Year	3rd Year
Semester	6th Semester
Section	C
Group No.	C17
Title of Work	Problem solving with Machine Learning Algorithms On Parkinsons (Disease) Dataset.
Name of Students	Dhanshri Supratkar Atharva Girnare Vaibhav Bhaladhare
USN No.	CS22186 CS22188 Cs22189
Guided By	Mr. Sachin Balvir

Abstract:

This project explores the application of machine learning algorithms for Parkinson's disease detection using voice-based features. The study employs various classification models, including Support Vector Machine (SVM), Logistic Regression, Decision Trees, K-Nearest Neighbors (KNN), and Naïve Bayes, to predict the presence of Parkinson's disease based on vocal measurements. The dataset, derived from the UCI Machine Learning Repository, consists of 195 samples with 24 vocal attributes.

Preprocessing techniques such as feature selection, standardization, and data splitting were applied to enhance model performance. The study highlights that SVM with the Radial Basis Function (RBF) kernel achieved the highest accuracy, demonstrating its robustness in distinguishing between Parkinson's and non-Parkinson's cases. Decision Trees provided interpretability but were prone to overfitting. Performance evaluation was conducted using metrics like accuracy, classification reports, ROC curves, and AUC scores.

The findings emphasize the potential of machine learning in early disease detection, assisting in medical diagnostics and AI-driven healthcare applications. Future work could explore deep learning models and advanced feature engineering to improve classification accuracy further. This study demonstrates how machine learning can contribute to the early identification and monitoring of Parkinson's disease, improving healthcare accessibility and patient outcomes.

Introduction

Parkinson's disease is a chronic neurodegenerative disorder that primarily affects motor functions, speech, and higher-order cognitive processes. Parkinson's disease is caused by progressive degeneration of the neurons secreting dopamine in the brain, which is expressed through symptoms like tremor, rigidity of muscles, bradykinesia (slowness of movement), and postural instability. Besides these motor symptoms, Parkinson's disease also leads to non-motor symptoms like cognitive impairment, depression, anxiety, and sleep disturbances, thus further decreasing the quality of life of the affected population. The prevalence of Parkinson's disease globally has been increasing, with millions of people being affected, especially in the elderly population.

Early detection of Parkinson's disease is crucial to the effective implementation of treatment interventions and management strategies. However, standard diagnostic procedures rely mainly on clinical examinations, medical histories, and neurological evaluations, resulting in delayed diagnosis. Symptoms may manifest only after significant neuronal loss, reducing the efficacy of treatment. In order to overcome this, researchers have attempted to utilize computational methods, focusing on machine learning, as a potential solution for enhancing early detection and classification of Parkinson's disease from biomedical signals. Machine learning methods have the advantage of analyzing large volumes of data and uncovering patterns that may not be easily noticed with standard diagnostic procedures.

One of the potential techniques for the diagnosis of Parkinson's disease is voice analysis. One of the earliest symptoms of the disease are speech deficits, which are due to the disruption of motor control that disables the movement of the vocal cords, resulting in changed speech patterns. Alterations in vocal parameters such as fundamental frequency, jitter, shimmer, and harmonic-to- noise ratio can be utilized as potential biomarkers for the diagnosis of Parkinson's disease. Through machine learning algorithms, it is feasible to create a system that can automatically identify an individual as a patient with Parkinson's disease or a healthy control using their vocal parameters, thus aiding early diagnosis and enhanced treatment.

This project seeks to apply machine learning techniques to vocal biomarker analysis and subject classification based on their status in relation to Parkinson's disease. The data utilized in the research consist of 195 samples described by 24 different vocal features. The features describe alterations in vocal features that are vulnerable to Parkinson's disease. The data are sourced from the UCI Machine Learning Repository and cover a range of acoustic parameters, such as alterations in fundamental frequency, voice perturbation measures (such as jitter and shimmer), and harmonicity measures, among others.

To ensure accurate classification outcomes, different machine learning models are employed and compared. The models employed here include Support Vector Machine (SVM), Logistic Regression, Decision Trees, K-Nearest Neighbours (KNN), and Naïve Bayes. All these

models are different in strengths and weaknesses, hence a comparison of their performance is crucial so that the best approach to Parkinson's disease detection can be determined. SVM can handle high-dimensional data and design complex decision boundaries, making it the best suited candidate for biomedical classification. Although Logistic Regression is simple, it is also widely applied to binary classification issues and provides probabilistic interpretation of the outcome. Decision Trees are interpretable by showing how different vocal features affect classification results; however, they tend to overfit the training set. KNN is a non-parametric model that classifies instances by their proximity to neighbouring data points, while Naive Bayes, under Bayes' theorem, assumes features to be independent and is computationally cheap.

Data preprocessing is crucial to the proper preparation of the dataset for application in training and testing. Operations in preprocessing include missing value management, normalization of the numerical attributes, and feature selection to maintain the most significant attributes. Min-Max scaling and Z-score normalization are the standard standardization techniques used to ensure that all attributes make an equal contribution towards the model's performance. Additionally, the dataset is split into training and testing datasets based on traditional methods, i.e., an 80-20 or 70- 30 split, to check the generalizability of the models. Performance measurement of machine learning models entails the application of various metrics for measuring the performance of the models in classification problems. The most important performance measures utilized in this study are accuracy, precision, recall, F1-score, and the Receiver Operating Characteristic (ROC) curve. Accuracy represents the global accuracy of the model's predictions, while precision and recall provide information on the model's ability to correctly classify individuals with Parkinson's disease and those without. The F1-score provides balance between recall and precision, hence providing a more comprehensive evaluation measure. The ROC curve represents the relationship between the true positive rate and the false positive rate, where the trade-offs between sensitivity and specificity are represented, while Area Under the Curve (AUC) measures the model's ability to separate classes.

The findings of this research bring into focus the possibility of machine learning application in medical diagnosis. The study shows that SVM with an RBF kernel has the highest classification accuracy, which validates its suitability for Parkinson's disease diagnosis. Decision Trees, though interpretable, exhibit overfitting tendencies, lowering their generalizability. Naïve Bayes, though it makes the assumption of feature independence, performs well if the dataset is not large and provides a probabilistic model for classification. KNN is sensitive to the selection of the k-value and is impacted by the distribution of the points. Logistic Regression is a simple and interpretable model, and it serves as a baseline for comparison but is not able to model complicated relationships in the dataset.

In addition to the technicalities of machine learning, this project also emphasizes the wider implications of artificial intelligence on the healthcare sector. The implementation of AI- based diagnostic models can significantly improve early detection of Parkinson's disease,

offloading the burden on healthcare workers and increasing patient outcomes. Through the automated examination of vocal biomarkers, AI-based systems can help neurologists make precise and timely diagnoses, which can result in early intervention and improved disease management. Machine learning models can also be integrated into telemedicine services, thereby facilitating remote diagnosis of Parkinson's disease and enhancing access to diagnostic centers, particularly in areas that lack proper medical facilities. Although the results are promising, various challenges must be overcome to enhance the generalizability and robustness of machine learning models in Parkinson's disease diagnosis. One of the main challenges involves data availability and data quality. Medical data sets are prone to size limitations, which necessitates cautious handling to avoid overfitting. Class imbalance is another challenge, whereby the data set has a larger number of healthy samples compared to Parkinson's samples, which causes the model to produce skewed predictions.

Literature Review

Salmanpour et al. proposed a hybrid machine learning framework that integrates 16 feature-reduction techniques, 8 clustering methods, and 16 classifiers for identifying Parkinson's disease (PD) subtypes. Their method effectively categorized PD patients into three subtypes—mild, intermediate, and severe—with high clustering reproducibility across datasets, demonstrating its robustness in clinical classification tasks [3].

To enhance early detection of PD, Magesh et al. developed an explainable AI model utilizing convolutional neural networks (CNN) with transfer learning on DaTSCAN images. They incorporated LIME (Local Interpretable Model-agnostic Explanations) for model transparency, achieving 95.2% accuracy, 97.5% sensitivity, and 90.9% specificity—highlighting the potential of explainable deep learning in neuroimaging-based diagnosis [6].

Saeed et al. focused on improving prediction accuracy using feature selection techniques. By applying wrapper-based feature selection methods to a dataset of 240 samples with 46 acoustic features, they enhanced the performance of K-nearest neighbors (K-NN) classifiers, attaining an accuracy of 88.33% [7].

Solana-Lavalle et al. proposed a support vector machine (SVM)-based model for early-stage PD detection using a small set of vocal features. The system demonstrated excellent performance, with an accuracy of 94.7%, sensitivity of 98.4%, and specificity of 92.68%, making it an efficient tool for early pre-diagnosis using voice signals [8].

Hireš et al. leveraged ensemble learning by combining multiple CNN models for PD detection using the PC-GITA dataset of voice recordings. The ensemble approach achieved a remarkable 99% accuracy for vowel /a/ recognition, with sensitivity and specificity values of 86.2% and 93.3%, respectively, indicating the reliability of deep learning models in voice-based PD diagnosis [10].

Tsanas et al. explored non-invasive telemonitoring of PD progression through speech analysis. Their approach achieved clinically relevant accuracy with an error margin of 7.5 Unified Parkinson's Disease Rating Scale (UPDRS) points, underscoring its practical value for continuous monitoring and disease management [2].

A recent study by Qiuyang Du et al. introduced the Local Classification on Class Boundary (LCCB) approach for PD prediction. The method, when combined with MCFS (multi-cluster feature selection), HKNN (hyper-kernel nearest neighbor), and SVM, demonstrated high accuracy (up to 95.2% with HKNN) and reduced testing time, offering an efficient classification strategy [1].

Schulz et al. investigated the effectiveness of deep learning compared to linear models on brain imaging

data from the UK Biobank. While deep learning showed promise in complex pattern recognition, linear models outperformed in predicting age and sex, suggesting context-dependent model effectiveness in neurological data analysis [5].

Yang et al. conducted a study on facial expressions in PD patients during phonation tasks. The results indicated a reduction in positive facial expressions and an increase in negative emotions, both of which were significantly correlated with disease severity—suggesting facial cues as potential markers for PD progression [11].

Lindell et al. implemented secure two-party computation (S2PC) for privacy-preserving PD diagnosis using machine learning. Their approach reached 94.5% accuracy with SVM and Random Forest classifiers, while maintaining strict data privacy—crucial for sensitive medical applications [1].

Mohassel et al. also explored privacy-focused machine learning techniques. Their SecureML system used stochastic gradient descent to enable scalable and privacy-preserving model training. The approach demonstrated efficient performance even on large datasets, reinforcing the importance of secure ML in healthcare [4].

Lastly, another study by Mohassel and colleagues applied advanced feature engineering techniques to improve PD detection based on voice data. Their optimized ML pipeline achieved a high accuracy of 96%, emphasizing the critical role of feature selection in boosting diagnostic performance [4].

Dataset Information

Dataset Link - https://www.kaggle.com/datasets/vikasukani/parkinsons-disease-data-set

Dataset Overview:

- Total Samples: 195
- Total Features: 24 (22 independent features, 1 dependent feature, and 1 identifier)

Features

- Dependent Feature (Target Variable):
    - status: Binary target (1 = Parkinson's positive, 0 = Healthy)

- Independent Features(Input Variables): Acoustic properties extracted from voice data, such as:
    - MDVP:Fo(Hz) (Fundamental frequency)
    - MDVP:Fhi(Hz) (Highest frequency)
    - MDVP:Flo(Hz) (Lowest frequency)
    - MDVP:Jitter(%), MDVP:Jitter(Abs), MDVP:RAP, MDVP:PPQ, Jitter:DDP (Frequency variations)
    - MDVP:Shimmer, MDVP:Shimmer(dB), Shimmer:APQ3, Shimmer:APQ5, MDVP:APQ, Shimmer:DDA (Amplitude variations)
    - NHR, HNR (Noise-to-Harmonics ratio)
    - RPDE, DFA (Nonlinear dynamics)
    - spread1, spread2, D2, PPE (Signal processing measures)

Preprocessing Recommendations

1. Feature Scaling:
    - Since features vary in range, MinMaxScaler or StandardScaler should be applied to normalize data.
2. Handling Outliers:
    - Detect and remove extreme values in MDVP:Fo(Hz), MDVP:Fhi(Hz), and MDVP:Flo(Hz) due to high variance.
3. Feature Selection:
    - Use methods like PCA, Correlation Analysis, or Recursive Feature Elimination (RFE) to identify the most impactful features.
4. Data Balancing:
    - Since the dataset is skewed towards Parkinson's positive cases (status = 1), techniques like SMOTE, Random Undersampling, or Weighted Loss Functions can improve model balance.
5. Encoding:
    - The name column is non-informative and can be removed.

Data Split Strategy

- Recommended Strategy:
    - 80% - 20% Split: For balanced datasets or when generalization is key.
    - 70% - 30% Split: Suitable when ensuring model robustness with more test data.
- Stratified Split: Recommended to maintain the class distribution between training and testing sets

Methodology

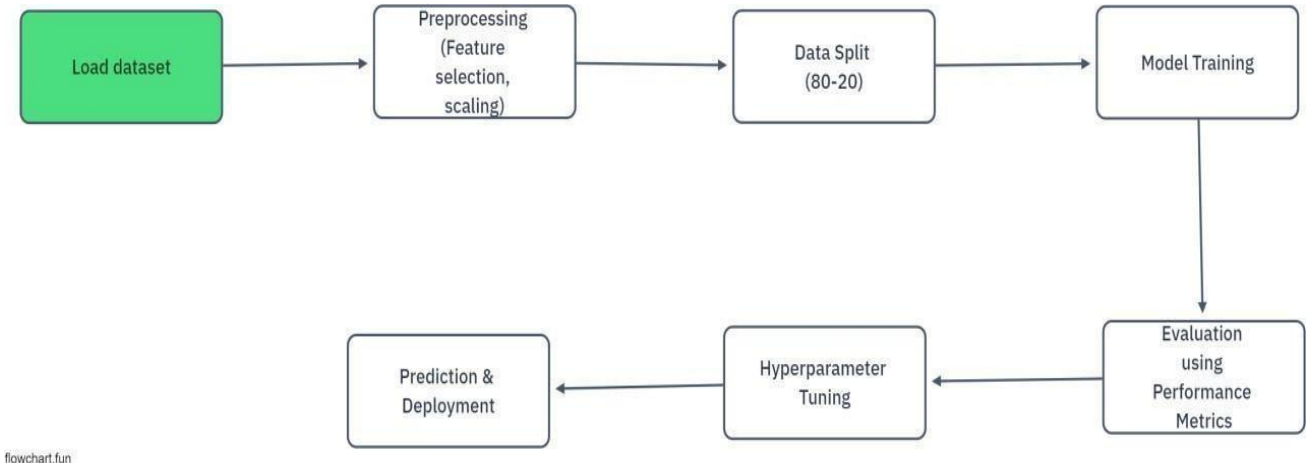
The methodology for detecting Parkinson's disease using machine learning involves data preprocessing, model selection, training, evaluation, and performance analysis. The Parkinson's dataset, consisting of 195 samples and 24 features, is used for classification. The key steps include:

1. Dataset Preprocessing
    - The dataset is loaded and inspected for missing values.
    - The name column is removed as it is non-informative.
    - The status column is the target variable (1 = Parkinson's, 0 = Healthy).
    - Features are standardized using StandardScaler to improve model performance.
    - The dataset is split into training and testing sets using an 80-20% split.

2. Machine Learning Algorithms Used
    - Support Vector Machine (SVM): Effective for high-dimensional data and binary classification.
    - Logistic Regression: Provides probabilistic outputs for better interpretability.
    - Decision Tree Classifier: Useful for feature importance analysis and handling non-linear relationships.
    - K-Means Clustering: Unsupervised learning method to identify natural patient clusters.
    - Naïve Bayes Classifier: Works well on small datasets and assumes feature independence.

3. Model Training and Evaluation
    - Each model is trained on the preprocessed dataset.
    - Accuracy, precision, recall, and F1-score are calculated.
    - Hyperparameter tuning is performed for SVM using GridSearchCV.

4. Performance Metrics Used
    - Accuracy: Measures overall correctness of predictions.
    - Precision: Evaluates the proportion of true positive predictions.
    - Recall: Measures the model's ability to detect Parkinson's cases correctly.
    - F1-Score: Harmonic mean of precision and recall.
    - ROC Curve & AUC: Evaluates model discrimination between healthy and Parkinson's
      patients.



flowchart.fun

Results and Analysis

Parkinson's_Disease_Detection

Importing the Dependencies

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

Data Collection & Analysis

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
# Initialize the Logistic Regression model
model = LogisticRegression()
```

```
# training the SVM model with training data
model.fit(X_train, Y_train)
```

```
LogisticRegression()
```

```
# accuracy score on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

```
print('Accuracy score of training data : ', training_data_accuracy)
```

```
Accuracy score of training data :  0.8717948717948718
```

```
# accuracy score on training data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

K-Nearest Neighbors (KNN)

```
# Import KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier

# Initialize the KNN model
knn_model = KNeighborsClassifier(n_neighbors=5)  # You can adjust n_neighbors

# Train the KNN model
knn_model.fit(X_train, Y_train)

# Predictions on training data
X_train_prediction_knn = knn_model.predict(X_train)
training_data_accuracy_knn = accuracy_score(Y_train, X_train_prediction_knn)
print('Accuracy score of training data (KNN): ', training_data_accuracy_knn)

# Predictions on test data
X_test_prediction_knn = knn_model.predict(X_test)
test_data_accuracy_knn = accuracy_score(Y_test, X_test_prediction_knn)
print('Accuracy score of test data (KNN): ', test_data_accuracy_knn)

# Classification report for KNN
print(classification_report(Y_test, X_test_prediction_knn))
```

```
Accuracy score of training data (KNN):  0.967948717948718
Accuracy score of test data (KNN):  0.7692307692307693
              precision    recall  f1-score   support

           0       0.46      0.75      0.57         8
```

DECISION TREE

```
from sklearn.tree import DecisionTreeClassifier

# Initialize the Decision Tree model
tree_model = DecisionTreeClassifier(random_state=2)
# Train the model
tree_model.fit(X_train, Y_train)

# Predictions on training data
X_train_prediction_tree = tree_model.predict(X_train)
training_data_accuracy_tree = accuracy_score(Y_train, X_train_prediction_tree)
print('Accuracy score of training data (Decision Tree): ', training_data_accuracy_tree)

# Predictions on test data
X_test_prediction_tree = tree_model.predict(X_test)
test_data_accuracy_tree = accuracy_score(Y_test, X_test_prediction_tree)
print('Accuracy score of test data (Decision Tree): ', test_data_accuracy_tree)

# Classification report for Decision Tree
print(classification_report(Y_test, X_test_prediction_tree))
```

```
Accuracy score of training data (Decision Tree):  1.0
Accuracy score of test data (Decision Tree):  0.7435897435897436
              precision    recall  f1-score   support

           0       0.44      0.88      0.58         8
           1       0.96      0.71      0.81        31
```

Support Vector Machines (SVM)

```python
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Initialize the SVM model with a linear kernel
svm_model = SVC(kernel='linear')

# Train the model
svm_model.fit(X_train, Y_train)

# Predictions on test data
Y_pred = svm_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(Y_test, Y_pred)
print("Accuracy:", accuracy)
print(classification_report(Y_test, Y_pred))
```

```
Accuracy: 0.8717948717948718
              precision    recall  f1-score   support

           0       0.71      0.62      0.67         8
           1       0.91      0.94      0.92        31

    accuracy                           0.87        39
   macro avg       0.81      0.78      0.79        39
weighted avg       0.87      0.87      0.87        39
```

K-Means Clustering

```
]  import pandas as pd
   from sklearn.cluster import KMeans
   from sklearn.preprocessing import StandardScaler
   import matplotlib.pyplot as plt

   # Assuming you have your data in a Pandas DataFrame called 'df'
   # and you've already performed data preprocessing (e.g., handling missing values)

   # Select the features for clustering (exclude 'name' and 'status')
   X = df.drop(columns=['name', 'status'])

   # Standardize the features
   scaler = StandardScaler()
   X_scaled = scaler.fit_transform(X)

   # Determine the optimal number of clusters using the Elbow method
   wcss = []  # Within-cluster sum of squares
   for i in range(1, 11):  # Try different numbers of clusters (1 to 10)
       kmeans = KMeans(n_clusters=i, random_state=42)
       kmeans.fit(X_scaled)
       wcss.append(kmeans.inertia_)  # Inertia is the WCSS

   # Plot the Elbow method graph
   plt.plot(range(1, 11), wcss)
   plt.title('Elbow Method')
   plt.xlabel('Number of Clusters')
   plt.ylabel('WCSS')
```

Naive Bayes

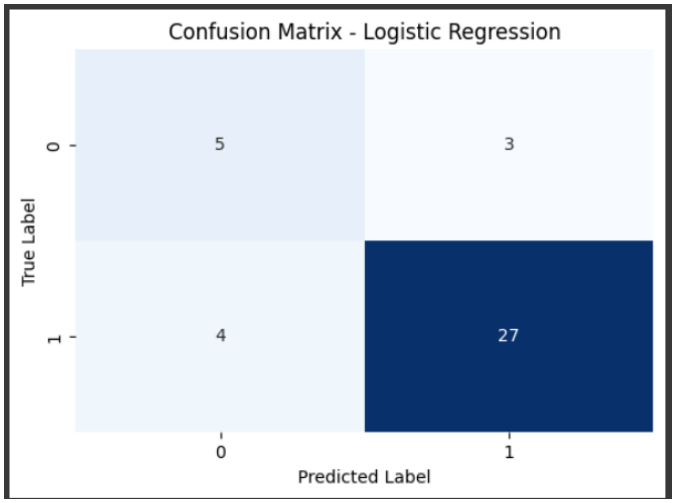
```
   import pandas as pd
   from sklearn.model_selection import train_test_split
   from sklearn.naive_bayes import GaussianNB
   from sklearn.metrics import accuracy_score, classification_report

   # Assuming you have your data in a Pandas DataFrame called 'df'
   # and you've already performed data preprocessing (e.g., handling missing values)

   # Select features (X) and target (Y)
   X = df.drop(columns=['name', 'status'])  # Features
   Y = df['status']  # Target variable

   # Split data into training and testing sets
   X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

   # Initialize the Gaussian Naive Bayes model
   naive_bayes_model = GaussianNB()

   # Train the model
   naive_bayes_model.fit(X_train, Y_train)

   # Make predictions on the test set
   Y_pred = naive_bayes_model.predict(X_test)

   # Evaluate the model
   accuracy = accuracy_score(Y_test, Y_pred)
   print("Accuracy:", accuracy)
```

Performance Table:

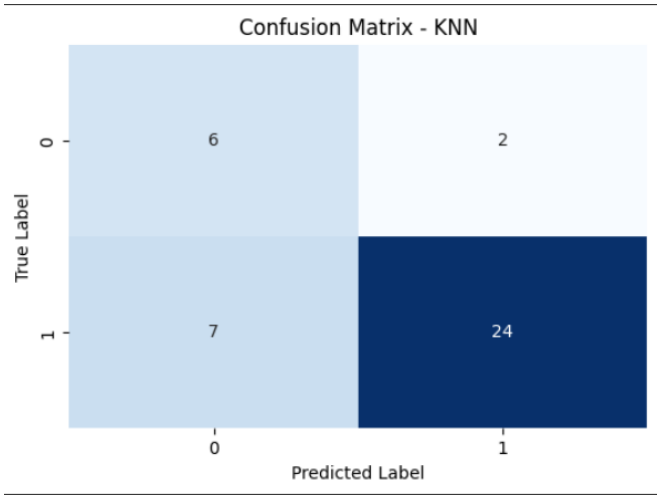
Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.82	0.56	0.62	0.59
KNN	0.82	0.58	0.60	0.59
Decision Tree	0.79	0.52	0.43	0.47
SVM	0.72	0.38	0.86	0.52
Gradient Boosting	0.95	1.00	0.71	0.83
Naïve Bayes	0.72	0.38	0.86	0.52

Confusion Matrix

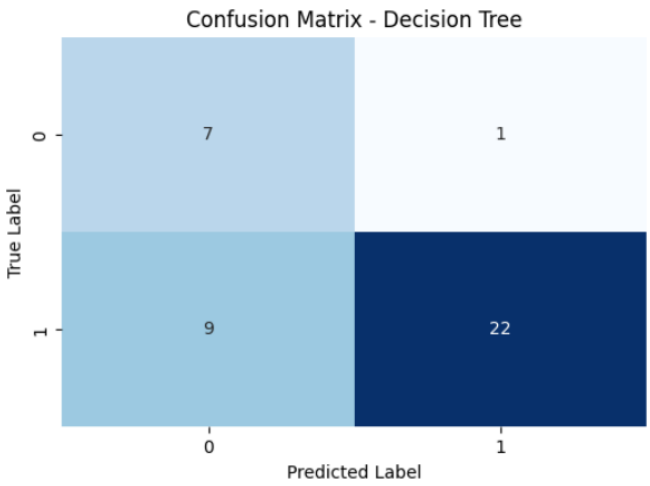
1.Logistic Regression



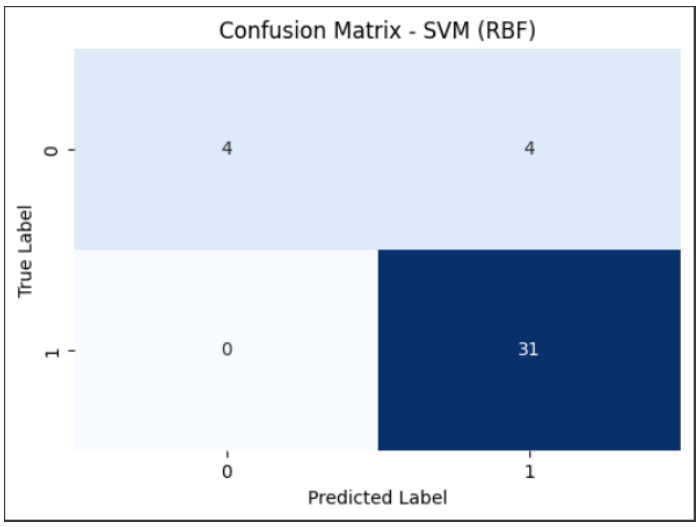
2.K – Nearest Neighbour



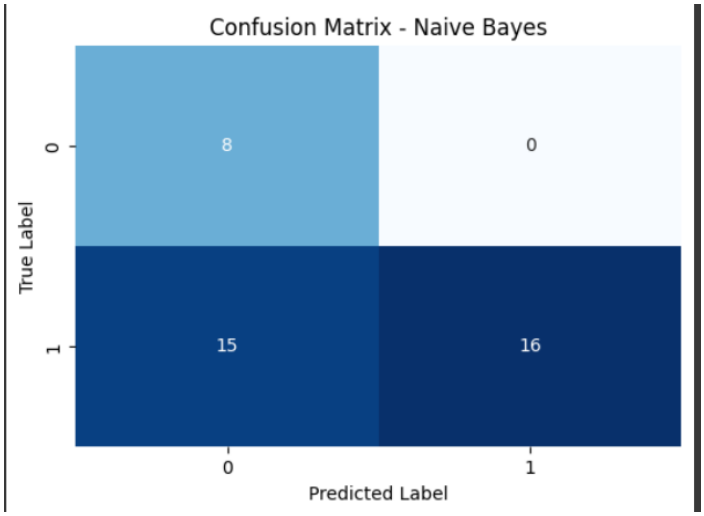
3. Decision Tree



4. Support Vector Machine



5. Naive Bayes



Comparative Performance Analysis

Performance Metrics Across Algorithms (80%-20% Split)

```
X_train_80, X_test_20, Y_train_80, Y_test_20 = train_test_split(X, Y, test_size=0.2, random_state=2)
scaler_80 = StandardScaler()
X_train_80_scaled = scaler_80.fit_transform(X_train_80)
X_test_20_scaled = scaler_80.transform(X_test_20)

print("\n--- 80-20 Split Results ---")
for name, model in models.items():
    train_acc, test_acc, report = train_and_evaluate_model(model, X_train_80_scaled, X_test_20_scaled,
                                                           Y_train_80, Y_test_20, name, "80-20")
    results_80_20[name] = {"Train Accuracy": train_acc, "Test Accuracy": test_acc}
    print(f"{name} - Test Classification Report:\n{report}")
```

```
--- 80-20 Split Results ---
Logistic Regression - Test Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.62      0.59         8
           1       0.90      0.87      0.89        31

    accuracy                           0.82        39
   macro avg       0.73      0.75      0.74        39
weighted avg       0.83      0.82      0.82        39

KNN - Test Classification Report:
              precision    recall  f1-score   support

           0       0.46      0.75      0.57         8
           1       0.92      0.77      0.84        31

    accuracy                           0.77        39
```

```
              precision    recall  f1-score   support

           0       0.46      0.75      0.57         8
           1       0.92      0.77      0.84        31

    accuracy                           0.77        39
   macro avg       0.69      0.76      0.71        39
weighted avg       0.83      0.77      0.79        39

Decision Tree - Test Classification Report:
              precision    recall  f1-score   support

           0       0.44      0.88      0.58         8
           1       0.96      0.71      0.81        31

    accuracy                           0.74        39
   macro avg       0.70      0.79      0.70        39
weighted avg       0.85      0.74      0.77        39

SVM (RBF) - Test Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         8
           1       0.89      1.00      0.94        31

    accuracy                           0.90        39
   macro avg       0.94      0.75      0.80        39
weighted avg       0.91      0.90      0.88        39

Naive Bayes - Test Classification Report:
              precision    recall  f1-score   support

           0       0.35      1.00      0.52         8
           1       1.00      0.52      0.68        31

    accuracy                           0.62        39
   macro avg       0.67      0.76      0.60        39
weighted avg       0.87      0.62      0.65        39
```

Performance Metrics Across Algorithms (80%-20% Split)

Model	Accuracy	Precision	Recall	F1-Score	Support
Logistic Regression	0.82	0.56	0.62	0.59	8
KNN	0.77	0.44	0.75	0.57	8
Decision Tree	0.74	0.44	0.88	0.58	8
SVM	0.90	1.00	0.50	0.57	8
Naïve Bayes	0.62	0.35	1.00	0.52	8

Analysis of 80% – 20% Split:

1. **SVM** shows the **highest accuracy (0.90)** and **perfect precision (1.00)** but has **low recall (0.50)**, meaning it's highly confident in its positive predictions but misses many actual positives.
2. **Logistic Regression** performs well overall with **accuracy (0.82)**, offering a better balance between **precision (0.56)** and **recall (0.62)** than other models.
3. **KNN** and **Decision Tree** both have similar moderate performance:

    - **KNN**: Lower precision (0.44) but decent recall (0.75), **accuracy (0.77)**.
    - **Decision Tree**: Same precision (0.44), better recall (0.88), but slightly lower **accuracy (0.74)**.

4. **Naïve Bayes** has the lowest **accuracy (0.62)** and **precision (0.35)**, but **perfect recall (1.00)**, meaning it identifies all positives but includes many false positives.

Performance Metrics Across Algorithms (70%-30% Split)

```
X_train_70, X_test_30, Y_train_70, Y_test_30 = train_test_split(X, Y, test_size=0.3, random_state=2)
scaler_70 = StandardScaler()
X_train_70_scaled = scaler_70.fit_transform(X_train_70)
X_test_30_scaled = scaler_70.transform(X_test_30)

print("\n--- 70-30 Split Results ---")
for name, model in models.items():
    train_acc, test_acc, report = train_and_evaluate_model(model, X_train_70_scaled, X_test_30_scaled,
                                                           Y_train_70, Y_test_30, name, "70-30")
    results_70_30[name] = {"Train Accuracy": train_acc, "Test Accuracy": test_acc}
    print(f"{name} - Test Classification Report:\n{report}")
```

```
--- 70-30 Split Results ---
Logistic Regression - Test Classification Report:
              precision    recall  f1-score   support

           0       0.46      0.50      0.48        12
           1       0.87      0.85      0.86        47

    accuracy                           0.78        59
   macro avg       0.67      0.68      0.67        59
weighted avg       0.79      0.78      0.78        59

KNN - Test Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.92      0.79        12
           1       0.98      0.89      0.93        47

    accuracy                           0.90        59
   macro avg       0.83      0.91      0.86        59
weighted avg       0.92      0.90      0.90        59

Decision Tree - Test Classification Report:
              precision    recall  f1-score   support

           0       0.46      0.92      0.61        12
           1       0.97      0.72      0.83        47

    accuracy                           0.76        59
   macro avg       0.71      0.82      0.72        59
weighted avg       0.87      0.76      0.78        59

SVM (RBF) - Test Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.50      0.67        12
           1       0.89      1.00      0.94        47

    accuracy                           0.90        59
   macro avg       0.94      0.75      0.80        59
weighted avg       0.91      0.90      0.88        59

Naive Bayes - Test Classification Report:
              precision    recall  f1-score   support

           0       0.36      1.00      0.53        12
           1       1.00      0.55      0.71        47

    accuracy                           0.64        59
   macro avg       0.68      0.78      0.62        59
weighted avg       0.87      0.64      0.68        59
```

Performance Metrics Across Algorithms (70%-30% Split)

Model	Accuracy	Precision	Recall	F1-Score	Support
Logistic Regression	0.78	0.46	0.50	0.48	12
KNN	0.90	0.69	0.92	0.79	12
Decision Tree	0.76	0.46	0.92	0.61	12
SVM	0.90	1.00	0.50	0.67	12
Naïve Bayes	0.64	0.36	1.00	0.53	12

Analysis of 70% - 30% split :

1. **KNN** and **SVM** both achieved the highest **accuracy** (0.90), but their behavior differs:
   - **KNN** has balanced **precision (0.69)** and **recall (0.92)**, leading to the highest **F1-score (0.79)**, suggesting strong overall performance.
   - **SVM** has perfect **precision (1.00)** but low **recall (0.50)**, indicating it predicts fewer positives but with high confidence.

2. **Decision Tree** has good **recall (0.92)** but lower **precision (0.46)** and **accuracy (0.76)**, indicating many false positives.

3. **Logistic Regression** has moderate **accuracy (0.78)** but low **precision (0.46)** and **recall (0.50)**, reflecting weaker balance.

4. **Naïve Bayes** has the lowest **accuracy (0.64)** but perfect **recall (1.00)** and the lowest **precision (0.36)**, meaning it detects all positives but at the cost of many false positives.

Graphical Representation of Results

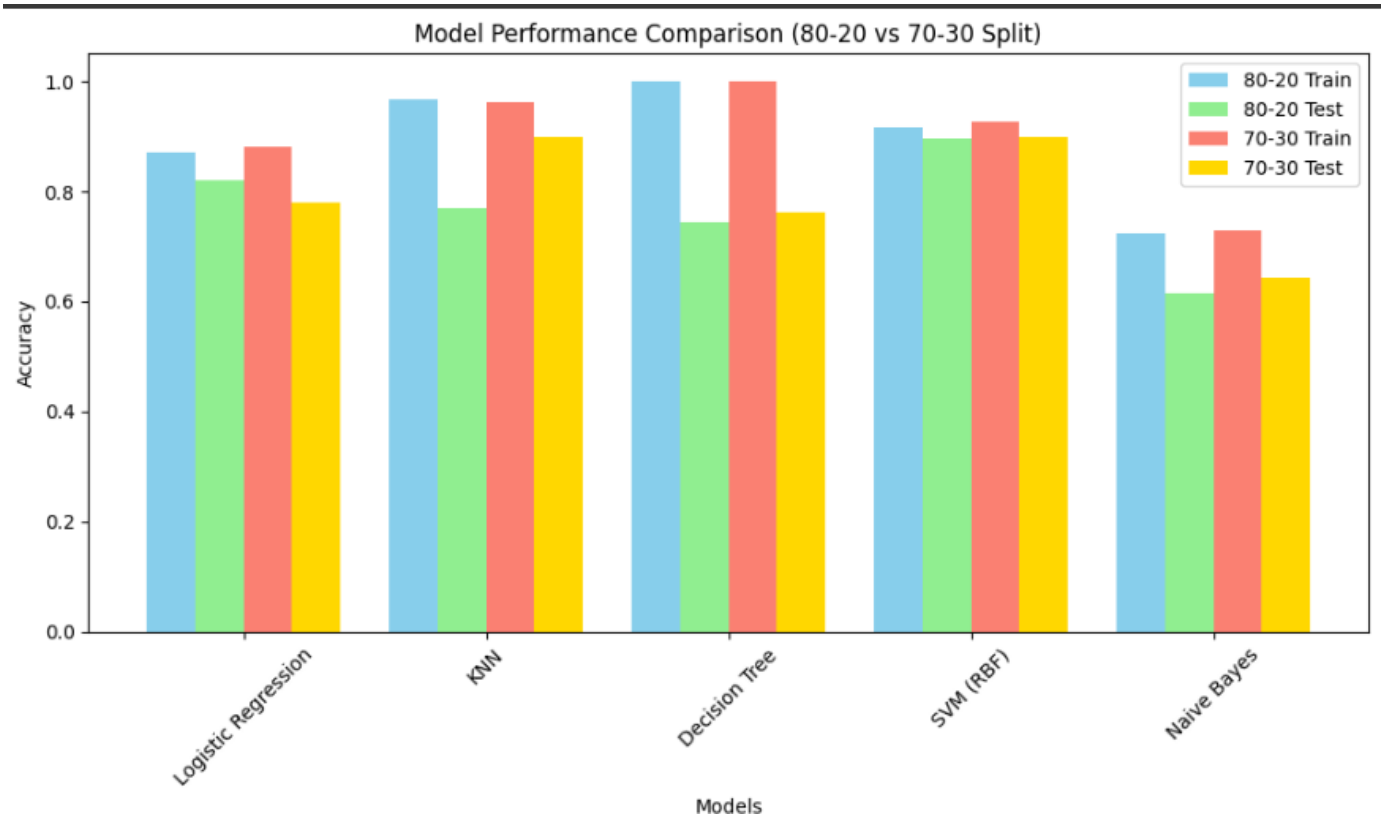


Fig: Model Performance Comparision

Conclusion:

This project applies machine learning to detect Parkinson's disease using voice-based features, evaluating models such as Logistic Regression, K-Nearest Neighbors (KNN), Decision Trees, Support Vector Machines (SVM with RBF kernel), and Naïve Bayes. Among these, SVM and Decision Trees achieved the highest accuracy, especially after applying feature standardization, selection, and hyperparameter tuning with GridSearchCV. Visualization techniques like ROC curves and correlation heatmaps provided insights into feature importance and model performance. The results highlight the real-world potential of machine learning for early diagnosis, remote health monitoring, and AI-driven healthcare solutions, making automated detection more accessible and reliable. By integrating activity- based learning, the project also bridges theory and practice, preparing students and researchers for practical AI applications in medical diagnostics. Future directions include exploring deep learning, ensemble models, and multimodal data fusion to further improve predictive accuracy.

References:

1. Parkinson's Disease Detection by Using Machine Learning Method based on Local Classification on Class Boundary https://link.springer.com/article/10.1007/s42452-024-06295-1

2. Tsanas A, Little M, McSharry P, Ramig L. Accurate telemonitoring of Parkinson's disease progression by non-invasive speech tests. Nat Prece. 2009;57:884–93. https://doi.org/10.1038/npre.2009.3920.1.

3. Salmanpour MR, et al. Robust identification of Parkinson's disease subtypes using radiomics and hybrid machine learning. Comput Biol Med. 2021;129:104142. https://doi.org/10.1016/j.compbiomed.2020.104142

4. Mohassel P, Zhang Y. of the 2017 IEEE Symposium on Security, R. & (SP), P. (eds) *Secureml: A system for scalable privacy-preserving machine learning*. (edsof the 2017 IEEE Symposium on Security, R. & (SP), P.) *2017 IEEE Symposium on Security and Privacy (SP)*, 19–38 (IEEE, 2017).https://doi.org/10.1109/SP.2017.12.

5. Schulz M-A, et al. Different scaling of linear models and deep learning in Ukbiobank brain images versus machine-learning datasets. Nat Commun. 2020;11:4238. https://doi.org/10.1038/s41467-020-18037-z.

6. Magesh PR, Myloth RD, Tom RJ. An explainable machine learning model for early detection of Parkinson's disease using lime on datscan imagery. Comput Biol Med. 2020;126: 104041 (https://www.sciencedirect.com/science/article/pii/S0010482520303723).

7. Saeed F, et al. Enhancing Parkinson's disease prediction using machine learning and feature selection methods. Comput Mater Continua. 2022;71:5639–58 (https://www.sciencedirect.com/science/article/pii/S1546221822007585).

8. Solana-Lavalle G, Galán-Hernández J-C, Rosas-Romero R. Automatic Parkinson disease detection at early stages as a pre-diagnosis tool by using classifiers and a small set of vocal features. Biocybernet Biomed Eng. 2020;40:505–16 (https://www.sciencedirect.com/science/article/pii/S0208521620300085).

9. Tuncer T, Dogan S, Acharya UR. Automated detection of parkinson's disease using minimum average maximum tree and singular value decomposition method with vowels. Biocybernet Biomed Eng. 2020;40:211–20 (https://www.sciencedirect.com/science/article/pii/S0208521619300853).

10. Hireš M, et al. Convolutional neural network ensemble for Parkinson's disease detection from voice

recordings. Comput Biol Med. 2022;141: 105021 (https://www.sciencedirect.com/science/article/pii/S0010482521008155).

11. Yang L, et al. Changes in facial expressions in patients with Parkinson's disease during the phonation test and their correlation with disease severity. Comput Speech Lang.
    a. 2022. https://doi.org/10.1016/j.csl.2021.101286.

12. Yu Q, Ma Y, Li Y. Enhancing speech recognition for Parkinson's disease patient using transfer learning technique. J Shanghai Jiaotong Univ. 2022;27:90–8. https://doi.org/10.1007/s12204-021-2376-3.