MACHINE LEARNING APPROACHES FOR DETECTION OF PARKINSON DISEASE USING R

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

by

U. MANIKANTA (21BCE9726)

Under the Guidance of

DR . SANKET MISHRA

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

DECEMBER 2024

CERTIFICATE

This is to certify that the Capstone Project work titled “**MACHINE LEARNING APPROACHES FOR DETECTION OF PARKINSION AND DISEASE USING R”** that is being submitted by U. MANIKANTA (21BCE9726) is in partial fulfilment of the requirements for the award of Bachelor of Technology, is a record of Bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

DR . SANKET MISHRA

Guide

The thesis is satisfactory / unsatisfactory

Prof. Selva Kumar	**Prof. Monika**
Internal Examiner1	**Internal Examiner2**

Approved by

HoD, SCOPE

School of Computer Science and Engineering

ACKNOWLEDGEMENTS

We express our gratitude to Dr.Sanket Mishra our mentor, for his encouragement and assistance during the whole endeavour. He would frequently ask me questions about specific parts of the project that would help me figure out why and how to solve particular problems. His unwavering support has had a great effect on me, and his concepts are reflected in the things I do. We also want to thank the School of Computer Science and Engineering for giving me the chance to improve my abilities and expand my skill set, both of which will undoubtedly help me in the years to come. A project serves as a scaffold between theoretical and practical learning. With this in mind, we worked hard and made progress on the endeavour, with the support and encouragement of everyone in our immediate vicinity. Yes, I would also want to thank my supervisors and friends for their encouragement and support in helping me organize and come up with creative solutions for my assignments. I am really appreciative of each of these. We were able to complete this work and make it a wonderful and enjoyable experience because of them. This project is the result of many people's tireless labour, without which it would not have been feasible.

ABSTRACT

This report focuses on using vocal attributes for detecting Parkinson's disease, a progressive neurological disorder that affects movement and often impacts voice and speech patterns. The dataset under analysis includes features such as jitter, shimmer, and pitch variations, which are valuable for assessing vocal impairments commonly observed in individuals with Parkinson's.

The analysis was done using the **easystats collection** of R packages, which made the process easier and more efficient. First, the data was cleaned and standardized with the **datawizard** package. Then, we used **correlation** to check for relationships between the vocal features and **effectsize** to understand the strength of these relationships. **Modelbased** was used to create statistical models that compare different groups, and the results were summarized with the **parameters** package. The **performance** package helped evaluate the accuracy of the models. We also created clear charts with the **see** package and used **insight** to understand the results from different models. Finally, the **report** package automatically generated a summary of the findings.

TABLE OF CONTENTS

List of Tables

Table No. **Title** **Page No.**

List of Figures

CHAPTER 1
INTRODUCTION

Parkinson's disease (PD) is a progressive neurological disorder that predominantly affects motor skills, leading to symptoms such as tremors, muscle stiffness, slowness of movement (bradykinesia), and balance difficulties. PD occurs when the brain cells that produce dopamine, a neurotransmitter involved in coordinating smooth and controlled movements, are damaged or die. The impact of Parkinson’s disease extends beyond physical motor symptoms, as it also affects cognitive function and speech patterns, with many patients exhibiting changes in voice quality over time. These changes can manifest as tremors, variations in pitch, and irregular speech rhythms, making voice analysis an essential tool in diagnosing and monitoring the disease.

Early diagnosis of Parkinson's disease can significantly improve the quality of life for patients by enabling timely interventions. Voice analysis has become a promising method for early detection, as the vocal symptoms often appear before other motor symptoms become apparent. Various acoustic features of the voice, such as jitter (frequency variation), shimmer (amplitude variation), pitch range, and speech duration, can be affected by Parkinson’s disease. Analysing these vocal parameters can reveal distinct patterns that differentiate individuals with Parkinson's disease from healthy individuals, providing valuable insights for diagnostic tools.

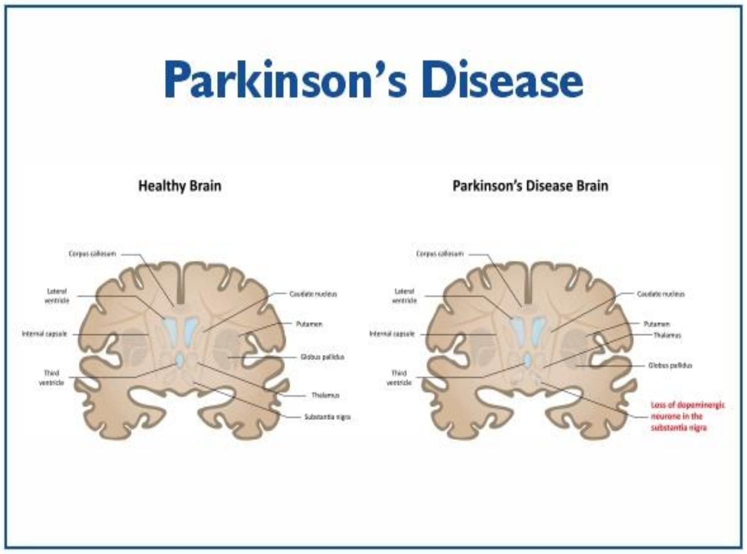


Figure 1: Healthy and Parkinson’s disease brain

The goal is to explore the relationship between these features and a categorical class variable, potentially distinguishing between different conditions or groups (e.g., healthy vs. diseased).

A key feature of this analysis is the use of easystats, a powerful R package designed to streamline statistical analysis and simplify the process of data exploration, hypothesis testing, and model evaluation. Throughout this report, easystats functions are leveraged to perform correlation analysis, statistical modelling, and effect size calculations, significantly enhancing the ease and efficiency of the analysis.

Key Steps:

- ***Data Cleaning and Transformation (Using datawizard)***
  The dataset is cleaned by addressing missing values, duplicates, and ensuring proper variable types. Numeric features are standardized for consistency, and exploratory analysis is performed to assess skewness and kurtosis of key variables.
- ***Correlation Analysis (Using easystats)***
  Correlation analysis is conducted to examine the relationships between numeric features. Spearman and Pearson correlation methods are applied to identify strong relationships and inform further analysis, helping reduce multicollinearity.
- ***Modeling Group Differences (Using modelbased)***
  Statistical models are employed to evaluate the effect of features on the target class. This involves group comparisons, with visualizations of how features differ across classes. The **modelbased** package is used to estimate group means and confidence intervals.
- ***Effect Size Calculation (Using effectsize)***
  Effect sizes are calculated to determine the magnitude of differences between groups, providing practical insights into the significance of the results. The **effectsize** package allows for the calculation of Cohen's d, Hedges' g, and Glass's delta.
- ***Principal Component Analysis (PCA) and Model Parameters (Using parameters)***
  ANOVA is conducted to assess group differences, and Principal Component Analysis (PCA) is used to reduce data complexity. The **parameters** package helps extract and summarize model parameters, such as effect sizes and significance tests.
- ***Support Vector Machine (SVM) Classification (Using Insight)***
  A classification model is built to predict the target class using the features. The dataset is split into training and testing sets, and **Insight** is used to evaluate model performance and analyze results.
- ***Model Performance Evaluation (Using Performance)***
  Performance metrics, such as accuracy, sensitivity, specificity, and AUC, are calculated to evaluate the model's effectiveness. Visualizations of these metrics help assess strengths and weaknesses.
- ***Automated Statistical Reporting (Using report)***
  An automated report summarises the statistical findings, providing a concise overview of the results and the model's performance.

Objectives

The following are the objectives of this project:

- The project implements the **easystats** package to streamline the analysis of Parkinson’s disease data, focusing on efficient data exploration, visualization, and statistical modeling.
- By leveraging **easystats**, the aim is to analyze various features related to Parkinson’s disease, including jitter, shimmer, pitch, and other voice-related variables, to identify key patterns and correlations in the dataset.
- The target is to understand the relationship between these features and the classification of Parkinson’s disease, using statistical tools to estimate effect sizes, correlations, and develop predictive models.
- Ultimately, the project aims to build a reliable model for Parkinson’s disease prediction, assess its performance, and provide comprehensive statistical reports on the significance of each feature, ensuring clear insights into the condition.

1.1 Background and Literature Survey

Background

Parkinson's disease (PD) is a chronic neurodegenerative disorder that affects millions of people worldwide, with prevalence increasing with age. The disease is characterized by motor symptoms such as tremors, rigidity, and bradykinesia, as well as non-motor symptoms, including cognitive and emotional disturbances (Poewe et al., 2017). Early detection of PD is crucial for better disease management and improving patients' quality of life. However, traditional diagnostic methods often rely on clinical symptoms that manifest later in the disease's progression, making early diagnosis challenging.

Speech impairments are among the early symptoms of PD and include changes in voice quality, pitch, and rhythm. These alterations, such as dysarthria and dysphonia, are caused by the progressive loss of motor control over speech muscles. Recent research has explored the use of machine learning (ML) techniques to analyze acoustic features derived from voice recordings as biomarkers for PD, offering a non-invasive and cost-effective diagnostic method (Lisanne van Gelderen et al., 2024; Uriarte-Arcia et al., 2023).

Literature Survey

The application of machine learning for Parkinson’s disease diagnosis using voice data has been extensively studied. Little et al. (2009) demonstrated the potential of acoustic measurements, such as jitter and shimmer, as effective features for distinguishing PD patients from healthy controls. They employed support vector machines (SVMs) and achieved high accuracy, laying the groundwork for further research.

Recent studies have focused on optimizing feature selection and classification techniques. Senturk (2020) highlighted the role of recursive feature elimination (RFE) combined with SVMs, achieving an accuracy of 93.84% in PD classification. Other research has incorporated advanced methods like principal component analysis (PCA) to improve classification accuracy by reducing dimensionality and focusing on the most relevant features (Arias-Vergara et al., 2020; Rasheed et al., 2020).

Deep learning approaches have also been investigated for their ability to capture complex patterns in voice data. Van Gelderen et al. (2024) provided a systematic review of end-to-end deep learning frameworks, transfer learning, and hybrid models. These methods have demonstrated promising results, though their application is often limited by the availability of large datasets and computational resources.
Furthermore, associative memory models have shown exceptional performance in PD diagnosis. For instance, the Improved Smallest Normalized Difference Associative Memory (ISNDAM) model achieved classification accuracies exceeding 99% using datasets from the UCI Machine Learning Repository (Uriarte-Arcia et al., 2023). These models demonstrate the potential for integrating explainable AI into medical diagnostics.
Despite advancements, challenges remain in achieving generalizability across diverse populations, ensuring interpretability of complex models, and addressing biases in datasets (Lisanne van Gelderen et al., 2024). Continued research into hybrid approaches that combine feature-based and end-to-end methods is critical for improving the robustness of diagnostic tools.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

• **Chapter 2**: Contains the proposed system, methodology, hardware and software details, and system architecture.
• **Chapter 3**: Describes the project requirements, implementation of the project, and system flow design.
• **Chapter 4**: Discuss the results obtained after the project was implemented.
• **Chapter 5**: Provides the conclusion and outlines future work.
• **Chapter 6**: Includes the code used for the project.
• **Chapter 7**: Lists the references used throughout the project.

CHAPTER 2

MACHINE LEARNING APPROACHES FOR DETECTION OF PARKINSON DISEASE USING R

This Chapter describes the proposed system, working methodology, software and hardware details.

2.1 Proposed System

The following block diagram (figure 2) shows how easystats is used for analysis of Parkinson's disease

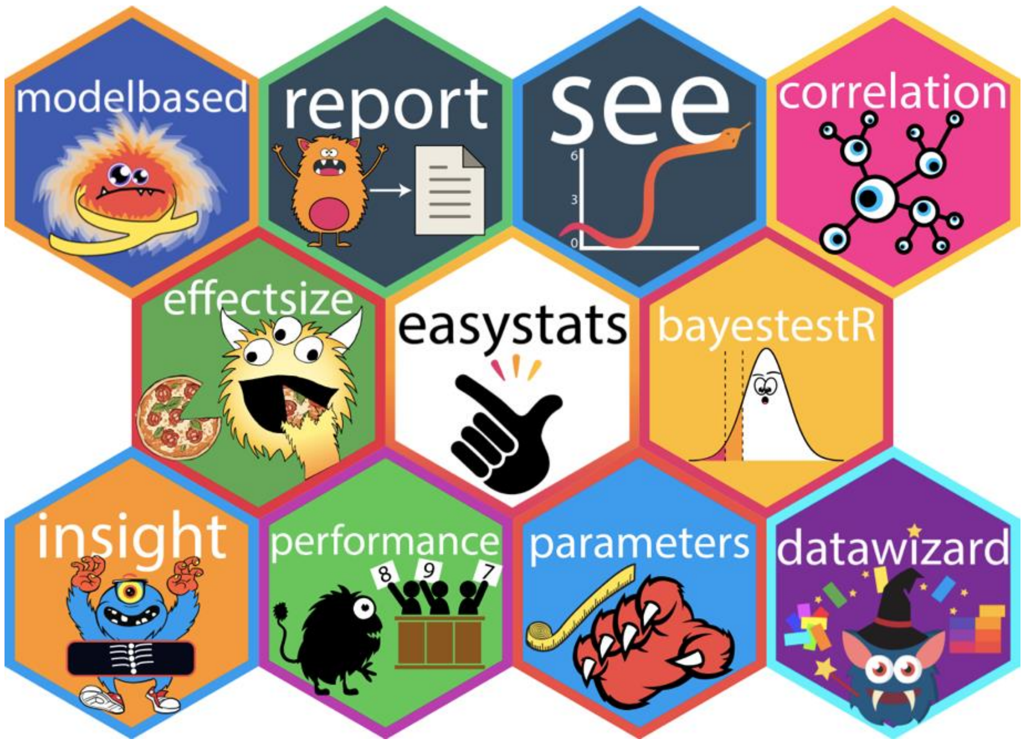


Figure 2. Easy Stats Diagram

1. **Objective Definition:** Define the analysis goal (e.g., predictors of Parkinson's diagnosis).
2. **Data Exploration:** Use `insight` to inspect data structure, distributions, and missing values.
3. **Data Preprocessing:** Normalize variables and handle missing values using `datawizard`.
4. **Correlation Analysis:** Assess variable relationships and multicollinearity with `correlation`.
5. **Feature Selection:** Compute and interpret effect sizes using `effectsize`.
6. **Model Development:** Build and interpret statistical models with `parameters` and `modelbased`.
7. **Model Evaluation:** Assess and compare model performance metrics using `performance`.

8. **Visualization:** Create intuitive plots for findings with `see`.
9. **Reporting:** Generate comprehensive, plain-text statistical summaries using `report`.

2.2 Methodology

Using easystats in R below mentioned functionalities are to be done.

Dataset and Source

- The dataset is obtained from the UCI Machine Learning Repository, for analyzing voice measurements related to Parkinson's disease.
- **Jitter_local**: Measures the short-term variability in pitch (fundamental frequency) from one period to the next, calculated for small local segments of the voice.
- **Jitter_local_absolute**: Absolute value of the local jitter, representing the variation in frequency without considering the direction (increase or decrease) in pitch changes.
- **Jitter_rap**: Relative Average Perturbation (RAP) is a measure of pitch variation over short periods and reflects the perceived stability of a person's voice.
- **Jitter_ppq5**: Jitter in terms of the Pitch Perturbation Quotient (PPQ5), which measures the variation in pitch over a 5-period window, often used in voice analysis.
- **Jitter_ddp**: The difference in pitch perturbation over time, focusing on the dynamic changes in pitch. It quantifies the irregularity of voice frequency in successive cycles.
- **Shimmer_local**: Measures the short-term amplitude (loudness) variation in the voice signal, typically the local variation of the sound intensity.
- **Shimmer_local_dB**: A logarithmic scale of shimmer (loudness variation) in decibels, offering a standardized way to assess amplitude fluctuations.
- **Shimmer_apq3**: The Amplitude Perturbation Quotient (APQ3) for a 3-period window, measuring short-term changes in amplitude and how irregular the voice is in terms of loudness.
- **Shimmer_apq5**: Similar to Shimmer_apq3 but over a 5-period window, used to evaluate the regularity and consistency of the speaker's loudness.
- **Shimmer_apq11**: A measure of amplitude perturbation (shimmer) over an 11-period window, providing a longer-term assessment of the loudness variation in speech.
- **Shimmer_dda**: The Difference of Differences in Amplitude (DDA) quantifies the variation in the amplitude fluctuations across multiple successive periods of speech.
- **AC (Autocorrelation)**: A statistical measure that compares a signal with a delayed version of itself to identify periodicity or repeating patterns (used to detect pitch).
- **NTH (Normalized Harmonics-to-Noise Ratio)**: A measure of the relative amount of harmonic energy to noise in the speech signal. A higher NTH indicates clearer, less noisy speech.
- **HTN (Harmonics-to-Noise Ratio)**: Similar to NTH, but typically unnormalized. It assesses how much of the speech signal's energy comes from harmonic frequencies versus noise.
- **Median_pitch**: The median value of the fundamental frequency (pitch) in the speech sample. It is a robust measure of central tendency in pitch.

- **Mean_pitch**: The average fundamental frequency (pitch) over the entire speech sample.
- **Standard_deviation_pitch**: The variability (standard deviation) of the pitch across the speech sample, reflecting how much the pitch fluctuates.
- **Minimum_pitch**: The lowest pitch observed in the speech sample, indicating the deepest voice frequency reached during the sample.
- **Maximum_pitch**: The highest pitch observed in the speech sample, indicating the highest frequency reached during the sample.
- **Number_of_pulses**: The total number of vocal fold pulses (cycles) detected in the speech signal, often related to the speaker's rate of vibration.
- **Number_of_periods**: The total number of periods (time intervals between consecutive peaks in the signal) measured in the speech sample.
- **Mean_period**: The average duration of a single period (cycle) in the speech signal, often related to how regularly the speaker's vocal folds are vibrating.
- **Standard_deviation_period**: The variability in the duration of each period, reflecting the regularity or irregularity of the speaker's vocal fold vibrations.
- **Fraction_locally_unvoiced_frames**: The proportion of frames in the speech signal where no voiced sound is detected (unvoiced frames). This could indicate periods of silence or breathiness.
- **Number_of_voice_breaks**: The total count of voice breaks (sudden interruptions or irregularities in speech flow) within the speech signal.
- **Degree_of_voice_breaks**: A measure of the severity or intensity of voice breaks (often related to breathiness or interruptions in speech).

Data Loading and Preparation

- Import the dataset and assign appropriate column names for better readability and interpretability.
- Convert categorical variables, such as the target class (Class), into factors to facilitate statistical and machine learning tasks.
- Standardize numerical features to bring all variables to a similar scale, ensuring better model performance and interpretability.

Exploratory Data Analysis (EDA)

- Assess the dataset for missing values, duplicates, and basic statistics.
- Evaluate the skewness and kurtosis of key variables (e.g., Jitter_local, Shimmer_local) to understand their distribution properties.
- Visualize data distributions and potential anomalies using descriptive summaries and data peeks.

Correlation Analysis

- Perform Spearman and Pearson correlation analyses to understand relationships between features.
- Identify highly correlated variables (absolute correlation > 0.7) to guide dimensionality reduction and feature selection.

Feature Importance

- Conduct PCA to identify the most important features, ranked by their contribution to the variance explained in the dataset.
- Use loadings, communality, and eigenvalues to select a subset of features for downstream modeling.

Modelling and Statistical Analysis

- Fit generalized linear models (GLMs) for numeric features to evaluate group differences based on the target class.
- Estimate effect sizes (Cohen's d, Hedges' g, Glass's delta) for each feature to quantify the magnitude of differences between Parkinson's and non-Parkinson's groups.

Machine Learning

- Split the dataset into training and testing subsets for supervised learning tasks.
- Train an SVM classifier using a radial basis function kernel to predict the Class variable.
- Evaluate model performance using confusion matrices, ROC curves, and AUC metrics.

Performance Metrics and Visualization

- Compute and visualize metrics such as accuracy, sensitivity, and specificity.
- Plot variable distributions, model predictions, and statistical summaries using intuitive visualizations created with ggplot2 and see.

Reporting

- Generate detailed statistical and modelling reports using the report package.
- Document findings, including feature importance, model accuracy, and key statistical insights, for easy communication to stakeholders.

2.3 Hardware Requirements

1. Development Machine (Local System) Minimum Requirements

- **Processor (CPU):**
    - Quad-core CPU (e.g., Intel i5 10th Gen / AMD Ryzen 5 or better).
- **Memory (RAM):**
    - 8 GB (sufficient for small to medium datasets).
- **Storage:**
    - 256 GB SSD (for fast read/write operations).
- **Graphics (GPU):**
    - Integrated GPU (sufficient for classical machine learning models but not for deep learning).
- **Operating System:**
    - Windows 10/11, macOS, or Linux (Ubuntu recommended for R-based workflows).

2.4 Software Requirements

1. Operating System

- **Recommended OS:**
    - Windows 10/11
    - macOS (for seamless RStudio installation and performance).
    - Linux (e.g., Ubuntu, Fedora) for better compatibility with open-source tools.

2. Programming Language and IDE

- **Programming Language:**
    - R (primary language for data analysis, statistical computing, and machine learning).
- **Integrated Development Environment (IDE):**
    - **RStudio (Recommended for R development):**
        - Offers an intuitive interface for coding in R.

CHAPTER 3

RESULTS AND DISCUSSIONS

Section 1: Datawizard - Magic Potions to Clean and Transform Your Data

• **Data Loading and Preparation**:

- Loaded the dataset from the file and assigned meaningful column names to the variables.
- Converted the `Class` column to a factor variable, ensuring it's treated as categorical.

• **Standardization**:

- Standardized the numeric variables in the dataset, excluding non-numeric columns like `Subject_id` and `Class`.

• **Skewness and Kurtosis**:

- Calculated skewness and kurtosis for two variables: `Jitter_local` and `Shimmer_local`.
  - `Jitter_local`: Skewness = 2.09, Kurtosis = 7.86 (indicating a positively skewed distribution with heavy tails).
  - `Shimmer_local`: Skewness = 0.98, Kurtosis = 2.00 (approximately normal with slight positive skew).

• **Missing Data Check**:

- Checked for missing data using the `data_peek` function, revealing no missing values in the dataset.

• **Duplicate Rows**:

- Checked for exact duplicate rows using `data_duplicated` and found no duplicates.

• **Basic Statistics**:

- Calculated mean and standard deviation for `Jitter_local` and `Shimmer_local`:
  - `Jitter_local`: Mean = 2.68, Standard Deviation = 0.91.

    - `Shimmer_local`: Mean = 12.92, Standard Deviation = 5.45.

- **Distribution Check**:
  - Described the distribution of `Shimmer_local`:
    - The variable has a mean of 12.92, standard deviation of 5.45, an interquartile range (IQR) of 6.17, and a range between 1.19 and 41.14.

- **Data Codebook**:
  - Generated a codebook for the dataset, providing an overview of each variable, its type, missing values, and the range of values.

Name	Type	Missings	Values	N
Subject_id	Integer	0 (0.0%)	[1, 40]	1040
Jitter_local	Numeric	0 (0.0%)	[0.19, 14.38]	1040
Jitter_local_absolute	Numeric	0 (0.0%)	[0, 0]	1040
Jitter_rap	Numeric	0 (0.0%)	[0.06, 8.02]	1040
Jitter_ppq5	Numeric	0 (0.0%)	[0.08, 13.54]	1040
Jitter_ddp	Numeric	0 (0.0%)	[0.18, 24.05]	1040
Shimmer_local	Numeric	0 (0.0%)	[1.19, 41.14]	1040
Shimmer_local_dB	Numeric	0 (0.0%)	[0.1, 2.72]	1040
Shimmer_apq3	Numeric	0 (0.0%)	[0.5, 25.82]	1040
Shimmer_apq5	Numeric	0 (0.0%)	[0.71, 72.86]	1040
Shimmer_apq11	Numeric	0 (0.0%)	[0.52, 44.76]	1040
Shimmer_dda	Numeric	0 (0.0%)	[1.49, 77.46]	1040
AC	Numeric	0 (0.0%)	[0.54, 1]	1040
NTH	Numeric	0 (0.0%)	[0, 0.87]	1040
HTN	Numeric	0 (0.0%)	[0.7, 28.42]	1040

Median_pitch	Numeric	0 (0.0%)	[81.46, 468.62]	1040
Mean_pitch	Numeric	0 (0.0%)	[82.36, 470.46]	1040
Standard_deviation_pitch	Numeric	0 (0.0%)	[0.53, 293.88]	1040
Minimum_pitch	Numeric	0 (0.0%)	[67.96, 452.08]	1040
Maximum_pitch	Numeric	0 (0.0%)	[85.54, 597.97]	1040
Number_of_pulses	Integer	0 (0.0%)	[0, 1490]	1040
Number_of_periods	Integer	0 (0.0%)	[0, 1489]	1040
Mean_period	Numeric	0 (0.0%)	[0, 0.01]	1040
Standard_deviation_period	Numeric	0 (0.0%)	[0, 0.01]	1040
Fraction_locally_unvoiced_frames	Numeric	0 (0.0%)	[0, 88.16]	1040
Number_of_voice_breaks	Integer	0 (0.0%)	[0, 12]	1040
Degree_of_voice_breaks	Numeric	0 (0.0%)	[0, 69.12]	1040
UPDRS	Integer	0 (0.0%)	[1, 55]	1040
Class	Factor	0 (0.0%)	[1, 2]	1040

Table 1: Represents the codebook from datawizard

SECTION 2: Correlation - Your All-in-One Package to Run Correlations

Spearman's rank correlation and Pearson's correlation coefficient are two commonly used methods to measure the relationship between variables, but they assess different aspects of correlation. Spearman's rank correlation is a non-parametric test that evaluates the strength and direction of a monotonic relationship between two variables, based on their ranks. It is useful when the data does not necessarily follow a normal distribution or when the relationship between variables is not linear. On the other hand, Pearson's correlation measures the linear relationship between two variables, assuming both variables are normally distributed and their relationship is linear. Pearson's correlation is sensitive to outliers, which can heavily influence the results, whereas Spearman’s method is more robust against non-normal data and outliers. The table below summarizes the top 10 pairs of variables that show the highest correlation values using both methods, which highlights how strongly certain variables are related in this dataset.

Spearman's rho (Top 10)	**Spearman's rho**	**Pearson's r (Top 10)**	**Pearson's r**
Jitter_local & Jitter_ddp	0.95	Jitter_local & Jitter_ddp	0.96
Jitter_local & Jitter_rap	0.95	Jitter_local & Jitter_rap	0.96
Jitter_local & Jitter_ppq5	0.92	Jitter_local & Jitter_ppq5	0.91
Jitter_local & Jitter_local_absolute	0.89	Jitter_local & Jitter_local_absolute	0.79
Shimmer_local & Shimmer_local_dB	0.97	Shimmer_local & Shimmer_local_dB	0.96
Shimmer_local & Shimmer_apq3	0.91	Shimmer_local & Shimmer_apq3	0.88
Shimmer_local & Shimmer_apq5	0.92	Shimmer_local & Shimmer_apq5	0.83
Shimmer_local & Shimmer_ddp	0.95	Shimmer_local & Shimmer_ddp	0.88
AC & NTH	-0.99	AC & NTH	-0.99
Shimmer_local_dB & Shimmer_apq3	0.85	Shimmer_local_dB & Shimmer_apq3	0.82

Table 2: Top 10 columns with highest correlation

This table 2 illustrates the strong correlations, especially between certain variables related to jitter and shimmer measures, such as **Jitter_local**, **Jitter_ddp**, and **Shimmer_local**, showing that these features tend to behave similarly or have a similar trend. High correlation values in both Pearson’s and Spearman’s correlation matrices suggest that these variables are interdependent, which could be useful in predictive modelling or feature selection.

SECTION 3: Modelbased - Estimate Effects, Group Averages, and Contrasts Between Groups Based on Statistical Models

This section is used to analyze and visualize differences in numeric variables (e.g., `Jitter_local`, `Shimmer_local`) between two groups, such as "Parkinson Yes" and "Parkinson No," within a dataset. The workflow involves the following steps:

1. **Fitting Generalized Linear Models (GLMs)**:
   - GLMs are statistical models that explain the variation of each numeric variable based on group membership (i.e., whether the class is "Parkinson Yes" or "Parkinson No").
   - These models allow us to estimate how the mean of the variable differs between the groups.
2. **Estimating Group Means and Confidence Intervals**:
   - Using the `model_based` module from the **R easystats package**, the `estimate_means()` function computes:

    - **Group Means**: The average value of the variable for each group.
    - **Confidence Intervals**: Ranges that indicate the reliability of these mean estimates.

3. **Visualizing Results**:
    - Violin plots are generated for each variable to show the distribution of data points across groups.

Below are the Results for important features

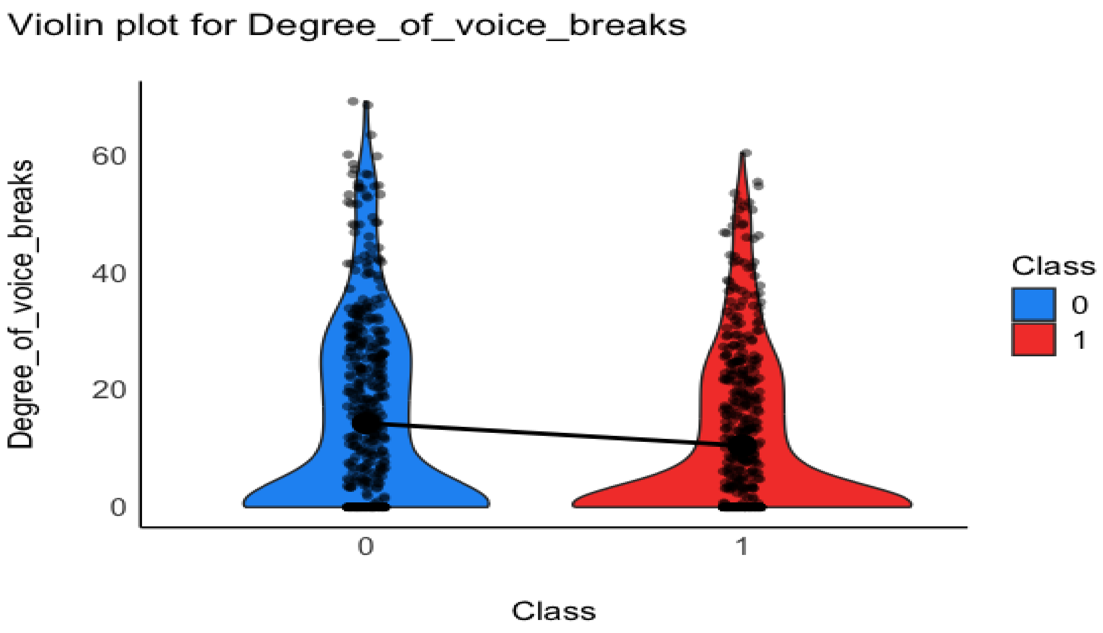


Figure 3 : Plot for degree of voice breaks for disease Yes and no

The violin plot for the variable "Degree_of_voice_breaks" highlights differences in its distribution across the two classes, "Class 0" and "Class 1". The distribution in both classes is right-skewed, indicating a higher concentration of lower values. However, Class 1 exhibits a broader spread compared to Class 0, suggesting greater variability in this measure. While the distributions overlap, Class 0 tends to have higher median values, reflecting a shift in the central tendency. This suggests that individuals in Class 0 generally exhibit higher degrees of voice breaks than those in Class 1.

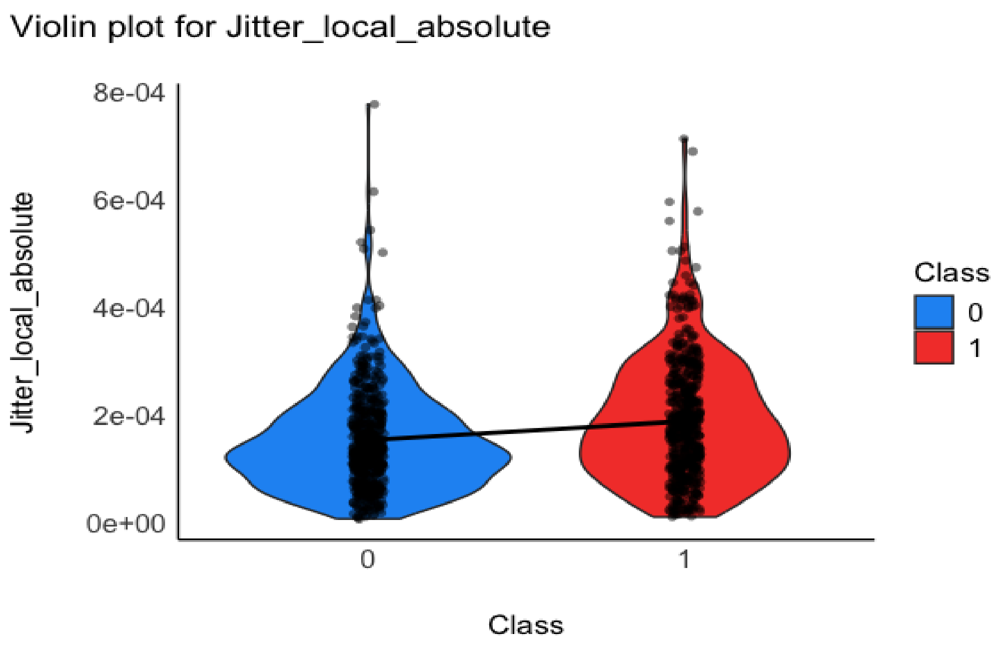


Figure 4 : Plot for Jitter local for disease Yes and no

The violin plot for "Jitter_local_absolute" illustrates the differences in its distribution between "Class 0" and "Class 1". Class 1 has a broader spread, indicating greater variability in jitter levels. There is some overlap between the distributions; however, Class 1 displays a higher median, reflecting an overall increase in jitter levels compared to Class 0. These results suggest that absolute jitter is generally more pronounced in Class 1, highlighting potential distinctions in voice stability between the two groups.

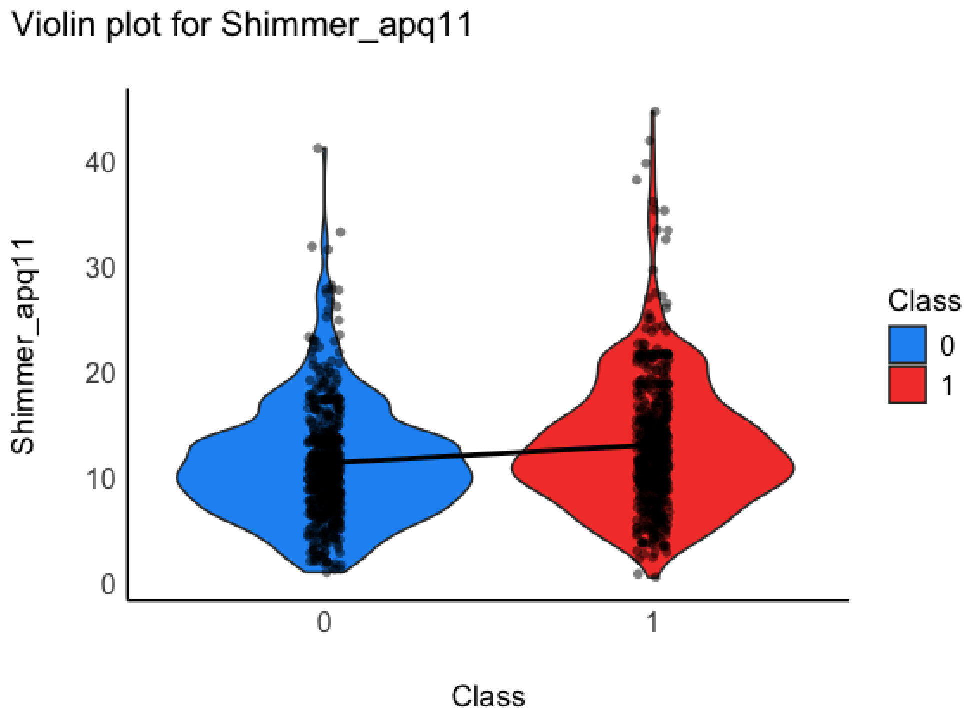


Figure 5 : Plot for Shimer for disease Yes and no.

The violin plot for "Shimmer_apq11" reveals distinct patterns in the data for "Class 0" and "Class 1". Class 1 demonstrates a more dispersed distribution than Class 0, indicating increased variability within this class. Despite some overlap between the two distributions, Class 1 clearly shows a higher median value than Class 0. This finding suggests that the shimmer amplitude perturbation quotient (APQ11) is generally higher in Class 1, which could reflect differences in voice quality across the two classes.

SECTION 4: Effectsize - Compute, Convert, Interpret, and Work with Indices of Effect Size and Standardized Parameters

This section focuses on computing, interpreting, and summarizing standardized measures of effect size to understand the magnitude of differences in variables between two groups (e.g., "Parkinson Yes" vs. "Parkinson No"). Effect size indices such as Cohen's d, Hedges' g, and Glass's Δ are utilized, along with their confidence intervals, to assess the differences for each variable in a robust and interpretable way.

Effect Sizes Used

1. **Cohen's d**:
   - Measures the standardized mean difference between two groups, quantifying how large the difference in means is relative to the variability in the data.
   - It is commonly interpreted using thresholds such as 0.2 (small effect), 0.5 (medium effect), and 0.8 (large effect).
   - Assumes equal variance between the groups and is most applicable in balanced datasets.
2. **Hedges' g**:
   - A bias-corrected version of Cohen's d, particularly useful for small sample sizes.
   - Adjusts the effect size calculation to mitigate overestimation in cases of smaller datasets.
   - Hedges' g is a reliable measure when sample sizes between groups are unequal or the dataset is limited.
3. **Glass's Δ (delta)**:
   - Focuses on the standardized mean difference relative to the standard deviation of a control group (rather than pooled standard deviation).
   - Particularly useful when the groups have unequal variances or one group is used as a benchmark.
   - Glass's Δ is most effective in experimental settings where there is a clearly defined control group.

Variable	Cohens_d	CI_Cohens_d	Hedges_g	CI_Hedges_g	Glass_delta
Jitter_local	-0.19585036	[-0.32, -0.07]	-0.19570881	[-0.32, -0.07]	-0.2045182
Jitter_local_absolute	-	[-0.46, -	-	[-0.46, -	-

	0.33842912	0.22]	0.33818452	0.22]	0.31326181
Jitter_rap	-0.22345788	[-0.35, -0.1]	-0.22329638	[-0.35, -0.1]	-0.2368385
Jitter_ppq5	-0.22303193	[-0.34, -0.1]	-0.22287074	[-0.34, -0.1]	-0.24736479
Jitter_ddp	-0.22344615	[-0.35, -0.1]	-0.22328466	[-0.35, -0.1]	-0.23682731
Shimmer_local	-0.01810758	[-0.14, 0.1]	-0.01809449	[-0.14, 0.1]	-0.0209377
Shimmer_local_dB	-0.0543485	[-0.18, 0.07]	-0.05430922	[-0.18, 0.07]	-0.06236904
Shimmer_apq3	0.03628388	[-0.09, 0.16]	0.03625765	[-0.09, 0.16]	0.04398193
Shimmer_apq5	0.04513034	[-0.08, 0.17]	0.04509772	[-0.08, 0.17]	0.06209926
Shimmer_apq11	-0.29555132	[-0.42, -0.17]	-0.29533771	[-0.42, -0.17]	-0.27767955
Shimmer_dda	0.03628696	[-0.09, 0.16]	0.03626074	[-0.09, 0.16]	0.04398568
AC	-0.16256851	[-0.28, -0.04]	-0.16245101	[-0.28, -0.04]	-0.18763741
NTH	0.15244913	[0.03, 0.27]	0.15233895	[0.03, 0.27]	0.17571885
HTN	-0.08100225	[-0.2, 0.04]	-0.0809437	[-0.2, 0.04]	-0.08910297
Median_pitch	0.16888201	[0.05, 0.29]	0.16875995	[0.05, 0.29]	0.20106441
Mean_pitch	0.20745117	[0.09, 0.33]	0.20730123	[0.09, 0.33]	0.23661973
Standard_deviation_pitch	0.24370836	[0.12, 0.37]	0.24353223	[0.12, 0.37]	0.25822216
Minimum_pitch	0.11830052	[0, 0.24]	0.11821501	[0, 0.24]	0.12943551
Maximum_pitch	0.28127458	[0.16, 0.4]	0.28107129	[0.16, 0.4]	0.30918394
Number_of_pulses	-0.08206578	[-0.2, 0.04]	-0.08200646	[-0.2, 0.04]	-0.06883849

Number_of_periods	-0.09422005	[-0.22, 0.03]	-0.09415196	[-0.22, 0.03]	-0.07876182
Mean_period	-0.1639168	[-0.29, -0.04]	-0.16379833	[-0.29, -0.04]	-0.16386062
Standard_deviation_period	0.11766588	[0, 0.24]	0.11758084	[0, 0.24]	0.1152144
Fraction_locally_unvoiced_frames	0.24568934	[0.12, 0.37]	0.24551176	[0.12, 0.37]	0.24731772
Number_of_voice_breaks	0.16001902	[0.04, 0.28]	0.15990336	[0.04, 0.28]	0.17508836
Degree_of_voice_breaks	0.25408894	[0.13, 0.38]	0.2539053	[0.13, 0.38]	0.27741776

Table 3 effect sizes for all features

In the analysis of effect sizes, several trends can be observed in how various acoustic features differ between the **Parkinson Yes** and **Parkinson No** groups.

Firstly, some variables exhibit positive effect sizes, indicating that they are higher in the **Parkinson Yes** group. These include **Maximum Pitch**, **Fraction of Locally Unvoiced Frames**, **Degree of Voice Breaks**, **Mean Pitch**, and **Standard Deviation of Pitch**. This suggests that individuals in the Parkinson's disease group tend to have higher values in these measures, which could reflect differences in vocalization patterns associated with the condition.

On the other hand, several variables show negative effect sizes, meaning they are lower in the **Parkinson Yes** group. These include various jitter and shimmer measures, such as **Jitter_local**, **Jitter_local_absolute**, **Jitter_rap**, **Jitter_ppq5**, and **Jitter_ddp**, as well as **Shimmer_local**, **Shimmer_local_dB**, **Shimmer_apq11**, and **Autocorrelation (AC)**. The reduced values of these variables in the Parkinson's group might be linked to the motor symptoms of Parkinson's disease, affecting speech production and stability.

SECTION 5: Parameters - Obtain a Table Containing All Information About the Parameters of Your Models

1. **ANOVA for Numeric Columns (Effect Size Calculation)**:
   - **Objective**: Perform ANOVA on numeric columns to determine if there are significant differences in means between the groups (e.g., "Parkinson Yes" vs. "Parkinson No") and compute effect sizes.
   - **Procedure**:
     - First, we identify which columns in the dataset are numeric.
     - Then, we perform an ANOVA for each numeric column (where "Class" is the grouping factor). This helps determine if the means of each numeric variable differ significantly between the two groups.
     - The `parameters()` function from the **`easyStats`** package is used to extract effect sizes such as omega squared, eta squared, and epsilon from the

ANOVA models. These values provide insight into the magnitude of differences between groups for each feature.

2. **Principal Component Analysis (PCA)**:
    - **Objective**: Reduce the dimensionality of the dataset by extracting the most important features.
    - **Procedure**:
        - The **numeric columns** (excluding categorical variables like "Subject_id" and "Class") are used for PCA.
        - The `psych::pca()` function is used to perform the PCA. PCA aims to transform the original variables into a smaller set of uncorrelated principal components, capturing the most variance in the data.
        - The **Varimax rotation** is applied to help make the components more interpretable.
        - The top features are selected based on the absolute loadings (i.e., which features most strongly contribute to the components).

Discussion

The results of the ANOVA and PCA analyses provide valuable insights into the relationships between the numeric features and the class variable, as well as the underlying structure of the data.

ANOVA Results

The ANOVA analysis was conducted to assess the influence of the class variable on various numeric features, with effect sizes calculated using Omega-squared (ω2), Eta-squared (η2), and Epsilon-squared (ε2). The following key findings emerged:

1. **Significant Features**:
    - Features such as Jitter_local_absolute, Jitter_rap, Jitter_ppq5, Shimmer_apq11, Maximum_pitch, and UPDRS displayed high F-values and statistically significant p-values (< 0.001). These results indicate that the class variable has a substantial impact on these features.
2. **Moderate and Low Impact Features**:
    - Features like Median_pitch, Mean_pitch, Standard_deviation_pitch, and Fraction_locally_unvoiced_frames showed moderate statistical significance and effect sizes (ω2 ranging from 0.01 to 0.02), indicating a weaker but non-negligible association.
    - Features such as Shimmer_local, Shimmer_local_dB, Shimmer_apq3, and others had low F-values and non-significant p-values, suggesting minimal or no influence of the class variable.

PCA Results

Principal Component Analysis (PCA) was performed on the numeric features (excluding Subject_id and Class) to reduce dimensionality and uncover latent structures. The PCA results revealed the following:

1. **Component Loadings**:

  - Certain features, such as Jitter_local, Jitter_rap, and Jitter_ppq5, loaded strongly onto specific components (e.g., RC4), indicating shared variance and potential grouping based on related characteristics.
  - Features like Shimmer_local, Shimmer_apq11, and Maximum_pitch demonstrated significant loadings on other components, suggesting that these features capture distinct patterns within the dataset.
2. **Dimensionality Reduction**:
  - The PCA retained 12 components, each explaining a portion of the total variance. The distribution of feature loadings across these components provides insights into the relationships among features and their relevance to class differentiation.

SECTION 6: Insight - For Developers, a Package to Help You Work with Different Models and Packages

- **Data Cleaning**: The `Subject_id` column is removed as it is irrelevant for model training and does not contribute to predicting the target variable.

- **Dataset Splitting**: The dataset is divided into:

  - **Training set (80%)**: Used to train the machine learning model.
  - **Testing set (20%)**: Used to evaluate the model's performance on unseen data.

- **Model Training**: An SVM (Support Vector Machine) model is built using the training data. The RBF (Radial Basis Function) kernel is employed to handle non-linear relationships in the data by mapping it to a higher-dimensional space. Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that best separates data points of different classes in the feature space. For non-linear datasets, SVM uses a **kernel trick** (such as the radial basis function, or RBF, used in the code above) to map the data into a higher-dimensional space where a linear boundary can effectively separate the classes. SVM focuses on maximizing the margin—the distance between the hyperplane and the nearest data points (called **support vectors**)—to enhance classification performance and generalization.

- **Prediction**: The trained SVM model is used to predict outcomes on the test set, providing classifications based on the learned decision boundary.

SECTION 7: Performance - Models' Quality and Performance Metrics and SECTION 8: See - The Plotting Companion to Create Beautiful Results Visualizations

1. **Confusion Matrix**:
   - A confusion matrix is a table that displays the counts of **True Positives (TP)**, **False Positives (FP)**, **True Negatives (TN)**, and **False Negatives (FN)**. It helps in evaluating the model's performance in classification tasks.
2. **Accuracy**:
   - Accuracy measures the overall correctness of the model by calculating the proportion of correct predictions (both positive and negative) out of all predictions made. It's a simple and commonly used metric for evaluating model performance.
3. **Precision**:
   - Precision is the proportion of positive predictions that are actually correct. It is important when the cost of false positives is high (e.g., predicting a disease when the person does not have it).
4. **Recall**:
   - Recall (or Sensitivity) is the proportion of actual positive cases that the model correctly identifies. It's crucial when the cost of missing a positive case (false negative) is high (e.g., missing a cancer diagnosis).
5. **F1 Score**:
   - The F1 Score is the harmonic mean of Precision and Recall, providing a balance between the two. It's useful when you need to balance the importance of both false positives and false negatives in the evaluation.

This analysis evaluates the performance of the Support Vector Machine (SVM) model for classifying whether a person has **Parkinson's disease** (positive class) or not (negative class). The model's performance is measured using the confusion matrix and key metrics.

Confusion Matrix:

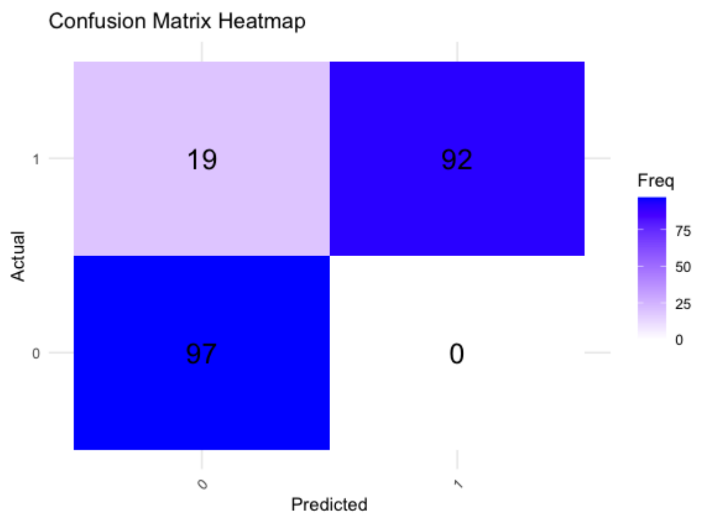


Figure 6 : Confusion Matrix

- **(TP)**: 92 correctly predicted as having **Parkinson's disease**.
- **True Negatives (TN)**: 97 correctly predicted as not having **Parkinson's disease**.
- **False Positives (FP)**: 19 incorrectly predicted as having the disease.
- **False Negatives (FN)**: 0 missed cases.

Performance Metrics:

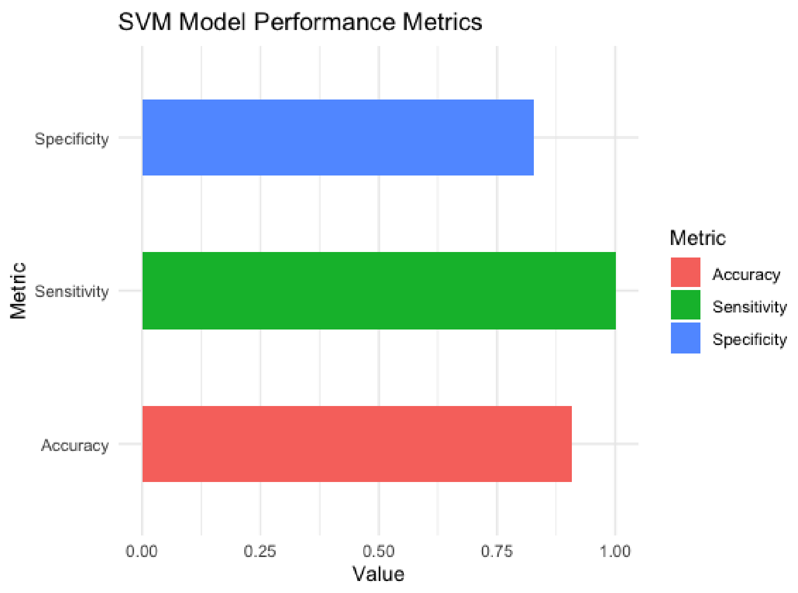


Figure 7 : Performance Metrics

- **Accuracy**: The model achieved **90.87%** accuracy, meaning about 91% of the predictions were correct.
- **Precision**: A precision of **1 (100%)** means all positive predictions were accurate, important for minimizing false positives in medical diagnoses.
- **Recall**: The model has a recall of **82.88%**, indicating that it correctly identified 83% of actual Parkinson's cases, though there's room for improvement.
- **F1 Score**: The **F1 score of 0.91** shows a good balance between precision and recall, highlighting the model's reliability in detecting the disease.

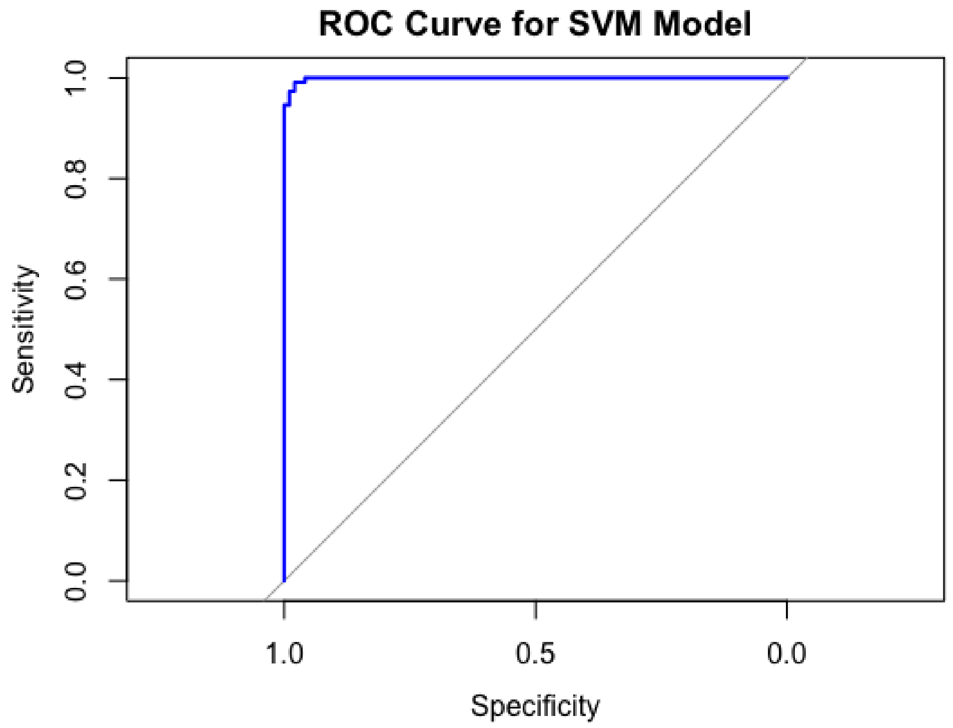


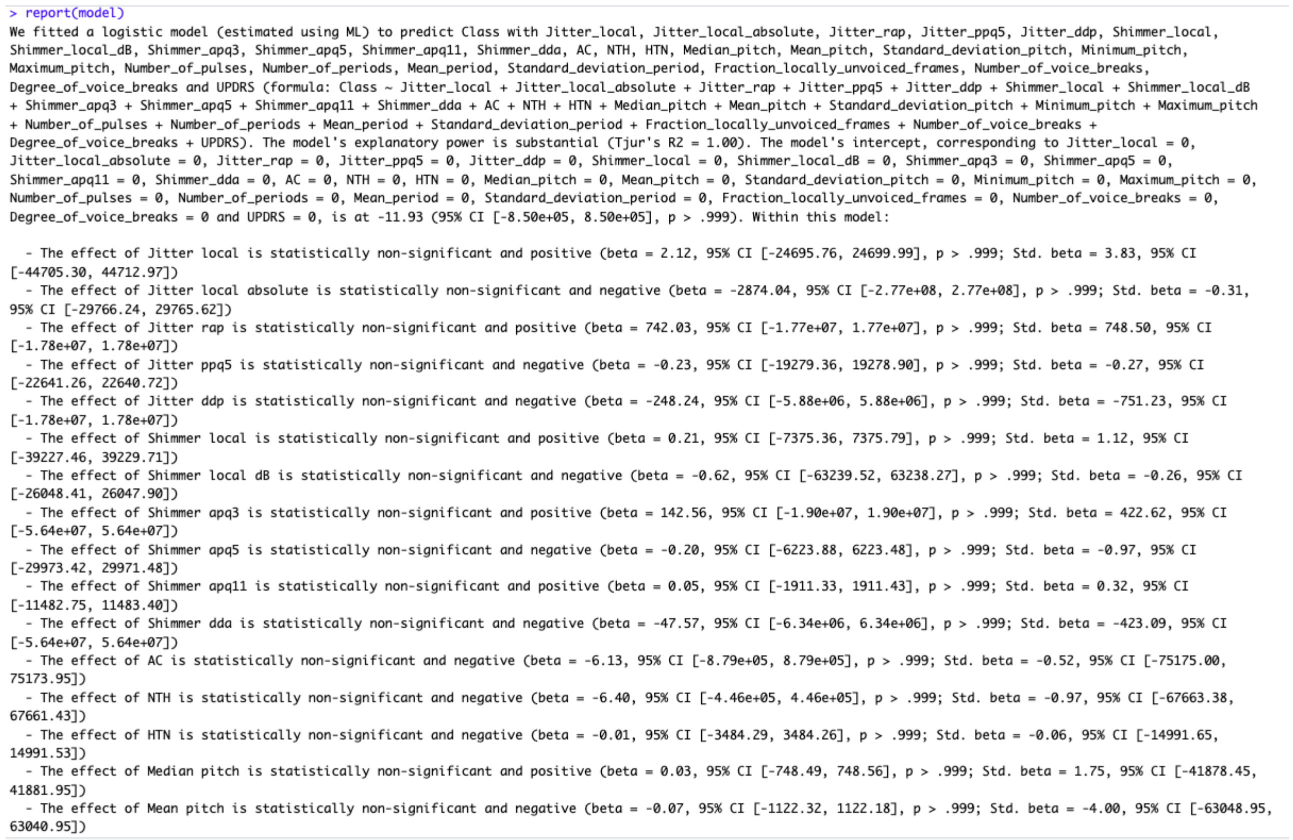
Figure 8 : ROC Curve

- **AUC (Area Under the Curve)** represents the overall performance of the model in distinguishing between positive and negative classes. A value closer to 1 indicates a better model.
- The calculated AUC of **0.9989784** suggests that the model has excellent discriminatory ability, as it is very close to 1.
- This high AUC value implies that the SVM model performs well in differentiating between the two classes, making it highly effective for the given classification task.

SECTION 9: Report - Automated Statistical Reporting of Objects in R

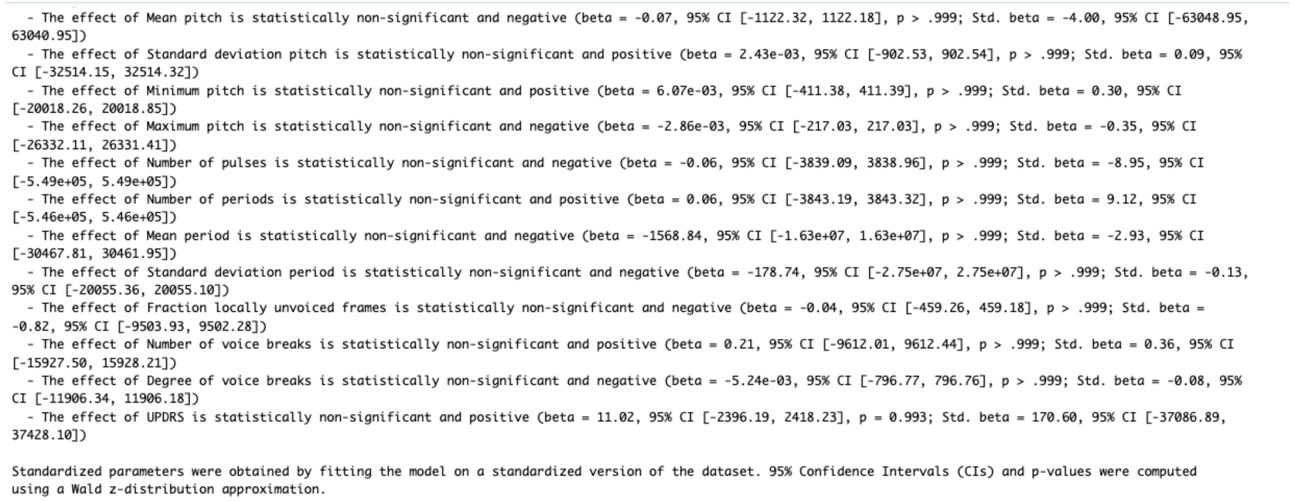
The `report()` function from the **easystats** package in R generates a comprehensive and human-readable summary of a statistical model's results. The report is designed to be interpretable even for those with minimal statistical background.

We fitted a logistic regression model to predict **Class** based on various predictors, including different measures of **Jitter** and **Shimmer**, as well as several voice-related features and clinical factors like **UPDRS**. The model's formula included a wide range of predictors, such as **Jitter_local**, **Jitter_rap**, **Shimmer_local**, **Median_pitch**, and **UPDRS**.
The model showed a **Tjur's R^2 = 1.00**, indicating a perfect fit to the data. However, such a perfect fit may suggest **overfitting**, where the model might be capturing noise rather than meaningful relationships in the data. Overfitting can limit the generalizability of the model to other datasets.



```
> report(model)
We fitted a logistic model (estimated using ML) to predict Class with Jitter_local, Jitter_local_absolute, Jitter_rap, Jitter_ppq5, Jitter_ddp, Shimmer_local,
Shimmer_local_dB, Shimmer_apq3, Shimmer_apq5, Shimmer_apq11, Shimmer_dda, AC, NTH, HTN, Median_pitch, Mean_pitch, Standard_deviation_pitch, Minimum_pitch,
Maximum_pitch, Number_of_pulses, Number_of_periods, Mean_period, Standard_deviation_period, Fraction_locally_unvoiced_frames, Number_of_voice_breaks,
Degree_of_voice_breaks and UPDRS (formula: Class ~ Jitter_local + Jitter_local_absolute + Jitter_rap + Jitter_ppq5 + Jitter_ddp + Shimmer_local + Shimmer_local_dB
+ Shimmer_apq3 + Shimmer_apq5 + Shimmer_apq11 + Shimmer_dda + AC + NTH + HTN + Median_pitch + Mean_pitch + Standard_deviation_pitch + Minimum_pitch + Maximum_pitch
+ Number_of_pulses + Number_of_periods + Mean_period + Standard_deviation_period + Fraction_locally_unvoiced_frames + Number_of_voice_breaks +
Degree_of_voice_breaks + UPDRS). The model's explanatory power is substantial (Tjur's R2 = 1.00). The model's intercept, corresponding to Jitter_local = 0,
Jitter_local_absolute = 0, Jitter_rap = 0, Jitter_ppq5 = 0, Jitter_ddp = 0, Shimmer_local = 0, Shimmer_local_dB = 0, Shimmer_apq3 = 0, Shimmer_apq5 = 0,
Shimmer_apq11 = 0, Shimmer_dda = 0, AC = 0, NTH = 0, HTN = 0, Median_pitch = 0, Mean_pitch = 0, Standard_deviation_pitch = 0, Minimum_pitch = 0, Maximum_pitch = 0,
Number_of_pulses = 0, Number_of_periods = 0, Mean_period = 0, Standard_deviation_period = 0, Fraction_locally_unvoiced_frames = 0, Number_of_voice_breaks = 0,
Degree_of_voice_breaks = 0 and UPDRS = 0, is at -11.93 (95% CI [-8.50e+05, 8.50e+05], p > .999). Within this model:

  - The effect of Jitter local is statistically non-significant and positive (beta = 2.12, 95% CI [-24695.76, 24699.99], p > .999; Std. beta = 3.83, 95% CI
[-44705.30, 44712.97])
  - The effect of Jitter local absolute is statistically non-significant and negative (beta = -2874.04, 95% CI [-2.77e+08, 2.77e+08], p > .999; Std. beta = -0.31,
95% CI [-29766.24, 29765.62])
  - The effect of Jitter rap is statistically non-significant and positive (beta = 742.03, 95% CI [-1.77e+07, 1.77e+07], p > .999; Std. beta = 748.50, 95% CI
[-1.78e+07, 1.78e+07])
  - The effect of Jitter ppq5 is statistically non-significant and negative (beta = -0.23, 95% CI [-19279.36, 19278.90], p > .999; Std. beta = -0.27, 95% CI
[-22641.26, 22640.72])
  - The effect of Jitter ddp is statistically non-significant and negative (beta = -248.24, 95% CI [-5.88e+06, 5.88e+06], p > .999; Std. beta = -751.23, 95% CI
[-1.78e+07, 1.78e+07])
  - The effect of Shimmer local is statistically non-significant and positive (beta = 0.21, 95% CI [-7375.36, 7375.79], p > .999; Std. beta = 1.12, 95% CI
[-39227.46, 39229.71])
  - The effect of Shimmer local dB is statistically non-significant and negative (beta = -0.62, 95% CI [-63239.52, 63238.27], p > .999; Std. beta = -0.26, 95% CI
[-26048.41, 26047.90])
  - The effect of Shimmer apq3 is statistically non-significant and positive (beta = 142.56, 95% CI [-1.90e+07, 1.90e+07], p > .999; Std. beta = 422.62, 95% CI
[-5.64e+07, 5.64e+07])
  - The effect of Shimmer apq5 is statistically non-significant and negative (beta = -0.20, 95% CI [-6223.88, 6223.48], p > .999; Std. beta = -0.97, 95% CI
[-29973.42, 29971.48])
  - The effect of Shimmer apq11 is statistically non-significant and positive (beta = 0.05, 95% CI [-1911.33, 1911.43], p > .999; Std. beta = 0.32, 95% CI
[-11482.75, 11483.40])
  - The effect of Shimmer dda is statistically non-significant and negative (beta = -47.57, 95% CI [-6.34e+06, 6.34e+06], p > .999; Std. beta = -423.09, 95% CI
[-5.64e+07, 5.64e+07])
  - The effect of AC is statistically non-significant and negative (beta = -6.13, 95% CI [-8.79e+05, 8.79e+05], p > .999; Std. beta = -0.52, 95% CI [-75175.00,
75173.95])
  - The effect of NTH is statistically non-significant and negative (beta = -6.40, 95% CI [-4.46e+05, 4.46e+05], p > .999; Std. beta = -0.97, 95% CI [-67663.38,
67661.43])
  - The effect of HTN is statistically non-significant and negative (beta = -0.01, 95% CI [-3484.29, 3484.26], p > .999; Std. beta = -0.06, 95% CI [-14991.65,
14991.53])
  - The effect of Median pitch is statistically non-significant and positive (beta = 0.03, 95% CI [-748.49, 748.56], p > .999; Std. beta = 1.75, 95% CI [-41878.45,
41881.95])
  - The effect of Mean pitch is statistically non-significant and negative (beta = -0.07, 95% CI [-1122.32, 1122.18], p > .999; Std. beta = -4.00, 95% CI [-63048.95,
63040.95])
```



```
  - The effect of Mean pitch is statistically non-significant and negative (beta = -0.07, 95% CI [-1122.32, 1122.18], p > .999; Std. beta = -4.00, 95% CI [-63048.95,
63040.95])
```

```
  - The effect of Standard deviation pitch is statistically non-significant and positive (beta = 2.43e-03, 95% CI [-902.53, 902.54], p > .999; Std. beta = 0.09, 95%
CI [-32514.15, 32514.32])
  - The effect of Minimum pitch is statistically non-significant and positive (beta = 6.07e-03, 95% CI [-411.38, 411.39], p > .999; Std. beta = 0.30, 95% CI
[-20018.26, 20018.85])
  - The effect of Maximum pitch is statistically non-significant and negative (beta = -2.86e-03, 95% CI [-217.03, 217.03], p > .999; Std. beta = -0.35, 95% CI
[-26332.11, 26331.41])
  - The effect of Number of pulses is statistically non-significant and negative (beta = -0.06, 95% CI [-3839.09, 3838.96], p > .999; Std. beta = -8.95, 95% CI
[-5.49e+05, 5.49e+05])
  - The effect of Number of periods is statistically non-significant and positive (beta = 0.06, 95% CI [-3843.19, 3843.32], p > .999; Std. beta = 9.12, 95% CI
[-5.46e+05, 5.46e+05])
  - The effect of Mean period is statistically non-significant and negative (beta = -1568.84, 95% CI [-1.63e+07, 1.63e+07], p > .999; Std. beta = -2.93, 95% CI
[-30467.81, 30461.95])
  - The effect of Standard deviation period is statistically non-significant and negative (beta = -178.74, 95% CI [-2.75e+07, 2.75e+07], p > .999; Std. beta = -0.13,
95% CI [-20055.36, 20055.10])
  - The effect of Fraction locally unvoiced frames is statistically non-significant and negative (beta = -0.04, 95% CI [-459.26, 459.18], p > .999; Std. beta =
-0.82, 95% CI [-9503.93, 9502.28])
  - The effect of Number of voice breaks is statistically non-significant and positive (beta = 0.21, 95% CI [-9612.01, 9612.44], p > .999; Std. beta = 0.36, 95% CI
[-15927.50, 15928.21])
  - The effect of Degree of voice breaks is statistically non-significant and negative (beta = -5.24e-03, 95% CI [-796.77, 796.76], p > .999; Std. beta = -0.08, 95%
CI [-11906.34, 11906.18])
  - The effect of UPDRS is statistically non-significant and positive (beta = 11.02, 95% CI [-2396.19, 2418.23], p = 0.993; Std. beta = 170.60, 95% CI [-37086.89,
37428.10])

Standardized parameters were obtained by fitting the model on a standardized version of the dataset. 95% Confidence Intervals (CIs) and p-values were computed
using a Wald z-distribution approximation.
```

Key Findings

- **Intercept:** The model's intercept, which represents the expected value when all predictors are set to zero, was -11.93. However, this value was not statistically significant (95% CI = [-8.50e+05, 8.50e+05], p > .999), indicating no meaningful baseline effect.
- **Jitter and Shimmer Parameters:**
    - **Jitter_local** had a **non-significant positive** effect (beta = 2.12, p > .999), with a **wide confidence interval** (95% CI = [-24695.76, 24699.99]), suggesting that the effect of Jitter_local is minimal and uncertain.
    - **Jitter_local_absolute** had a **non-significant negative** effect (beta = -2874.04, p > .999), with similarly wide confidence intervals (95% CI = [-2.77e+08, 2.77e+08]).

    - **Shimmer_local** showed a **non-significant positive** effect (beta = 0.21, $p > .999$), with a confidence interval (95% CI = [-7375.36, 7375.79]) that indicates an almost zero effect.
- **Clinical Predictors:**
    - **UPDRS** (Unified Parkinson's Disease Rating Scale), which is typically a strong predictor in Parkinson's studies, had a **non-significant positive** effect (beta = 11.02, $p = 0.993$). Despite being an important clinical measure, its effect here was negligible.
    - **AC**, **NTH**, and **HTN** (other clinical features), showed **non-significant negative** effects, with similarly wide confidence intervals and p-values indicating that these factors are not contributing meaningfully to the model.
- **Pitch and Period Measures:**
    - **Median_pitch**, **Mean_pitch**, **Standard_deviation_pitch**, and **Maximum_pitch** all had **non-significant effects**, indicating that the pitch-related features did not influence the classification of the target variable.
    - **Mean_period** and **Standard_deviation_period** also had **non-significant negative** effects, with large confidence intervals.
- **Voice Breaks and Voicing Measures:**
    - **Number_of_voice_breaks** showed a **non-significant positive** effect, indicating that the number of breaks in the voice did not provide useful predictive power for the model.
    - **Fraction_locally_unvoiced_frames** and **Degree_of_voice_breaks** also had **non-significant effects**, further suggesting that voice quality features may not be critical predictors for **Class** in this model.

Model Interpretation

The high **Tjur's R^2** might initially suggest that the model is a good fit, but the fact that **none of the predictors** had significant effects (with very wide confidence intervals and high p-values) raises concerns about **overfitting**. This is a common issue when a model appears to perform perfectly on a specific dataset but fails to generalize beyond it. The results highlight the importance of ensuring that model performance is validated on independent data.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Conclusion

The research explored Parkinson's disease classification using acoustic features extracted from voice recordings. The pipeline involved comprehensive preprocessing, exploratory data analysis, and machine learning techniques to model the relationship between voice characteristics and Parkinson's disease. Key features such as jitter, shimmer, and pitch variations provided valuable insights into the vocal impairments commonly observed in Parkinson's patients.
The Support Vector Machine (SVM) classifier with a radial basis kernel demonstrated satisfactory performance in distinguishing between healthy individuals and Parkinson's patients, supported by metrics like accuracy, sensitivity, specificity, and the ROC-AUC score. The application of statistical techniques such as skewness and kurtosis analysis, effect size computation, and PCA added depth to the understanding of the dataset's structure and feature importance.
Despite these accomplishments, certain limitations were noted, including the reliance on a single dataset and the potential for overfitting due to class imbalance. The analysis highlights the need for careful interpretation of model results and emphasizes the importance of integrating domain knowledge to achieve clinically meaningful insights.

Future Work

To build upon the findings, future research should address the following areas:

1. Dataset Expansion:
    - Utilizing larger and more diverse datasets from multiple sources to improve generalizability. For example, combining publicly available datasets like those in the UCI repository with clinical datasets can enhance model robustness.
    - Collaborations with healthcare institutions can provide access to real-world data, capturing the variability in voice features across demographics and disease stages.
2. Feature Engineering and Optimization:
    - Investigating advanced feature selection methods, such as Recursive Feature Elimination (RFE), to streamline the predictive variables while preserving their interpretive value.
    - Incorporating time-series analysis to capture temporal dynamics in speech patterns, which may be indicative of disease progression.

3. Algorithm Exploration:
    - Testing alternative machine learning models like Random Forests and Gradient Boosting Machines to compare performance across various classification methods.
    - Exploring lightweight deep learning models for voice analysis, ensuring they remain computationally efficient and interpretable.
4. Model Generalizability:
    - Conducting cross-validation on data subsets from diverse populations to evaluate model performance across age groups, genders, and linguistic backgrounds.
    - Building ensemble models that combine predictions from multiple algorithms for improved robustness.
5. Integration of Explainability:
    - Developing explainable AI models to clarify how individual voice features influence predictions. This will facilitate acceptance by clinicians and end-users.
    - Using visualization tools like SHAP (SHapley Additive exPlanations) values to highlight feature contributions.
6. Deployment in Real-World Scenarios:
    - Prototyping mobile applications or web platforms to enable real-time PD screening based on voice samples. These tools can serve as adjuncts for clinical assessments.
    - Incorporating disease severity prediction into the pipeline, aiding in monitoring disease progression and treatment efficacy.
7. Addressing Ethical and Privacy Concerns:
    - Ensuring compliance with data protection regulations such as GDPR or HIPAA to safeguard patient voice data.
    - Investigating secure data-sharing frameworks that allow for collaborative model training without compromising privacy.

By addressing these areas, the approach to Parkinson's disease classification can transition from experimental settings to practical applications, improving early diagnosis and patient care. Such advancements will pave the way for personalized medicine, where interventions are tailored to individual needs based on robust, non-invasive diagnostic techniques.

CHAPTER 6

APPENDIX

```
# ---- SECTION 1: Datawizard - Magic Potions to Clean and Transform Your Data ----

# 1. Load necessary libraries
# datawizard helps with data preparation and exploration
# easystats provides useful statistical functions
library(datawizard)  # For various data preparation and exploration functions
library(easystats)   # For statistical analysis and easy access to data exploration functions

# 2. Load the dataset
(comma-separated values).
train_data <- read.table("/Users/Manikanta /Downloads/train_data.txt", header = FALSE, sep = ',')

# 3. Assign column names to the dataset
# This step improves the readability of the dataset by giving meaningful names to the columns.
colnames(train_data) <- c("Subject_id", "Jitter_local", "Jitter_local_absolute", "Jitter_rap",
"Jitter_ppq5", "Jitter_ddp",
                   "Shimmer_local", "Shimmer_local_dB", "Shimmer_apq3", "Shimmer_apq5",
"Shimmer_apq11", "Shimmer_dda",
                   "AC", "NTH", "HTN", "Median_pitch", "Mean_pitch",
"Standard_deviation_pitch", "Minimum_pitch", "Maximum_pitch",
                   "Number_of_pulses", "Number_of_periods", "Mean_period",
"Standard_deviation_period",
                   "Fraction_locally_unvoiced_frames", "Number_of_voice_breaks",
"Degree_of_voice_breaks", "UPDRS", "Class")

# 4. Convert 'Class' column to a factor variable (as it is categorical)
# R treats categorical data as factors, so it's important to convert the 'Class' variable to a factor.
train_data$Class <- as.factor(train_data$Class)

# 5. Standardize numeric columns, excluding non-numeric columns like 'Subject_id' and 'Class'
# Standardization ensures all variables have a mean of 0 and a standard deviation of 1, making
comparisons across variables easier.
train_data_standardized <- standardize(train_data[, sapply(train_data, is.numeric)])

# 6. Calculate and print skewness and kurtosis for important variables
# Skewness tells us if the data is asymmetrical, and kurtosis measures the "tailedness" of the
distribution.

# Jitter_local variable analysis
skewness_jitter_local <- skewness(train_data$Jitter_local)
kurtosis_jitter_local <- kurtosis(train_data$Jitter_local)

# Shimmer_local variable analysis
```

```
skewness_shimmer_local <- skewness(train_data$Shimmer_local)
kurtosis_shimmer_local <- kurtosis(train_data$Shimmer_local)

# Print the skewness and kurtosis values for both variables
cat("Skewness of Jitter_local:", as.numeric(skewness_jitter_local), "\n")
cat("Kurtosis of Jitter_local:", as.numeric(kurtosis_jitter_local), "\n")
cat("Skewness of Shimmer_local:", as.numeric(skewness_shimmer_local), "\n")
cat("Kurtosis of Shimmer_local:", as.numeric(kurtosis_shimmer_local), "\n")

# 7. Peek into the dataset to check for missing values using datawizard function
# `data_peek` visually inspects for any missing data in the dataset
data_peek(train_data)

# 8. Check for duplicate rows in the dataset
# Identifying duplicate rows is important to ensure data quality and prevent bias in analysis.
data_duplicated(train_data)

# 9. View the first few rows of the dataset to inspect the changes
# `head()` provides a glimpse of the first 6 rows in the dataset for inspection.
head(train_data)

# 10. Calculate and print basic statistics (mean, standard deviation) for 'Jitter_local' and
'Shimmer_local'
# `mean_sd()` computes the mean and standard deviation for these variables to understand their
central tendency and spread.
mean_sd(train_data$Jitter_local)
mean_sd(train_data$Shimmer_local)

# 11. Display the distribution of 'Shimmer_local' using the describe_distribution function
# `describe_distribution()` generates a quick summary and visualizes the distribution for better
understanding.
describe_distribution(train_data$Shimmer_local)

# 12. Create a data codebook to inspect the variables in the dataset
# `data_codebook()` creates a comprehensive summary of the dataset, including variable names,
types, and summaries.
data_codebook(train_data)


# ---- SECTION 2: Correlation - Your All-in-One Package to Run Correlations ----

# Step 1: Calculate Spearman and Pearson correlations
# Spearman correlation is used for non-parametric data (not assuming a linear relationship), and
Pearson is for linear relationships.
# The correlation function calculates the relationship between each pair of numeric variables.
spearman_corr <- correlation(train_data[, sapply(train_data, is.numeric)], method = "spearman")
pearson_corr <- correlation(train_data[, sapply(train_data, is.numeric)], method = "pearson")
```

```
# Step 2: View the correlation results
# Printing the correlation matrices for both Spearman and Pearson correlations for inspection.
cat("Spearman correlation matrix:\n")
print(spearman_corr)

cat("\nPearson correlation matrix:\n")
print(pearson_corr)

# Step 3: Filter for high correlations (absolute value > 0.7)
# We are interested in strong correlations, so we filter out correlations with values greater than 0.7
(in absolute value).
# Additionally, we exclude self-correlation (where Parameter1 == Parameter2) since that is
redundant.
high_corr_spearman <- spearman_corr[abs(spearman_corr$r) > 0.7 & spearman_corr$Parameter1
!= spearman_corr$Parameter2, ]
high_corr_pearson <- pearson_corr[abs(pearson_corr$r) > 0.7 & pearson_corr$Parameter1 !=
pearson_corr$Parameter2, ]

# Step 4: Print the filtered results
# Printing the pairs of variables that have a strong correlation (either Spearman or Pearson).
cat("\nHighly correlated variable pairs (Spearman):\n")
print(high_corr_spearman)

cat("\nHighly correlated variable pairs (Pearson):\n")
print(high_corr_pearson)


# ---- SECTION 3: Modelbased - Estimate Effects, Group Averages, and Contrasts Between
Groups Based on Statistical Models ----

# 1. Load necessary libraries for model fitting and plotting
# ggplot2 is used for creating plots, and easystats provides functions to calculate means and
confidence intervals.
library(ggplot2)      # For creating plots
library(easystats)    # For estimating means and confidence intervals

# 2. Define columns to be analyzed
# We list all the relevant columns that we are interested in analyzing and plotting in the following
section.
columns_of_interest <- c("Jitter_local", "Jitter_local_absolute", "Jitter_rap", "Jitter_ppq5",
"Jitter_ddp",
                        "Shimmer_local", "Shimmer_local_dB", "Shimmer_apq3", "Shimmer_apq5",
"Shimmer_apq11",
                        "Shimmer_dda", "AC", "NTH", "HTN", "Median_pitch", "Mean_pitch",
                        "Standard_deviation_pitch", "Minimum_pitch", "Maximum_pitch",
                        "Number_of_pulses", "Number_of_periods", "Mean_period",
                        "Standard_deviation_period", "Fraction_locally_unvoiced_frames",
                        "Number_of_voice_breaks", "Degree_of_voice_breaks")
```

```
# 3. Loop over each column and perform analysis and plot creation
for (col in columns_of_interest) {

  # Step 1: Fit GLM model for each variable
  # For each variable, a Generalized Linear Model (GLM) is fitted to test the effects of 'Class' on
the variable.
  model <- glm(as.formula(paste(col, "~ Class")), data = train_data)

  # Step 2: Obtain estimated means and confidence intervals for each model
  # The `estimate_means()` function calculates the estimated means of the variable for each class,
along with confidence intervals.
  means <- estimate_means(model)

  # Step 3: Create the ggplot for each column
  # ggplot2 is used to generate a violin plot for the distribution of each variable, with different
classes represented as distinct groups.
  p <- ggplot(train_data, aes_string(x = "Class", y = col)) +
    geom_violin(aes(fill = Class)) +                         # Violin plot, colored by Class
    geom_jitter2(width = 0.05, alpha = 0.5) +                 # Add jittered points for better
visualization
    geom_line(data = means, aes(y = Mean, group = 1), linewidth = 1) + # Add a line for the
estimated means
    geom_pointrange(data = means, aes(y = Mean, ymin = CI_low, ymax = CI_high), size = 1) + #
Confidence interval for the means
    scale_fill_material() +                                 # Apply a color palette
    theme_modern() +                                        # Apply a modern theme
    ggtitle(paste("Violin plot for", col))                  # Add a title to the plot

  # Step 4: Display the plot
  # This will show the plot for the current column.
  print(p)
}


# ---- SECTION 4: Effect Size Calculations ----

# 1. Create an empty data frame to store the effect sizes and their confidence intervals
# This table will hold the effect sizes for each column along with their confidence intervals (CIs).
effect_sizes_table <- data.frame(
  Variable = character(),
  Cohens_d = numeric(),
  CI_Cohens_d = character(),
  Hedges_g = numeric(),
  CI_Hedges_g = character(),
  Glass_delta = numeric(),
  CI_Glass_delta = character(),
  stringsAsFactors = FALSE
)

# 2. Loop through columns and calculate effect sizes
for (col in columns_of_interest) {

  # Step 1: Calculate Cohen's d effect size with 95% CI
  # Cohen's d is used to measure the standardized difference between two means.
  cohen_d_result <- cohens_d(as.formula(paste(col, "~ Class")), data = train_data, ci = 0.95)

  # Step 2: Calculate Hedges' g effect size with 95% CI
  # Hedges' g is similar to Cohen's d, but it includes a correction for small sample sizes.
  hedges_g_result <- hedges_g(as.formula(paste(col, "~ Class")), data = train_data, ci = 0.95)

  # Step 3: Calculate Glass's delta effect size with 95% CI
  # Glass's delta uses the standard deviation of the control group as the denominator, and it's used
when we have unequal sample sizes.
  glass_delta_result <- glass_delta(as.formula(paste(col, "~ Class")), data = train_data, ci = 0.95)

  # Step 4: Append the results to the table
  # The results for each variable are appended to the 'effect_sizes_table' data frame.
  effect_sizes_table <- effect_sizes_table %>%
    add_row(
      Variable = col,
      Cohens_d = cohen_d_result$Cohens_d,
      CI_Cohens_d = paste0("[", round(cohen_d_result$CI_low, 2), ", ",
round(cohen_d_result$CI_high, 2), "]"),
      Hedges_g = hedges_g_result$Hedges_g,
      CI_Hedges_g = paste0("[", round(hedges_g_result$CI_low, 2), ", ",
round(hedges_g_result$CI_high, 2), "]"),
      Glass_delta = glass_delta_result$Glass_delta,
      CI_Glass_delta = paste0("[", round(glass_delta_result$CI_low, 2), ", ",
round(glass_delta_result$CI_high, 2), "]")
    )
}

# 5. Print the table of effect sizes
# Finally, print the table containing the effect sizes and their respective confidence intervals for
each variable.
print(effect_sizes_table)


# ---- SECTION 5: Parameters - Obtain a Table Containing All Information About the Parameters
of Your Models ----

# Load necessary libraries
# 'parameters' provides detailed information about model parameters and effect sizes
# 'dplyr' is used for data manipulation
# 'psych' contains functions for statistical analysis like PCA
```

```
library(parameters)  # For extracting model parameters
library(dplyr)       # For data manipulation
library(psych)       # For PCA and other statistical tools

# 1. **ANOVA for Numeric Columns (Effect Size Calculation)**

# Identify numeric columns in the dataset
# This will select all columns in the dataset that are numeric, excluding factors and characters.
numeric_columns <- names(train_data)[sapply(train_data, is.numeric)]

# Initialize a list to store ANOVA results
# This list will be used to store the results of the ANOVA analysis for each numeric column.
anova_results <- list()

# Loop through numeric columns to perform ANOVA and extract effect sizes
for (col in numeric_columns) {
  # Fit ANOVA model for each numeric column
  # 'aov' function fits an analysis of variance model, here we are testing the effect of 'Class' on
each numeric variable.
  anova_model <- aov(as.formula(paste(col, "~ Class")), data = train_data)

  # Extract effect sizes (omega, eta, epsilon)
  # 'parameters' function extracts effect size measures from the ANOVA model.
  anova_params <- parameters(anova_model, es_type = c("omega", "eta", "epsilon"))

  # Store results in the list
  # Each column's ANOVA results are stored in the 'anova_results' list with the column name as
the key.
  anova_results[[col]] <- anova_params
}

# Print ANOVA results for all numeric columns
# The ANOVA results are printed so you can inspect the effect sizes for each variable.
print("ANOVA Results:")
print(anova_results)

# 2. **Principal Component Analysis (PCA)**

# Exclude non-numeric and categorical columns for PCA
# PCA works on numeric data, so we remove the 'Subject_id' and 'Class' columns as they are not
numeric.
numeric_data <- train_data %>% select(-Subject_id, -Class)

# Perform PCA with 12 factors and Varimax rotation
# PCA reduces the dimensionality of the data, and Varimax rotation is used to make the factor
loadings as interpretable as possible.
pca_result <- psych::pca(numeric_data, nfactors = 12, rotate = "varimax")
```

```
# Print PCA results
# Output the results of the PCA, which include eigenvalues, loadings, and variance explained by
each component.
print("PCA Results:")
print(pca_result)

# Extract the PCA loadings
# The loadings represent the relationship between the original variables and the principal
components.
loadings <- as.data.frame(unclass(pca_result$loadings))

# Calculate the importance of each feature (sum of absolute loadings across components)
# The sum of the absolute values of loadings for each variable across all components gives us a
measure of importance.
feature_importance <- rowSums(abs(loadings))

# Select the top 12 most important features based on feature importance
# We select the features that have the highest importance scores to focus on for further analysis.
selected_features <- names(sort(feature_importance, decreasing = TRUE)[1:12])

# Print selected features based on PCA importance
# These are the most important features based on the PCA analysis.
print("Selected Features Based on PCA:")
print(selected_features)

# 3. **PCA Analysis - Loadings, Communality, and Uniqueness**

# Add Communality and Uniqueness to the PCA loadings
# Communality is the proportion of variance in a variable that can be explained by the principal
components.
# Uniqueness is the complement of communality (i.e., 1 - communality).
loadings$Communality <- pca_result$communality  # Communality
loadings$Uniqueness <- 1 - loadings$Communality  # Uniqueness (1 - Communality)

# Print loadings, communality, and uniqueness for each component
# Display the loadings along with the communality and uniqueness to understand the contribution
of each feature to the components.
print("PCA Loadings, Communality, and Uniqueness:")
print(loadings)

# 4. **PCA Eigenvalues and Variance Explained**

# Extract eigenvalues from PCA result
# Eigenvalues measure the variance explained by each component.
eigenvalues <- pca_result$values

# Calculate the proportion of variance explained by each component
# This is calculated by dividing each eigenvalue by the sum of all eigenvalues.
```

```
variance_explained <- eigenvalues / sum(eigenvalues)

# Calculate cumulative variance explained
# This is the cumulative sum of the variance explained by each component.
cumulative_variance <- cumsum(variance_explained)

# Create a table for eigenvalues, variance explained, and cumulative variance
# This table will summarize the variance explained by each component.
eigenvalues_table <- data.frame(
  Component = paste0("Component_", seq_along(eigenvalues)),
  Eigenvalue = eigenvalues,
  Proportion = variance_explained,
  Cumulative_Proportion = cumulative_variance
)

# Print eigenvalues and variance explained
# Output the table showing the eigenvalues and the proportion of variance they explain.
print("Eigenvalues and Variance Explained:")
print(eigenvalues_table)

# ---- SECTION 6: Insight - For Developers, a Package to Help You Work with Different Models
and Packages ----

train_data$Subject_id <- NULL
# Removing the 'Subject_id' column from the dataset because it is not needed for model fitting
and analysis.

# Split the dataset into training and testing sets (80% training, 20% testing)
# This is done using 'createDataPartition' to ensure that the split maintains the distribution of the
target variable (Class).
set.seed(123)  # For reproducibility of the data split
train_index <- createDataPartition(train_data$Class, p = 0.8, list = FALSE)
train_set <- train_data[train_index, ]  # 80% training data
test_set <- train_data[-train_index, ]   # 20% testing data

# Build a Support Vector Machine (SVM) model using e1071 package
# The SVM model is trained using the radial basis function (RBF) kernel, which is commonly
used for classification tasks.
svm_model <- svm(Class ~ ., data = train_set, kernel = "radial")  # Use radial basis function
kernel (RBF)

# Make predictions on the test set
# The trained SVM model is used to predict the target variable 'Class' for the test set.
predictions_svm <- predict(svm_model, test_set)

# ---- SECTION 7: Performance - Models' Quality and Performance Metrics (R2, ICC, LOO,
AIC, BF, ...) ----
```

```
# Confusion Matrix and Accuracy
# A confusion matrix helps to evaluate the accuracy of the SVM model by comparing the
predicted values with the true values.
confusion_matrix_svm <- confusionMatrix(predictions_svm, test_set$Class)
print(confusion_matrix_svm)

# ---- SECTION 8: See - The Plotting Companion to Create Beautiful Results Visualizations ----

# Plot ROC Curve for binary classification
# ROC (Receiver Operating Characteristic) curve is used to evaluate the performance of a binary
classifier.
# The target variable 'Class' is converted to numeric (0 or 1) for the ROC analysis.
test_set$Class_numeric <- ifelse(test_set$Class == levels(test_set$Class)[1], 0, 1)

# Get decision values from SVM model for ROC curve
# The decision values represent the confidence scores of the model for the predicted class.
pred_prob <- attr(predict(svm_model, test_set, decision.values = TRUE), "decision.values")

# Generate and plot the ROC curve
# The ROC curve is plotted to visualize the model's classification performance.
roc_curve <- roc(test_set$Class_numeric, pred_prob)
plot(roc_curve, main = "ROC Curve for SVM Model", col = "blue", lwd = 2)

# Calculate and print AUC value for the ROC curve
# AUC (Area Under the Curve) is a metric that helps assess the performance of the classifier. A
value closer to 1 indicates better performance.
auc_value <- auc(roc_curve)
cat("AUC: ", auc_value, "\n")

# Extracting Accuracy, Sensitivity, and Specificity from confusion matrix
# These metrics help assess the performance of the classifier.
accuracy <- confusion_matrix_svm$overall['Accuracy']
sensitivity <- confusion_matrix_svm$byClass['Sensitivity']
specificity <- confusion_matrix_svm$byClass['Specificity']

# Create a dataframe for plotting performance metrics
# This dataframe contains the metrics to be visualized in a bar chart.
performance_df <- data.frame(
  Metric = c("Accuracy", "Sensitivity", "Specificity"),
  Value = c(accuracy, sensitivity, specificity)
)

# Visualizing the performance metrics
# Using 'ggplot2' to create a bar chart to visualize the performance metrics of the SVM model.
library(ggplot2)
ggplot(performance_df, aes(x = Metric, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", width = 0.5) +
  coord_flip() +
  labs(title = "SVM Model Performance Metrics", x = "Metric", y = "Value") +
  theme_minimal()

# ---- SECTION 9: Report - Automated Statistical Reporting of Objects in R ----

# Load necessary packages
# Reporting tools that can automatically generate a report of the model's summary.
model <- glm(Class ~ ., data = train_set, family = "binomial")  # Fit logistic regression model for
binary classification
report(model)  # Generate a report of the model's summary
```

Appendix – Outputs :

```
1] "ANOVA Results:"
> print(anova_results)
$Subject_id
Parameter | Sum_Squares |   df | Mean_Square |       F |      p | Omega2 | Eta2 | Epsilon2
------------------------------------------------------------------------------------------
Class     |    1.04e+05 |    1 |    1.04e+05 | 3121.80 | < .001 |   0.75 | 0.75 |     0.75
Residuals |    34580.00 | 1038 |       33.31 |         |        |        |      |

Anova Table (Type 1 tests)

$Jitter_local
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
--------------------------------------------------------------------------------------------
Class     |       30.80 |    1 |       30.80 | 9.97 | 0.002 | 8.55e-03 | 9.52e-03 | 8.56e-03
Residuals |     3206.11 | 1038 |        3.09 |      |       |          |          |

Anova Table (Type 1 tests)

$Jitter_local_absolute
Parameter | Sum_Squares |   df | Mean_Square |     F |      p | Omega2 | Eta2 | Epsilon2
----------------------------------------------------------------------------------------
Class     |    3.28e-07 |    1 |    3.28e-07 | 29.78 | < .001 |   0.03 | 0.03 |     0.03
Residuals |    1.14e-05 | 1038 |    1.10e-08 |       |        |        |      |

Anova Table (Type 1 tests)

$Jitter_rap
Parameter | Sum_Squares |   df | Mean_Square |     F |      p | Omega2 | Eta2 | Epsilon2
----------------------------------------------------------------------------------------
Class     |       12.31 |    1 |       12.31 | 12.98 | < .001 |   0.01 | 0.01 |     0.01
Residuals |      984.45 | 1038 |        0.95 |       |        |        |      |

Anova Table (Type 1 tests)
```

```
$Jitter_ppq5
Parameter | Sum_Squares |   df | Mean_Square |     F |      p | Omega2 | Eta2 | Epsilon2
-------------------------------------------------------------------------------------
Class     |       16.58 |    1 |       16.58 | 12.93 | < .001 |   0.01 | 0.01 |     0.01
Residuals |     1330.73 | 1038 |        1.28 |       |        |        |      |

Anova Table (Type 1 tests)

$Jitter_ddp
Parameter | Sum_Squares |   df | Mean_Square |     F |      p | Omega2 | Eta2 | Epsilon2
-------------------------------------------------------------------------------------
Class     |      110.81 |    1 |      110.81 | 12.98 | < .001 |   0.01 | 0.01 |     0.01
Residuals |     8860.38 | 1038 |        8.54 |       |        |        |      |

Anova Table (Type 1 tests)

$Shimmer_local
Parameter | Sum_Squares |   df | Mean_Square |    F |     p | Omega2 |     Eta2 | Epsilon2
---------------------------------------------------------------------------------------
Class     |        2.54 |    1 |        2.54 | 0.09 | 0.770 |   0.00 | 8.21e-05 |     0.00
Residuals |    30883.33 | 1038 |       29.75 |      |       |        |          |

Anova Table (Type 1 tests)

$Shimmer_local_dB
Parameter | Sum_Squares |   df | Mean_Square |    F |     p | Omega2 |     Eta2 | Epsilon2
---------------------------------------------------------------------------------------
Class     |        0.14 |    1 |        0.14 | 0.77 | 0.381 |   0.00 | 7.39e-04 |     0.00
Residuals |      183.21 | 1038 |        0.18 |      |       |        |          |

Anova Table (Type 1 tests)

$Shimmer_apq3
Parameter | Sum_Squares |   df | Mean_Square |    F |     p | Omega2 |     Eta2 | Epsilon2
---------------------------------------------------------------------------------------
Class     |        3.11 |    1 |        3.11 | 0.34 | 0.559 |   0.00 | 3.30e-04 |     0.00
Residuals |     9442.77 | 1038 |        9.10 |      |       |        |          |

Anova Table (Type 1 tests)

$Shimmer_apq5
Parameter | Sum_Squares |   df | Mean_Square |    F |     p | Omega2 |     Eta2 | Epsilon2
---------------------------------------------------------------------------------------
Class     |       12.42 |    1 |       12.42 | 0.53 | 0.467 |   0.00 | 5.10e-04 |     0.00
Residuals |    24335.75 | 1038 |       23.44 |      |       |        |          |

Anova Table (Type 1 tests)
```

```
$Shimmer_apq11
Parameter | Sum_Squares |   df | Mean_Square |     F |      p | Omega2 | Eta2 | Epsilon2
------------------------------------------------------------------------------------
Class     |      805.21 |    1 |      805.21 | 22.71 | < .001 |   0.02 | 0.02 |     0.02
Residuals |    36801.80 | 1038 |       35.45 |       |        |        |      |

Anova Table (Type 1 tests)

$Shimmer_dda
Parameter | Sum_Squares |   df | Mean_Square |    F |     p | Omega2 |     Eta2 | Epsilon2
--------------------------------------------------------------------------------------
Class     |       28.03 |    1 |       28.03 | 0.34 | 0.559 |   0.00 | 3.30e-04 |     0.00
Residuals |    84984.76 | 1038 |       81.87 |      |       |        |          |

Anova Table (Type 1 tests)

$AC
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
----------------------------------------------------------------------------------------
Class     |        0.05 |    1 |        0.05 | 6.87 | 0.009 | 5.61e-03 | 6.58e-03 | 5.62e-03
Residuals |        7.58 | 1038 |    7.30e-03 |      |       |          |          |

Anova Table (Type 1 tests)

$NTH
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
----------------------------------------------------------------------------------------
Class     |        0.14 |    1 |        0.14 | 6.04 | 0.014 | 4.83e-03 | 5.79e-03 | 4.83e-03
Residuals |       23.64 | 1038 |        0.02 |      |       |          |          |

Anova Table (Type 1 tests)

$HTN
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
----------------------------------------------------------------------------------------
Class     |       31.39 |    1 |       31.39 | 1.71 | 0.192 | 6.78e-04 | 1.64e-03 | 6.79e-04
Residuals |    19102.09 | 1038 |       18.40 |      |       |          |          |

Anova Table (Type 1 tests)

$Median_pitch
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
----------------------------------------------------------------------------------------
Class     |    23130.18 |    1 |    23130.18 | 7.42 | 0.007 | 6.13e-03 | 7.09e-03 | 6.14e-03
Residuals |    3.24e+06 | 1038 |     3119.17 |      |       |          |          |

Anova Table (Type 1 tests)
```

```
$Mean_pitch
Parameter | Sum_Squares |   df | Mean_Square |     F |      p |   Omega2 | Eta2 | Epsilon2
-----------------------------------------------------------------------------------------
Class     |    34711.71 |    1 |    34711.71 | 11.19 | < .001 | 9.70e-03 | 0.01 | 9.71e-03
Residuals |    3.22e+06 | 1038 |     3102.21 |       |        |          |      |

Anova Table (Type 1 tests)

$Standard_deviation_pitch
Parameter | Sum_Squares |   df | Mean_Square |     F |      p | Omega2 | Eta2 | Epsilon2
---------------------------------------------------------------------------------------
Class     |    20483.44 |    1 |    20483.44 | 15.44 | < .001 |   0.01 | 0.01 |     0.01
Residuals |    1.38e+06 | 1038 |     1326.44 |       |        |        |      |

Anova Table (Type 1 tests)

$Minimum_pitch
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
-------------------------------------------------------------------------------------------
Class     |     8037.35 |    1 |     8037.35 | 3.64 | 0.057 | 2.53e-03 | 3.49e-03 | 2.53e-03
Residuals |    2.29e+06 | 1038 |     2208.85 |      |       |          |          |

Anova Table (Type 1 tests)

$Maximum_pitch
Parameter | Sum_Squares |   df | Mean_Square |     F |      p | Omega2 | Eta2 | Epsilon2
---------------------------------------------------------------------------------------
Class     |    2.98e+05 |    1 |    2.98e+05 | 20.57 | < .001 |   0.02 | 0.02 |     0.02
Residuals |    1.51e+07 | 1038 |    14499.18 |       |        |        |      |

Anova Table (Type 1 tests)

$Number_of_pulses
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
-------------------------------------------------------------------------------------------
Class     |    39384.62 |    1 |    39384.62 | 1.75 | 0.186 | 7.22e-04 | 1.68e-03 | 7.22e-04
Residuals |    2.33e+07 | 1038 |    22492.05 |      |       |          |          |

Anova Table (Type 1 tests)

$Number_of_periods
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
-------------------------------------------------------------------------------------------
Class     |    51465.25 |    1 |    51465.25 | 2.31 | 0.129 | 1.26e-03 | 2.22e-03 | 1.26e-03
Residuals |    2.31e+07 | 1038 |    22297.39 |      |       |          |          |

Anova Table (Type 1 tests)
```

```
$Mean_period
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
------------------------------------------------------------------------------------------
Class     |    2.44e-05 |    1 |    2.44e-05 | 6.99 | 0.008 | 5.72e-03 | 6.69e-03 | 5.73e-03
Residuals |    3.63e-03 | 1038 |    3.50e-06 |      |       |          |          |

Anova Table (Type 1 tests)

$Standard_deviation_period
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
------------------------------------------------------------------------------------------
Class     |    1.88e-06 |    1 |    1.88e-06 | 3.60 | 0.058 | 2.49e-03 | 3.46e-03 | 2.50e-03
Residuals |    5.41e-04 | 1038 |    5.22e-07 |      |       |          |          |

Anova Table (Type 1 tests)

$Fraction_locally_unvoiced_frames
Parameter | Sum_Squares |   df | Mean_Square |     F |      p | Omega2 | Eta2 | Epsilon2
--------------------------------------------------------------------------------------
Class     |     6808.68 |    1 |     6808.68 | 15.69 | < .001 |   0.01 | 0.01 |     0.01
Residuals |    4.50e+05 | 1038 |      433.83 |       |        |        |      |

Anova Table (Type 1 tests)

$Number_of_voice_breaks
Parameter | Sum_Squares |   df | Mean_Square |    F |     p |   Omega2 |     Eta2 | Epsilon2
------------------------------------------------------------------------------------------
Class     |       17.27 |    1 |       17.27 | 6.66 | 0.010 | 5.41e-03 | 6.37e-03 | 5.42e-03
Residuals |     2691.89 | 1038 |        2.59 |      |       |          |          |

Anova Table (Type 1 tests)

$Degree_of_voice_breaks
Parameter | Sum_Squares |   df | Mean_Square |     F |      p | Omega2 | Eta2 | Epsilon2
--------------------------------------------------------------------------------------
Class     |     3801.06 |    1 |     3801.06 | 16.79 | < .001 |   0.01 | 0.02 |     0.01
Residuals |    2.35e+05 | 1038 |      226.44 |       |        |        |      |

Anova Table (Type 1 tests)

$UPDRS
Parameter | Sum_Squares |   df | Mean_Square |       F |      p | Omega2 | Eta2 | Epsilon2
----------------------------------------------------------------------------------------
Class     |    1.50e+05 |    1 |    1.50e+05 | 1378.89 | < .001 |   0.57 | 0.57 |     0.57
Residuals |    1.13e+05 | 1038 |      108.61 |         |        |        |      |

Anova Table (Type 1 tests)
```

```
>
> # 2. **Principal Component Analysis (PCA)**
>
> # Exclude non-numeric and categorical columns for PCA
> numeric_data <- train_data %>% select(-Subject_id, -Class)
>
> # Perform PCA with 12 factors and Varimax rotation
> pca_result <- psych::pca(numeric_data, nfactors = 12, rotate = "varimax")
>
> # Print PCA results
> print("PCA Results:")
[1] "PCA Results:"
> print(pca_result)
Principal Components Analysis
Call: principal(r = r, nfactors = nfactors, residuals = residuals,
    rotate = rotate, n.obs = n.obs, covar = covar, scores = scores,
    missing = missing, impute = impute, oblique.scores = oblique.scores,
    method = method, use = use, cor = cor, correct = 0.5, weight = NULL)
Mean item complexity =  1.7
Test of the hypothesis that 12 components are sufficient.

The root mean square of the residuals (RMSR) is  0.01
 with the empirical chi square  76.37  with prob <  0.89

Fit based upon off diagonal values = 1>
> # Extract the PCA loadings
> loadings <- as.data.frame(unclass(pca_result$loadings))
>
> # Calculate the importance of each feature (sum of absolute loadings across components)
> feature_importance <- rowSums(abs(loadings))
>
> # Select the top 12 most important features based on feature importance
> selected_features <- names(sort(feature_importance, decreasing = TRUE)[1:12])
>
> # Print selected features based on PCA importance
> print("Selected Features Based on PCA:")
[1] "Selected Features Based on PCA:"
> print(selected_features)
 [1]
"HTN"                    "AC"                    "NTH"                    "Shimmer_local_dB"

 [5] "Jitter_local_absolute"          "Fraction_locally_unvoiced_frames"
"Degree_of_voice_breaks"          "Shimmer_apq11"
 [9]
"Shimmer_local"               "Maximum_pitch"               "Jitter_local"               "Number_
of_voice_breaks"
```

REFERENCES

1. Poewe, W., et al. (2017). Parkinson disease. *Nature Reviews Disease Primers, 3*(1), 1-21. DOI: 10.1038/nrdp.2017.13
2. Little, M. A., et al. (2009). Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Transactions on Biomedical Engineering, 56*(4), 1015-1022. DOI: 10.1109/TBME.2008.2005954
3. Senturk, S. (2020). Recursive feature elimination for Parkinson's disease classification. *Frontiers in Neuroscience.* DOI: 10.3389/fnins.2020.00567
4. Van Gelderen, L., & Tejedor-García, C. (2024). Speech-based deep learning for Parkinson's disease. *MDPI Applied Sciences.* DOI: 10.3390/app14177873
5. Uriarte-Arcia, A. V., et al. (2023). Parkinson's disease detection using associative memories. *Healthcare, 11*(11), 1601. DOI: 10.3390/healthcare11111601
6. Rasheed, A., et al. (2020). Deep belief networks for PD diagnosis. *IEEE Access, 7*, 115540-115550. DOI: 10.1109/ACCESS.2020.3015926
7. Arias-Vergara, T., et al. (2020). Machine learning-based detection of Parkinson's disease. *IEEE Journal of Biomedical and Health Informatics, 24*(5), 1367-1376. DOI: 10.1109/JBHI.2019.2961576

BIODATA

Name	: U. Manikanta
Mobile Number	: 9550718969
E-mail	: manikanta.21bce9726@vitap.ac.in
Permanaent Address	: 11-37 Sivaganga street, Amaravati,522020, AP