```

```
Out[11]: <catboost.core.CatBoostClassifier at 0x27dd1d30988>
```

```
In [13]: y_pred=new_model.predict(x_test)
         acc2=accuracy_score(y_test,y_pred)
         acc2*100
```

```
Out[13]: 96.61016949152543
```

```
In [14]: print(classification_report(y_test,y_pred))
         math=mat(y_test,y_pred)
         math*100
```

```
              precision    recall  f1-score   support

           0       0.94      0.94      0.94        16
           1       0.98      0.98      0.98        43

    accuracy                           0.97        59
   macro avg       0.96      0.96      0.96        59
weighted avg       0.97      0.97      0.97        59
```

```
Out[14]: 91.42441860465115
```

```
In [15]: confusion_matrix(y_test,y_pred)
```

```
Out[15]: array([[15,  1],
                [ 1, 42]], dtype=int64)
```

```
In [37]: roc_auc_score(y_test,y_pred)
```

```
Out[37]: 0.9571220930232558
```

```
In [ ]:
```

```
In [ ]:
```

```
In [6]: XGB_model=XGBClassifier()
        params_xgb={
         "learning_rate"    : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30 ] ,
         "max_depth"        : [ 3, 4, 5, 6, 8, 10, 12, 15],
         "min_child_weight" : [ 1, 3, 5, 7 ],
         "gamma"            : [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
         "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ]

        }
        Grid_xgb = GridSearchCV(estimator=XGB_model, param_grid = params_xgb, cv = 3, n_jobs=-1, scoring='roc_auc')
        Grid_xgb.fit(X_res, y_res)
```

```
Out[6]: GridSearchCV(cv=3, estimator=XGBClassifier(), n_jobs=-1,
                     param_grid={'colsample_bytree': [0.3, 0.4, 0.5, 0.7],
                                 'gamma': [0.0, 0.1, 0.2, 0.3, 0.4],
                                 'learning_rate': [0.05, 0.1, 0.15, 0.2, 0.25, 0.3],
                                 'max_depth': [3, 4, 5, 6, 8, 10, 12, 15],
                                 'min_child_weight': [1, 3, 5, 7]},
```

```
                 scoring='roc_auc')
```

```
In [7]: print(" Results from Grid Search " )
        print("\n The best estimator across ALL searched params:\n",Grid_xgb.best_estimator_)
        print("\n The best score across ALL searched params:\n",Grid_xgb.best_score_)
        print("\n The best parameters across ALL searched params:\n",Grid_xgb.best_params_)

         Results from Grid Search

         The best estimator across ALL searched params:
         XGBClassifier(colsample_bytree=0.3, gamma=0.0, learning_rate=0.3)

         The best score across ALL searched params:
         0.9972468987595038

         The best parameters across ALL searched params:
         {'colsample_bytree': 0.3, 'gamma': 0.0, 'learning_rate': 0.3, 'max_depth': 3, 'min_child_weight': 1}
```

```
In [16]: xgb=XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                       colsample_bynode=1, colsample_bytree=0.5, gamma=0.0,
                       learning_rate=0.2, max_delta_step=0, max_depth=6,
                       min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                       nthread=None, objective='binary:logistic', random_state=0,
                       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
                       silent=None, subsample=1, verbosity=1)
         xgb.fit(X_res,y_res)

Out[16]: XGBClassifier(colsample_bytree=0.5, gamma=0.0, learning_rate=0.2, max_depth=6)
```

```
In [17]: predict_xgb=xgb.predict(x_test)
         accu_xgb=accuracy_score(y_test,predict_xgb)
```

```
In [18]: accu_xgb

Out[18]: 0.9491525423728814
```

```
In [19]: print(classification_report(y_test,predict_xgb))
         math2=mat(y_test,predict_xgb)
         math2

                       precision    recall  f1-score   support

                    0       0.88      0.94      0.91        16
                    1       0.98      0.95      0.96        43

             accuracy                           0.95        59
            macro avg       0.93      0.95      0.94        59
         weighted avg       0.95      0.95      0.95        59


Out[19]: 0.8746154593939881
```

```
In [ ]:
```

In [38]:
```
random_model=RandomForestClassifier()
params={'max_depth':[3,4,5,6,8,9,10],
        'n_estimators':[100,150,200,250,400,500,700,800,900,1000,1050],
        'criterion':['gini','entropy']
       }
grid_model=GridSearchCV(estimator=random_model,param_grid=params,n_jobs=-1,cv=3,return_train_score=True,scoring='roc_auc')
grid_model.fit(X_res,y_res)
```

Out[38]:
```
GridSearchCV(cv=3, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 4, 5, 6, 8, 9, 10],
                         'n_estimators': [100, 150, 200, 250, 400, 500, 700,
                                          800, 900, 1000, 1050]},
             return_train_score=True, scoring='roc_auc')
```

In [39]:
```
print(" Results from Grid Search " )
print("\n The best estimator across ALL searched params:\n",grid_model.best_estimator_)
print("\n The best score across ALL searched params:\n",grid_model.best_score_)
print("\n The best parameters across ALL searched params:\n",grid_model.best_params_)
```

```
 Results from Grid Search 

 The best estimator across ALL searched params:
 RandomForestClassifier(criterion='entropy', max_depth=8, n_estimators=150)

 The best score across ALL searched params:
 0.9969827931172469

 The best parameters across ALL searched params:
 {'criterion': 'entropy', 'max_depth': 8, 'n_estimators': 150}
```

In [ ]:

In [20]:
```
model=RandomForestClassifier(criterion='gini',max_depth= 9, n_estimators= 200)
model.fit(X_res,y_res)
```

Out[20]:
```
RandomForestClassifier(max_depth=9, n_estimators=200)
```

In [21]:
```
predict_random=model.predict(x_test)
accu_random=accuracy_score(y_test,predict_random)
accu_random
```

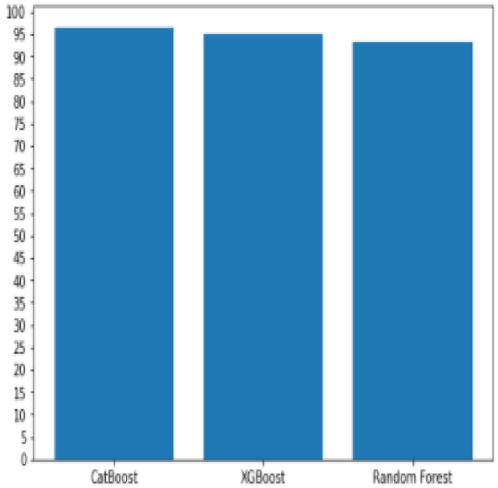
Out[21]:
```
0.9322033898305084
```

In [22]:
```
print(classification_report(y_test,predict_random))
math3=mat(y_test,predict_random)
math3
```

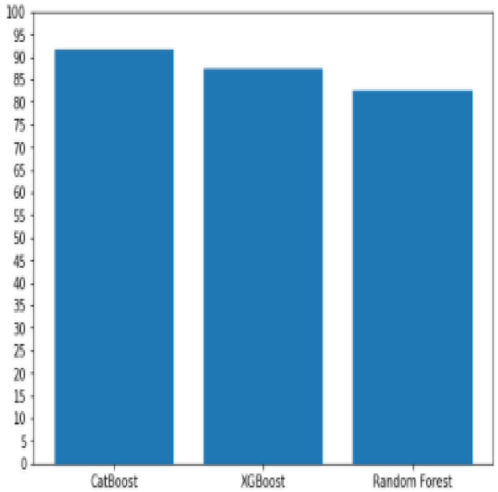
```
              precision    recall  f1-score   support

           0       0.93      0.81      0.87        16
           1       0.93      0.98      0.95        43

    accuracy                           0.93        59
```

Out[22]: 0.8247747100021534

In [37]:
```
fig = plt.figure()

ax = fig.add_axes([0,0,1,1])
models = ['CatBoost','XGBoost','Random Forest']
accuracy = [acc2*100,accu_xgb*100,accu_random*100]
ax.bar(models,accuracy)
plt.yticks(np.arange(0, 105, 5))
plt.show()
```



In [38]:
```
fig = plt.figure()

ax = fig.add_axes([0,0,1,1])
models = ['CatBoost','XGBoost','Random Forest']
accuracy = [math*100,math2*100,math3*100]
ax.bar(models,accuracy)
plt.yticks(np.arange(0, 105, 5))
plt.show()
```



In [ ]:

CHAPTER 5 : CONCLUSION & FUTURE SCOPE

Early detection of Parkinson's diseases is very useful as it will helps to prevent the patients from worst stage. From this study we analyse the different machine learning algorithm like CatBoost, XGBoost and Random Forest and got a efficient Parkinson's Disease prediction model with high accuracy-96.61% and high MCC-91.42% which will help to predict parkinon before getting it to worst.

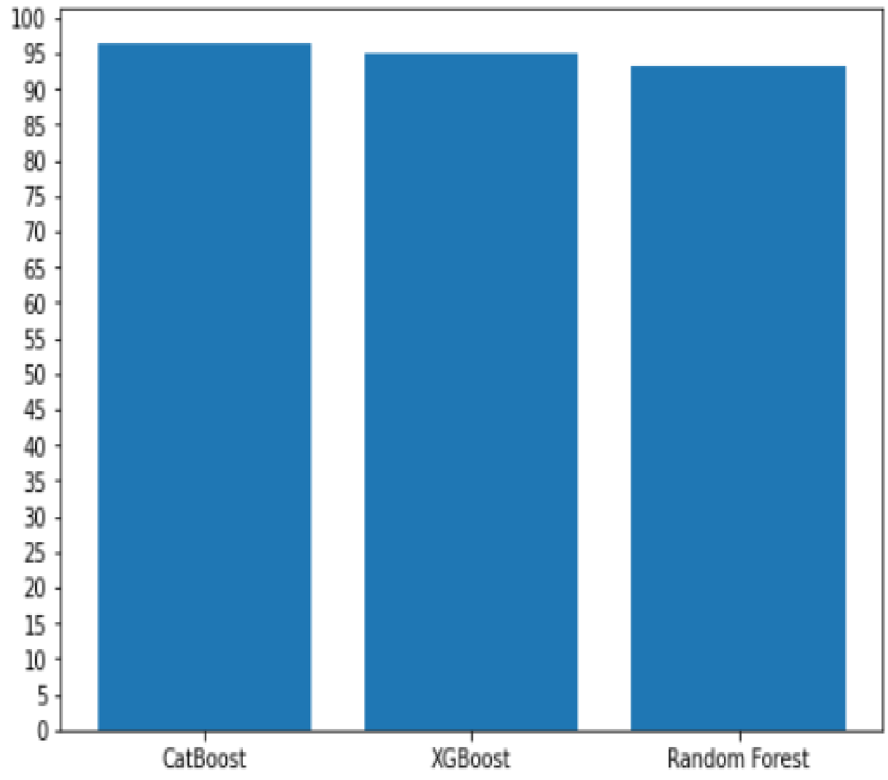


Figure 5.1 Accuracy Comparision

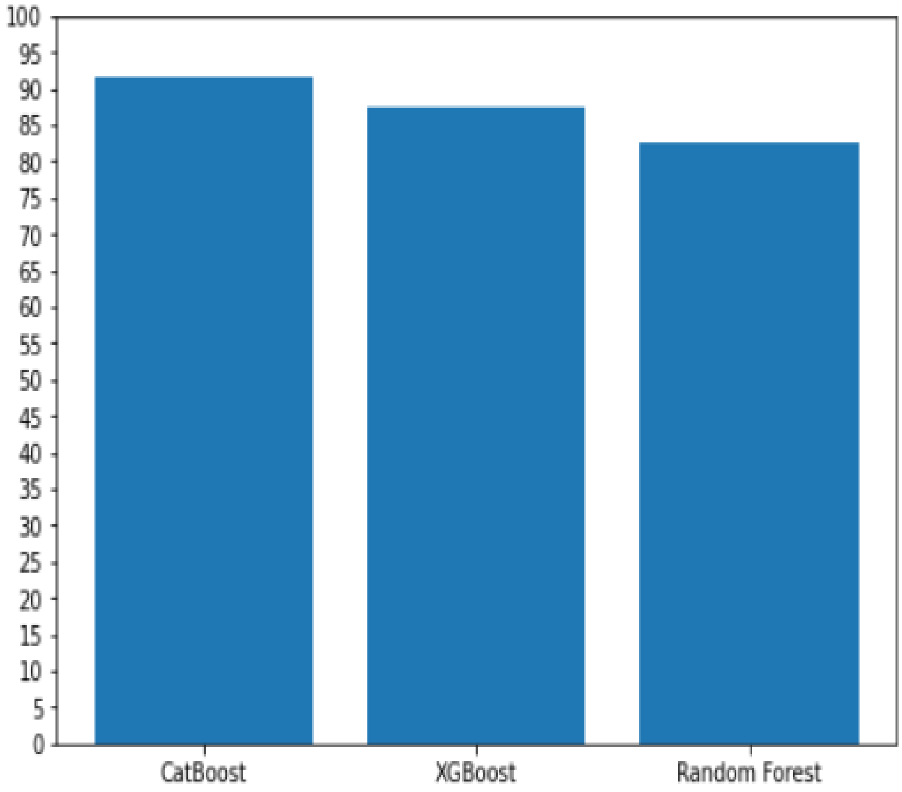


Figure 5.2 MCC Comparision

REFERENCES

[1] https://www.geeksforgeeks.org/decision-tree/

[2] https://dataconomy.com/2015/01/whats-the-difference-between-supervised-and-unsupervised-learning/

[3] https://www.geeksforgeeks.org/supervised-unsupervised-learning/

[4] https://towardsdatascience.com/types-of-machine-learning-algorithms- you-should-know-953a08248861

[5] https://www.displayr.com/what-is-hierarchical-clustering/#:~:text=Hierarchical%20clustering%2C%20also%20known%20as,broadly%20similar%20to%20each%20other

[6] https://www.javatpoint.com/unsupervised-machine-learning