**DETECTION OF PARKINSON’S DISEASE USING MACHINE LEARNING**

## A project report submitted in partial fulfilment of the requirement for the Award of the Degree of

**BACHELOR OF ENGINEERING**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**by**

## T Kamaljeet (160718733037)

**P Saritha (160718733047)**

**R Vishal (160718733040)**

***Under the Guidance of***

## Mr. P. V. Ramanaiah , Assistant Professor, Dept. of CSE



**Department of Computer Science and Engineering Methodist College of Engineering and Technology, King Koti, Abids, Hyderabad-500001.**

## 2021-2022

**Methodist College of Engineering and Technology, King Koti, Abids, Hyderabad-500001.**

**Department of Computer Science and Engineering**



**DECLARATION BY THE CANDIDATES**

I, **T Kamaljeet (160718733037), P Saritha (160718733047)** and **R Vishal (160717733040)** students of Methodist College of Engineering and Technology, pursuing Bachelor’s degree in Computer Science and Engineering, hereby declare that this Project report entitled “**Detection Of Parkinson’s Disease Using Machine Learning",** carried out under the guidance of **Mr. P. V Ramanaiah** submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science. This is a record work carried out by us and the results embodied in this report have not been reproduced/copied from any source.

#### T Kamaljeet (160718733037)

**P Saritha (160718733047)**

**R Vishal (160718733040)**

**Methodist College of Engineering and Technology, King Koti, Abids, Hyderabad-500001.**

**Department of Computer Science and Engineering**



**CERTIFICATE BY THE SUPERVISOR**

This is to certify that this Internship report entitled “**Detection Of Parkinson’s Disease Using Machine Learning**” being submitted *by* T Kamaljeet (160718733037), P Saritha (160718733047) and R Vishal (160718733040) submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering, during the academic year 2021-2022, is a bonafide record of work carried out by them.

#### Mr. P.V. Ramanaiah

#### Assistant Professor,

#### Date: Dept. of CSE.

**Methodist College of Engineering and Technology,**

**King Koti , Abids , Hyderabad-500001.**

**Department of Computer Science and Engineering**



**CERTIFICATE BY HEAD OF THE DEPARTMENT**

This is to certify that this Internship report entitled “**Detection Of Parkinson’s Disease Using Machine Learning**” by **T Kamaljeet (160717733037), P Saritha (160718733047) and R Vishal (160718733040)** submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering of the Osmania University, Hyderabad, during the academic year 2021-2022, is a bonafide record of work carried out by them.

**Dr. P. Lavanya**

Professor &

Date: Head of the Department

**Methodist College of Engineering and Technology,**

**King Koti, Abids, Hyderabad-500001.**

**Department of Computer Science and Engineering**



**PROJECT APPROVAL CERTIFICATE**

This is to certify that this project report entitled “**Detection Of Parkinson’s Disease Using Machine Learning**” by **T Kamaljeet (160717733037), P Saritha (160718733047) and R Vishal (160718733040)**, submitted in partial fulfillment of the requirements for the degree of Bachelor of Engineering in Computer Science and Engineering of the Osmania University, Hyderabad, during the academic year 2019-2020, is a bonafide record of work carried out by them.

#### INTERNAL EXTERNAL HOD

**ACKNOWLEDGEMNTS**

We would like to express our sincere gratitude to my Project guide **Mr.P.V.Ramanaiah, Assistant Professor,** for giving us the opportunity to work on this topic. It would never be possible for us to take this project to this level without his innovative ideas and his relentless support and encouragement.

We would like to thank our project coordinator **Mr. T. Praveen Kumar, Assistant Professor,** who helped us by being an example of high vision and pushing towards greater limits of achievement.

Our sincere thanks to **Dr. P. Lavanya, Professor** and **Head of the Department of Computer Science and Engineering,** for her valuable guidance and encouragement which has played a major role in the completion of the project and for helping us by being an example of high vision and pushing towards greater limits of achievement.

We would like to express a deep sense of gratitude towards the **Dr. Prabhu G. Benakop, Principal, Methodist College of Engineering and Technology,** for always being an inspiration and for always encouraging us in every possible way.

We would like to express a deep sense of gratitude towards the **Dr. Lakshmipathi Rao, Director, Methodist College of Engineering and Technology,** for always being an inspiration and for always encouraging us in every possible way.

We are indebted to the Department of Computer Science & Engineering and Methodist College of Engineering and Technology for providing us with all the required facility to carry our work in a congenial environment. We extend our gratitude to the CSE Department staff for providing us to the needful time to time whenever requested.

We would like to thank our parents for allowing us to realize our potential, all the support they have provided us over the years was the greatest gift anyone has ever given us and also for teaching us the value of hard work and education. Our parents has offered us with tremendous support and encouragement, thanks to our parents for all the moral support and the amazing opportunities they have given us over the years.



**METHODIST COLLEGE OF**

**ENGINEERING & TECHNOLOGY**

Approved by AICTE,Affiliated to Osmania University, - College Code – 1607

**Department of Computer Science & Engineering**

## Vision & Mission

**VISION**

To become a leader in providing Computer Science & Engineering education with emphasis on knowledge and innovation.

## MISSION

**M1:** To offer flexible programs of study with collaborations to suit industry needs

**M2:** To provide quality education and training through novel pedagogical practices

**M3: To** Expedite high performance of excellence in teaching, research and innovations. **M4:** To impart moral, ethical valued education with social responsibility.

**Program Educational Objectives**

#### Graduates of Compute Science and Engineering at Methodist College of Engineering and Technology will be able to:

**PEO1:** Apply technical concepts, Analyze, Synthesize data to Design and create novel products and solutions for the real life problems.

**PEO2:** Apply the knowledge of Computer Science Engineering to pursue higher education with due consideration to environment and society.

**PEO3:** Promote collaborative learning and spirit of team work through multidisciplinary projects

**PEO4:** Engage in life-long learning and develop entrepreneurial skills.

**Program Specific Outcomes**

#### At the end of 4 years, Compute Science and Engineering graduates at MCET will be able to:

**PSO1:** Apply the knowledge of Computer Science and Engineering in various domains like networking and data mining to manage projects in multidisciplinary environments.

**PSO2:** Develop software applications with open-ended programming environments**.**

**PSO3:** Design and develop solutions by following standard software engineering principle and implement by using suitable programming languages and platforms

METHODIST COLLEGE OF ENGINEERING & TECHNOLOGY

Approved by AICTE,Affiliated to Osmania University, - College Code – 1607

**PROGRAM OUTCOMES**

**PO1:**Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2:**Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO3:**Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO4:**Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5:**Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6:**The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO7:**Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO8:**Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9:**Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO10**: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO11:**Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO12:**Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

**ABSTRACT**

In present-day life, Parkinson's disease is a brain disorder that leads to shaking, stiffness, and difficulty with walking, balance, and coordination. Parkinson's symptoms usually begin gradually and get worse over time. As the disease progresses, people may have difficulty walking and talking. Accurately detecting Parkinson's disease at an early stage is certainly indispensable for slowing down its progress and providing patients the possibility of accessing to disease-modifying therapy. Most Parkinson Disease patients suffer from vocal cord disorders. Speech impairment is an early indicator of Parkinson Disease.

Biomarkers derived from human voice can offer in-sight into neurological disorders, such as Parkinson's disease (PD), because of their underlying cognitive and neuromuscular function. PD is a progressive neurodegenerative disorder that affects about one million people in the the United States, with approximately sixty thousand new clinical diagnoses made each year. Historically, PD has been difficult to quantity and doctors have tended to focus on some symptoms while ignoring others, relying primarily on subjective rating scales. Due to the decrease in motor control that is the hallmark of the disease, voice can be used as a means to detect and diagnose PD. With advancements in technology and the prevalence of audio collecting devices in daily lives, reliable models that can translate this audio data into a diagnostic tool for healthcare professionals would potentially provide diagnoses that are cheaper and more accurate. We provide evidence to validate this concept here using a voice dataset collected from people with and without PD. This paper explores the effectiveness of using supervised classification algorithms, such as deep neural networks, to accurately diagnose individuals with the disease. Our peak accuracy of 85% provided by the machine learning models exceed the average clinical diagnosis accuracy of non-experts (73.8%) and average accuracy of movement disorder specialists (79.6% without follow-up, 83.9% after follow-up) with pathological post-mortem examination as ground truth.

# TABLE OF CONTENTS

1. Introduction …………………………………………………………………………......1
2. [Literature](#_TOC_250028) Survey………………………………………………………………….…......4
3. [Design analysis…………………………………………………………………..….…...7](#_TOC_250027)

3.1 [System Architecture…………………………………………………………..……..](#_TOC_250026).7

3.2 UML Diagrams……………………………………………………………….……...9

3.2.1 [Class diagram ……………………………………………………………...…..9](#_TOC_250025)

3.2.2 Use Case diagram………………………………………….…………….…...10

3.2.3 [Sequence diagram…………………………………………………………….11](#_TOC_250023)

3.2.4 Activity diagram……………………………………………………...……....12

3.2.5 Collaboration diagram……………………………………….……………….13

3.2.6 Component diagram……………………………………………..……………14

1. Modules and Implementation……………………………………………………………15

4.1 [List Of Modules…………………………………………………………….……….16](#_TOC_250020)

4.2 [Libraries………………………………………….……………………….………….](#_TOC_250019)17

4.3 [Logistic Regression…………………………………………………………………..25](#_TOC_250011)

4.4 KNN…………………………………………………………….……………………28

4.5 SVM………………………………………………………………………………….32

4.6 Stacking………………………………………………………………….…………...37

4.7 Random Forest…………………………………………………………..……………38

4.8 Adaptive Boosting……………………………………………………………………42

5. Testing……………………………………………………………….…………………..44

5.1 [Unit testing…………………………………………………………………………...45](#_TOC_250007)

5.2 [Integration testing…………………………………………………………………....45](#_TOC_250005)

5.3 Validation testing…………………………………………………………………….46

5.4 System testing……………………………………………………………….……….46

5.5 Acceptance testing………………………………………..………………………….47

5.6 Integrated Manual testing……………………………………………………………48

1. Output Screenshots………………………………………………………………………49
2. Conclusion and Future Scope ……………………………..………….…………………55
3. References……………………………………………………………….………………57

8.1Appendix A: Technology used……………………………………………………....59

* 1. Appendix B: Software and hardware requirements………………………………….63
  2. [Appendix C: Sample Code…………………………………………………………..65](#_TOC_250000)

1. QR code scanner……………………………………………………………………….107

**CHAPTER 1**

**INTRODUCTION**

1. **INTRODUCTION**

In Parkinson’s disease is a neurodegenerative disorder of central nervous system that causes partial or full loss of motor reflexes, speech, behavior, mental processing, and other vital functions. It is generally observed in elderly people and causes disorders in speech and motor abilities of 90% of the patients. People with Parkinson’s disease suffer from speech impairments like dysphonia (defective use of the voice), hypophonia (reduced volume), monotone (reduced pitch range), and dysarthria (difficulty with articulation of sounds or syllables). In this project voice parameters of Parkinson’s disease patients and healthy subjects will be analyzed to predict the presence of Parkinson’s disease. To accomplish improved accuracy of the models used for detecting and predicting Parkinson’s disease. To build supervised machine learning algorithms to train the models on cleaned UCI Parkinson's dataset. To take voice features as parametric input and check if a person has Parkinson’s disease or not.To implement and monitor activities of patient suffering from Parkinson's disease and stores their history.

Parkinson’s disease is a disorder that affects small regions in the brain that control movement, posture and balance. It is a complex disease that has many different symptoms, so that not everyone with the condition suffers from the same problems.

Parkinson’s disease is named after the British doctor who wrote the first book about the disease, in 1817, that made it an easily recognized entity. Parkinson called it, “The Shaking palsy,” or “paralysis agitans.” In his day, the term “agitans” referred to tremors. “Palsy” meant weakness and “paralysis” meant paralyzed, so the condition was considered a disorder of weakness and tremors, which is not completely true, as we shall see. Parkinson was famous in his day because of his political activities as an advocate for the poor, his scientific publications on geology and his invention of the truss, in the days before surgery for hernias was available.

It took 100 years before the major brain changes were first noted in the brains of people with Parkinson’s disease, and another 50 years after that before most experts agreed that these changes were, in fact, the disease process itself. In the early 1960’s the understanding of the importance of the brain chemical, dopamine (**d**i-**o**rtho-**p**henyl-**a**lanine), came about, followed shortly thereafter by the first substantive treatment for Parkinson’s disease, L-Dopa. The

treatment for Parkinson’s Disease was very poor before this. Parkinson himself recommended that a vertical incision be made in the back of the neck and that cork should be inserted to keep the wound from healing. He thought that the pus that developed in the infection represented the diseased fluids that were causing the disease so that drawing off this material would improve the patient. As you probably would guess, this never became too popular. L-Dopa wasn’t very popular at first either, even though it had a much better scientific rationale, because it caused so much nausea and vomiting.

Carbodopa was developed soon thereafter, to prevent the vomiting, leading to the combined medication, carbidopa/levodopa, marketed as Sinemet (sine=without, emesis=vomiting). This drug remains the single best drug we have for treating the symptoms of Parkinson’s disease.

PD is common. It affects about 500,000- one million Americans, or about 1% of people over the age of 60. After Alzheimer’s disease, it is the second most common progressive, neurological disorder in the US. In the state of Rhode Island alone, with a population of only one million people, there are an estimated 1-2,000 people with PD. Although there is a large amount of research on PD, we still don’t know what causes it. And we even have some trouble diagnosing it at times.

The more that Parkinson’s Disease is studied, the more intricate we find the condition to be. Although most people think of it as a disorder of movement, posture and balance, it may also affect behavior. Only over the past 15 years or so have doctors started to investigate this important area. Treatment od PD have focused until recently only the movements, but we now see the disorder in a more realistic, holistic way. Quality of life has become a more central focus as we deal with a disease we cannot yet cure.

# 

# 

# CHAPTER 2

# LITERATURE SURVEY

# 2.LITERATURE SURVEY

There are several applications on Diseases available on the internet and several publications on Parkinson’s Disease Detection. Some of them which were related to our project were listed.

**Betul Erdogdu Sakar, M. Erdem Isenkul**, a Introduced a dataset which was collection of voice samples from 40 individuals out of which 20 were healthy and 20 were suffering from Parkinson disease. Compared the leave one subject out and summarised leave one out validation schemes on classification algorithms KNN and SVM. Results showed that s-loso performed much better than the loo method. Mean and standard deviation were found to better metrics for summarization of the features obtained from the voice samples rather than considering each and every sample of a subject. Main drawback is the model gave low accuracy results with KNN using loso and s-loso schemes since vowels carried more weight on PD classification rather than sentences or words

**Ali KARATUTLU**, developed Random subspace ensemble method using knn to improve the performance of individual classifiers. With the help of k-fold cross validation KNN, was used as the base classifiers. These were applied on normalized features of the Istanbul PD dataset. They were tested on varied number of features and with number of KNN, and it was found that ensemble of KNN learners out performed. It was seen that with k valued 10 and with 114 KNN classifiers on 7 dimensional subspaces demonstrated lowest classification error. It was also shown that variation and the type of base classifier varied the performance of the random subspace classifier ensemble method. Main drawback is random selection of feature subspaces because some of the randomly selected subsets might have poor discrimination capability 4.

**Sajid Ullah Khan** Proposed idea cluster analysis which is an iterative process was applied to modify data pre-processing and model parameters until the required properties are achieved. • The data is passed through the data pre-processing phases that are data cleaning, recovering missing values and transformed. After which the three clustering techniques are applied. • The techniques that are use are K-NN, Random Forest and Ada-Boost This is done to get the accurate model for detecting disease.The accuracy achieved were 90.25%, 87.17, 88.71% for K-NN, Random Forest and Ada-Boost respectively. It was observed from the results that K-NN was the best model for classification with highest accuracy.

**Miran Brezocnik** Developed Multiple ANNs were used on PD Dataset using LOSO scheme. LOSO has far less bias, and provides practically unbiased prediction. • The method involved feature selection and then selecting best result from multiple ANN classifiers using majority voting technique. Feature selection was done using Pearsons, Kendalls correlation coefficient, pca and self-organizing maps. • The number 4 and short sentence 4 from the voice samples contained more value in classifying PD according to Pearson and Kendall’s correlation coefficients. NN is fine-tuned, and a test accuracy of 86.47% was achieved. • Main drawback is performance of ANNs’ could be improved by using other feature selection procedures and by additional work on fine-tuning. Several vocal tests in other languages were not included in the study and performing the classification on those datasets would help increase the reach of the models in predicting PD.

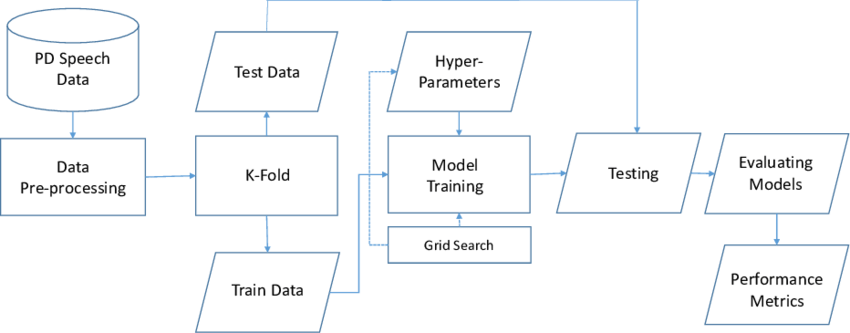
The system proposes a method for detecting Parkinson’s disease using the voice features using supervised machine learning classification algorithms – K Nearest Neighbor, Support Vector Machines, Random forest and Decision Trees.

An application where data about patients can be recorded which can be accessed and reviewed by doctors to monitor the condition of the patients is also proposed.

# CHAPTER 3 DESIGN ANALYSIS

**3. DESIGN ANALYSIS**

## 3.1 System Architecture

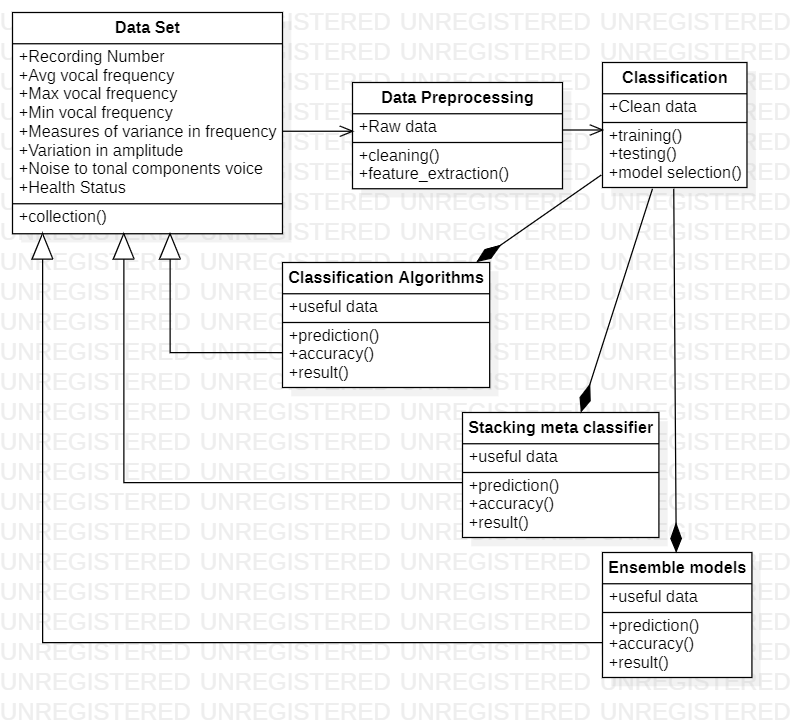


**Fig 3.1 Architecture Overview**

## A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.The system Architecture contains many modules where data flows from PD speech data to performance metrics.

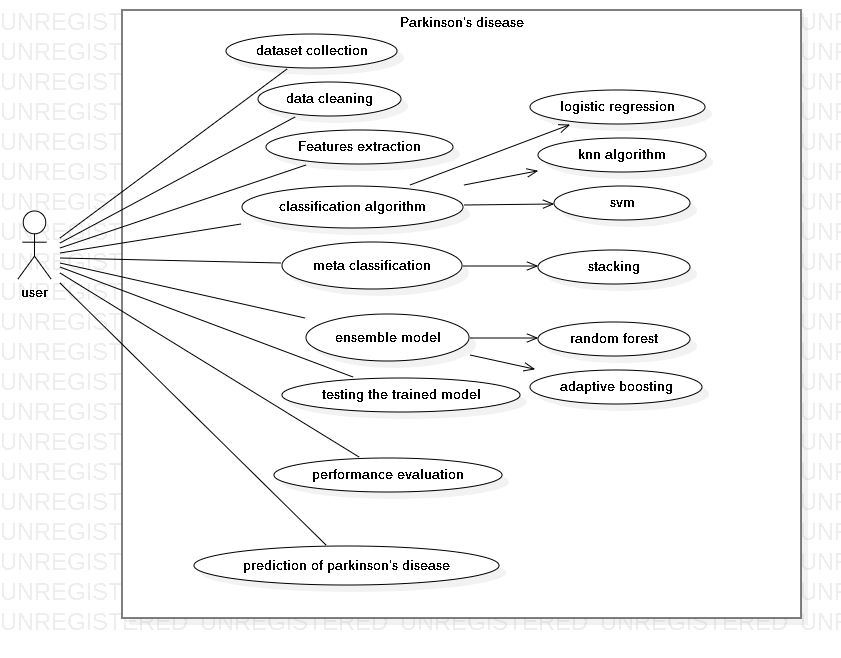
## 3.2 UML Diagrams

**3.2.1 Class Diagram**



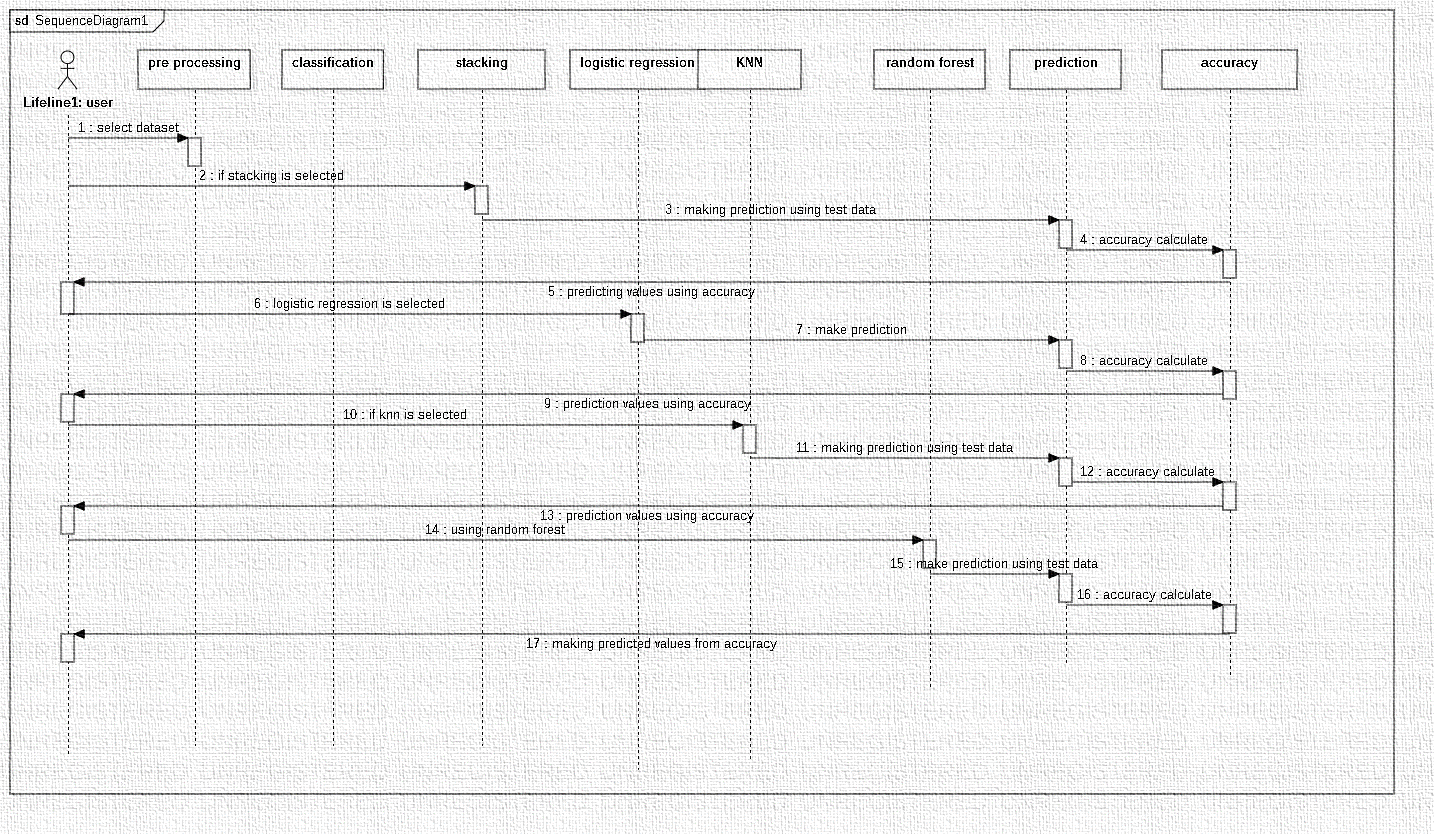
These figure shows the class diagram. Here dataset is the main class which includes the above specified attributes and operations. The data pre processing, contains the attribute and operations, where coming to classification algorithm which contains different algorithms to calculate the accuracy

**3.2.2 Use Case Diagram**



The figure shows use case diagram. Here the only actor is the user. He performs the various operations like dataset collection, data cleaning ,features extraction ,classification algorithm ,meta classification, ensemble model ,testing the trained model, performance evaluation, prediction of Parkinson disease

**3.2.3 Sequence Diagram**



The figure shows sequence diagram. The user gives the input to the data pre processing where it contains the data set in which it perform the different algorithm to get the accuracy of the patience.

**3.2.4 Activity Diagram**



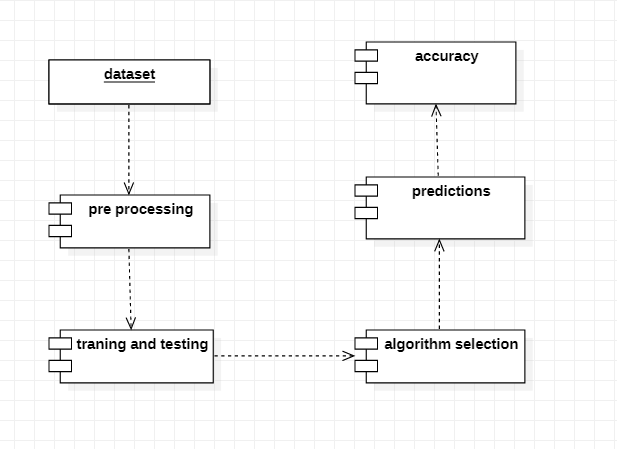
Figure shows the activity Diagram .The user gives it input then the pre processing check that the input is valid or not then it divide the algorithm for it like classification algorithm, meta classifier, ensemble models, which then combine to calculate the accuracy.

#### 3.2.5 Collaboration Diagram

#### 

The figure show the collaboration diagram. The user sends the data set to the pre processor for predicting the accuracy.

**3.2.6 Component diagram**



The component diagram includes the steps involved in the development process of project, that is the machine learning components.

# CHAPTER 4

**MODULES AND IMPLEMENTATION**

**4.MODULES AND IMPLEMENTATION**

**4.1 Component Design / Module Decomposition**

**Machine Learning Components**

**Dataset collection:** In this module, we collect the data from UCI dataset archives. This dataset contains the information of voice parameters and the presence of Parkinson’s disease.

**Data Cleaning:** In this module data cleaning is done to prepare the data for analysis by removing or modifying the data that may be incorrect, incomplete, duplicated or improperly formatted.

**Feature Extraction:** This is done to reduce the number of attributes in the dataset hence providing advantages like speeding up the training and accuracy improvements. Component Design / Module Decomposition.

**Model training :** In this module we use supervised classification algorithms like knn, svm, random forest to train the model on the cleaned dataset after dimensionality reduction.

**Testing the trained model:** In this module we test the trained machine learning model using the test dataset.

**Performance Evaluation:** In this module, we evaluate the performance of trained machine learning model using performance evaluation criteria such as F1 score, accuracy and classification error. In case the model performs poorly, we optimize the machine learning algorithms to improve the performance.

**Prediction of Parkinson’s disease:** In this module we use trained and optimized machine learning model to predict whether the patient has Parkinson’s disease or not using the voice features.

**4.2 LIBRARIES:**

**NUMPY**

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.

The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.

A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python-based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

**WHY Numpy fast?**

Vectorization describes the absence of any explicit looping, indexing, etc., in the code - these things are taking place, of course, just “behind the scenes” in optimized, pre-compiled C code. Vectorized code has many advantages, among which are:

vectorized code is more concise and easier to read

fewer lines of code generally means fewer bugs

the code more closely resembles standard mathematical notation (making it easier, typically, to correctly code mathematical constructs)

vectorization results in more “Pythonic” code. Without vectorization, our code would be littered with inefficient and difficult to read for loops.

Broadcasting is the term used to describe the implicit element-by-element behavior of operations; generally speaking, in NumPy all operations, not just arithmetic operations, but logical, bit-wise, functional, etc., behave in this implicit element-by-element fashion, i.e., they broadcast. Moreover, in the example above, a and b could be multidimensional arrays of the same shape, or a scalar and an array, or even two arrays of with different shapes, provided that the smaller array is “expandable” to the shape of the larger in such a way that the resulting broadcast is unambiguous.

**BENEFITS OF NUMPY** :

More speed: NumPy uses algorithms written in C that complete in nanoseconds rather than seconds.

Fewer loops: NumPy helps you to reduce loops and keep from getting tangled up in iteration indices.

Clearer code: Without loops, your code will look more like the equations you’re trying to calculate.

Better quality: There are thousands of contributors working to keep NumPy fast, friendly, and bug free.

**CONCLUSION**

No matter how many dimensions your data lives in, NumPy gives you the tools to work with it. You can store it, reshape it, combine it, filter it, and sort it, and your code will read like you’re operating on only one number at a time rather than hundreds or thousands.

**PANDA :**

Python Pandas is defined as an open-source library that provides high-performance data manipulation in Python. This tutorial is designed for both beginners and professionals.

It is used for data analysis in Python and developed by Wes McKinney in 2008. Our Tutorial provides all the basic and advanced concepts of Python Pandas, such as Numpy, Data operation and Time Series

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It is used for data analysis in Python and developed by Wes McKinney in 2008.

Data analysis requires lots of processing, such as restructuring, cleaning or merging, etc. There are different tools are available for fast data processing, such as Numpy, Scipy, Cython, and Panda. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools. Pandas is built on top of the **Numpy** package, means **Numpy** is required for operating the Pandas.

Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyze.

Features of Pandas

It has a fast and efficient DataFrame object with the default and customized indexing.

Used for reshaping and pivoting of the data sets.

Group by data for aggregations and transformations.

It is used for data alignment and integration of the missing data.

Provide the functionality of Time Series.

Process a variety of data sets in different formats like matrix data, tabular heterogeneous, time series.

Handle multiple operations of the data sets such as subsetting, slicing, filtering, groupBy, re-ordering, and re-shaping.

It integrates with the other libraries such as SciPy, and scikit-learn.

Provides fast performance, and If you want to speed it, even more, you can use the Cython.

**Benefits of Pandas**

The benefits of pandas over using other language are as follows:

Data Representation: It represents the data in a form that is suited for data analysis through its DataFrame and Series.

Clear code: The clear API of the Pandas allows you to focus on the core part of the code. So, it provides clear and concise code for the user.

**SEABORN:**

Seaborn is a Python visualization library for statistical plotting. It comes equipped with preset styles and color palettes so you can create complex, aesthetically pleasing charts with a few lines of code. It's designed to work with Numpy and Pandas.) data structures and to support statistical tasks completed in SciPy and Statsmodels.

Seaborn is built on top of Python's core visualization library matpoltlib, but it's meant to serve as a complement, not a replacement. In most cases, you'll still use matplotlib for simple plotting, and you'll need a knowledge of matplotlib to tweak Seaborn's default plots.

In the world of Analytics, the best way to get insights is by visualizing the data. Data can be visualized by representing it as plots which is easy to understand, explore and grasp. Such data helps in drawing the attention of key elements.

To analyse a set of data using Python, we make use of Matplotlib, a widely implemented 2D plotting library. Likewise, Seaborn is a visualization library in Python. It is built on top of Matplotlib.

## Seaborn Vs Matplotlib

It is summarized that if Matplotlib “tries to make easy things easy and hard things possible”, Seaborn tries to make a well-defined set of hard things easy too.”

Seaborn helps resolve the two major problems faced by Matplotlib; the problems are −

Default Matplotlib parameters

Working with data frames

As Seaborn compliments and extends Matplotlib, the learning curve is quite gradual. If you know Matplotlib, you are already half way through Seaborn.

### Features of Seaborn

Seaborn is built on top of Python’s core visualization library Matplotlib. It is meant to serve as a complement, and not a replacement. However, Seaborn comes with some very important features. Let us see a few of them here. The features help in −

Built in themes for styling matplotlib graphics

Visualizing univariate and bivariate data

Fitting in and visualizing linear regression models

Plotting statistical time series data

Seaborn works well with NumPy and Pandas data structures

It comes with built in themes for styling Matplotlib graphics

In most cases, you will still use Matplotlib for simple plotting. The knowledge of Matplotlib is recommended to tweak Seaborn’s default plots.

**Benefits of seaborn :**

Data visualization

Use to show data as a line plot.

Use to create high level informative and attractive plots to show the data.

**MATPOLTLIB:**

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc. Human minds are more adaptive for the visual representation of data rather than textual data. We can easily understand things when they are visualized. It is better to represent the data through the graph where we can analyze the data more efficiently and make the specific decision according to data analysis. Before learning the matplotlib, we need to understand data visualization and why data visualization is important. Matplotlib is easy to use and an amazing visualizing library in Python. It is built on NumPy arrays and designed to work with the broader SciPy stack and consists of several plots like line, bar, scatter, histogram, etc. In this article, we will learn about Python plotting with Matplotlib from basics to advance with the help of a huge dataset containing information about different types of plots and their customizations. Matplotlib is the most widely used data visualization and plotting library in all of Python. In fact, as we've said before, many of the other libraries in this course utilize attributes of Matplotlib to display the plots they generate.

Much of Matplotlib's popularity comes from the fact that it is highly customizable, with users able to edit almost every aspect of a Matplotlib plot.

Matplotlib plots are comprised of a hierarchy of objects. At the top level of the plot, the Figure is what contains the rest of the plot elements. The intermediate and lower level plot elements are objects and elements like the Axes, Labels, Ticks, and Legends. All of these elements can be tweaked by the user.

In this section, we'll cover the features of Matplotlib, and when you would want to use it. We'll then move on to covering the layout and elements that comprise a Matplotlib plot, demonstrating how to customize these elements. We'll then go over some examples of the visualizations that you can create with Matplotlib.

Finally, we'll explore the Collatz Conjecture and learn how simple number sequences can have profound visualizations.

**Features of matplotlib**

# 1. Semantic way to generate complex, subplot grids.

# 2.Setting the aspect ratio of the axes box.

# 3. Colored labels in legends

# 4.  Ticks and labels

# 5. rcParams can be passed as Decorators

**Benefits of matplotlib :**

Simple and easy to grasp for beginners.

Easier to use for people who have had prior experience with Matlab or other graph plotting tools.

It provides high-quality images and plots in various formats such as png, pdf, pgf, etc.

Provides control to various elements of a figure such as DPI, figure colour, figure size.

**SCIKIT LEARN :**

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

Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

## Origin of Scikit-Learn

It was originally called *scikits.learn* and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project at another level and made the first public release (v0.1 beta) on 1st Feb. 2010.

### **Components of scikit-learn:**

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

**Supervised learning algorithms:** Think of any supervised machine learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of machine learning algorithms is one of the big reasons for the high usage of scikit-learn. I started using scikit to solve supervised learning problems and would recommend that to people new to scikit / machine learning as well.

**Cross-validation:** There are various methods to check the accuracy of supervised models on unseen data using sklearn.

**Unsupervised learning algorithms:** Again there is a large spread of machine learning algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.

**Various toy datasets:** This came in handy while learning scikit-learn. I had learned SAS using various academic datasets (e.g. IRIS dataset, Boston House prices dataset). Having them handy while learning a new library helped a lot.

**Feature extraction:** Scikit-learn for extracting features from images and text (e.g. Bag of words)

Important features of scikit-learn:

Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.

Accessible to everybody and reusable in various contexts.

Built on the top of NumPy, SciPy, and matplotlib.

Open source, commercially usable – BSD license.

**Benefits of scikit learn :**

The main benefits of scikit learn are its open-source anyone can use it any time this library and it is easy to use.

The benefit of scikit learn are :

Open source

Easy to use

Free

Properly documented

Versatile used

Here is a brief description of the benefits of scikit learn:

Open source: scikit learn is an open-source library that is available for public use. This is openly available for use or even in this the user can modify or redirect the whole code.

Easy to use: As we know scikit learn is an open-source library anyone can use this anytime. Many of the research organizations use scikit learn in their operation and they all were agreed that the scikit learn library is easy to use.

Free: scikit learn library is free to use the people don’t take any license for running this library. The user would not have to worry when they work on the application.

# 

# ALGORITHMS

**4.3 LOGISTIC REGRESSION:**

# Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



## Logistic Function (Sigmoid Function):

The sigmoid function is a mathematical function used to map the predicted values to probabilities.

It maps any real value into another value within a range of 0 and 1.

The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.

In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

## Assumptions for Logistic Regression:

The dependent variable must be categorical in nature.

The independent variable should not have multi-collinearity.

## Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

We know the equation of the straight line can be written as:

Logistic Regression in Machine Learning

In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

Logistic Regression in Machine Learning

But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

Logistic Regression in Machine Learning

The above equation is the final equation for Logistic Regression.

Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a very important aspect of Logistic regression and is dependent on the classification problem itself. The decision for the value of the threshold value is majorly affected by the values of precision and recall. Ideally, we want both precision and recall to be 1, but this seldom is the case. Machine-Learning-Course In the case of a Precision-Recall tradeoff, we use the following arguments to decide upon the threshold:-

1. Low Precision/High Recall: In applications where we want to reduce the number of false negatives without necessarily reducing the number of false positives, we choose a decision value that has a low value of Precision or a high value of Recall. For example, in a cancer diagnosis application, we do not want any affected patient to be classified as not affected without giving much heed to if the patient is being wrongfully diagnosed with cancer. This is because the absence of cancer can be detected by further medical diseases but the presence of the disease cannot be detected in an already rejected candidate.

2. High Precision/Low Recall: In applications where we want to reduce the number of false positives without necessarily reducing the number of false negatives, we choose a decision value that has a high value of Precision or a low value of Recall. For example, if we are classifying customers whether they will react positively or negatively to a personalized advertisement, we want to be absolutely sure that the customer will react positively to the advertisement because otherwise, a negative reaction can cause a loss of potential sales from the customer.

## Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

**Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

**Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"

**Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

### Applications of Logistic Regression

1. Predicting a probability of a person having a heart attack

2. Predicting a customer’s propensity to purchase a product or halt a subscription.

3. Predicting the probability of failure of a given process or product.

**4.4 KNN :**

# K-Nearest Neighbor (KNN) Algorithm

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

**Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

## Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



## How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

**Step-1:** Select the number K of the neighbors

**Step-2:** Calculate the Euclidean distance of **K number of neighbors**

**Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.

**Step-4:** Among these k neighbors, count the number of the data points in each category.

**Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.

**Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



Firstly, we will choose the number of neighbors, so we will choose the k=5.

Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

## How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.

A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.

Large values for K are good, but it may find some difficulties.

## Advantages of KNN Algorithm:

It is simple to implement.

It is robust to the noisy training data

It can be more effective if the training data is large.

## Disadvantages of KNN Algorithm:

Always needs to determine the value of K which may be complex some time.

The computation cost is high because of calculating the distance between the data points for all the training samples.

# 4.5 Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



**Example:** SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:

SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

## Types of SVM

**SVM can be of two types:**

**Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

**Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

## Hyperplane and Support Vectors in the SVM algorithm:

**Hyperplane:** There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

**Support Vectors:**

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

## How does SVM works?

**Linear SVM:**

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



**Non-Linear SVM:**

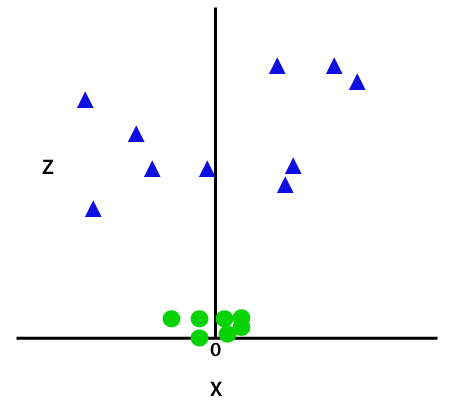
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

z=x2 +y2

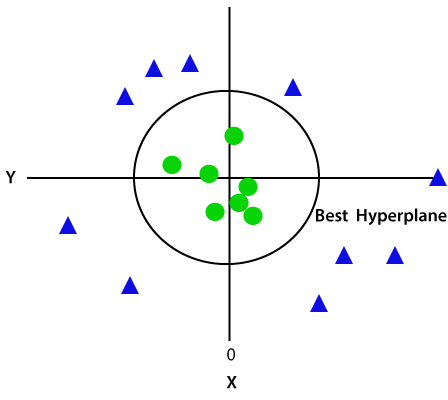
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



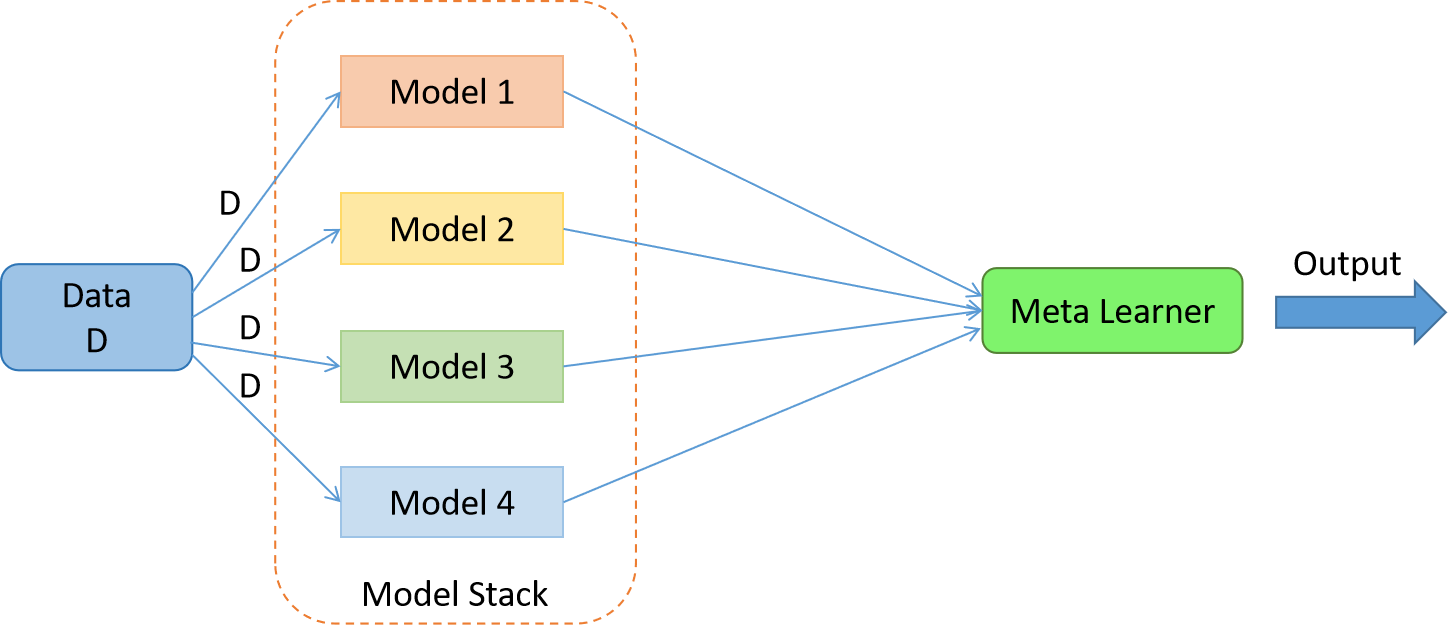
Hence we get a circumference of radius 1 in case of non-linear data.

**4.6 Stacking**

 While bagging and boosting used homogenous weak learners for ensemble, Stacking often considers heterogeneous weak learners, learns them in parallel, and combines them by training a meta-learner to output a prediction based on the different weak learner’s predictions. A meta learner inputs the predictions as the features and the target being the ground truth values in data, it attempts to learn how to best combine the input predictions to make a better output prediction.

In averaging ensemble eg. Random Forest, the model combines the predictions from multiple trained models. A limitation of this approach is that each model contributes the same amount to the ensemble prediction, irrespective of how well the model performed. An alternate approach is a weighted average ensemble, which weighs the contribution of each ensemble member by the trust on their contribution in giving the best predictions. The weighted average ensemble provides an improvement over the model average ensemble.

A further generalization of this approach is replacing the linear weighted sum with Linear Regression (regression problem) or Logistic Regression (classification problem) to combine the predictions of the sub-models with any learning algorithm. This approach is called Stacking.



# 4.7 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

**The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**

The below diagram explains the working of the Random Forest algorithm:

50.3M

997

Exception Handling in Java



## Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.

The predictions from each tree must have very low correlations.

## Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

<="" li="">

It takes less training time as compared to other algorithms.

It predicts output with high accuracy, even for the large dataset it runs efficiently.

It can also maintain accuracy when a large proportion of data is missing.

## How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

**Step-1:** Select random K data points from the training set.

**Step-2:** Build the decision trees associated with the selected data points (Subsets).

**Step-3:** Choose the number N for decision trees that you want to build.

**Step-4:** Repeat Step 1 & 2.

**Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The working of the algorithm can be better understood by the below example:

**Example:** Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



## Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

**Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.

**Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.

**Land Use:** We can identify the areas of similar land use by this algorithm.

**Marketing:** Marketing trends can be identified using this algorithm.

## Advantages of Random Forest

Random Forest is capable of performing both Classification and Regression tasks.

It is capable of handling large datasets with high dimensionality.

It enhances the accuracy of the model and prevents the overfitting issue.

## Disadvantages of Random Forest

Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

**BOOSTING:**Boosting is an ensemble modelling technique that was first presented by Freund and Schapire in the year 1997, since then, Boosting has been a prevalent technique for tackling binary classification problems. These algorithms improve the prediction power by converting a number of weak learners to strong learners.

The principle behind boosting algorithms is first we built a model on the training dataset, then a second model is built to rectify the errors present in the first model. This procedure is continued until and unless the errors are minimized, and the dataset is predicted correctly.

Let’s take an example to understand this, suppose you built a decision tree algorithm on the Titanic dataset and from there you get an accuracy of 80%. After this, you apply a different algorithm and check the accuracy and it comes out to be 75% for KNN and 70% for Linear Regression.

We see the accuracy differs when we built a different model on the same dataset. But what if we use combinations of all these algorithms for making the final prediction? We’ll get more accurate results by taking the average of results from these models. We can increase the prediction power in this way.

Boosting algorithms works in a similar way, it combines multiple models (weak learners) to reach the final output (strong learners).

There are mainly 3 types of boosting algorithms:

AdaBoost algorithm

Gradient descent algorithm

Xtreme gradient descent algorithm

Here I will be focusing on the AdaBoost algorithm.

**4.8 ADABOOST (ADAPTIVE BOOSTING):**

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called Decision Stumps.

What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lowe error is received.

An adaBoost algorithm can be used to boost the performance of any machine learning algorithm. Machine Learning has become a powerful tool which can make predictions based on a large amount of data. It has become so popular in recent times that the application of machine learning can be found in our day-to-day activities. A common example of it is getting suggestions for products while shopping online based on the past items bought by the customer. Machine Learning, often referred to as  predictive modeling, can be defined as the ability of computers to learn without being programmed explicitly. Instead, it use s programmed algorithms to analyze input data to predict output within an acceptable range.

### What is AdaBoost Algorithm?

In machine learning, boosting originated from the question of whether a set of weak classifiers could be converted to a strong classifier. A weak learner or classifier is a learner who is better than random guessing. This will be robust in over-fitting as in a large set of weak classifiers, each weak classifier being better than random. As a weak classifier, a simple threshold on a single feature is generally used. If the feature is above the threshold than predicted, it belongs to positive otherwise belongs to negative.

AdaBoost stands for ‘Adaptive Boosting’, which transforms weak learners or predictors to strong predictors in order to solve problems of classification

### How AdaBoost Algorithm Works

AdaBoost can be used to improve the performance of machine learning alogorithms. It is used best with weak learners, and these models achieve high accuracy above random chance on a classification pro1blem. The common algorithms with AdaBoost used are decision trees with level one. A weak learner is a classifier or predictor which performs relatively poor in terms of accuracy. Also, it can be implied that the weak learners are simple to compute, and many instances of algorithms are combined to create a strong classifier through boosting. -1 represents negative class, and 1 indicate positive. First, we have to select the weak classifier with the lowest weighted classification error by fitting the weak classifiers to the data set.

### What is AdaBoost Algorithm Used for?

AdaBoost can be used for face detection as it seems to be the standard algorithm for face detection in images. It uses a rejection cascade consisting of many layers of classifiers. When the detection window is not recognized at any layer as a face, it is rejected. The first classifier in the window discards the negative window keeping the computational cost to a minimum. Though AdaBoost combines the weak classifiers, the principles of AdaBoost are also used to find the best features to use in each layer of the cascade.

### Pros and Cons

One of the many advantages of the AdaBoost Algorithm is it is fast, simple and easy to program. Also, it has the flexibility to be combined with any machine learning algorithm, and there is no need to tune the parameters except for T. It has been extended to learning problems beyond binary classification, and it is versatile as it can be used with text or numeric data.

AdaBoost also has few disadvantages, such as it is from empirical evidence and particularly vulnerable to uniform noise. Weak classifiers being too weak can lead to low margins and overfitting.

# 

# CHAPTER 5 TESTING

# 5.TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing

requirement.

The testing process overview is as follows:

**Types of Testing:-**

**5.1 Unit Testing:-**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business

process, application, and/or system conFigure. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**5.2 Integration Testing:-**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields.

Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**5.3 Validation Testing:-**

An engineering validation test (EVT) is performed on first engineering prototypes, to ensure that the basic unit performs to design goals and specifications. It is important in identifying design problems, and solving them as early in the design cycle as possible, is the key to keeping projects on time and within budget. Too often, product design and performance problems are not detected until late in the product development cycle — when the product is ready to be shipped. The old adage holds true: It costs a penny to make a change in engineering, a dime in production and a dollar after a product is in the field.

Verification is a Quality control process that is used to evaluate whether or not a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase. Verification can be in development, scale-up, or production. This is often an internal process.

Validation is a Quality assurance process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance of fitness for purpose with end users and other product stakeholders.

**5.4 System Testing:-**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s).

System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS).

System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

**5.5 Acceptance Testing:**

Acceptance Testing is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system’s compliance with the business requirements and assess whether it is acceptable for delivery.

There are various forms of acceptance testing:

 User acceptance Testing

 Business acceptance Testing

 Alpha Testing

 Beta Testing

**5.6 Integrated Manual Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Test Data(input)** | Expected Result | Actual Result | Pass/ Fail |
| **1** | Attributes entered by a patient suffering from Parkinson’s disesase | Parksinsons is detected | Parksinsons is detected | **Pass** |
| **2** | Attributes entered by a healthy person. | Parkinsons is not detected | Parkinsons is not detected | **Pass** |
| **3** | User leave a field empty while entering voice attribute values | Generate the error “empty field detected” | Model doesn’t process the input because of wrong number of attributes passed and an alert is released to enter values of all fields | **Pass** |
| **4** | User enters negative value as one of the input value for prediction | Generate the error “negative value entered” | Model process the input with the negative values and an alert is thrown to enter only positive numerical values | **Pass** |

# CHAPTER 6 SCREENSHOTS

# SCREENSHOTS

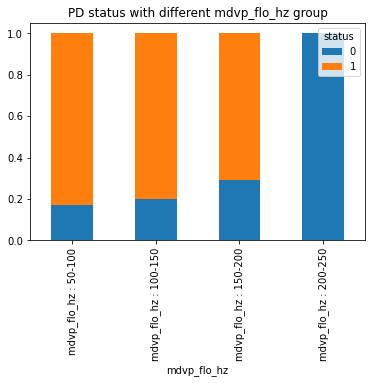
*# Five point summary of each attribute*

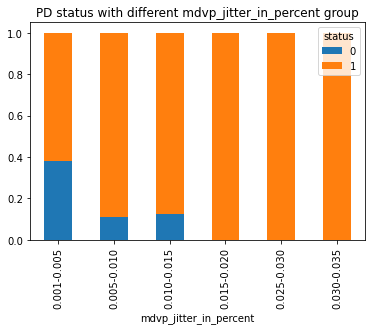
pdData.describe().T

|  | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **mdvp\_fo\_hz** | 195.0 | 154.228641 | 41.390065 | 88.333000 | 117.572000 | 148.790000 | 182.769000 | 260.105000 |
| **mdvp\_fhi\_hz** | 195.0 | 197.104918 | 91.491548 | 102.145000 | 134.862500 | 175.829000 | 224.205500 | 592.030000 |
| **mdvp\_flo\_hz** | 195.0 | 116.324631 | 43.521413 | 65.476000 | 84.291000 | 104.315000 | 140.018500 | 239.170000 |
| **mdvp\_jitter\_in\_percent** | 195.0 | 0.006220 | 0.004848 | 0.001680 | 0.003460 | 0.004940 | 0.007365 | 0.033160 |
| **mdvp\_jitter\_abs** | 195.0 | 0.000044 | 0.000035 | 0.000007 | 0.000020 | 0.000030 | 0.000060 | 0.000260 |
| **mdvp\_rap** | 195.0 | 0.003306 | 0.002968 | 0.000680 | 0.001660 | 0.002500 | 0.003835 | 0.021440 |
| **mdvp\_ppq** | 195.0 | 0.003446 | 0.002759 | 0.000920 | 0.001860 | 0.002690 | 0.003955 | 0.019580 |
| **jitter\_ddp** | 195.0 | 0.009920 | 0.008903 | 0.002040 | 0.004985 | 0.007490 | 0.011505 | 0.064330 |
| **mdvp\_shimmer** | 195.0 | 0.029709 | 0.018857 | 0.009540 | 0.016505 | 0.022970 | 0.037885 | 0.119080 |
| **mdvp\_shimmer\_db** | 195.0 | 0.282251 | 0.194877 | 0.085000 | 0.148500 | 0.221000 | 0.350000 | 1.302000 |
| **shimmer\_apq3** | 195.0 | 0.015664 | 0.010153 | 0.004550 | 0.008245 | 0.012790 | 0.020265 | 0.056470 |
| **shimmer\_apq5** | 195.0 | 0.017878 | 0.012024 | 0.005700 | 0.009580 | 0.013470 | 0.022380 | 0.079400 |
| **mdvp\_apq** | 195.0 | 0.024081 | 0.016947 | 0.007190 | 0.013080 | 0.018260 | 0.029400 | 0.137780 |
| **shimmer\_dda** | 195.0 | 0.046993 | 0.030459 | 0.013640 | 0.024735 | 0.038360 | 0.060795 | 0.169420 |
| **nhr** | 195.0 | 0.024847 | 0.040418 | 0.000650 | 0.005925 | 0.011660 | 0.025640 | 0.314820 |
| **hnr** | 195.0 | 21.885974 | 4.425764 | 8.441000 | 19.198000 | 22.085000 | 25.075500 | 33.047000 |
| **rpde** | 195.0 | 0.498536 | 0.103942 | 0.256570 | 0.421306 | 0.495954 | 0.587562 | 0.685151 |
| **dfa** | 195.0 | 0.718099 | 0.055336 | 0.574282 | 0.674758 | 0.722254 | 0.761881 | 0.825288 |
| **spread1** | 195.0 | -5.684397 | 1.090208 | -7.964984 | -6.450096 | -5.720868 | -5.046192 | -2.434031 |
| **spread2** | 195.0 | 0.226510 | 0.083406 | 0.006274 | 0.174351 | 0.218885 | 0.279234 | 0.450493 |
| **d2** | 195.0 | 2.381826 | 0.382799 | 1.423287 | 2.099125 | 2.361532 | 2.636456 | 3.671155 |
| **ppe** | 195.0 | 0.206552 | 0.090119 | 0.044539 | 0.137451 | 0.194052 | 0.252980 | 0.527367 |
| **status** | 195.0 | 0.753846 | 0.431878 | 0.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |

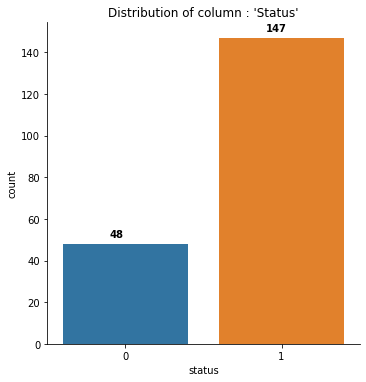
# 

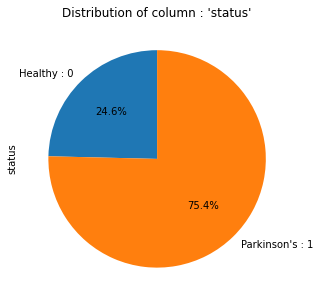
# 

****

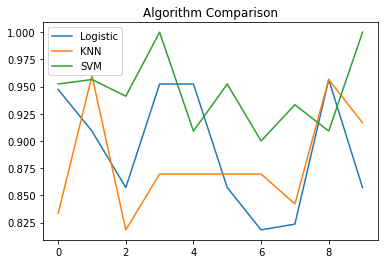
****

>

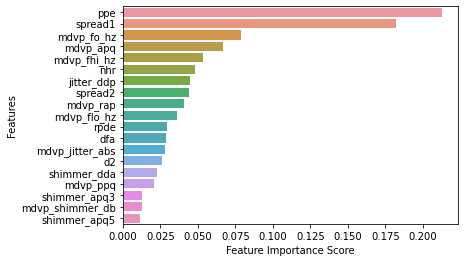
****

****

>

****

**features importance**

****

# CHAPTER 7 CONCLUSION AND FUTURE SCOPE

# 7.CONCLUSION AND FUTURE SCOPE

This project aims to provide a better prediction for Parkinson’s disease by optimizing and tuning the parameters of KNN, SVM and Random Forest Algorithms. This project also provides a reliable health monitoring system for those suffering from Parkinson’s disease.

To help in early detection of Parkinson’s disease which would help in early diagnosis thus slowing down disease progression.

Provide machine learning models which would increase accuracy of detection of the Parkinson’s disease. To help or aid current research to find a cure for the Parkinson’s disease.

# CHAPTER 8

# REFERENCES

**8.REFERENCES**

[1] B. E. Sakar et al., "Collection and Analysis of a Parkinson Speech Dataset With Multiple Types of Sound Recordings," in IEEE Journal of Biomedical and Health Informatics, vol. 17, no. 4, pp. 828-834, July 2013, doi: 10.1109/JBHI.2013.2245674.

[2] Benba, A., Jilbab, A. & Hammouch, A. Analysis of multiple types of voice recordings in cepstral domain using MFCC for discriminating between patients with Parkinson’s disease and healthy people. Int J Speech Technol 19, 449–456 (2016).

[3] Ö. Eskıdere, A. Karatutlu and C. Ünal, "Detection of Parkinson's disease from vocal features using random subspace classifier ensemble," 2015 Twelve International Conference on Electronics Computer and Computation (ICECCO), Almaty, 2015, pp. 1-4, doi: 10.1109/ICECCO.2015.7416886.

[4] Nissar, Iqra & Rizvi, Danish & Masood, Sarfaraz & Mir, Aqib. (2018). Voice-Based Detection of Parkinson’s Disease through Ensemble Machine Learning Approach: A Performance Study. EAI Endorsed Transactions on Pervasive Health and Technology.

[5] 162806. 10.4108/eai.13- 7-2018.162806. [5]Y. Li, C. Zhang, Y. Jia, P. Wang, X. Zhang and T. Xie, "Simultaneous learning of speech feature and segment for classification of Parkinson disease," 2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom), Dalian, 2017, pp. 1-6, doi: 10.1109/HealthCom.2017.8210820.

[6] Caliskan, Abdullah & Badem, Hasan & Basturk, Alper & Yüksel, Meltem. (2017). Diagnosis of the Parkinson disease by using deep neural network classifier. Istanbul University - Journal of Electrical and Electronics Engineering. 17. 3311-3318.

[7] Berus L, Klancnik S, Brezocnik M, Ficko M. Classifying Parkinson's Disease Based on Acoustic Measures Using Artificial Neural Networks. Sensors (Basel). 2018;19(1):16. Published 2018 Dec 20. doi:10.3390

[8] https://www.apdaparkinson.org//article//introduction-parkinsons--disease/

# APPENDIX A

# TECHNOLOGIES USED

# 

# 8.1 APPENDIX A

# TECHNOLOGIES USED

**Python:**

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until stepping down as leader in July 2018. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural. It also has a comprehensive standard library are freely available in source or binary form for all major platforms from the Python Website and may be freely disturbed. The same site also contains distributions of and pointers to many free third-party Python modules, programs and tools, and additional documentation.

Python is simple to use, but it is a real programming language, offering much more structure and support for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much more error checking than C, and, being a very-high-level language, it has high-level data types built in, such as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those languages. It comes with a large collection of standard modules that you can use as the basis of your programs — or as examples to start learning to program in Python. Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like Tk.

Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throwaway programs, or to test functions during bottom-up program development. It is also a handy desk calculator. Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throwaway programs, or to test functions during bottom-up program development. It is also a handy desk calculator. Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throwaway programs, or to test functions during bottom-up program development. It is also a handy desk calculator.

Python’s features include-

**Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

**Easy-to-read** − Python code is more clearly defined and visible to the eyes.

**Easy-to-maintain** − Python's source code is fairly easy-to-maintain.

**Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

**Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

**Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

**Databases** − Python provides interfaces to all major commercial databases.

**GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**Scalable** − Python provides a better structure and support for large programs than shell scripting.

**ANACONDA SOFTWARE**

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics) that aims to simply the package management and deployment. The distribution includes the data science packages suitable for Windows, Linux, macOS. It is developed and maintained by Anaconda, Inc., which has founded by Peter Wang and Travis Oilphant in 2012.Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda and the virtual environment manager that includes GUI, Anaconda Navigator as a graphical alternative to the command line interface (CLI). The main difference between conda and the pip package manager is that in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS . Conda is an open-source, cross-platform, package management system. Anaconda comes with great tools like JupyterLab, Jupyter Notebook, Qt Console, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code.

# APPENDIX B

**SOFTWARE AND HARDWARE REQUIREMENTS**

**8.2 APPENDIX B**

**SOFTWARE AND HARDWARE REQUIREMENTS**

### Hardware Requirements:-

Processor : Any Processor above 500 MHz

RAM : 8 GB

Hard Disk : 1 TB

**Software Requirements:-**

Operating System : Windows 10

Platform : Jupyter Notebook

Language : Python

# APPENDIX C

# SAMPLE CODE

**8.3 APPENDIX C**

**SAMPLE CODE**

CODE

**import** numpy **as** np *# linear algebra*

**import** pandas **as** pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

**import** seaborn **as** sns *# Data Visualizations*

**import** matplotlib.pyplot **as** plt *# Necessary module for plotting purpose*

**import** warnings *# importing warning library*

​

*# add graphs into jupyter notebook*

**%**matplotlib inline

warnings.filterwarnings('ignore')*# for ignoring warnings in notebook*

​

**import** statsmodels.api **as** sm *# importing statsmodel api*

**from** sklearn **import** model\_selection *# For model\_selection*

**from** sklearn.model\_selection **import** train\_test\_split *# For train-test split*

​

*# getting methods for confusion matrix, F1 score, Accuracy Score*

**from** sklearn **import** metrics

**from** sklearn.metrics **import** confusion\_matrix,f1\_score,accuracy\_score,classification\_report,roc\_curve,auc,average\_precision\_score

**from** sklearn.linear\_model **import** LogisticRegression *# For logistic Regression*

**from** sklearn.naive\_bayes **import** GaussianNB *# For Naive Bayes classifier*

**from** sklearn.neighbors **import** KNeighborsClassifier *# For K-NN Classifier*

**from** sklearn.svm **import** SVC *# For support vector machine based classifier*

​

*#Scaling*

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.ensemble **import** StackingClassifier

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.ensemble **import** AdaBoostClassifier

pdDataOrg **=** pd.read\_csv("C:\\Users\\kamaljeet\\Downloads\\parkinsons.data") *# using pandas read\_csv*

*function to load dataset into pdData variable*

pdDataOrg.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | **name** | **MDVP:Fo(Hz)** | **MDVP:Fhi(Hz)** | **MDVP:Flo(Hz)** | **MDVP:Jitter(%)** | **MDVP:Jitter(Abs)** | **MDVP:RAP** | **MDVP:PPQ** | **Jitter:DDP** | **MDVP:Shimmer** | **...** | **Shimmer:DDA** | **NHR** | **HNR** | **status** | **RPDE** | **DFA** | **spread1** | **spread2** | **D2** | **PPE** | | **0** | phon\_R01\_S01\_1 | 119.992 | 157.302 | 74.997 | 0.00784 | 0.00007 | 0.00370 | 0.00554 | 0.01109 | 0.04374 | ... | 0.06545 | 0.02211 | 21.033 | 1 | 0.414783 | 0.815285 | -4.813031 | 0.266482 | 2.301442 | 0.284654 | | **1** | phon\_R01\_S01\_2 | 122.400 | 148.650 | 113.819 | 0.00968 | 0.00008 | 0.00465 | 0.00696 | 0.01394 | 0.06134 | ... | 0.09403 | 0.01929 | 19.085 | 1 | 0.458359 | 0.819521 | -4.075192 | 0.335590 | 2.486855 | 0.368674 | | **2** | phon\_R01\_S01\_3 | 116.682 | 131.111 | 111.555 | 0.01050 | 0.00009 | 0.00544 | 0.00781 | 0.01633 | 0.05233 | ... | 0.08270 | 0.01309 | 20.651 | 1 | 0.429895 | 0.825288 | -4.443179 | 0.311173 | 2.342259 | 0.332634 | | **3** | phon\_R01\_S01\_4 | 116.676 | 137.871 | 111.366 | 0.00997 | 0.00009 | 0.00502 | 0.00698 | 0.01505 | 0.05492 | ... | 0.08771 | 0.01353 | 20.644 | 1 | 0.434969 | 0.819235 | -4.117501 | 0.334147 | 2.405554 | 0.368975 | | **4** | phon\_R01\_S01\_5 | 116.014 | 141.781 | 110.655 | 0.01284 | 0.00011 | 0.00655 | 0.00908 | 0.01966 | 0.06425 | ... | 0.10470 | 0.01767 | 1 |  |  |  |  |  |  |  | |

5 rows × 24 columns

'''

To use columns of pdDataOrg df more conveniently following are some changes

a. pushing target column i.e 'status' to last column

b. converting all column names in lower case

c. replacing spaces in column names with '\_'

d. replacing ':' in column names with '\_'

e. replacing '(' in column names with '\_'

f. replacing ')' in column names with '' i.e blank

g. replacing '%' in column names with 'in\_percent'

'''

​

pdData **=** pdDataOrg.copy() *# creating a copy of loanDataOrg into loanData*

​

targetCol **=** 'status' *# defining target column*

targetColDf **=** pdData.pop(targetCol) *# popping target column from loanData df*

pdData.insert(len(pdData.columns),targetCol, targetColDf)*# inserting target column to last column*

*# deleting variables that were used for changing column position of target column*

**del** targetCol

**del** targetColDf

​*# converting column names into lower case*

pdData.columns **=** [c.lower() **for** c **in** pdData.columns]

*# replacing spaces in column names with '\_'*

pdData.columns **=** [c.replace(' ', '\_') **for** c **in** pdData.columns]

*# replacing ':' in column names with '\_'*

pdData.columns **=** [c.replace(':', '\_') **for** c **in** pdData.columns]

*# replacing '(' in column names with '\_'*

pdData.columns **=** [c.replace('(', '\_') **for** c **in** pdData.columns]

*# replacing ')' in column names with '' i.e blank*

pdData.columns **=** [c.replace(')', '') **for** c **in** pdData.columns]

*# replacing '%' in column names with 'in\_percent'*

pdData.columns **=** [c.replace('%', 'in\_percent') **for** c **in** pdData.columns]

​

*# to check the above printing top 5 rows*

pdData.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **name** | **mdvp\_fo\_hz** | **mdvp\_fhi\_hz** | **mdvp\_flo\_hz** | **mdvp\_jitter\_in\_percent** | **mdvp\_jitter\_abs** | **mdvp\_rap** | **mdvp\_ppq** | **jitter\_ddp** | **mdvp\_shimmer** | **...** | **shimmer\_dda** | **nhr** | **hnr** | **rpde** | **dfa** | **spread1** | **spread2** | **d2** | **ppe** | **status** |
| **0** |  | phon\_R01\_S01\_1 | 119.992 | 157.302 | 74.997 | 0.00784 | 0.00007 | 0.00370 | 0.00554 | 0.01109 | 0.04374 | ... | 0.06545 | 0.02211 | 21.033 | 0.414783 | 0.815285 | -4.813031 | 0.266482 | 2.301442 | 0.284654 | 1 |
| **1** |  | phon\_R01\_S01\_2 | 122.400 | 148.650 | 113.819 | 0.00968 | 0.00008 | 0.00465 | 0.00696 | 0.01394 | 0.06134 | ... | 0.09403 | 0.01929 | 19.085 | 0.458359 | 0.819521 | -4.075192 | 0.335590 | 2.486855 | 0.368674 | 1 |
| **2** |  | phon\_R01\_S01\_3 | 116.682 | 131.111 | 111.555 | 0.01050 | 0.00009 | 0.00544 | 0.00781 | 0.01633 | 0.05233 | ... | 0.08270 | 0.01309 | 20.651 | 0.429895 | 0.825288 | -4.443179 | 0.311173 | 2.342259 | 0.332634 | 1 |
| **3** |  | phon\_R01\_S01\_4 | 116.676 | 137.871 | 111.366 | 0.00997 | 0.00009 | 0.00502 | 0.00698 | 0.01505 | 0.05492 | ... | 0.08771 | 0.01353 | 20.644 | 0.434969 | 0.819235 | -4.117501 | 0.334147 | 2.405554 | 0.368975 | 1 |
| **4** |  | phon\_R01\_S01\_5 | 116.014 | 141.781 | 110.655 | 0.01284 | 0.00011 | 0.00655 | 0.00908 | 0.01966 | 0.06425 | ... | 0.10470 | 0.01767 | 19.649 | 0.417356 | 0.823484 | -3.747787 | 0.234513 | 2.332180 | 0.410335 | 1 |

5 rows × 24 columns

**Attribute Information:**

​name - ASCII subject name and recording number.

​

mdvp\_fo\_hz - Average vocal fundamental frequency (Actualy column name MDVP:Fo(Hz) )

​

mdvp\_fhi\_hz - Maximum vocal fundamental frequency (Actualy column name MDVP:Fhi(Hz) )

​

mdvp\_flo\_hz - Minimum vocal fundamental frequency (Actualy column name MDVP:Flo(Hz)

​

mdvp\_jitter\_in\_percent, mdvp\_jitter\_abs, mdvp\_rap, mdvp\_ppq, jitter\_ddp - Several measures of variation in fundamental frequency (Actualy column names MDVP:Jitter(%), MDVP:Jitter(Abs), MDVP:RAP, MDVP:PPQ, Jitter:DDP respectively)

​

mdvp\_shimmer, mdvp\_shimmer\_db, shimmer\_apq3, shimmer\_apq5, mdvp\_apq, shimmer\_dda - Several measures of variation in amplitude (Actualy column names MDVP:Shimmer, MDVP:Shimmer(dB), Shimmer:APQ3, Shimmer:APQ5, MDVP:APQ, Shimmer:DDA respectively)

​

nhr, hnr - Two measures of ratio of noise to tonal components in the voice (Actualy column names NHR, HNR respectively)

​

rpde, d2 - Two nonlinear dynamical complexity measures (Actualy column names RPDE, D2 respectively)

​

dfa - Signal fractal scaling exponent (Actualy column name DFA )

​

spread1, spread2, ppe - Three nonlinear measures of fundamental frequency variation (Actualy column names spread1, spread2, PPE respectively)

​

status - Health status of the subject (one) - Parkinson's, (zero) - healthy (Target Varibale / attribute)

print('\033[1mThe Parkinson\'s disease dataset having "{0}" rows and "{1}" columns\033[0 m.'.format(pdData.shape[0],pdData.shape[1]))

**The Parkinson's disease dataset having "195" rows and "24" columns**.

pdData.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 195 entries, 0 to 194

Data columns (total 24 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 name 195 non-null object

1 mdvp\_fo\_hz 195 non-null float64

2 mdvp\_fhi\_hz 195 non-null float64

3 mdvp\_flo\_hz 195 non-null float64

4 mdvp\_jitter\_in\_percent 195 non-null float64

5 mdvp\_jitter\_abs 195 non-null float64

6 mdvp\_rap 195 non-null float64

7 mdvp\_ppq 195 non-null float64

8 jitter\_ddp 195 non-null float64

9 mdvp\_shimmer 195 non-null float64

10 mdvp\_shimmer\_db 195 non-null float64

11 shimmer\_apq3 195 non-null float64

12 shimmer\_apq5 195 non-null float64

13 mdvp\_apq 195 non-null float64

14 shimmer\_dda 195 non-null float64

15 nhr 195 non-null float64

16 hnr 195 non-null float64

17 rpde 195 non-null float64

18 dfa 195 non-null float64

19 spread1 195 non-null float64

20 spread2 195 non-null float64

21 d2 195 non-null float64

22 ppe 195 non-null float64

23 status 195 non-null int64

dtypes: float64(22), int64(1), object(1)

memory usage: 36.7+ KB

Setting 'name' attribute as index of the pdData dataframe as the attribute / column does not have an significance towards identifying patients have Parkinson's disease or not i.e 'status' column [Health status of the subject (one) - Parkinson's, (zero) - healthy]

*# setting name column as index column*

pdData.set\_index('name',inplace=True)

*# after setting column 'name' as index now we have less columns to confirm that printing number of rows and column once again*

print('\033[1mAfter setting \'name\' column as index of the Dataset,\033[0m now there are \033[1m"{0}"\033[0m Rows and \033[1m"{1}"\033[0m Columns in the given Dataset.'.format(pdData.shape[0],pdData.shape[1]))

**After setting 'name' column as index of the Dataset,** now there are **"195"** Rows and **"23"** Columns in the given Dataset.

*# printing top 5 rows once again to check*

pd.options.display.max\_columns **=** **None**

pdData.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **mdvp\_fo\_hz** | **mdvp\_fhi\_hz** | **mdvp\_flo\_hz** | **mdvp\_jitter\_in\_percent** | **mdvp\_jitter\_abs** | **mdvp\_rap** | **mdvp\_ppq** | **jitter\_ddp** | **mdvp\_shimmer** | **mdvp\_shimmer\_db** | **shimmer\_apq3** | **shimmer\_apq5** | **mdvp\_apq** | **shimmer\_dda** | **nhr** | **hnr** | **rpde** | **dfa** | **spread1** | **spread2** | **d2** | **ppe** | **status** |
| **name** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **phon\_R01\_S01\_1** | 119.992 | 157.302 | 74.997 | 0.00784 | 0.00007 | 0.00370 | 0.00554 | 0.01109 | 0.04374 | 0.426 | 0.02182 | 0.03130 | 0.02971 | 0.06545 | 0.02211 | 21.033 | 0.414783 | 0.815285 | -4.813031 | 0.266482 | 2.301442 | 0.284654 | 1 |
| **phon\_R01\_S01\_2** | 122.400 | 148.650 | 113.819 | 0.00968 | 0.00008 | 0.00465 | 0.00696 | 0.01394 | 0.06134 | 0.626 | 0.03134 | 0.04518 | 0.04368 | 0.09403 | 0.01929 | 19.085 | 0.458359 | 0.819521 | -4.075192 | 0.335590 | 2.486855 | 0.368674 | 1 |
| **phon\_R01\_S01\_3** | 116.682 | 131.111 | 111.555 | 0.01050 | 0.00009 | 0.00544 | 0.00781 | 0.01633 | 0.05233 | 0.482 | 0.02757 | 0.03858 | 0.03590 | 0.08270 | 0.01309 | 20.651 | 0.429895 | 0.825288 | -4.443179 | 0.311173 | 2.342259 | 0.332634 | 1 |
| **phon\_R01\_S01\_4** | 116.676 | 137.871 | 111.366 | 0.00997 | 0.00009 | 0.00502 | 0.00698 | 0.01505 | 0.05492 | 0.517 | 0.02924 | 0.04005 | 0.03772 | 0.08771 | 0.01353 | 20.644 | 0.434969 | 0.819235 | -4.117501 | 0.334147 | 2.405554 | 0.368975 | 1 |
| **phon\_R01\_S01\_5** | 116.014 | 141.781 | 110.655 | 0.01284 | 0.00011 | 0.00655 | 0.00908 | 0.01966 | 0.06425 | 0.584 | 0.03490 | 0.04825 | 0.04465 | 0.10470 | 0.0177 | 19.64 | 0.4173 | 0.8234 | -3.747 | 0.2345 | 2.3321 | 0.4103 | 1 |

print("\033[1m\*"**\***100)

print("a.\nColumn\_Names Data\_Types")

print("\*"**\***30)

print("\033[0m{0}\033[1m".format(pdData.dtypes))

print("\*"**\***30)

print()

​*# printing No of Columns having different Types of Datatype*

​print("\*"**\***100)

print("b.\nNumber of Columns with each DataTypes as follows :")

print("\*"**\***50)

print("Column\_Names No\_of\_Columns\033[0m")

print("\*"**\***30)

print(pdData.dtypes.value\_counts())

print("\033[1m\*"**\***30)

print("\033[0m")

​*# printing Different Column Names of the dataset*

​print("\033[1m\*"**\***100)

print("c.\nEach Column Names of the dataset")

print("\*"**\***80)

print("\033[0m{0}\033[1m".format(pdData.columns))

print("\*"**\***80)

print("\033[0m")

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**a.**

**Column\_Names Data\_Types**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

mdvp\_fo\_hz float64

mdvp\_fhi\_hz float64

mdvp\_flo\_hz float64

mdvp\_jitter\_in\_percent float64

mdvp\_jitter\_abs float64

mdvp\_rap float64

mdvp\_ppq float64

jitter\_ddp float64

mdvp\_shimmer float64

mdvp\_shimmer\_db float64

shimmer\_apq3 float64

shimmer\_apq5 float64

mdvp\_apq float64

shimmer\_dda float64

nhr float64

hnr float64

rpde float64

dfa float64

spread1 float64

spread2 float64

d2 float64

ppe float64

status int64

dtype: object

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**b.**

**Number of Columns with each DataTypes as follows :**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Column\_Names No\_of\_Columns**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

float64 22

int64 1

dtype: int64

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**c. Each Column Names of the dataset**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Index(['mdvp\_fo\_hz', 'mdvp\_fhi\_hz', 'mdvp\_flo\_hz', 'mdvp\_jitter\_in\_percent',

'mdvp\_jitter\_abs', 'mdvp\_rap', 'mdvp\_ppq', 'jitter\_ddp', 'mdvp\_shimmer',

'mdvp\_shimmer\_db', 'shimmer\_apq3', 'shimmer\_apq5', 'mdvp\_apq',

'shimmer\_dda', 'nhr', 'hnr', 'rpde', 'dfa', 'spread1', 'spread2', 'd2',

'ppe', 'status'],

dtype='object')

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

All columns except 'status' are Numeric column.

Columns 'status' is Nominal Categorical column with binary response.

*# checking missing values in dataset for each attributes / columns*

​print("\033[1m\*"**\***100)

print("Column\_Name No\_of\_Missing\_Values")

print("\*"**\***50)

print("\033[0m{0}".format(pdData.isnull().sum()))

print("\033[1m\*"**\***50)

print()

​*# checking if any duplicate rows available in the dataset*

​print("\*"**\***100)

print("Showing Duplicate rows if any in the dataset: ")

print("\*"**\***50)

print("\033[0m{0}".format(pdData[pdData.duplicated()]))

print("\033[1m\*"**\***100)

print("\033[0m")

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Column\_Name No\_of\_Missing\_Values**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

mdvp\_fo\_hz 0

mdvp\_fhi\_hz 0

mdvp\_flo\_hz 0

mdvp\_jitter\_in\_percent 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

rpde 0

dfa 0

spread1 0

spread2 0

d2 0

ppe 0

status 0

dtype: int64

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Showing Duplicate rows if any in the dataset:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Empty DataFrame

Columns: [mdvp\_fo\_hz, mdvp\_fhi\_hz, mdvp\_flo\_hz, mdvp\_jitter\_in\_percent, mdvp\_jitter\_abs, mdvp\_rap, mdvp\_ppq, jitter\_ddp, mdvp\_shimmer, mdvp\_shimmer\_db, shimmer\_apq3, shimmer\_apq5, mdvp\_apq, shimmer\_dda, nhr, hnr, rpde, dfa, spread1, spread2, d2, ppe, status]

Index: []

As shown above, (a.) There are no missing values and (b.) No duplicate rows in the given dataset

*# checking skewness of the data*

pdData.skew().sort\_values(ascending**=False**)

nhr 4.220709

jitter\_ddp 3.362058

mdvp\_rap 3.360708

mdvp\_jitter\_in\_percent 3.084946

mdvp\_ppq 3.073892

mdvp\_jitter\_abs 2.649071

mdvp\_apq 2.618047

mdvp\_fhi\_hz 2.542146

mdvp\_shimmer\_db 1.999389

shimmer\_apq5 1.798697

mdvp\_shimmer 1.666480

shimmer\_dda 1.580618

shimmer\_apq3 1.580576

mdvp\_flo\_hz 1.217350

ppe 0.797491

mdvp\_fo\_hz 0.591737

spread1 0.432139

d2 0.430384

spread2 0.144430

dfa -0.033214

rpde -0.143402

hnr -0.514317

status -1.187727

dtype: float64

As from above we understand the following:

Independent variables are measured in different units e.g. Hz, dB, % and absoulute etc i.e variation in units of data exists and gap between feature values extreamly high. Requires data scalling techniques to scale different quantities of measurements. Symmetrical distribution : Values close to 0 MDVP:Fo(Hz) spread1 spread2 PPE Negative skewness and Tail is larger towards the left hand side of the distribution HNR status RPDE DFA Positive skewness and Tail is larger towards the Right hand side of the distribution All other attributes have a very high distribution towards right of the median

Using univariate & bivariate analysis to check the individual attributes for their basic statistics such as central values, spread, tails, relationships between variables etc. mention your observations.

* 'mdvp\_fo\_hz' attribute : (MDVP:Fo(Hz) - Average vocal fundamental frequency )

feature **=** 'mdvp\_fo\_hz'

meanData **=** 'Mean : ' **+** str(round(pdData[feature].mean(),4)) *# variable to contain mean of the attribute*

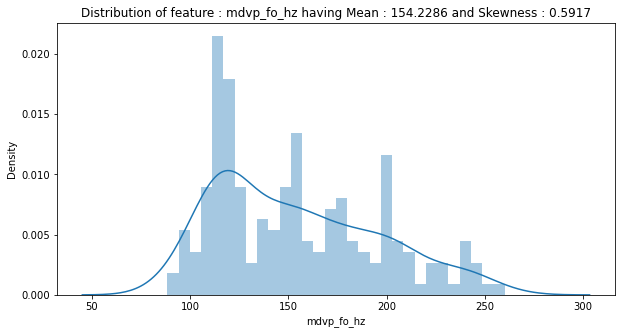
skewData **=** 'Skewness : ' **+** str(round(pdData[feature].skew(),4)) *# variable to contain skewness of the attribute*

plt.figure(figsize**=**(10,5)) *# setting figure size with width = 10 and height = 5*

fig **=** sns.distplot(pdData[feature], bins**=**30, kde**=True**) *# seaborn distplot to examine distribution of the feature*

plt.title("Distribution of feature : "**+**feature**+**" having "**+**meanData**+**" and "**+**skewData) *# setting title of the figure*

plt.show()

****

From above we can understand the following about Average vocal fundamental frequency (mdvp\_fo\_hz) attribute of the dataset:

Mean value of the attribute is 154.2286 with skewness of 0.5917, which shows that the datapoints of the attribute is slightly right / positive skewed. Maximum datapoints are ranging from 110 to 130 Hz.

plt.figure(figsize**=**(10,5)) *# setting figure size with width = 10 and height = 5*

*# seaborn distplot to examine distribution of the feature of healthy patient*

fig **=** sns.distplot(pdData[pdData['status'] **==** 0][feature], bins**=**30, kde**=True**, label**=**'Healthy')

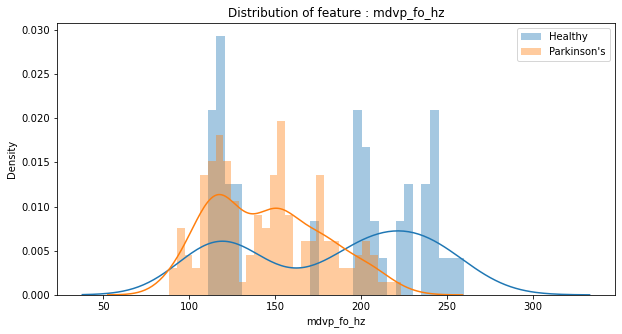
*# seaborn distplot to examine distribution of the feature of Parkinson's patient*

fig **=** sns.distplot(pdData[pdData['status'] **==** 1][feature], bins**=**30, kde**=True**, label**=**'Parkinson\'s')

plt.legend()

plt.title("Distribution of feature : "**+**feature) *# setting title of the figure*

plt.show()

****

From the above we can observe that, most of the patients with Parkinson's disease have Average vocal fundamental frequency (mdvp\_fo\_hz) between 90 to 190 hz. Even though some of healthy patients have Average vocal fundamental frequency between 110-130 Hz and 170-180 Hz.

Lets bucket Average vocal fundamental frequency (mdvp\_fo\_hz) and check w.r.t the different status i.e Healthy or Parkinson's:

bins **=** [50,100,150,200,250,300] *# defining mdvp\_fo\_hz bins,*

*# defining labels of mdvp\_fo\_hz groups as per bins defined as above*

mdvp\_fo\_hz\_group **=** ['mdvp\_fo\_hz : 50-100', 'mdvp\_fo\_hz : 100-150', 'mdvp\_fo\_hz : 150-200', 'mdvp\_fo\_hz : 200-250', 'mdvp\_fo\_hz : 250-300']

pdData\_mdvp\_fo\_hz\_bin **=** pd.cut(pdData.mdvp\_fo\_hz,bins,labels**=**mdvp\_fo\_hz\_group) *# segmenting data as per bins defined*

​*# putting into pandas crosstab and applying lambda function to take percentage and assigning to mdvp\_fo\_hz\_group\_col variable*

mdvp\_fo\_hz\_group\_col **=** pd.crosstab(pdData\_mdvp\_fo\_hz\_bin,pdData.status).apply(**lambda** r: r**/**r.sum()**\***100, axis**=**1)

print(mdvp\_fo\_hz\_group\_col) *# printing above crosstab*

​

*# plotting a stacked bar chart to show PD status for different mdvp\_fo\_hz group*

mdvp\_fo\_hz\_group\_col.div(mdvp\_fo\_hz\_group\_col.sum(1).astype(float), axis**=**0).plot(kind**=**'bar',stacked**=True**)

plt.title("PD status with different mdvp\_fo\_hz group") *# setting title of the figure*

status 0 1

mdvp\_fo\_hz

mdvp\_fo\_hz : 50-100 0.000000 100.000000

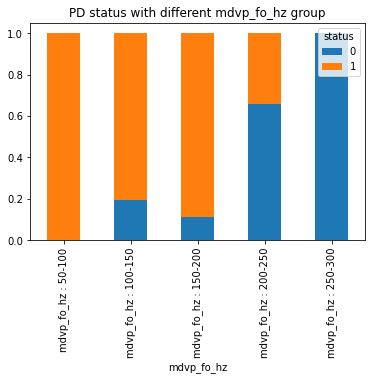
mdvp\_fo\_hz : 100-150 19.565217 80.434783

mdvp\_fo\_hz : 150-200 11.475410 88.524590

mdvp\_fo\_hz : 200-250 65.625000 34.375000

mdvp\_fo\_hz : 250-300 100.000000 0.000000

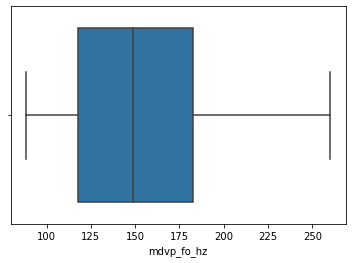
Text(0.5, 1.0, 'PD status with different mdvp\_fo\_hz group')

****

We can find out the following from the above crosstab: All the patient with Average vocal fundamental frequency (mdvp\_fo\_hz) group between 50-100 are having Parkinson's disease. Average vocal fundamental frequency (mdvp\_fo\_hz) group between 150-200 having second higest Parkinson's patient with percentage of 88.525, followed by Average vocal fundamental frequency (mdvp\_fo\_hz) group between 100-150 having Parkinson's patient with percentage of 80.435 . Average vocal fundamental frequency (mdvp\_fo\_hz) group between 200-250 having Healthy patient with percentage of 65.625 . All the patient with Average vocal fundamental frequency (mdvp\_fo\_hz) group between 250-300 are Healthy.

Let's check outliers for the attribute :

ax **=** sns.boxplot(x**=**pdData[feature]) *# seaborn boxplot to examine outliers of the feature*

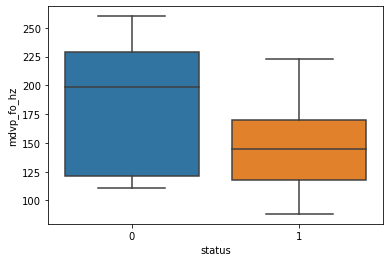
****

There are no outliers presnt in the 'mdvp\_fo\_hz' feature / attribute as we can see from above boxplot.

Now will check if any outliers present for different target attributes i.e 'status'

sns.boxplot(x**=**pdData['status'],y**=**pdData[feature])

<AxesSubplot:xlabel='status', ylabel='mdvp\_fo\_hz'>

****

From above, it is observed there are no outliers present in mdvp\_fo\_hz attribute for different 'status' attributes.

* 'ppe' attribute : (PPE - Nonlinear measures of fundamental frequency variation )

feature **=** 'ppe'

meanData **=** 'Mean : ' **+** str(round(pdData[feature].mean(),6)) *# variable to contain mean of the attribute*

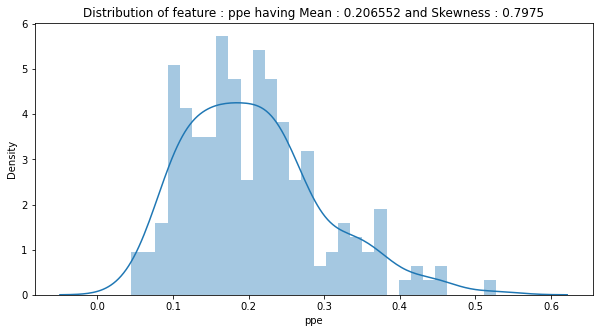
skewData **=** 'Skewness : ' **+** str(round(pdData[feature].skew(),4)) *# variable to contain skewness of the attribute*

plt.figure(figsize**=**(10,5)) *# setting figure size with width = 10 and height = 5*

fig **=** sns.distplot(pdData[feature], bins**=**30, kde**=True**) *# seaborn distplot to examine distribution of the feature*

plt.title("Distribution of feature : "**+**feature**+**" having "**+**meanData**+**" and "**+**skewData) *# setting title of the figure*

plt.show()

From above we can understand the following about ppe (PPE) attribute of the datasetMean value of the skewness of 0.7975, which shows that the datapoints of the attribute is slightly right / positive skewed. Maximum datapoints are ranging from 0.1 to 0.27 .

plt.figure(figsize**=**(10,5)) *# setting figure size with width = 10 and height = 5*

*# seaborn distplot to examine distribution of the feature of healthy patient*

fig **=** sns.distplot(pdData[pdData['status'] **==** 0][feature], bins**=**30, kde**=True**, label**=**'Healthy')

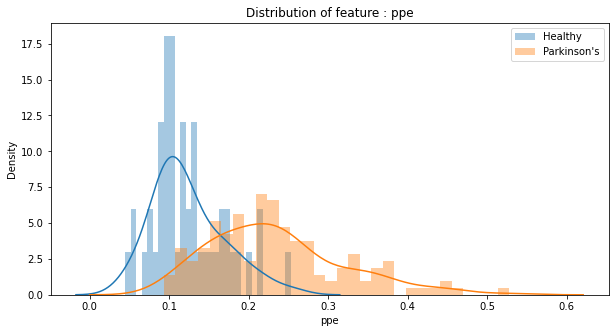
*# seaborn distplot to examine distribution of the feature of Parkinson's patient*

fig **=** sns.distplot(pdData[pdData['status'] **==** 1][feature], bins**=**30, kde**=True**, label**=**'Parkinson\'s')

plt.legend()

plt.title("Distribution of feature : "**+**feature) *# setting title of the figure*

plt.show()



From the above we can observe that, patients with ppe (PPE) values greater than 0.16 are more likly to have Parkinson's disease.

Lets bucket ppe (PPE) and check w.r.t the different status i.e Healthy or Parkinson's:

bins **=** [0.0, 0.1, 0.2, 0.3, 0.4, 0.5] *# defining ppe bins,*

*# defining labels of ppe groups as per bins defined as above*

ppe\_group **=** ['0.0-0.1', '0.1-0.2', '0.2-0.3', '0.3-0.4', '0.4-0.5']

pdData\_ppe\_bin **=** pd.cut(pdData[feature],bins,labels**=**ppe\_group) *# segmenting data as per bins defined*

​

*# putting into pandas crosstab and applying lambda function to take percentage and assigning to ppe\_group\_col variable*

ppe\_group\_col **=** pd.crosstab(pdData\_ppe\_bin,pdData.status).apply(**lambda** r: r**/**r.sum()**\***100, axis**=**1)

print(ppe\_group\_col) *# printing above crosstab*

​

*# plotting a stacked bar chart to show PD status for different mdvp\_fo\_hz group*

ppe\_group\_col.div(ppe\_group\_col.sum(1).astype(float), axis**=**0).plot(kind**=**'bar',stacked**=True**)

plt.title("PD status with different ppe group") *# setting title of the figure*

status 0 1

ppe

0.0-0.1 94.444444 5.555556

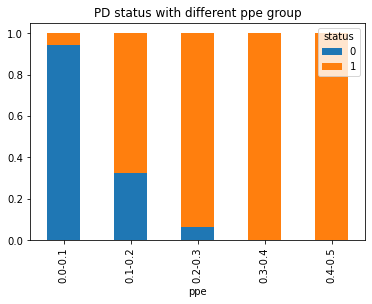
0.1-0.2 32.530120 67.469880

0.2-0.3 6.250000 93.750000

0.3-0.4 0.000000 100.000000

0.4-0.5 0.000000 100.000000

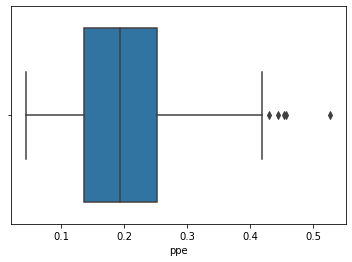
Text(0.5, 1.0, 'PD status with different ppe group')



We can find out the following from the above crosstab: All patient with ppe (PPE) (DFA) group with value more than 0.3 are having Parkinson's disease. ppe (PPE) group between 0.2-0.3 having Parkinson's patient with percentage of 93.750 followed by ppe (PPE) group between 0.1-0.2 having Parkinson's patient with percentage of 67.470 . ppe (PPE) group between 0.0-0.1 having very less Parkinson's patient with percentage of 5.556 .

Let's check outliers for the attribute :

ax **=** sns.boxplot(x**=**pdData[feature]) *# seaborn boxplot to examine outliers of the feature*



In the 'ppe' attribute some outliers are present, let's check for the same :

Q1 **=** pdData[feature].quantile(0.25) *# evaluating lower / first quartile*

Q3 **=** pdData[feature].quantile(0.75) *# evaluating upper / third quartile*

IQR **=** Q3 **-** Q1 *# evaluating Inter Quartile Range i.e IQR*

'''

finding outliers which are mild outliers (Lower quartile - 1.5 times IQR) or

extreme outliers (Upper quartile + 1.5 times IQR)

'''

outliers **=** pdData[((pdData[feature] **<** (Q1 **-** 1.5 **\*** IQR)) **|**(pdData[feature] **>** (Q3 **+** 1.5 **\*** IQR)))][feature]

​

print("\*"**\***125)

*# printing mean, median and IQR for the feature*

print("\033[1mFeature {0} : Mean = {1}, Median = {2} and Inter-Quartile-Range (IQR) = {3}\033[0m"

.format(feature,round(np.mean(pdData[feature]),6),round(np.median(pdData[feature]),6),round(IQR,6))

)

print()

print("\*"**\***125)

*# printing No of outliers, percentage of the data points are outliers and the values of the outliers*

print("There are \033[1m{0} outliers\033[0m ({1} % of the data points) in \033[1m{2}\033[0m feature and the values are \033[1m{3}\033[0m"

.format(outliers.shape[0],round(((outliers.shape[0]**/**pdData[feature].shape[0])**\***100),3),feature,outliers.tolist()))

print("\*"**\***125)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Feature ppe : Mean = 0.206552, Median = 0.194052 and Inter-Quartile-Range (IQR) = 0.115529**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

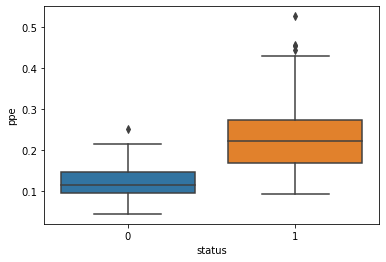
There are **5 outliers** (2.564 % of the data points) in **ppe** feature and the values are **[0.430788, 0.444774, 0.457533, 0.527367, 0.454721]**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now will check if any outliers present for different target attributes i.e 'status'

sns.boxplot(x**=**pdData['status'],y**=**pdData[feature])

<AxesSubplot:xlabel='status', ylabel='ppe'>



From above, it is observed that for 'ppe' attribute patients with Parkinson's disease have more outliers than Healthy patients.

* 'status' attribute : (Health status of the subject (one) - Parkinson's, (zero) - healthy)

plt.figure(figsize**=**(10,5)) *# setting figure size with width = 10 and height = 5*

*# seaborn count catplot to examine distribution of the status*

ax **=** sns.catplot(x**=**'status', kind**=**"count", data**=**pdData)

plt.title("Distribution of column : 'Status'") *# setting title of the figure*

y **=** [] *# creating a null or empty array*

**for** val **in** range(pdData.status.nunique()): *# looping for number of unique values in the status*

*# appending count of each unique values from status to array y*

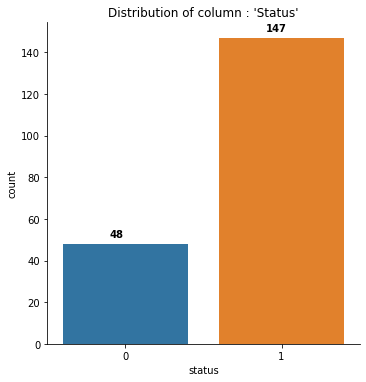
y.append(pdData.groupby(pdData.status,sort**=False**)['status'].count()[val])

**for** i, v **in** enumerate(y): *# looping count of each unique value in the status*

*# including count of each unique values in the plot*

plt.annotate(str(v), xy**=**(i,float(v)), xytext**=**(i**-**0.1, v**+**3), color**=**'black', fontweight**=**'bold')

<Figure size 720x360 with 0 Axes>



Lets check the percentage and plot a pie chart to show :

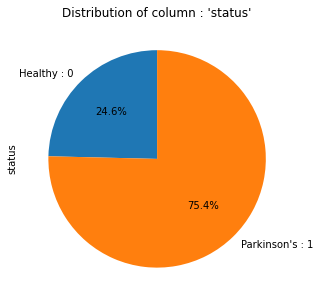
plt.figure(figsize**=**(5,5)) *# setting figure size with width = 10 and height = 5*

*# seaborn pie chart to examine distribution of the status*

pdData.groupby(['status']).status.count().plot(kind**=**'pie',labels**=**['Healthy : 0','Parkinson\'s : 1']

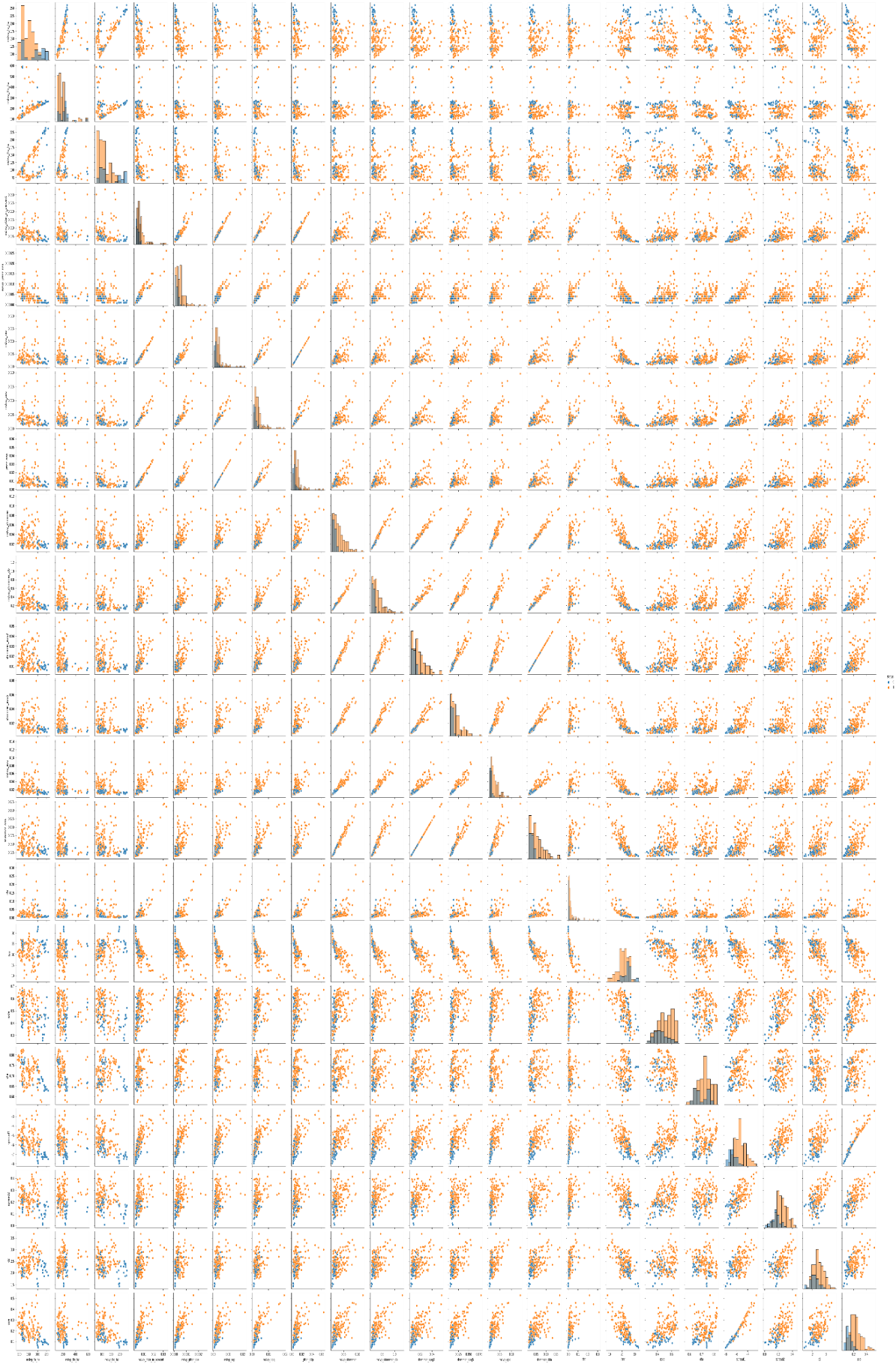
plt.title("Distribution of column : 'status'") *# setting title of the figure*

Text(0.5, 1.0, "Distribution of column : 'status'")



From above we can see out of 195 patients, 48 patients (24.6 %) are healthy and 147 patients (75.4%) patients are having Parkinson's disease.

sns.pairplot(pdData,hue**=**'status',diag\_kind**=**'hist') <seaborn.axisgrid.PairGrid at 0x2794fdf0>



plt.figure(figsize**=**(20,7))

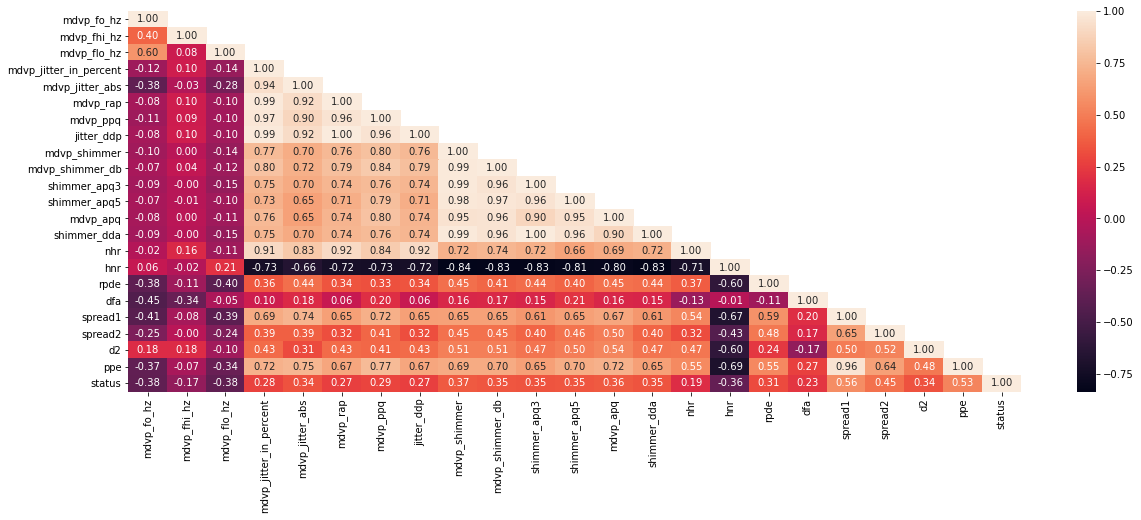
*# create a mask so we only see the correlation values once*

mask **=** np.zeros\_like(pdData.corr())

mask[np.triu\_indices\_from(mask, 1)] **=** **True**

a **=** sns.heatmap(pdData.corr(),mask**=**mask, annot**=True**, fmt**=**'.2f')

rotx **=** a.set\_xticklabels(a.get\_xticklabels(), rotation**=**90)



We can observe from the above pairplot and heatmap of correlation of different attributes:

Split the dataset into training and test set in the ratio of 70:30 (Training:Test)

*#Split the data into training and test set in the ratio of 70:30 respectively*

X **=** pdData.drop(['status'],axis**=**1)

y **=** pdData['status']

​*# split data into train subset and test subset*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.3, random\_state**=**47)

​*# checking the dimensions of the train & test subset*

*# printing dimension of train set*

print(X\_train.shape)

*# printing dimension of test set*

print(X\_test.shape)

(136, 22)

(59, 22)

Prepare the data for training - Scale the data if necessary, get rid of missing values (if any) etc

As we have seen earlier, there are no missing values in the dataset As from the earlier correlation heatmap of different attributes we found that mdvp\_jitter\_in\_percent (MDVP:Jitter(%)) have high correlation with mdvp\_jitter\_abs (MDVP:Jitter(Abs) ), mdvp\_rap (MDVP:RAP), mdvp\_ppq (MDVP:PPQ), jitter\_ddq (Jitter:DDQ) and nhr (NHR). So, in this case we will drop mdvp\_jitter\_in\_percent (MDVP:Jitter(%)).

X\_train.drop(['mdvp\_jitter\_in\_percent'],axis**=**1,inplace**=True**)

X\_test.drop(['mdvp\_jitter\_in\_percent'],axis**=**1,inplace**=True**)

Also from the earlier correlation heatmap of different attributes we found that mdvp\_shimmer (MDVP:Shimmer) have high correlation with mdvp\_shimmer\_db (MDVP:Shimmer(dB)), shimmer\_apq3 (Shimmer:APQ3), shimmer\_apq5 (Shimmer:APQ5), mdvp\_apq (MDVP:APQ), shimmer\_dda (Shimmer:DDA). So, in this case we will drop mdvp\_shimmer (MDVP:Shimmer).

X\_train.drop(['mdvp\_shimmer'],axis**=**1,inplace**=True**)

X\_test.drop(['mdvp\_shimmer'],axis**=**1,inplace**=True**)

Also we will drop hnr (HNR).

X\_train.drop(['hnr'],axis**=**1,inplace**=True**)

X\_test.drop(['hnr'],axis**=**1,inplace**=True**)

*# re checking the dimensions of the train & test subset after dropping several columns from the subsets*

*# printing dimension of train set*

print(X\_train.shape)

*# printing dimension of test set*

print(X\_test.shape)

(136, 19)

(59, 19)

*# Let us scale train as well as test data using StandardScaler*

scaler **=** StandardScaler()

​X\_train\_scaled **=** scaler.fit\_transform(X\_train)

X\_test\_scaled **=** scaler.fit\_transform(X\_test)

Training at least 3 standard classification algorithms - Logistic Regression, SVM, KNN etc, and noting down their accuracies on the test data

**A. Logistic Regression:**

*# Train and Fit model*

lr **=** LogisticRegression(random\_state**=**0)

lr.fit(X\_train\_scaled, y\_train)

​*#predict status for X\_test\_scaled dataset*

lr\_y\_pred **=** lr.predict(X\_test\_scaled)

​*# Confusion Matrix for the Logistic Regression Model*

print("Confusion Matrix : Logistic Regression")

print(confusion\_matrix(y\_test,lr\_y\_pred))

​*# Classification Report for the Logistic Regression Model*

classRep **=** classification\_report(y\_test, lr\_y\_pred, digits**=**2)

print(classRep)

Confusion Matrix : Logistic Regression

[[ 9 4] [ 4 42]]

precision recall f1-score support

0 0.69 0.69 0.69 13

1 0.91 0.91 0.91 46

accuracy 0.86 59

macro avg 0.80 0.80 0.80 59

weighted avg 0.86 0.86 0.86 59

From the above Logistic Regression Model, we can find out the following details:

Accuracy of the model:- 86% Re-call of the model:- 91% Precision of the model:- 91% F1-Score of the model:- 91%

**B. K-nearest neighbors:**

First let's find out the value of neighbors.

*# creating odd list of K for KNN*

myList **=** list(range(3,40,2))

​*# creating empty list for F1 scores od different value of K*

f1ScoreList **=** []

​*# perform accuracy metrics for values from 3,5....29*

**for** k **in** myList:

knn **=** KNeighborsClassifier(n\_neighbors**=**k)

knn.fit(X\_train\_scaled, y\_train)

*# predict the response*

y\_pred **=** knn.predict(X\_test\_scaled)

*# evaluate F1 Score*

f1Score **=** f1\_score(y\_test, y\_pred)

f1ScoreList.append(f1Score)

​

*# changing to misclassification error*

MSE **=** [1 **-** x **for** x **in** f1ScoreList]

​*# determining best k*

bestk **=** myList[MSE.index(min(MSE))]

print("The optimal number of neighbors is %d" **%** bestk)

The optimal number of neighbors is 29

*# instantiate learning model (k = 29)*

knn **=** KNeighborsClassifier(n\_neighbors **=** 29, weights **=** 'uniform', metric**=**'euclidean')

​*# fitting the model*

knn.fit(X\_train\_scaled, y\_train)

​*# predict the response*

knn\_y\_pred **=** knn.predict(X\_test\_scaled)

​*# Confusion Matrix for the K-nearest neighbors Model*

print("Confusion Matrix : K-nearest neighbors")

print(confusion\_matrix(y\_test,knn\_y\_pred))

​*# Classification Report for the K-nearest neighbors Model*

classRep **=** classification\_report(y\_test, knn\_y\_pred, digits**=**2)

print(classRep)

Confusion Matrix : K-nearest neighbors

[[ 8 5]

[ 0 46]]

precision recall f1-score support

0 1.00 0.62 0.76 13

1 0.90 1.00 0.95 46

accuracy 0.92 59

macro avg 0.95 0.81 0.86 59

weighted avg 0.92 0.92 0.91 59

From the above K-nearest neighbors Model, we can find out the following details:

Accuracy of the model:- 92% Re-call of the model:- 100% Precision of the model:- 90% F1-Score of the model:- 95%

**C. SVM (Support Vector Machine):**

svm **=** SVC(gamma**=**0.05, C**=**70,random\_state**=**47)

svm.fit(X\_train\_scaled , y\_train)

​*# predict the response*

svm\_y\_pred **=** svm.predict(X\_test\_scaled)

​*# Confusion Matrix for the Support Vector Machine Model*

print("Confusion Matrix : Support Vector Machine")

print(confusion\_matrix(y\_test,svm\_y\_pred))

​*# Classification Report for the Support Vector Machine Model*

classRep **=** classification\_report(y\_test, svm\_y\_pred, digits**=**2)

print(classRep)

Confusion Matrix : Support Vector Machine

[[10 3]

[ 0 46]]

precision recall f1-score support

0 1.00 0.77 0.87 13

1 0.94 1.00 0.97 46

accuracy 0.95 59

macro avg 0.97 0.88 0.92 59

weighted avg 0.95 0.95 0.95 59

From the above Support Vector Machine Model, we can find out the following details:

Accuracy of the model:- 95% Re-call of the model:- 100% Precision of the model:- 94% F1-Score of the model:- 97%

Determining which standard model performed better

*#Using K fold to check how the above algorighms varies throughout the dataset with 10 different subset of equal bins*

models **=** []

models.append(('Logistic Regression', LogisticRegression(random\_state**=**47)))

models.append(('K-NN', KNeighborsClassifier(n\_neighbors **=** 29, weights **=** 'uniform', metric**=**'euclidean')))

models.append(('SVM', SVC(gamma**=**0.05, C**=**70,random\_state**=**47)))

​

*# evaluate each model*

results **=** []

names **=** []

scoring **=** 'f1'

**for** name, model **in** models:

kfold **=** model\_selection.KFold(n\_splits**=**10,shuffle**=True**, random\_state**=**47)

cv\_results **=** model\_selection.cross\_val\_score(model, X\_train\_scaled, y\_train, cv**=**kfold, scoring**=**scoring)

results.append(cv\_results)

names.append(name)

print("\033[1m{0}\033[0m model have \033[1mmean F1-Score\033[0m of {1} and \033[1mSD F1-Score\033[0m of {2}".format(name, cv\_results.mean(), cv\_results.std()))

**Logistic Regression** model have **mean F1-Score** of 0.8930882775410977 and **SD F1-Score** of 0.05346316304648212

**K-NN** model have **mean F1-Score** of 0.8805069690035365 and **SD F1-Score** of 0.04633386023424363

**SVM** model have **mean F1-Score** of 0.9453975265995727 and **SD F1-Score** of 0.033233692576697435

plt.title('Algorithm Comparison')

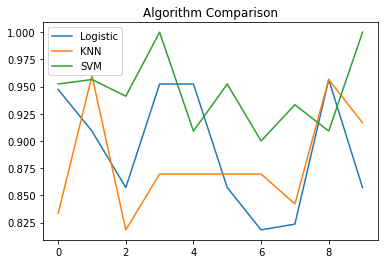
plt.plot(results[0],label**=**'Logistic')

plt.plot(results[1],label**=**'KNN')

plt.plot(results[2],label**=**'SVM')

plt.legend()

<matplotlib.legend.Legend at 0x2796cf86d60>



From the above comparision of different algorithms (Logistic Regression, K-nearest neighbors and Support Vector Machine) we can conclude that SVM (Support Vector Machine) performed slightly better than other algorithms.

**Train a meta-classifier and note the accuracy on test data**

**STACKING:**

*# defining level hetrogenious model*

level0 **=** list()

level0.append(('lr', LogisticRegression(random\_state**=**47)))

level0.append(('knn', KNeighborsClassifier(n\_neighbors **=** 29, weights **=** 'uniform', metric**=**'euclidean')))

level0.append(('cart', DecisionTreeClassifier()))

level0.append(('svm', SVC(gamma**=**0.05, C**=**70,random\_state**=**47)))

level0.append(('bayes', GaussianNB()))

​*# define meta learner model*

level1 **=** SVC(gamma**=**0.05, C**=**3,random\_state**=**47)

​*# define the stacking ensemble with cross validation of 5*

Stack\_model **=** StackingClassifier(estimators**=**level0, final\_estimator**=**level1, cv**=**5)

​*# predict the response*

Stack\_model.fit(X\_train\_scaled, y\_train)

prediction\_Stack **=** Stack\_model.predict(X\_test\_scaled)

​

*# Confusion Matrix for the Stacking Model*

print("Confusion Matrix : Stacking")

print(confusion\_matrix(y\_test,prediction\_Stack))

​*# Classification Report for the Stacking Model*

print(classification\_report(y\_test, prediction\_Stack, digits**=**2))

Confusion Matrix : Stacking

[[10 3]

[ 0 46]]

precision recall f1-score support

0 1.00 0.77 0.87 13

1 0.94 1.00 0.97 46

accuracy 0.95 59

macro avg 0.97 0.88 0.92 59

weighted avg 0.95 0.95 0.95 59

**AUC-ROC for stacking**

*#determining false positive rate and True positive rate, threshold*

fpr, tpr, threshold **=** metrics.roc\_curve(y\_test, prediction\_Stack)

roc\_auc\_stack **=** metrics.auc(fpr, tpr)

​*#plotting ROC curve*

plt.title('Receiver Operating Characteristic')

plt.plot(fpr, tpr, 'b', label **=** 'AUC = %0.2f' **%** roc\_auc\_stack)

plt.legend(loc **=** 'lower right')

plt.plot([0, 1], [0, 1],'r--')

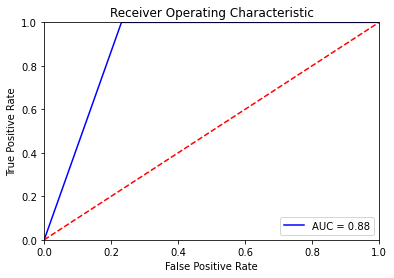
plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()



From the above Stacked meta classifier Model, we can find out the following details:

Accuracy of the model:- 95% Re-call of the model:- 100% Precision of the model:- 94% F1-Score of the model:- 97% ROC-AUC : 88%

Training at least one standard Ensemble model - Random forest, Bagging, Boosting etc, and noting the accuracy A. Random Forest

*#creating model of Random Forest*

RandomForest **=** RandomForestClassifier(n\_estimators **=** 100,criterion**=**'entropy',max\_features**=**10,random\_state**=**47)

RandomForest **=** RandomForest.fit(X\_train\_scaled, y\_train)

​

*# predict the response*

RandomForest\_pred **=** RandomForest.predict(X\_test\_scaled)

​

*# Confusion Matrix for the Random Forest Model*

print("Confusion Matrix : Random Forest")

print(confusion\_matrix(y\_test,RandomForest\_pred))

​

*# Classification Report for the Randome Forest Model*

print(classification\_report(y\_test, RandomForest\_pred, digits**=**2))

Confusion Matrix : Random Forest

[[ 9 4]

[ 1 45]]

precision recall f1-score support

0 0.90 0.69 0.78 13

1 0.92 0.98 0.95 46

accuracy 0.92 59

macro avg 0.91 0.84 0.86 59

weighted avg 0.91 0.92 0.91 59

**AUC-ROC for Random Forest**

*#determining false positive rate and True positive rate, threshold*

fpr, tpr, threshold **=** metrics.roc\_curve(y\_test, RandomForest\_pred)

roc\_auc\_rf **=** metrics.auc(fpr, tpr)

​*#plotting ROC curve*

plt.title('Receiver Operating Characteristic')

plt.plot(fpr, tpr, 'b', label **=** 'AUC = %0.2f' **%** roc\_auc\_rf)

plt.legend(loc **=** 'lower right')

plt.plot([0, 1], [0, 1],'r--')

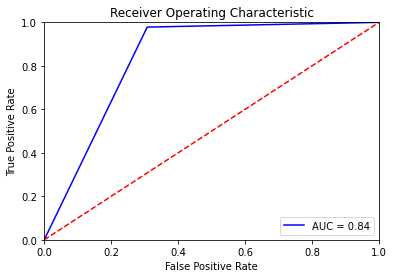
plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()



From the above Random Forest Model, we can find out the following details:

Accuracy of the model:- 92% Re-call of the model:- 98% Precision of the model:- 92% F1-Score of the model:- 95% ROC-AUC : 84%

*# Lets check features importance*

feature\_imp **=** pd.Series(RandomForest.feature\_importances\_,index**=**X\_train.columns).sort\_values(ascending**=False**)

feature\_imp

ppe 0.212646

spread1 0.181815

mdvp\_fo\_hz 0.078539

mdvp\_apq 0.066849

mdvp\_fhi\_hz 0.053630

nhr 0.048251

jitter\_ddp 0.044719

spread2 0.044320

mdvp\_rap 0.041004

mdvp\_flo\_hz 0.036010

rpde 0.029126

dfa 0.028516

mdvp\_jitter\_abs 0.028306

d2 0.025957

shimmer\_dda 0.022860

mdvp\_ppq 0.020985

shimmer\_apq3 0.012780

mdvp\_shimmer\_db 0.012384

shimmer\_apq5 0.011302

dtype: float64

*# Creating a bar plot*

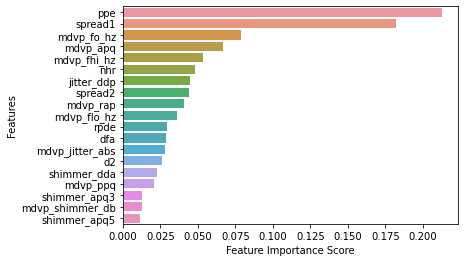
sns.barplot(x**=**feature\_imp, y**=**feature\_imp.index)

*# Add labels to your graph*

plt.xlabel('Feature Importance Score')

plt.ylabel('Features')

Text(0, 0.5, 'Features')



**Adaptive Boosting**

#Creating Model for Adaptive Boosting

AdBs **=** AdaBoostClassifier( n\_estimators**=** 50)

AdBs **=** AdBs.fit(X\_train\_scaled, y\_train)

​*# predict the response*

AdBs\_y\_pred **=** AdBs.predict(X\_test\_scaled)

​*# Confusion Matrix for the Adaptive Boosting Model*

print("Confusion Matrix : Adaptive Boosting")

print(confusion\_matrix(y\_test,AdBs\_y\_pred))

​*# Classification Report for the Adaptive Boosting Model*

print(classification\_report(y\_test, AdBs\_y\_pred, digits**=**2))

Confusion Matrix : Adaptive Boosting

[[ 9 4]

[ 2 44]]

precision recall f1-score support

0 0.82 0.69 0.75 13

1 0.92 0.96 0.94 46

accuracy 0.90 59

macro avg 0.87 0.82 0.84 59

weighted avg 0.89 0.90 0.90 59

*#AUC-ROC for AdaBoost*

*#determining false positive rate and True positive rate, threshold*

fpr, tpr, threshold **=** metrics.roc\_curve(y\_test, AdBs\_y\_pred)

roc\_auc\_ada **=** metrics.auc(fpr, tpr)

​

*#plotting ROC curve*

plt.title('Receiver Operating Characteristic')

plt.plot(fpr, tpr, 'b', label **=** 'AUC = %0.2f' **%** roc\_auc\_ada)

plt.legend(loc **=** 'lower right')

plt.plot([0, 1], [0, 1],'r--')

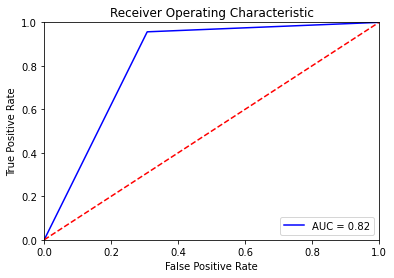
plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()



From the above Adaptive Boosting Model, we can find out the following details:

Accuracy of the model:- 90% Re-call of the model:- 96% Precision of the model:- 92% F1-Score of the model:- 94% ROC-AUC : 82%

Comparing all the models (minimum 5) and picking the best one among them

*#Using K fold to check how the various algorithms varies throughout the dataset with 10 different subset of*

*equal bins*

models **=** []

models.append(('Logistic Regression', LogisticRegression(random\_state**=**47)))

models.append(('K-NN', KNeighborsClassifier(n\_neighbors **=** 29, weights **=** 'uniform', metric**=**'euclidean')))

models.append(('SVM', SVC(gamma**=**0.05, C**=**70,random\_state**=**47)))

models.append(('Stacking', StackingClassifier(estimators**=**level0, final\_estimator**=**level1, cv**=**5)))

models.append(('Random Forest', RandomForestClassifier(n\_estimators **=** 100,criterion**=**'entropy',max\_features**=**10,random\_state**=**47)))

models.append(('Adaptive Boosting', AdaBoostClassifier( n\_estimators**=** 50)))

​

*# evaluate each model with scoring method accuracy*

print("\*"**\***125)

print("Accuracy scoring of the Models")

print("\*"**\***125)

results **=** []

names **=** []

scoring **=** 'accuracy'

**for** name, model **in** models:

kfold **=** model\_selection.KFold(n\_splits**=**10,shuffle**=True**, random\_state**=**47)

cv\_results **=** model\_selection.cross\_val\_score(model, X\_train\_scaled, y\_train, cv**=**kfold, scoring**=**scoring)

results.append(cv\_results)

names.append(name)

print("\033[1m{0}\033[0m model have \033[1mmean Accuracy\033[0m of {1} and \033[1mSD Accuracy\033[0m of {2}"

.format(name, round(cv\_results.mean(),2), round(cv\_results.std(),2)))

​

​

print()

print("\*"**\***125)

print("F1 scoring of the Models")

print("\*"**\***125)

​

*# evaluate each model with scoring method f1*

results **=** []

names **=** []

scoring **=** 'f1'

**for** name, model **in** models:

kfold **=** model\_selection.KFold(n\_splits**=**10,shuffle**=True**, random\_state**=**47)

cv\_results **=** model\_selection.cross\_val\_score(model, X\_train\_scaled, y\_train, cv**=**kfold, scoring**=**scoring)

results.append(cv\_results)

names.append(name)

print("\033[1m{0}\033[0m model have \033[1mmean F1-Score\033[0m of {1} and \033[1mSD F1-Score\033[0m of {2}"

.format(name, round(cv\_results.mean(),2), round(cv\_results.std(),2)))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Accuracy scoring of the Models

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Logistic Regression** model have **mean Accuracy** of 0.84 and **SD Accuracy** of 0.08

**K-NN** model have **mean Accuracy** of 0.8 and **SD Accuracy** of 0.07

**SVM** model have **mean Accuracy** of 0.92 and **SD Accuracy** of 0.05

**Stacking** model have **mean Accuracy** of 0.9 and **SD Accuracy** of 0.06

**Random Forest** model have **mean Accuracy** of 0.9 and **SD Accuracy** of 0.07

**Adaptive Boosting** model have **mean Accuracy** of 0.88 and **SD Accuracy** of 0.07

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

F1 scoring of the Models

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Logistic Regression** model have **mean F1-Score** of 0.89 and **SD F1-Score** of 0.05

**K-NN** model have **mean F1-Score** of 0.88 and **SD F1-Score** of 0.05

**SVM** model have **mean F1-Score** of 0.95 and **SD F1-Score** of 0.03

**Stacking** model have **mean F1-Score** of 0.95 and **SD F1-Score** of 0.05

**Random Forest** model have **mean F1-Score** of 0.93 and **SD F1-Score** of 0.05

**Adaptive Boosting** model have **mean F1-Score** of 0.92 and **SD F1-Score** of 0.05

We can conclude from the above Accuracy and F1 scoring method that, Stacking Model performs better than other models.

Stacking Model have mean Accuracy of 90% with standard deviation of 6% . And, Stacking Model have mean F1-Score of 95% with standard deviation of 5%.

1. **QR Code Scanner**



By

T Kamaljeet (160718733037)

P Saritha (160718733047)

R Vishal (160718733040)