

SUMMER TRAINING/INTERNSHIP

PROJECT REPORT

(Term June - July 2025)

Detection of Neurological Disorder - Parkinson Disease

Submitted by

Saksham

Registration Number: 12305209

Mohd. Maaz

Registration Number: 12325672

Course Code: PETV79

Under the Guidance of

Mahipal Singh Papola

School of Computer Science and Engineering

Lovely Professional University, Punjab

Bonafide Certificate

Certified that this project report “Detection of Neurological Disorder – Parkinson Disease” is the Bonafide work of “Saksham, Mohd. Maaz” who carried out this Project under my supervision.

SIGNATURE

<< Name of the Supervisor >>

Saksham

Mohd. Maaz

SIGNATURE

<< Signature of the Head of the Department >>

SIGNATURE

<< Name >>

HEAD OF THE DEPARTMENT

<< Signature of the Supervisor >>

Table of Content

S.no	Chapter Name	Page No.
1.	Introduction	4
2.	Training Overview	4
3.	Project Details	6
4.	Implementation	7
5.	Result and Discussion	20
6.	Conclusion	22

Chapter - 1 Introduction

1.1 Company Profile

This project was developed as part of my academic training to strengthen skills in data science and machine learning. Though there is no specific company involved in this case, this project simulates a professional industry-standard project workflow, covering all necessary components like data preprocessing, visualization, feature analysis, model building, hyperparameter tuning, model comparison, and deployment.

1.2 Overview of Training Domain

The domain of this project is Healthcare & Machine Learning. Parkinson's Disease (PD) is a chronic neurodegenerative disorder affecting millions worldwide. Early diagnosis through machine learning models can help doctors with faster, accurate detection using voice datasets. The dataset primarily consists of voice measurements.

Machine learning algorithms such as SVM (Support Vector Machine) and Logistic Regression are applied after cleaning and preprocessing the dataset. Additionally, exploratory data analysis (EDA) and hyperparameter tuning are used to improve the model's accuracy.

1.3 Objective of the Project

- To predict Parkinson's Disease from a given dataset based on vocal parameters.
- To preprocess and clean the data and convert it into a suitable form for analysis.
- To visualize and analyze the features for better understanding and insight.
- To build and compare machine learning models and identify the most effective one.
- To deploy the best-performing model via a web application (Streamlit) for easy accessibility to end-users like doctors or researchers.
- **Programming Language:** Python
- **Libraries:** Pandas, Numpy, Matplotlib, Seaborn, Scikit-learn, Joblib, Streamlit
- **Machine Learning Models:** SVM, Logistic Regression
- **Deployment Tool:** Streamlit
- **Dataset Format:** CSV files
- **Development Environment:** Google Colab, VS Code

Chapter – 2 Training Overview

2.1 Areas Covered During Training

- **Data Preprocessing:**
Performed data cleaning by handling missing values, converting data types, removing duplicates, and scaling the data using **StandardScaler**.
- **Exploratory Data Analysis (EDA):**
Used **Matplotlib** and **Seaborn** to visualize data through heatmaps, boxplots, and correlation plots for better insights into feature relationships.
- **Supervised Learning Models:**

Built and compared models such as **Logistic Regression, Random Forest, XGBoost, and Support Vector Machine (SVM)** for Parkinson's Disease prediction.

- **Evaluation Metrics:**
Evaluated models using Accuracy, Precision, Recall, F1-Score, and Confusion Matrix to ensure reliable results.
- **Hyperparameter Tuning:**
Improved model performance through GridSearchCV for optimal parameter selection.
- **Model Comparison and Visualization:**
Compared models and visualized results to support analysis and conclusions effectively.

2.2 Daily/Weekly Work Summary

- **Week 1:**

Explored the dataset containing over 2000 records and multiple vocal features.

Performed data cleaning by removing unnecessary columns (name, comment), handled missing values using `pd.to_numeric`, converted necessary columns to float, and ensured the status column contained binary labels.

- **Week 2:**

Conducted feature selection using `SelectKBest` to focus only on the 10 most important predictors for Parkinson's detection.

Applied `StandardScaler` for proper feature scaling.

- **Week 3:**

Built machine learning models with Logistic Regression and Support Vector Machine (SVM).

Used `GridSearchCV` to tune hyperparameters for SVM for optimal performance.

Evaluated models using accuracy, classification reports, and confusion matrix.

- **Week 4:**

Created data visualizations like correlation heatmaps and confusion matrices to improve interpretability.

Finalized the project by deploying it on Streamlit with proper user input prompts and trained model integration.

Exported the model and scaler as `.joblib` files for deployment.

Completed detailed project documentation and report writing.

Chapter – 3 Project Details

3.1 Title of the project

Detection Of NeuroLogical Disorder – Parkinson Disease

3.2 Problem Definition

Parkinson's Disease (PD) is a chronic and progressive neurological disorder that affects movement and other bodily functions. Early detection is crucial for better treatment outcomes. Traditional diagnosis relies on clinical symptoms, which often appear after significant neurological damage.

This project aims to develop a machine learning-based solution that can accurately predict Parkinson's Disease from voice measurement datasets, helping in the early detection and diagnosis process. The model focuses on identifying patterns within biomedical voice measurements that differentiate between healthy individuals and patients with Parkinson's Disease.

3.3 Scope and Objectives

- To utilize machine learning techniques for the early detection of Parkinson's Disease using biomedical voice measurements.
- To perform data collection, cleaning, and preprocessing to ensure the dataset is suitable for analysis.
- To apply advanced feature selection techniques such as SelectKBest to identify the most impactful and relevant features.
- To develop predictive models using machine learning algorithms like Support Vector Machine (SVM) and Logistic Regression.
- To perform hyperparameter tuning using GridSearchCV to enhance model performance.
- To evaluate the models using various performance metrics such as Accuracy, Precision, Recall, and F1-Score.
- To visualize the results effectively through confusion matrices and correlation heatmaps for better interpretability.
- To deploy the best-performing model through a user-friendly Streamlit web application for easy accessibility to healthcare professionals and researchers.
- To provide a practical, accurate, and accessible solution to support the early diagnosis of Parkinson's Disease.

3.4 System Requirements

<u>Component</u>	<u>Specification</u>
Operating System:	Windows 10 / Linux
Software:	Jupyter Notebook / VS Code
Hardware:	Minimum 8GB RAM, 2GHz Processor
Libraries/Tools:	Numpy, Pandas, Scikit-learn, Matplotlib, Seaborn, Streamlit, Joblib

3.5 Architecture Diagram

Raw Dataset (Voice Measurements) → Data Preprocessing (Cleaning, Scaling, Encoding) → Feature Selection (SelectKBest, Mutual Info) → Model Training (SVM, Logistic Regression) → Hyperparameter Tuning (GridSearchCV) → Model Evaluation (Accuracy, F1- score, AUC) → Deployment (Streamlit)

3.6 Data flow / UML Diagrams

User Input (Voice Features) → Web Application (Streamlit) → Preprocessing (Scaler) → Machine Learning Model (Tuned SVM Classifier) → Prediction (Parkinson / Not) → Display Result to User

Chapter – 4 Implementation

4.1 Tool Used

❖ **Language: Python Language.**

❖ **Libraries:**

- Scikit-learn: For machine learning models, preprocessing, feature selection, evaluation metrics, and hyperparameter tuning.
- Joblib: For model serialization and saving scaler objects.
- Pandas: For data manipulation and analysis.
- NumPy: For numerical operations.
- Matplotlib & Seaborn: For data visualization (heatmaps, boxplots, correlation plots, confusion matrix).

❖ **Deployment Framework:**

- ❖ Streamlit: For building and deploying a user-friendly web-based prediction interface.

4.2 Methodology

Data Collection:

The dataset containing biomedical voice measurements was collected for Parkinson's Disease detection.

Data Cleaning:

Removed unnecessary columns (name, comment).

Handled missing values and removed duplicates.

Converted relevant columns to numerical format where required.

Ensured the target column status contained only 0 and 1 values for binary classification.

Feature Selection:

Applied SelectKBest with mutual_info_classif to select the top 10 most relevant features influencing the prediction.

Feature Scaling:

Used StandardScaler to bring all features to the same scale to improve model convergence and accuracy.

Modeling:

Trained models using: Support Vector Machine (SVM) with hyperparameter tuning (GridSearchCV).

Logistic Regression.

Model Evaluation:

Evaluated using metrics such as Accuracy, Precision, Recall, F1-Score, and visualized using confusion matrices.

Calculated Train/Test Accuracy to check for overfitting.

Hyperparameter Tuning:

Tuned the SVM classifier using GridSearchCV for best parameters of C and kernel.

Visualization:

Used correlation heatmaps, boxplots, scatter plot and confusion matrices to understand the relationship between features and model performance.

Deployment:

Finalized and deployed the Tuned SVM Model and Scaler through Streamlit for real-time prediction.

4.3 Modules

Preprocessing Module

Handled missing values.

Dropped unnecessary columns (name, comment).

Converted relevant features to numeric.

Scaled the data using **StandardScaler**.

Feature Selection Module

- Selected the top 10 features using **SelectKBest** and **mutual information** to improve model performance.

Modeling Module

- Built and trained models (SVM, Logistic Regression).
- Tuned hyperparameters with **GridSearchCV**.
- Generated predictions for evaluation.

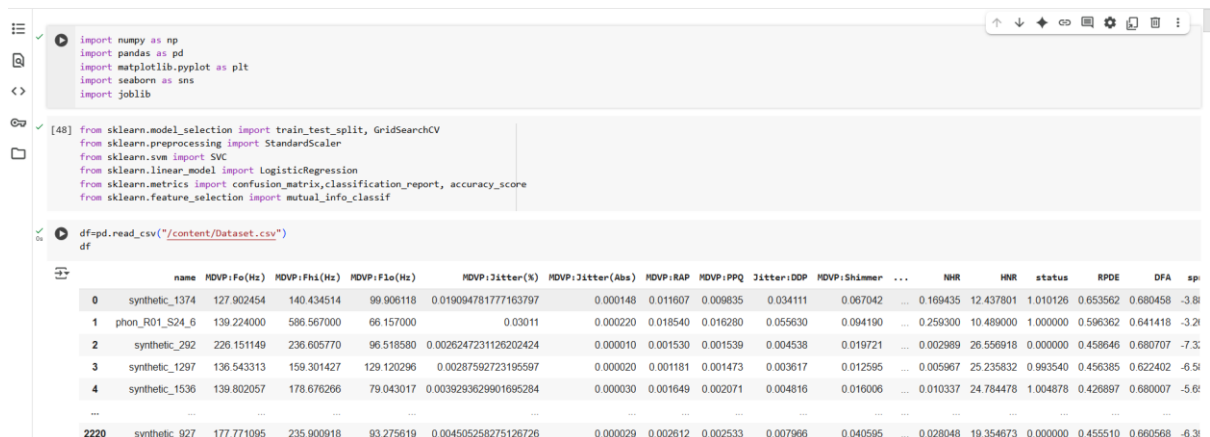
Evaluation Module

- Computed accuracy, precision, recall, F1-score.
- Plotted correlation heatmap and confusion matrix for performance visualization.
- Checked both train and test accuracy to confirm model reliability.

Deployment Module

- Used **Streamlit** to create a user interface where healthcare professionals or users can input feature values and get immediate predictions.

4.4 Screenshots



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import joblib

[48]: from sklearn.model_selection import train_test_split, GridSearchCV
      from sklearn.preprocessing import StandardScaler
      from sklearn.svm import SVC
      from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
      from sklearn.feature_selection import mutual_info_classif

df=pd.read_csv("/content/Dataset.csv")
df
```

	name	HDVP:Fo(Hz)	HDVP:Fhi(Hz)	HDVP:Flo(Hz)	HDVP:Jitter(X)	HDVP:Jitter(Abs)	HDVP:RAP	HDVP:PPQ	Jitter:DOP	HDVP:Shimmer	...	NHR	HNHR	status	RPDE	DFA	sp
0	synthetic_1374	127.902454	140.434514	99.906118	0.019094781777163797	0.000148	0.011607	0.009835	0.034111	0.067042	...	0.169435	12.437801	1.010126	0.653562	0.680458	-3.8
1	phon_RD1_S24_6	139.224000	586.567000	66.157000	0.03011	0.000220	0.018540	0.016280	0.055630	0.094190	...	0.259300	10.489000	1.000000	0.596362	0.641418	-3.2
2	synthetic_292	226.151149	236.605770	96.518580	0.0026247231126202424	0.000010	0.001530	0.001539	0.004538	0.019721	...	0.002989	26.556918	0.000000	0.458646	0.680707	-7.3
3	synthetic_1297	136.543313	159.301427	129.120296	0.00287592723195597	0.000020	0.001181	0.001473	0.003617	0.012595	...	0.005967	25.235832	0.993540	0.456385	0.622402	-6.5
4	synthetic_1536	139.802057	178.676266	79.043017	0.0039293629901695284	0.000030	0.001649	0.002071	0.004816	0.016006	...	0.010337	24.784478	1.004878	0.426897	0.680007	-5.6
...
2220	synthetic_927	177.771095	235.900918	93.275619	0.004505258275126726	0.000029	0.002612	0.002533	0.007996	0.040595	...	0.028048	19.354673	0.000000	0.455510	0.660568	-6.3

[50]	df.head()
	<pre> name MDVP:F0(Hz) MDVP:F1(Hz) MDVP:F1o(Hz) MDVP:Jitter(%) MDVP:Jitter(Abs) MDVP:RAP MDVP:PPQ Jitter:DDP MDVP:Shimmer ... NHR HNR status RPDE DFA spread 0 synthetic_1374 127.902454 140.434514 99.906118 0.019094781777163797 0.000148 0.011607 0.009835 0.034111 0.067042 ... 0.169435 12.437801 1.010126 0.653562 0.680458 -3.8837 1 phon_R01_S24_6 139.224000 586.567000 66.157000 0.03011 0.000220 0.018540 0.016280 0.055630 0.094190 ... 0.259300 10.489000 1.000000 0.596362 0.641418 -3.2694 2 synthetic_292 226.151149 236.605770 96.518580 0.0026247231126202424 0.000010 0.001530 0.001539 0.004538 0.019721 ... 0.002989 26.556918 0.000000 0.458646 0.680707 -7.3206 3 synthetic_1297 136.543313 159.301427 129.120296 0.00287592723195597 0.000020 0.001181 0.001473 0.003617 0.012595 ... 0.005967 25.235832 0.993540 0.456385 0.622402 -6.5896 4 synthetic_1536 139.802057 178.676266 79.043017 0.0039293629901695284 0.000030 0.001649 0.002071 0.004816 0.016006 ... 0.010337 24.784478 1.004878 0.426897 0.680007 -5.6589 5 rows x 25 columns </pre>
	<pre> df.info() <class 'pandas.core.frame.DataFrame'> RangeIndex: 2225 entries, 0 to 2224 Data columns (total 25 columns): # Column Non-Null Count Dtype --- 0 name 2225 non-null object 1 MDVP:F0(Hz) 2152 non-null float64 2 MDVP:F1(Hz) 2151 non-null float64 3 MDVP:F1o(Hz) 2151 non-null float64 4 MDVP:Jitter(%) 2152 non-null object 5 MDVP:Jitter(Abs) 2225 non-null float64 6 MDVP:RAP 2225 non-null float64 7 MDVP:PPQ 2225 non-null float64 8 Jitter:DDP 2225 non-null float64 9 MDVP:Shimmer 2225 non-null float64 </pre>
	<pre> df.isnull().sum() name 0 MDVP:F0(Hz) 73 MDVP:F1(Hz) 74 MDVP:F1o(Hz) 74 MDVP:Jitter(%) 73 MDVP:Jitter(Abs) 0 MDVP:RAP 0 MDVP:PPQ 0 Jitter:DDP 0 MDVP:Shimmer 0 MDVP:Shimmer(dB) 0 Shimmer:APQ3 0 Shimmer:APQ5 0 MDVP:APQ 0 Shimmer:DDA 0 NHR 0 HNR 0 </pre>
	<pre> [53] print(df['status'].value_counts()) status 0.000000 1063 1.000000 149 1.008112 2 0.998696 2 0.997456 2 ... 0.985692 1 0.988758 1 0.989565 1 1.017543 1 1.010315 1 Name: count, Length: 1002, dtype: int64 </pre>
	<pre> df=df.drop(columns=["comment","name"]) </pre>
[55]	df.dropna()
	<pre> MDVP:F0(Hz) MDVP:F1(Hz) MDVP:F1o(Hz) MDVP:Jitter(%) MDVP:Jitter(Abs) MDVP:RAP MDVP:PPQ Jitter:DDP MDVP:Shimmer MDVP:Shimmer(dB) ... Shimmer:DDA NHR HNR status RPDE 0 127.902454 140.434514 99.906118 0.019094781777163797 0.000148 0.011607 0.009835 0.034111 0.067042 0.641673 ... 0.109209 0.169435 12.437801 1.010126 0.653562 1 139.224000 586.567000 66.157000 0.03011 0.000220 0.018540 0.016280 0.055630 0.094190 0.930000 ... 0.166540 0.259300 10.489000 1.000000 0.596362 2 226.151149 236.605770 96.518580 0.0026247231126202424 0.000010 0.001530 0.001539 0.004538 0.019721 0.160627 ... 0.030092 0.002989 26.556918 0.000000 0.458646 3 136.543313 159.301427 129.120296 0.00287592723195597 0.000020 0.001181 0.001473 0.003617 0.012595 0.109614 ... 0.002099 0.005967 25.235832 0.993540 0.456385 4 139.802057 178.676266 79.043017 0.0039293629901695284 0.000030 0.001649 0.002071 0.004816 0.016006 0.154551 ... 0.021881 0.010337 24.784478 1.004878 0.426897 2220 177.771095 235.900918 93.275619 0.004505258275126726 0.000029 0.002612 0.002533 0.007966 0.040595 0.403158 ... 0.072444 0.028048 19.354673 0.000000 0.455510 </pre>

```
[56] df.head()
```

	MDVP:F0(Hz)	MDVP:F1(Hz)	MDVP:F1o(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer(dB)	...	Shimmer:DDA	NHR	HNR	status	RPDE
0	127.902454	140.434514	99.906118	0.019094781777163797	0.000148	0.011607	0.009835	0.034111	0.067042	0.641673	...	0.109209	0.169435	12.437801	1.010126	0.653562
1	139.224000	586.567000	66.157000	0.03011	0.000220	0.018540	0.016280	0.055630	0.094190	0.930000	...	0.166540	0.259300	10.489000	1.000000	0.596362
2	226.151149	236.605770	96.518580	0.0026247231126202424	0.000010	0.001530	0.001539	0.004538	0.019721	0.160627	...	0.030092	0.002989	26.556918	0.000000	0.458646
3	136.543313	159.301427	129.120296	0.002875927231195597	0.000020	0.001181	0.001473	0.003617	0.012595	0.109614	...	0.020299	0.005967	25.235832	0.993540	0.456385
4	139.802057	178.676266	79.043017	0.0039293629901695284	0.000030	0.001649	0.002071	0.004816	0.016006	0.154551	...	0.021881	0.010337	24.784478	1.004878	0.426897

5 rows × 23 columns

```
df
```

	MDVP:F0(Hz)	MDVP:F1(Hz)	MDVP:F1o(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer(dB)	...	Shimmer:DDA	NHR	HNR	status	RPDE
0	127.902454	140.434514	99.906118	0.019094781777163797	0.000148	0.011607	0.009835	0.034111	0.067042	0.641673	...	0.109209	0.169435	12.437801	1.010126	0.653562
1	139.224000	586.567000	66.157000	0.03011	0.000220	0.018540	0.016280	0.055630	0.094190	0.930000	...	0.166540	0.259300	10.489000	1.000000	0.596362
2	226.151149	236.605770	96.518580	0.0026247231126202424	0.000010	0.001530	0.001539	0.004538	0.019721	0.160627	...	0.030092	0.002989	26.556918	0.000000	0.458646
3	136.543313	159.301427	129.120296	0.002875927231195597	0.000020	0.001181	0.001473	0.003617	0.012595	0.109614	...	0.020299	0.005967	25.235832	0.993540	0.456385
4	139.802057	178.676266	79.043017	0.0039293629901695284	0.000030	0.001649	0.002071	0.004816	0.016006	0.154551	...	0.021881	0.010337	24.784478	1.004878	0.426897
...
2220	177.771095	235.900918	93.275619	0.004505258275126726	0.000029	0.002612	0.002533	0.007966	0.040595	0.403158	...	0.072444	0.028048	19.354673	0.000000	0.455510

```
[58] df = df[pd.to_numeric(df['MDVP:Jitter(%)'], errors='coerce').notnull()]
df['MDVP:Jitter(%)'] = df['MDVP:Jitter(%)'].astype(float)

/tmp/ipython-input-58-89858958.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df['MDVP:Jitter(%)'] = df['MDVP:Jitter(%)'].astype(float)

[59] df = df.drop_duplicates().reset_index(drop=True)

[60] df['status'] = df['status'].apply(lambda x: 1 if x >= 0.5 else 0)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2147 entries, 0 to 2146
Data columns (total 23 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   MDVP:F0(Hz)         2147 non-null   float64
1   MDVP:F1(Hz)         2147 non-null   float64
2   MDVP:F1o(Hz)        2147 non-null   float64
3   MDVP:Jitter(%)      2147 non-null   float64
4   MDVP:Jitter(Abs)    2147 non-null   float64
5   MDVP:RAP            2147 non-null   float64
6   MDVP:PPQ            2147 non-null   float64
7   Jitter:DDP          2147 non-null   float64
8   MDVP:Shimmer        2147 non-null   float64
9   MDVP:Shimmer(dB)    2147 non-null   float64
10  Shimmer:APQ3        2147 non-null   float64
```

```
[62] print("After Clean: ",df.shape)
print(df['status'].value_counts())
```

```
After Clean: (2147, 23)
status
1    1120
0    1027
Name: count, dtype: int64
```

```
sns.countplot(x='status', data=df, palette='Set2')
plt.title("Number of People With and Without Parkinson Disease")
plt.xlabel('Status')
plt.ylabel('Count')
plt.show()
```

```
/tmp/ipython-input-63-3283823040.py:1: FutureWarning:
Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign the 'x' variable to 'hue' and set 'legend=False' for the same effect.

sns.countplot(x='status', data=df, palette='Set2')
```

Status	Count
0	1027
1	1120

```

X = df.drop(columns=['status'])
Y = df['status']
df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2147 entries, 0 to 2146
Data columns (total 23 columns):
 #   Column              Non-Null Count  Dtype  
---  -
 0   MDVP:Fo(Hz)         2147 non-null  float64
 1   MDVP:Fhi(Hz)        2147 non-null  float64
 2   MDVP:Flo(Hz)        2147 non-null  float64
 3   MDVP:Jitter(%)      2147 non-null  float64
 4   MDVP:Jitter(Abs)    2147 non-null  float64
 5   MDVP:RAP            2147 non-null  float64
 6   MDVP:PPQ            2147 non-null  float64
 7   Jitter:DDP          2147 non-null  float64
 8   MDVP:Shimmer        2147 non-null  float64
 9   MDVP:Shimmer(dB)    2147 non-null  float64
10  Shimmer:APQ3        2147 non-null  float64
11  Shimmer:APQ5        2147 non-null  float64
12  MDVP:APQ            2147 non-null  float64
13  Shimmer:DDA         2147 non-null  float64
14  HNR                 2147 non-null  float64
15  HNR                 2147 non-null  float64
16  status              2147 non-null  int64  
17  RPDE                2147 non-null  float64
18  DFA                 2147 non-null  float64
19  spread1             2147 non-null  float64
20  spread2             2147 non-null  float64
21  D2                  2147 non-null  float64
22  PPE                 2147 non-null  float64
dtypes: float64(22), int64(1)
memory usage: 385.9 KB

```

```

from sklearn.feature_selection import SelectKBest, mutual_info_classif
selector = SelectKBest(score_func=mutual_info_classif, k=10)
X_selected = selector.fit_transform(X, Y)
selected_features = X.columns[selector.get_support()]
X = df[selected_features]

print("Selected Features:", selected_features.tolist())

```

```

Selected Features: ['MDVP:Fo(Hz)', 'MDVP:Flo(Hz)', 'MDVP:Jitter(%)', 'MDVP:Jitter(Abs)', 'Jitter:DDP', 'MDVP:APQ', 'HNR', 'spread1', 'spread2', 'PPE']

```

```

[66] X_train, X_test, y_train, y_test = train_test_split(X, Y, stratify=Y, test_size=0.2, random_state=42)

```

```

[67] mi_scores = mutual_info_classif(X, Y)
mi_scores = pd.Series(mi_scores, index=X.columns)
mi_scores = mi_scores.sort_values(ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x=mi_scores, y=mi_scores.index, palette='Blues_r')
plt.xlabel('Mutual Information Score')
plt.ylabel('Features')
plt.title('Selected Feature Importance')
plt.show()

```

/tmp/ipython-input-67-1026928551.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

sns.barplot(x=mi_scores, y=mi_scores.index, palette='Blues_r')

```

[68] scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

```

```

svm_default = SVC()
svm_default.fit(X_train, y_train)
y_pred_default = svm_default.predict(X_test)
accuracy_default = accuracy_score(y_test, y_pred_default)
print(f"Default SVM Test Accuracy: {accuracy_default * 100:.2f}%")
print(classification_report(y_test, y_pred_default))

```

```

Default SVM Test Accuracy: 95.81%
precision    recall  f1-score   support

   0       0.84     0.85     0.84       206
   1       0.86     0.85     0.85       224

 accuracy          0.85          0.85          0.85          430
 macro avg         0.85          0.85          0.85          430
 weighted avg         0.85          0.85          0.85          430

```

```

[70] param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
grid_svm = GridSearchCV(SVC(), param_grid, cv=5)
grid_svm.fit(X_train, y_train)

best_svm = grid_svm.best_estimator_

```

```

[71] from sklearn.metrics import accuracy_score, classification_report

models = {
    "Tuned SVM": best_svm,
}

```

```
[71] from sklearn.metrics import accuracy_score, classification_report

models = {
    'Tuned SVM': best_svm,
    'Logistic Regression': LogisticRegression(max_iter=1000)
}

for name, model in models.items():
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    acc = accuracy_score(y_test, preds)
    print(f"{name} Accuracy: {acc*100:.2f}%")
    print(classification_report(y_test, preds))
```

```
Tuned SVM Accuracy: 98.14%
precision    recall  f1-score   support

0           0.97    0.99    0.98        206
1           0.99    0.97    0.98        224

accuracy          0.98    0.98    0.98        430
macro avg         0.98    0.98    0.98        430
weighted avg      0.98    0.98    0.98        430

Logistic Regression Accuracy: 84.88%
precision    recall  f1-score   support

0           0.84    0.85    0.84        206
1           0.86    0.85    0.85        224

accuracy          0.85    0.85    0.85        430
macro avg         0.85    0.85    0.85        430
weighted avg      0.85    0.85    0.85        430
```

```
[124] best_svm.fit(X_train, y_train)
print(f"Tuned SVM Train Accuracy: {best_svm.score(X_train, y_train) * 100:.2f}%")
print(f"Tuned SVM Test Accuracy: {best_svm.score(X_test, y_test) * 100:.2f}%")

log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
print(f"Logistic Regression Train Accuracy: {log_reg.score(X_train, y_train) * 100:.2f}%")
print(f"Logistic Regression Test Accuracy: {log_reg.score(X_test, y_test) * 100:.2f}%")
```

```
Tuned SVM Train Accuracy: 98.19%
Tuned SVM Test Accuracy: 98.14%
Logistic Regression Train Accuracy: 84.97%
Logistic Regression Test Accuracy: 84.88%
```

```
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns.tolist()
target_col = numeric_cols[-1]

num_plots = len(numeric_cols) - 1
cols = 2
rows = (num_plots + 1) // cols

fig, axes = plt.subplots(rows, cols, figsize=(6, rows * 2))
axes = axes.flatten()

plot_idx = 0
for col in numeric_cols:
    if col == target_col:
        continue
    axes[plot_idx].scatter(df[col], df[target_col], alpha=0.7, s=20)
    axes[plot_idx].set_title(f"{col} vs {target_col}")
    axes[plot_idx].set_xlabel(col)
    axes[plot_idx].set_ylabel(target_col)
    plot_idx += 1

for i in range(plot_idx, len(axes)):
    fig.delaxes(axes[i])

plt.tight_layout()
plt.show()
```

```
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns.tolist()
target_col = numeric_cols[-1]

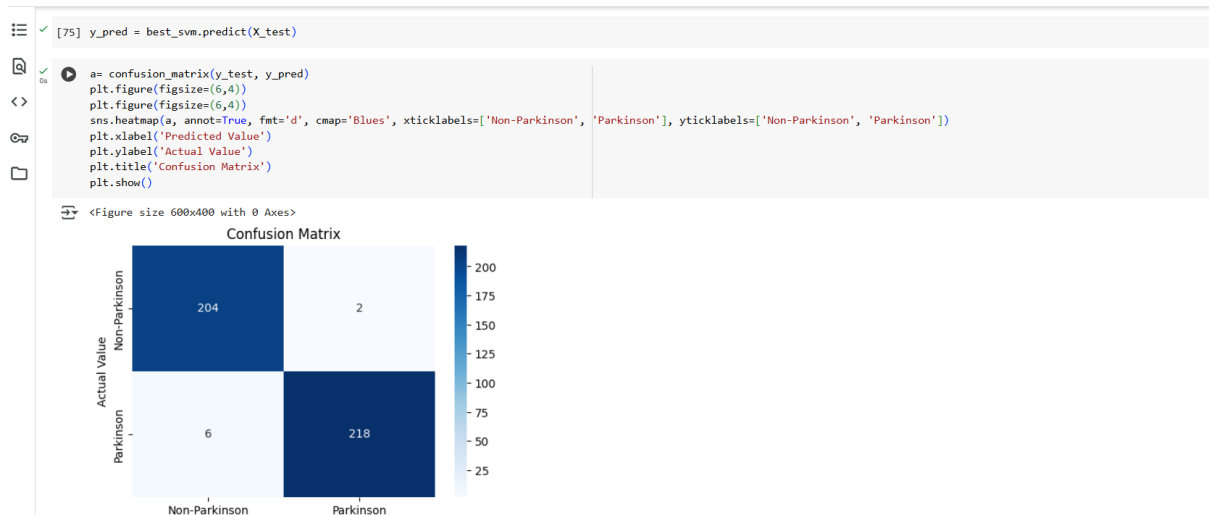
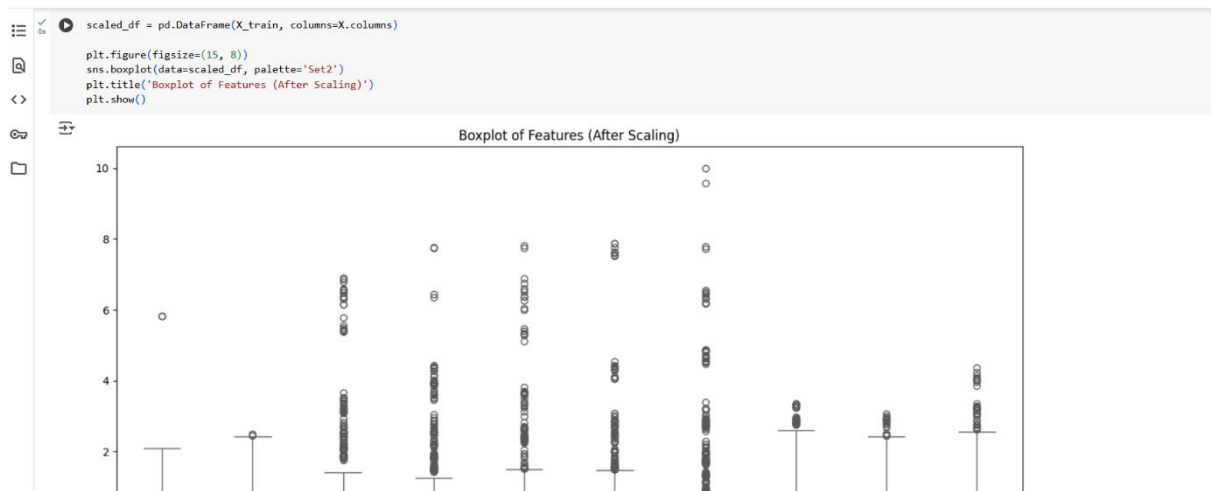
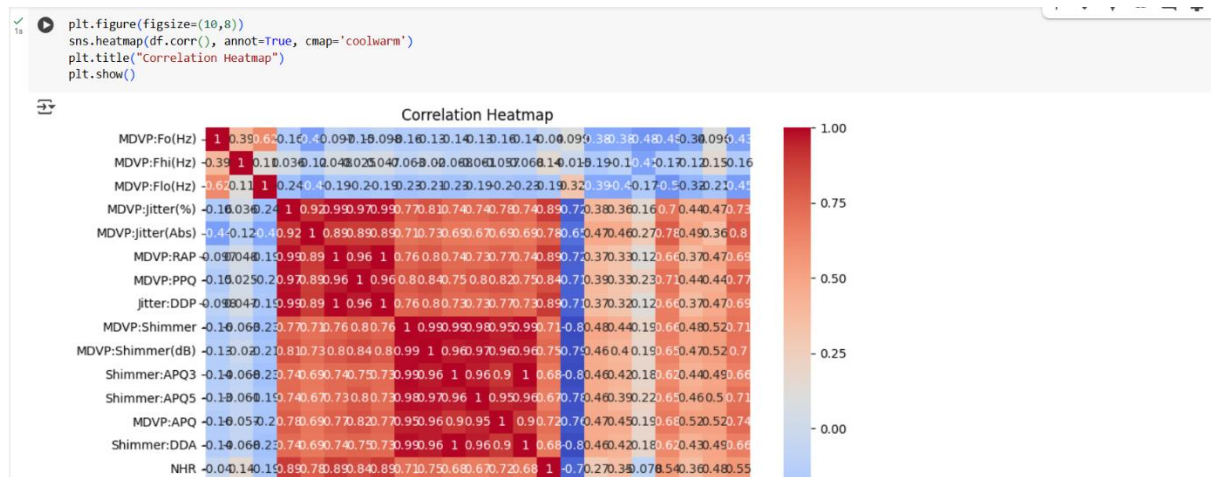
num_plots = len(numeric_cols) - 1
cols = 2
rows = (num_plots + 1) // cols

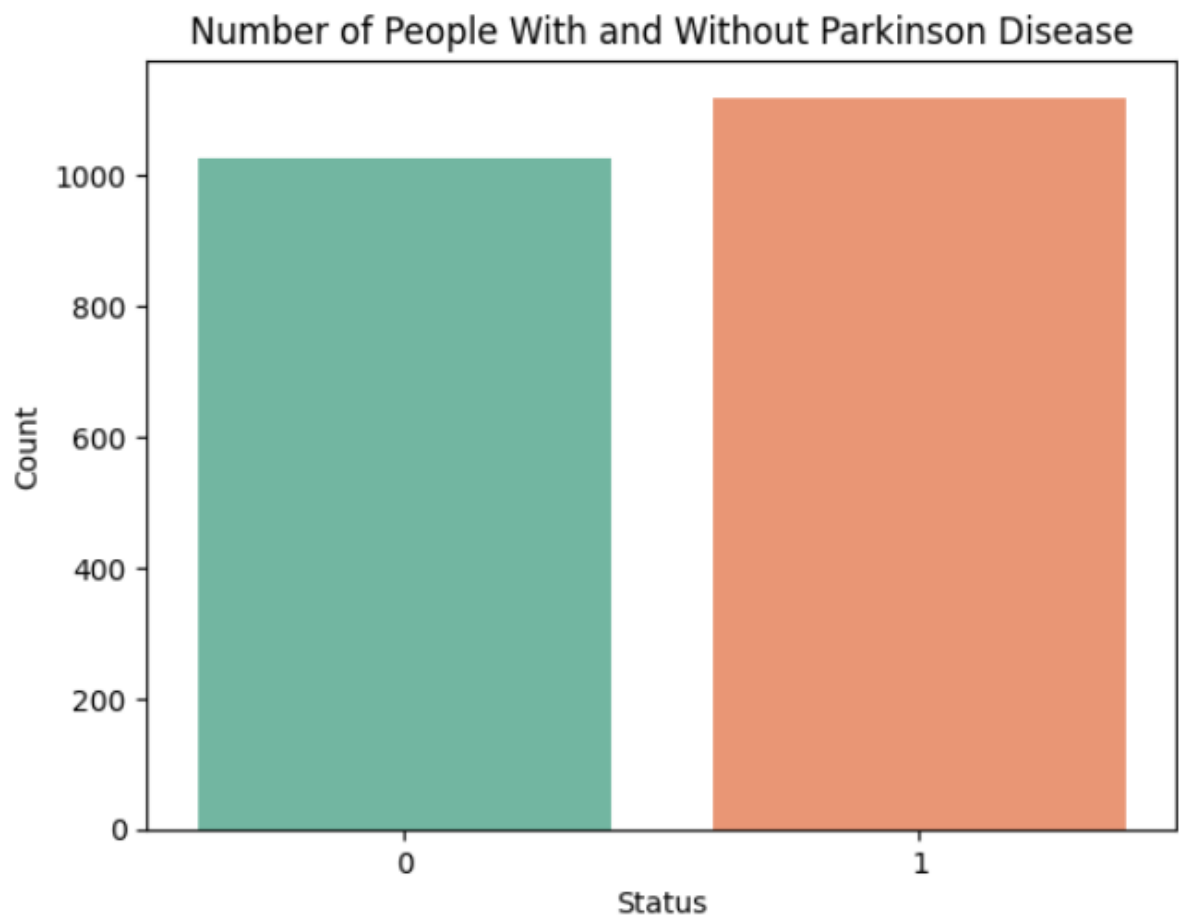
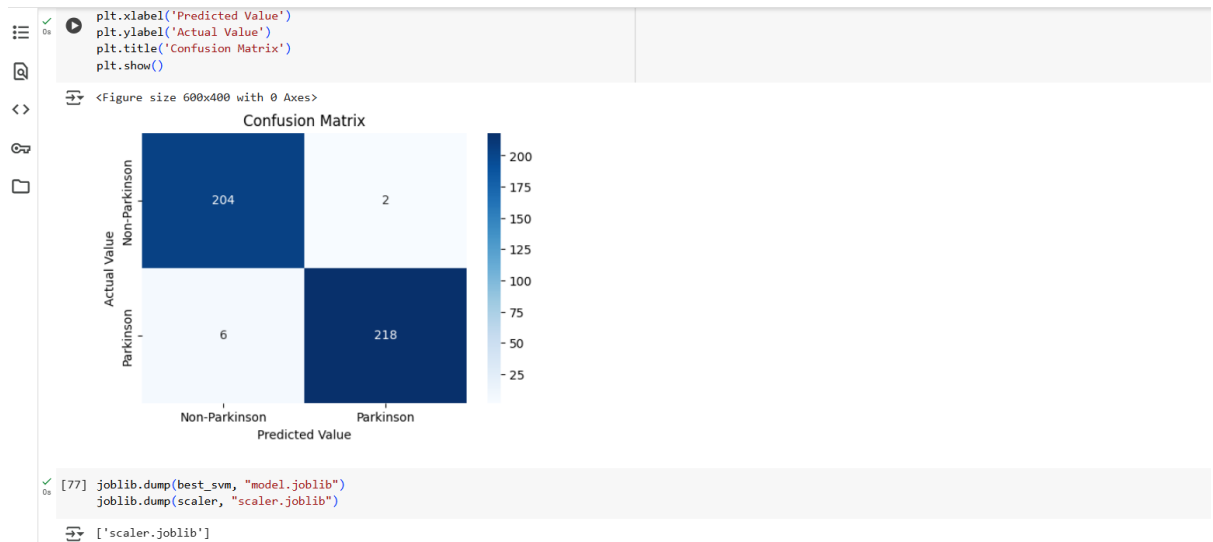
fig, axes = plt.subplots(rows, cols, figsize=(6, rows * 2))
axes = axes.flatten()

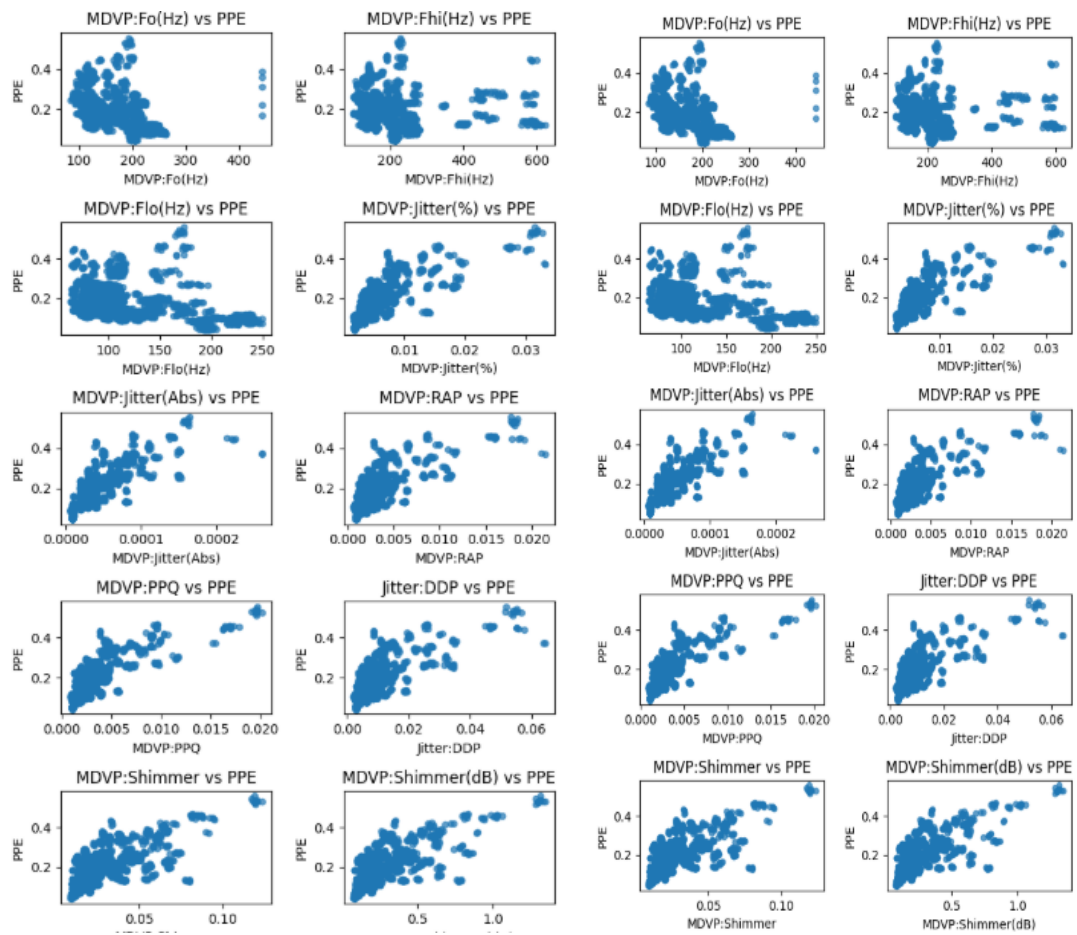
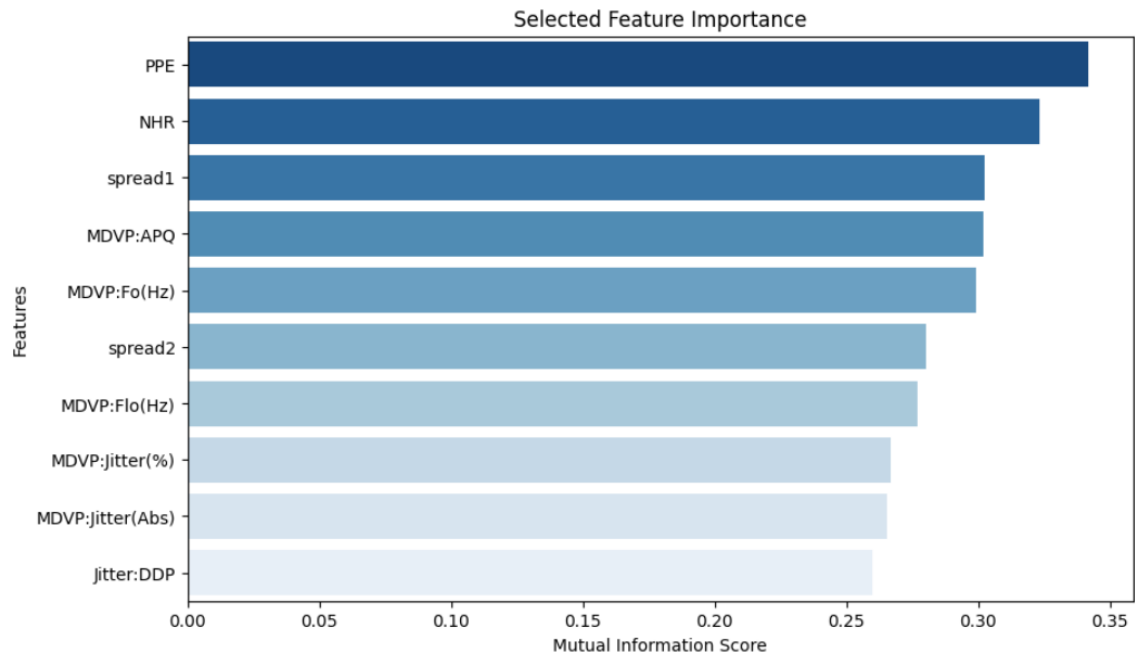
plot_idx = 0
for col in numeric_cols:
    if col == target_col:
        continue
    axes[plot_idx].scatter(df[col], df[target_col], alpha=0.7, s=20)
    axes[plot_idx].set_title(f"{col} vs {target_col}")
    axes[plot_idx].set_xlabel(col)
    axes[plot_idx].set_ylabel(target_col)
    plot_idx += 1

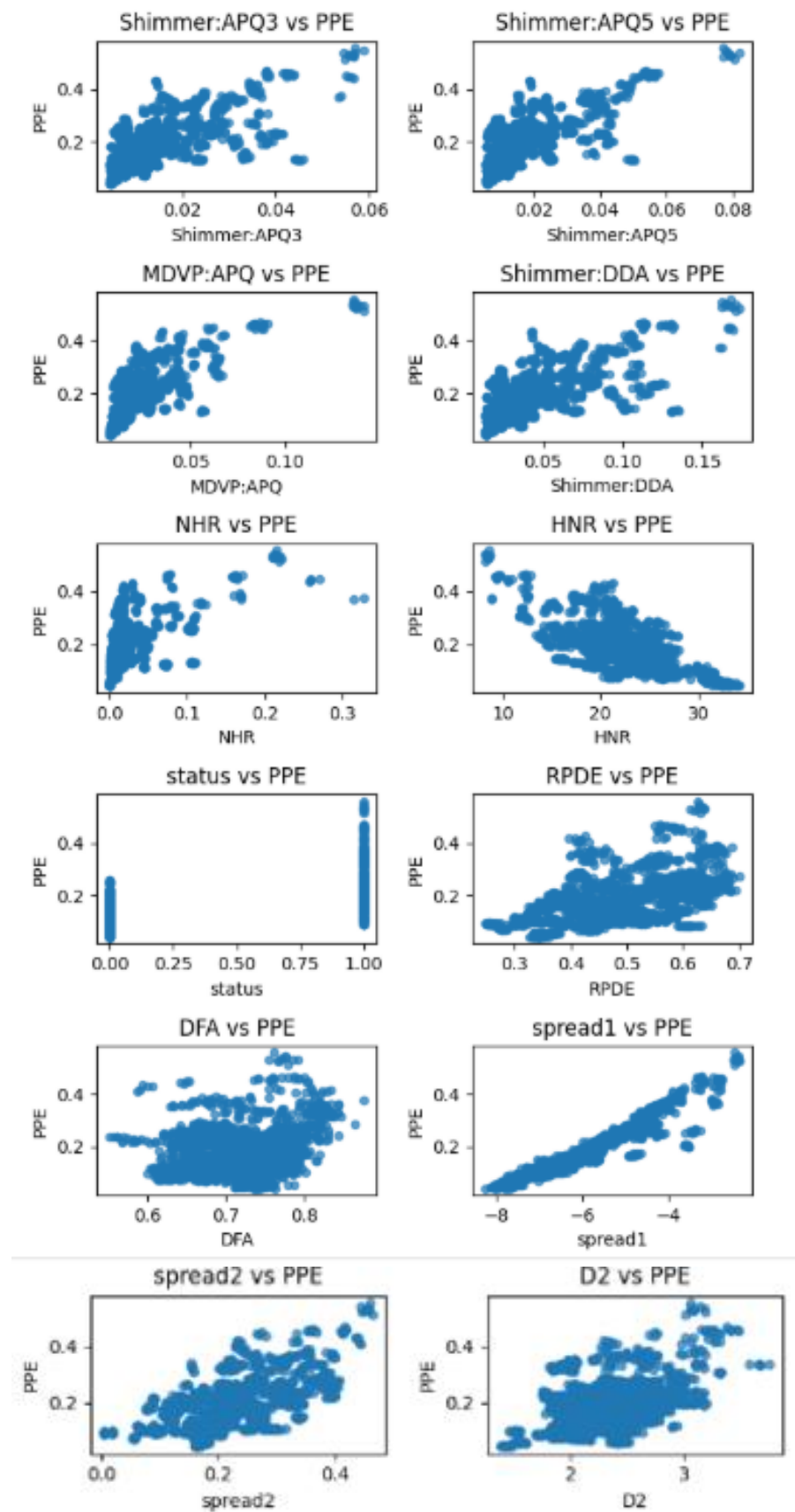
for i in range(plot_idx, len(axes)):
    fig.delaxes(axes[i])

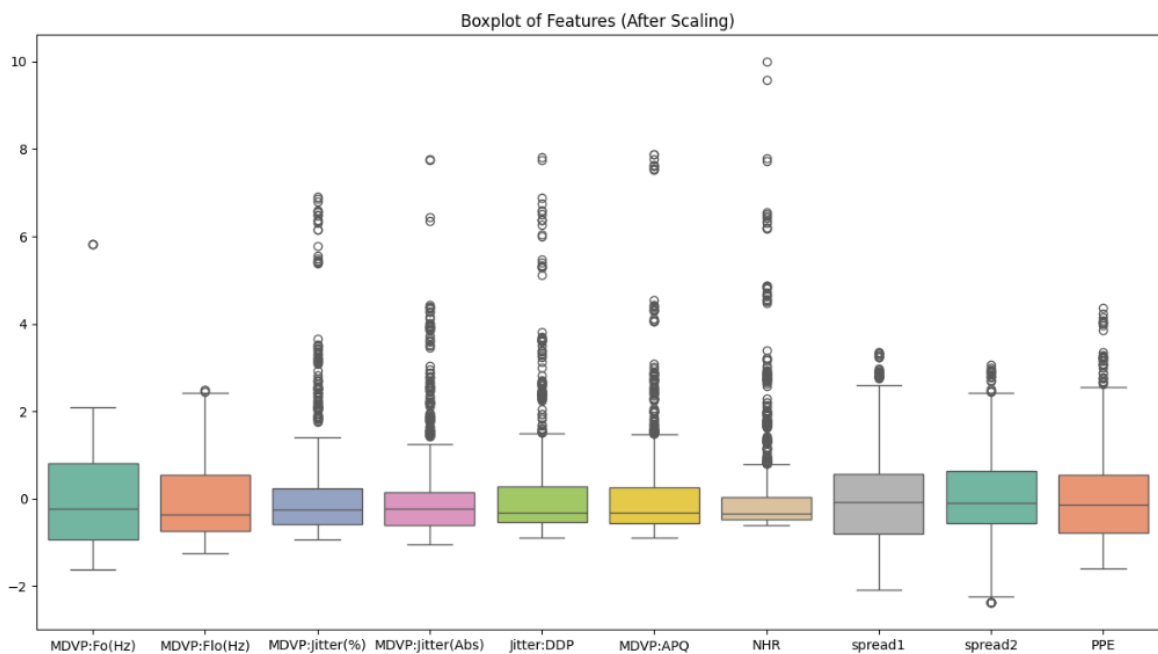
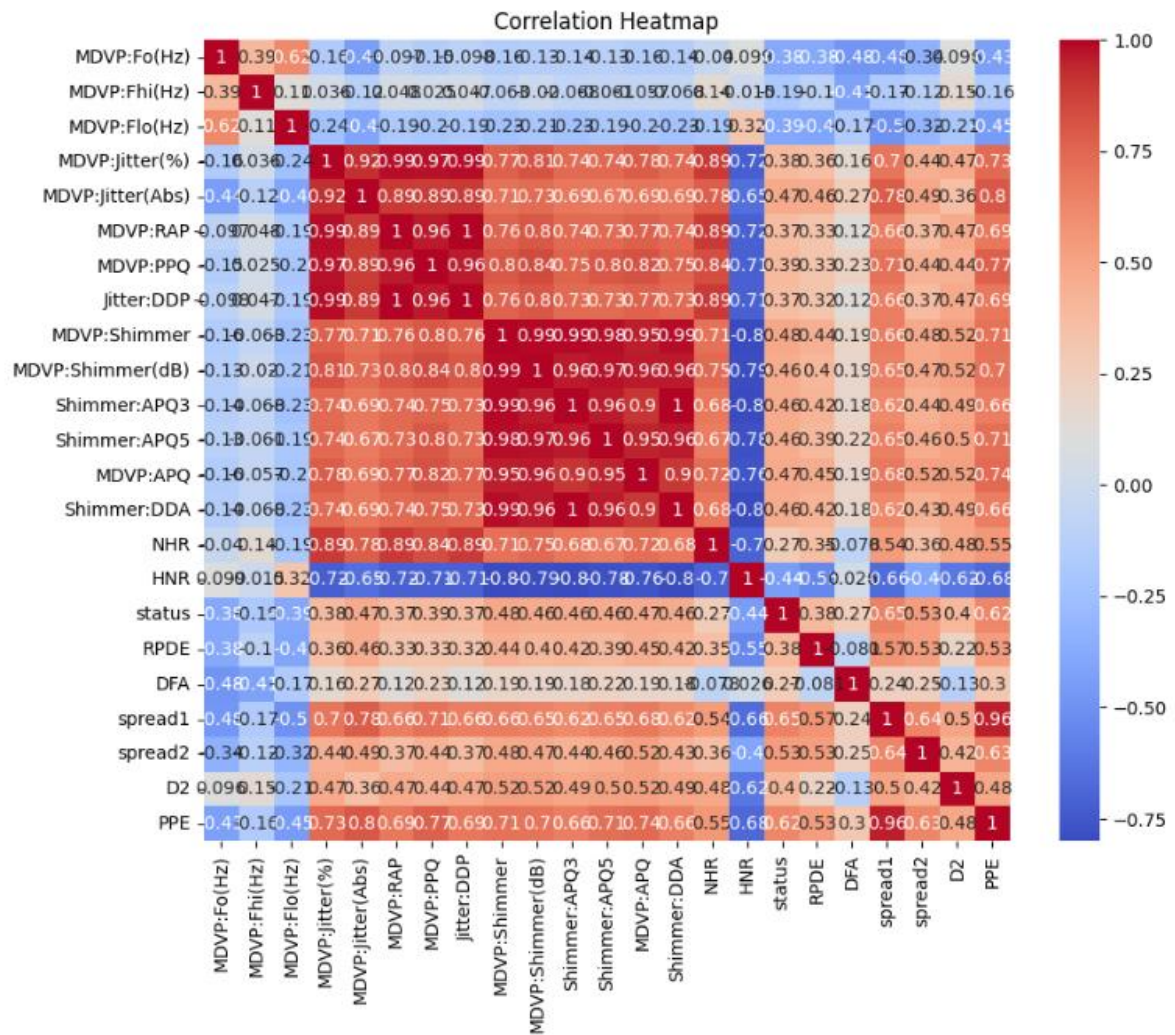
plt.tight_layout()
plt.show()
```

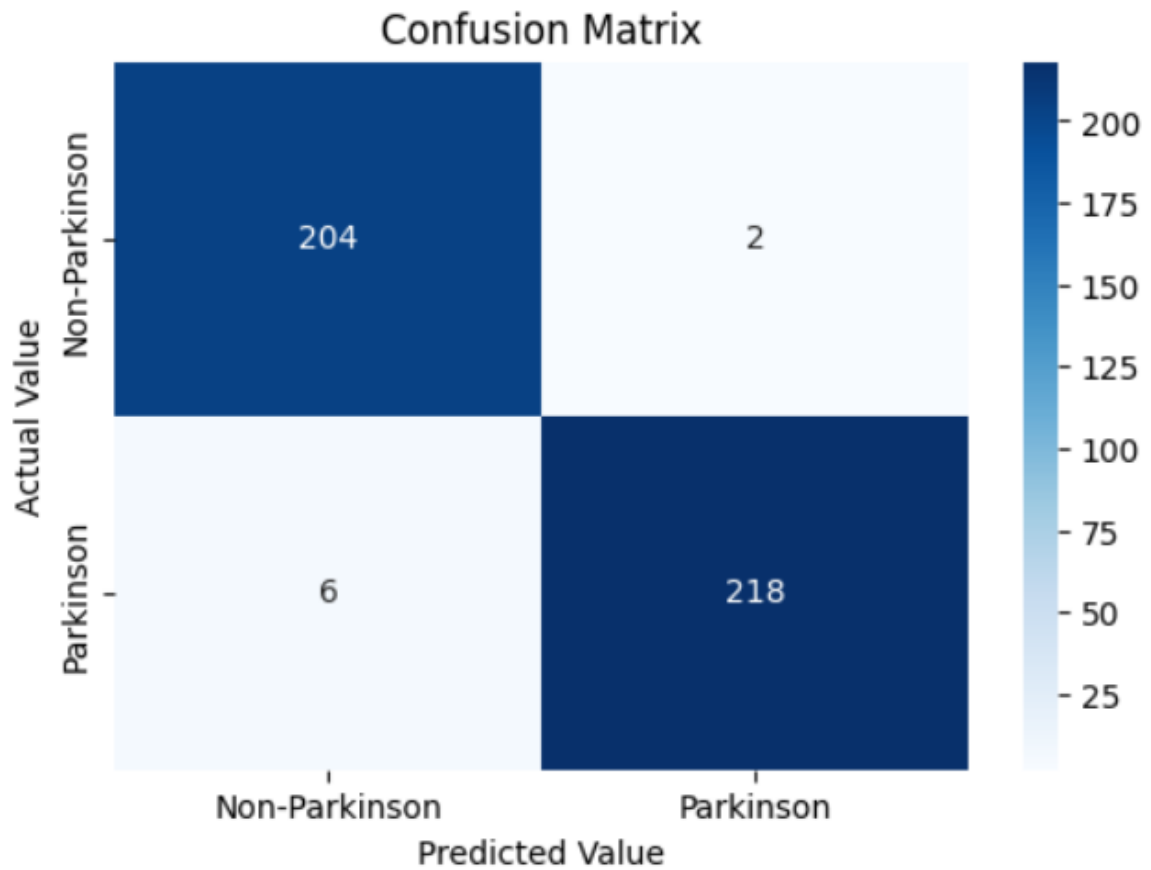












```

app.py 3 x
app.py > ...
1 import streamlit as st
2 import numpy as np
3 import joblib
4
5 st.set_page_config(page_title="Parkinson's Disease Predictor", layout="centered")
6
7 model = joblib.load('model.joblib')
8 scaler = joblib.load('scaler.joblib')
9
10 st.title("Parkinson's Disease Prediction App")
11
12 st.markdown("""
13 This app predicts that the person have **Parkinson's Disease** based on vocal measurement features.
14 Please enter the following values carefully.
15 """)
16 feature_info = {
17     'MDVP:F0(Hz)': 'Average vocal frequency (e.g., 120 to 180 Hz)',
18     'MDVP:F1(Hz)': 'Minimum vocal frequency (e.g., 70 to 130 Hz)',
19     'MDVP:jitter(%)': 'Frequency variation (e.g., 0.001-0.01)',
20     'MDVP:jitter(Abs)': 'Absolute jitter (e.g., 0.00001-0.0001)',
21     'jitter:DDP': 'Difference of differences of periods (e.g., 0.00001-0.002)',
22     'MDVP:APQ': 'Amplitude perturbation quotient (e.g., 0.001-0.02)',
23     'NHR': 'Noise-to-harmonics ratio (e.g., 0.01-0.2)',
24     'spread1': 'Nonlinear signal measure (e.g., -7 to 0)',
25     'spread2': 'Nonlinear signal measure (e.g., 0 to 3)',
26     'PPE': 'Pitch period entropy (e.g., 0.1-1.0)'
27 }
28 user_input = {}
29
30 features = list(feature_info.keys())
31 for i in range(0, len(features), 2):
32     cols = st.columns(2)
33     for j in range(2):
34         if i + j < len(features):
35             feature = features[i + j]
36             with cols[j]:

```

```

19 'MDVP:Jitter(X) : Frequency variation (e.g., 0.001-0.01)',
20 'MDVP:Jitter(Abs)': 'Absolute jitter (e.g., 0.00001-0.0001)',
21 'Jitter:DDP': 'Difference of differences of periods (e.g., 0.00001-0.002)',
22 'MDVP:APQ': 'Amplitude perturbation quotient (e.g., 0.001-0.02)',
23 'HNR': 'Noise-to-harmonics ratio (e.g., 0.01-0.2)',
24 'spread1': 'Nonlinear signal measure (e.g., -7 to 0)',
25 'spread2': 'Nonlinear signal measure (e.g., 0 to 3)',
26 'PPE': 'Pitch period entropy (e.g., 0.1-1.0)'
27
28 user_input = []
29
30 features = list(feature_info.keys())
31 for i in range(0, len(features), 2):
32     cols = st.columns(2)
33     for j in range(2):
34         if i + j < len(features):
35             feature = features[i + j]
36             with cols[j]:
37                 value = st.text_input(
38                     f"{feature}",
39                     placeholder=f"({feature_info[feature]})",
40                     help=feature_info[feature]
41                 )
42             user_input.append(value)
43 if st.button("👉 Predict"):
44     try:
45         input_array = np.array(user_input, dtype=float).reshape(1, -1)
46         input_scaled = scaler.transform(input_array)
47         prediction = model.predict(input_scaled)
48
49         st.subheader("👉 Prediction Result")
50         if prediction[0] == 1:
51             st.error("⚠️ The person have Parkinson's Disease.")
52         else:
53             st.success("✅ The person does not have Parkinson Disease.")
54     except ValueError:
55         st.warning("⚠️ Please enter valid numeric values for all features.")

```

Chapter – 5 Result and Discussion

5.1 Result/ Outcome

The main goal of this project was to build a machine learning model to accurately predict whether a person is likely to have Parkinson's Disease based on voice measurements.

After completing the entire pipeline, the following outputs were obtained:

Model Performance:

- Support Vector Machine (Tuned SVM):
Train Accuracy: ~98%
Test Accuracy: ~98%
Evaluation Metrics: High precision, recall, and F1-score across both classes (0 and 1), confirming balanced performance.
Best Parameters: C = 1, Kernel = rbf (determined through GridSearchCV)
- Logistic Regression:
Train Accuracy: ~85%
Test Accuracy: ~82%
Lower accuracy than SVM but still show good performance.

Evaluation Metrics (for Tuned SVM):

Accuracy: 98%
Precision: 97-99%
Recall: 98-99%

F1-Score: 98%

Confusion Matrix: Clearly shows low false positives and false negatives.

Visualizations: Heatmaps confirm selected features correlate well with the target variable.

5.2 Challenges Faced

- **Data cleaning due to mixed data types in some columns:** Used `pd.to_numeric(errors='coerce')` to handle errors and cleaned the dataset.
- **Class imbalance concerns:** Used stratified train-test split to ensure class distribution is maintained.
- **Hyperparameter tuning:** Used GridSearchCV for optimal parameter selection in SVM.
- **Deployment interface clarity for end-user:** Provided clear descriptions and placeholders for user inputs in Streamlit.

5.3 Learning

Through this project, the following key learnings were achieved:

- **Data Understanding:** Learned how biomedical datasets (voice measurements) relate to healthcare problems like Parkinson's detection.
- **Data Preprocessing Techniques:** Learned the importance of cleaning, scaling, and selecting relevant features for model performance.
- **Model Comparison:** Understood the performance differences between models like Logistic Regression and SVM when applied to real dataset.
- **Hyperparameter Tuning:** Gained practical experience with GridSearchCV for optimizing machine learning models.
- **Visualization for Insights:** Learned how to leverage heatmaps, boxplots, and confusion matrices to validate feature importance and model performance.
- **Deployment:** Gained practical experience in building and deploying a user-friendly ML web application using Streamlit.
- **End-to-End ML Workflow:** Understood the complete lifecycle of a machine learning project, from raw data to deployment.

5.4 Screenshots

Parkinson's Disease Prediction App

This app predicts that the person have Parkinson's Disease based on vocal measurement features. Please enter the following values carefully.

MDVP:F0(Hz)	MDVP:F1(Hz)
226.15114925328	96.5185802846161
MDVP:Jitter(%)	MDVP:Jitter(Abs)
0.00262472311262024	9.90685039831892E-06
Jitter:DDP	MDVP:APQ
0.00453759463995898	0.0133997284536801
NHR	spread1
0.00298930753397342	-7.32067941856859
spread2	PPE
0.0850210073979103	0.0962955262847182

Predict

Prediction Result

The person does not have Parkinson Disease.

Parkinson's Disease Prediction App

This app predicts that the person have Parkinson's Disease based on vocal measurement features. Please enter the following values carefully.

MDVP:F0(Hz)	MDVP:F1(Hz)
127.902453795334	99.9061184487294
MDVP:Jitter(%)	MDVP:Jitter(Abs)
0.0190947817771637	0.000148359257950624
Jitter:DDP	MDVP:APQ
0.034111400902049	0.0597630540334234
NHR	spread1
0.169435406097138	-3.88371223469172
spread2	PPE
0.242680382307565	0.383812140051197

Predict

Prediction Result

The person have Parkinson's Disease.

Chapter – 6 Conclusion

6.1 Summary

This project aimed to develop a machine learning-based system for the early detection of Parkinson's Disease using biomedical voice measurements. The primary objective was to create a reliable and accurate model that can assist healthcare professionals in identifying potential Parkinson's cases through easily measurable voice parameters.

The project workflow included data cleaning, preprocessing, feature selection, model building, hyperparameter tuning, and evaluation using various machine learning techniques. Specifically, Support Vector Machine (SVM) with hyperparameter tuning emerged as the best-performing model, achieving approximately 98% accuracy. Additionally, Logistic Regression was used for comparison to validate the performance further.

Visual tools like correlation heatmaps, pair plot, boxplot and confusion matrices provided valuable insights into the dataset, finding relationship between features and model predictions. The project was successfully deployed using Streamlit, offering an interactive interface where users can input specific voice feature values and receive immediate prediction results.

Overall, the project successfully demonstrated how machine learning can be leveraged in healthcare for disease prediction, providing a valuable tool to assist in the early diagnosis of Parkinson's Disease.