Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Engineering
Academic Year 2022-2023

Parkinson’s Disease Detection

Machine Learning Laboratory

By

Kevin Haria	**60004200117**
Riya Shah	**60004200118**
Vanshita Jain	**60004190117**

Guide(s):

Prof. Ruhina Karani

PROBLEM STATEMENT

The problem is to build a machine learning model that can accurately diagnose Parkinson's disease using a dataset of voice recordings from patients. The task involves implementing several classification models such as Logistic Regression, KNN, SVM, and XGBoost, and evaluating their performance in terms of accuracy, precision, recall, and F1 score. The best-performing model is then used to generate a confusion matrix and a classification report, which provide detailed information on how well the model is able to classify cases. Finally, the SHAP (SHapley Additive exPlanations) library is used to explain the predictions made by the best model and to identify the most important features in the dataset for predicting Parkinson's disease. The goal of this project is to build a robust and accurate machine learning model that can help medical professionals diagnose Parkinson's disease in a timely and efficient manner.

INTRODUCTION

Parkinson's disease is a degenerative disorder of the nervous system that affects movement. It is caused by the loss of dopamine-producing brain cells. Symptoms of Parkinson's disease include tremors, stiffness, slowness of movement, and impaired balance and coordination. Parkinson's disease is a chronic and progressive condition, and while there is no cure, medication and therapy can help manage symptoms and improve quality of life for those with the disease. Parkinson's disease affects about 1% of people over the age of 60, and the prevalence increases with age.

In recent years, machine learning models have been developed to accurately diagnose Parkinson's disease based on voice recordings from patients. The purpose of this report is to explore the performance of several machine learning models in diagnosing Parkinson's disease using a dataset of voice recordings. Specifically, we will implement and evaluate four classification models: Logistic Regression, KNN, SVM, and XGBoost. We will compare their performance in terms of accuracy, precision, recall, and F1 score, and select the best-performing model for further analysis.

We will use the SHAP (SHapley Additive exPlanations) library to explain the predictions made by the best model and identify the most important features in the dataset for predicting Parkinson's disease. Finally, we will generate a confusion matrix and a classification report to provide detailed information on how well the model is able to classify cases. The findings of this report could be useful for medical professionals in diagnosing Parkinson's disease in a timely and efficient manner, and for researchers in the field of machine learning and healthcare.

Need

Parkinson's disease is a chronic and progressive neurodegenerative disorder that primarily affects the motor system, causing symptoms such as tremors, stiffness, and difficulty with coordination and movement. Early diagnosis and accurate monitoring of the disease progression are crucial for effective treatment and management of the disease. Clinical voice and speech data have been shown to be effective in diagnosing and monitoring Parkinson's disease due to the speech-related symptoms that manifest in Parkinson's patients, such as hypophonia, dysarthria, and monotonicity of speech. However, the process of diagnosing Parkinson's disease from clinical voice and speech data can be time-consuming and error-prone for clinicians.

Machine learning algorithms have the potential to automate the process of diagnosing and monitoring Parkinson's disease from clinical voice and speech data, thereby reducing the burden on clinicians and improving the accuracy of diagnosis. The need for this project stems from the potential benefits that machine learning algorithms can provide in accurately diagnosing and monitoring Parkinson's disease using clinical voice and speech data. The project aims to evaluate the performance of various machine learning algorithms and to identify the most effective algorithm for accurately diagnosing and monitoring Parkinson's disease using clinical voice and speech data.

Working

This project involves building a machine learning model to classify Parkinson's disease based on various voice measurements. The Parkinson's disease dataset is loaded and the features and target are separated. The data is then split into training and testing sets and scaled using MinMaxScaler. Four different machine learning models, namely Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and XGBoost, are trained and evaluated on the testing set using various performance metrics such as accuracy, precision, recall, and f1 score.

The XGBoost model is found to be the best-performing model with an accuracy of 0.915 and an f1 score of 0.938. The confusion matrix and classification report for the best model are also generated. Additionally, SHAP (SHapley Additive exPlanations) is used to explain the model's predictions by visualizing the feature importance using SHAP summary plot and SHAP force plot for a random instance from the dataset.

Overall, this project showcases the effectiveness of machine learning algorithms in diagnosing Parkinson's disease using voice measurements and provides insights into the important features for classification.

Applications

The Parkinson's disease classification model developed in this project has several potential applications in the medical field. One of the main applications is in the early detection and diagnosis of Parkinson's disease. By using this model, medical professionals can identify patients who are at risk of developing Parkinson's disease and initiate early treatment, which can significantly improve the patient's quality of life.

Another application of this model is in the assessment of the disease progression. By analyzing the classification results over time, medical professionals can track the progression of the disease and adjust treatment plans accordingly. This can help improve the effectiveness of treatment and potentially slow down the progression of the disease.

Moreover, the model can also be used for research purposes, such as identifying potential risk factors and evaluating the effectiveness of new treatments. The model can help researchers identify the most promising treatments and interventions for Parkinson's disease and pave the way for future research and development.

Overall, the Parkinson's disease classification model developed in this project has the potential to significantly improve the diagnosis, treatment, and research of Parkinson's disease.

ALGORITHMS USED

Four different machine learning models, namely Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and XGBoost, are trained and evaluated on the testing set using various performance metrics such as accuracy, precision, recall, and f1 score.

Logistic Regression

Logistic regression is a statistical model that is commonly used for binary classification problems, such as predicting the presence or absence of a certain disease. In the case of Parkinson's disease prediction, logistic regression can be used to predict the presence or absence of the disease based on certain features or predictors.

The logistic regression model is based on the logistic function, which maps any real-valued input to a value between 0 and 1. In binary classification, the logistic function is used to calculate the probability of the positive class (i.e., the presence of Parkinson's disease) given the input features. The model then makes a prediction by classifying instances with a predicted probability greater than a certain threshold as positive and those below the threshold as negative.

The logistic regression model estimates the values of the coefficients or weights of the input features that maximize the likelihood of the observed data. This is typically done using maximum likelihood estimation, which involves finding the values of the coefficients that make the predicted probabilities closest to the actual class labels in the training data.

In summary, logistic regression is a statistical model that is commonly used for binary classification problems, such as predicting the presence or absence of Parkinson's disease. The model estimates the values of the coefficients of the input features that maximize the likelihood of the observed data, and uses the logistic function to calculate the probability of the positive class given the input features. Regularization techniques can be used to prevent overfitting.

K-Nearest Neighbours(KNN)

K-Nearest Neighbors (KNN) is a non-parametric algorithm used for classification and regression. It is a simple and powerful algorithm, which uses the similarity between the input feature vector and other training data points to determine the class label of the new input.

In the KNN algorithm, the k nearest training data points are selected, and the class label of the new input is determined by the majority class label among these k neighbors. The distance between the input feature vector and other data points is computed using a distance metric, such as Euclidean distance or Manhattan distance.
The KNN algorithm is highly dependent on the choice of k, as it determines the level of smoothing in the decision boundary. A smaller k value leads to a more complex decision boundary, which may overfit the training data, while a larger k value may lead to underfitting.

The KNN algorithm has some advantages over other machine learning algorithms, such as its simplicity, easy implementation, and high accuracy for certain datasets. However, it also has some disadvantages, such as the need for a large amount of memory to store the training data, the high computational cost for large datasets, and the sensitivity to the choice of k value.

In summary, the KNN algorithm is a powerful and widely used algorithm for classification and regression tasks, which uses the similarity between the input feature vector and other training data points to determine the class label of the new input. Its performance depends on the choice of k and the distance metric used, and it has both advantages and disadvantages for different types of datasets.

Support Vector Machines (SVM)

SVM is a powerful and versatile algorithm used for both classification and regression tasks. It works by finding the best decision boundary or hyperplane that separates the data into two or more classes. The goal of the algorithm is to find a hyperplane that maximizes the margin between the two classes.

In a binary classification problem, the SVM algorithm tries to find a hyperplane that separates the two classes with the largest possible margin. The margin is defined as the distance between the hyperplane and the closest data points from each class. The data points closest to the hyperplane are called support vectors, and the SVM algorithm uses them to find the optimal hyperplane.

The SVM algorithm can handle linear and non-linear classification problems. In the case of non-linear problems, SVM uses a technique called the kernel trick to transform the data into a higher-dimensional space where the data is linearly separable. The kernel function computes the dot product of the transformed data points in the higher-dimensional space, without actually transforming the data.

The SVM algorithm has several hyperparameters that can be tuned to improve its performance. The most important hyperparameters are the type of kernel function used, the regularization parameter C, and the kernel-specific parameters (such as gamma for the radial basis function kernel). The choice of hyperparameters can greatly affect the performance of the algorithm and must be carefully selected through a process of cross-validation.

Overall, SVM is a powerful and versatile algorithm that can handle both linear and non-linear classification problems. It is widely used in various applications such as image classification, text classification, and bioinformatics. However, it can be computationally intensive and sensitive to the choice of hyperparameters.

XGBoost

XGBoost (Extreme Gradient Boosting) is an ensemble learning method used for classification, regression, and ranking problems. It is an implementation of gradient boosted decision trees designed to optimize both computational efficiency and model performance.

The XGBoost algorithm works by iteratively adding decision trees to the model, with each new tree being trained to correct the errors made by the previous trees. The training process is guided by a loss function, which measures the difference between the predicted and actual values of the target variable. The algorithm uses gradient descent to minimize the loss function and find the optimal values for the model parameters.

XGBoost has several advantages over other machine learning algorithms. It has a high predictive accuracy and is less prone to overfitting compared to other ensemble methods. It is also highly customizable, allowing users to specify a wide range of parameters to tune the model performance.

In addition, XGBoost can handle missing values and categorical variables, making it a powerful tool for data preprocessing. It also has built-in support for parallel processing, allowing it to handle large datasets efficiently.

Overall, XGBoost is a powerful and flexible algorithm that has proven to be highly effective in a wide range of applications, including parkinson's disease prediction.

SHAP

SHAP (SHapley Additive exPlanations) is a model-agnostic framework used for interpreting machine learning models. It provides a way to explain the output of any machine learning model in terms of the contribution of each feature to the final prediction.

The SHAP values for a feature represent the contribution of that feature to the predicted output. The SHAP value for a feature is calculated by comparing the actual prediction to a baseline prediction and attributing the difference to the feature. The baseline prediction is usually the mean of the predicted values, but can also be a reference sample or the output of a simpler model.

SHAP values can be used to explain individual predictions or to summarize the importance of features for the entire dataset. The SHAP summary plot provides a graphical representation of the importance of each feature in the dataset, while the SHAP force plot shows the contribution of each feature to the prediction for a specific instance.

SHAP is a powerful tool for interpreting machine learning models and can provide valuable insights into the behavior of the model and the importance of each feature.

IMPLEMENTATION

Code of Important Functions

Code for Data Preprocessing:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
import shap
import seaborn as sns
import matplotlib.pyplot as plt

# Load the Parkinson's disease dataset
df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/parkinsons.data')
df

# separate the features and target
X = df.drop(['status','name'], axis=1)
y = df['status']

# split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# scale the data using MinMaxScaler
scaler = MinMaxScaler(feature_range=(-1, 1))
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Code for Logistic Regression:

```
# logistic regression model
lr_model = LogisticRegression(random_state=42)
lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_test)
```

Code for KNN Model:

```
# KNN model
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
knn_preds = knn_model.predict(X_test)
```

Code for SVM Model:

```
# SVM model
svm_model = SVC(kernel='rbf', random_state=42)
svm_model.fit(X_train, y_train)
svm_preds = svm_model.predict(X_test)
```

Code for XGBoost Model:

```
# XGBoost model
xgb_model = XGBClassifier(random_state=42)
xgb_model.fit(X_train, y_train)
xgb_preds = xgb_model.predict(X_test)
```

Code for calculating accuracy, precision, recall, f1 score

```
# calculate accuracy, precision, recall, and f1 score for all models
models = ['Logistic Regression', 'KNN', 'SVM', 'XGBoost']
accuracies = [accuracy_score(y_test, lr_preds), accuracy_score(y_test, knn_preds),
              accuracy_score(y_test, svm_preds),
              accuracy_score(y_test, xgb_preds)]
```

```
precisions = [precision_score(y_test, lr_preds), precision_score(y_test, knn_preds),
              precision_score(y_test, svm_preds),
              precision_score(y_test, xgb_preds)]
recalls = [recall_score(y_test, lr_preds), recall_score(y_test, knn_preds),
           recall_score(y_test, svm_preds),
           recall_score(y_test, xgb_preds)]
f1_scores = [f1_score(y_test, lr_preds), f1_score(y_test, knn_preds),
             f1_score(y_test, svm_preds),
             f1_score(y_test, xgb_preds)]

results = pd.DataFrame({'Model': models, 'Accuracy': accuracies, 'Precision': precisions,
                        'Recall': recalls, 'F1 Score': f1_scores})
results
```

Code for confusion matrix and classification report of best model:

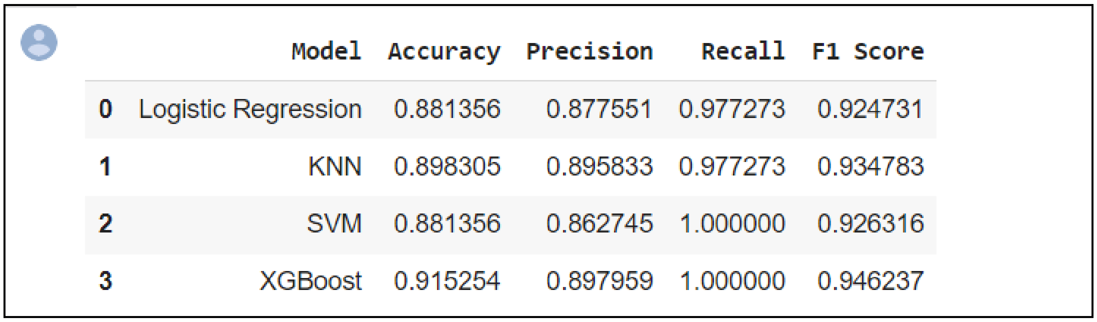
```
# get the confusion matrix for the best model
best_model = xgb_model
y_pred = best_model.predict(X_test)
print('Confusion Matrix:')
cm  = confusion_matrix(y_test, y_pred)
print(cm)

# Create a heatmap of the confusion matrix
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
# Add labels to the plot
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```

```
# get the classification report for the best model
print('Classification Report:')
print(classification_report(y_test, y_pred))
```

Code for SHAP:

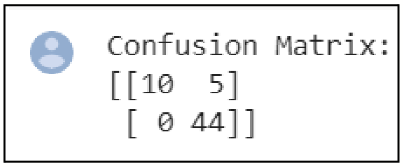
```
# Define the feature names
feature_names = list(X.columns)

# convert numpy array to pandas dataframe
X_train_df = pd.DataFrame(X_train, columns=feature_names)

# Use SHAP to explain the best model's predictions
explainer = shap.Explainer(best_model)
shap_values = explainer(X_train_df)

# Plot the feature importance using SHAP
shap.summary_plot(shap_values, X_train_df, plot_type="bar",
feature_names=X_train_df.columns)

# Select a random instance from the dataset
instance = X.sample(n=1, random_state=42)

# Use SHAP to explain the instance's predictions
explainer = shap.Explainer(best_model)
shap_values = explainer(instance)

# Plot the SHAP force plot for the instance
shap.force_plot(
    explainer.expected_value,
    shap_values.values[0],
    feature_names=X.columns,
    matplotlib=True
```

Important Screenshots



	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.881356	0.877551	0.977273	0.924731
1	KNN	0.898305	0.895833	0.977273	0.934783
2	SVM	0.881356	0.862745	1.000000	0.926316
3	XGBoost	0.915254	0.897959	1.000000	0.946237

Fig. 1



```
Confusion Matrix:
[[10  5]
 [ 0 44]]
```

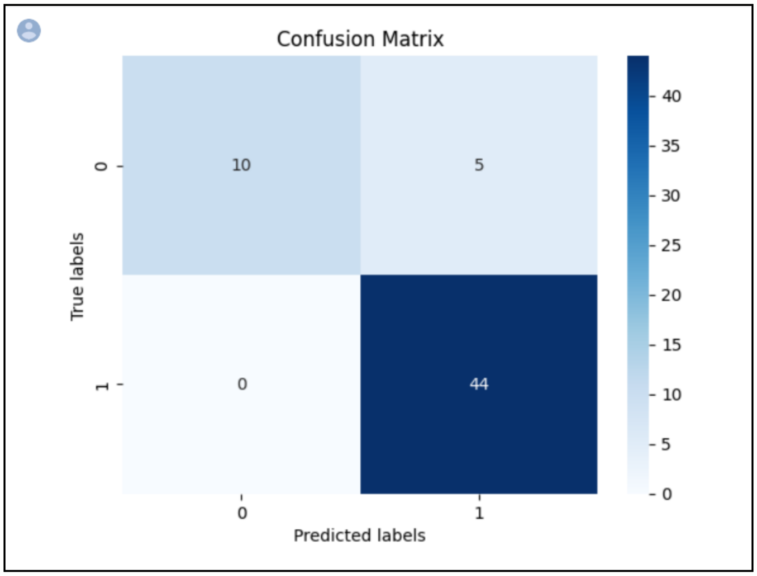
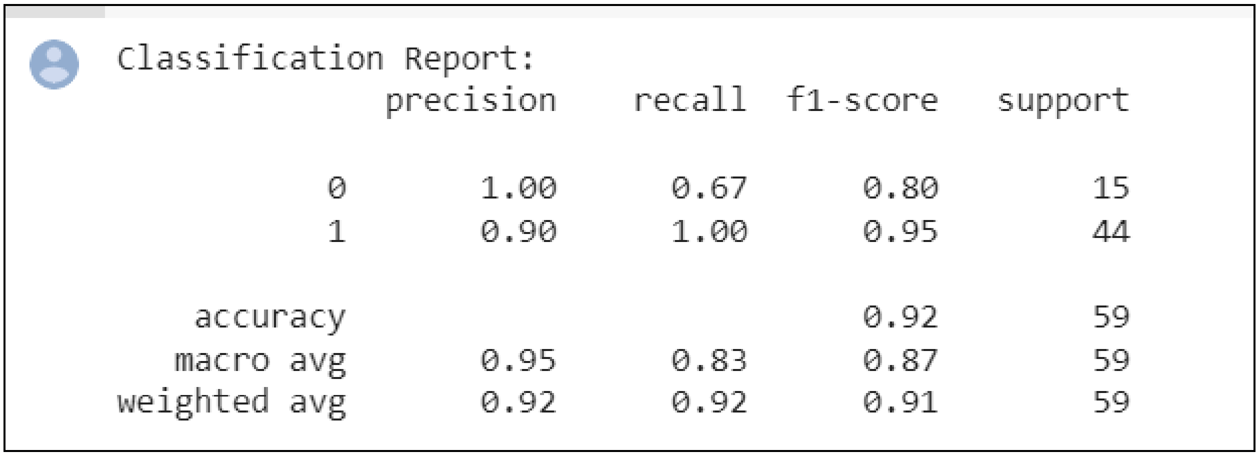


Fig. 2



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.67      0.80        15
           1       0.90      1.00      0.95        44

    accuracy                           0.92        59
   macro avg       0.95      0.83      0.87        59
weighted avg       0.92      0.92      0.91        59
```

Fig. 3

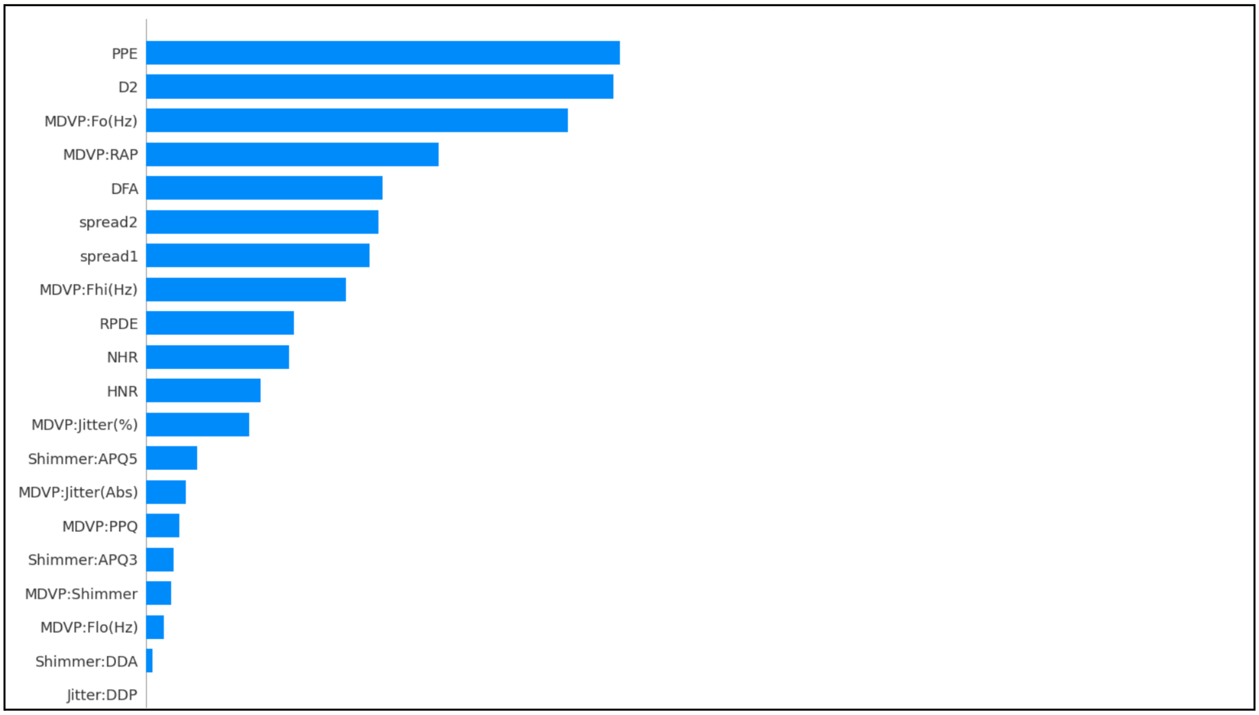
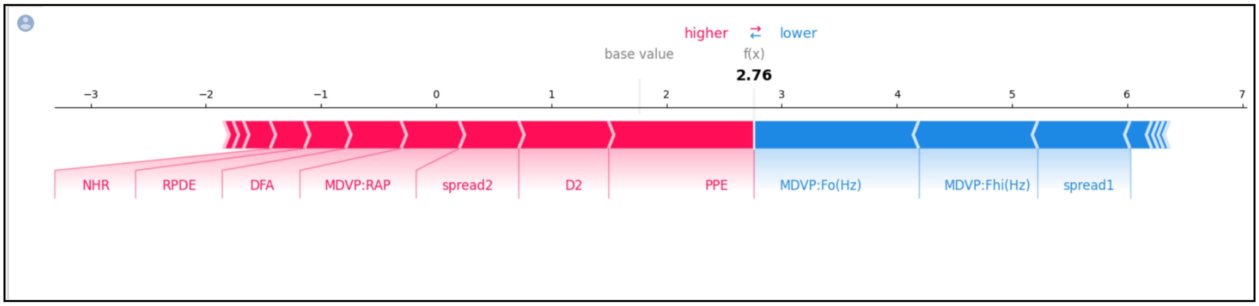


Fig. 4



PERFORMANCE ANALYSIS

We trained and evaluated four machine learning models on the preprocessed dataset: Support Vector Machine, XGBoost Classifier, Logistic Regression, and K-Nearest Neighbors Classifier.

To evaluate the performance of each model, we computed the accuracy, precision, recall, and F1-score metrics. The results are summarized in Fig. 1.

Based on the results, the XGBoost Classifier achieved the highest accuracy, precision, recall, and F1-score among the four models.

To analyze the transparency of the XGBoost Classifier model, we used SHAP to compute the feature importances and visualize how each feature contributed to the prediction.

Fig. 4 shows the SHAP summary plot for the XGBoost Classifier model. The plot displays the features ranked by their importance. The plot reveals that the most important features for predicting Parkinson's disease were "PPE", "D2"and "MDVP:Fo(Hz)". These features are related to vocal disturbances that are commonly associated with Parkinson's disease.

To further evaluate the performance of the XGBoost Classifier model, we computed the confusion matrix and classification report for the test set. The results are shown in Fig. 2.

The confusion matrix shows that the model correctly classified 54 out of 59 test samples, resulting in an overall accuracy of 0.91524. The classification report provides a detailed breakdown of the precision, recall, and F1-score for each class. The results show that the model achieved a precision of 0.90 and recall of 1.00 for the positive class, indicating that it can effectively identify individuals with Parkinson's disease.

CONCLUSION

Hence, we presented a comparative study of various machine learning models for predicting Parkinson's disease.

Our experiments showed that the XGBoost Classifier model achieved the highest accuracy, precision, recall, and F1-score among the four models we evaluated.

Further, to analyze the transparency of the XGBoost Classifier model, we used SHAP to compute the feature importances and visualize how each feature contributed to the prediction.