ISSN PRINT 2319 1775 Online 2320 7876

Research paper © 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, Iss 12, Dec 2022

A Systematic Approach to Detect Parkinson's Disease using Traditional and Ensemble Machine Learning Techniques

Dr. V. Rama Chandran[1], Professor & HOD, Department of CSE,
Vasireddy Venkatadri Institute of Technology, Nambur, Guntur Dt., Andhra Pradesh.

G. Hemanth[2], **E. Jahnavi**[3], **A. Vasundhara**[4], **B. Raj Kumar**[5]
[2,3,4,5] UG Students, Department of CSE,
Vasireddy Venkatadri Institute of Technology, Nambur, Guntur Dt., Andhra Pradesh.
[1,2,3,4,5] vrc.bhatt@vvit.net, hemanth17.goli@gmail.com, jahnavieluri1011@gmail.com, aramallavasundhara01@gmail.com, bharatharajkumar@gmail.com

DOI:10.48047/IJFANS/V11/I12/217

Abstract

Parkinson's disease (PD) is a neurodegenerative condition that worsens with time and affects both the neurological system and body components under the control of the nervous system. This condition causes slow movements, tremors, balance problems, and more. Currently, we have no proper cure or treatment available, but it can sometimes be cured with medication if it is diagnosed in its initial stages. Voice deterioration is also a common symptom, which often presents in the initial stages of the disease. As a result, the project 'A Systematic Approach to Detect Parkinson's Disease Using Traditional and Ensemble Machine Learning Techniques' is used to detect PD using voice data. In order to create a model that is capable of accurately identifying the disease's existence in a person's body, this project makes use of a variety of machine learning techniques, ensemble learning approaches, and Python libraries. This work aims to compare various machine learning models in the successful prediction of PD and develop an effective and accurate model to help detect the disease at an earlier stage, which could help doctors assist in the cure and recovery of PD patients. This project showed 97% efficiency. For this purpose, we plan to use the Parkinson's disease dataset in [5] , which is acquired from the UCIML repository.

Keywords: Parkinson's disease, PD, Machine Learning, Ensemble Learning, Logistic Regression, K-Nearest Neighbour, Support Vector Machine, Random Forest, Adaboost, Stacking, XGBoost.

Introduction

Parkinson's disease (PD) is one of the neurological disorders that affects an estimated 10 million people worldwide. It is caused by a decrease in dopamine levels in the brain, which can be attributed to the degeneration of dopaminergic neurons. Symptoms include unintentional or involuntary movements such as shaking, stiffness, and problems with balance and coordination. The symptoms appear gradually. But when the illness progresses, the symptoms have a bigger impact on the body. The noticeable early symptoms of PD are rigidity, tremors and trouble in walking, although psychological issues can also be developed as the disease worsens. Of all diagnosed cases of PD, 90% show some sort of vocal impairment, which consists of deterioration in the normal production of vocal sounds, medically termed dysphonia. There is no cure for Parkinson's disease. The challenge currently faced is the early detection of PD in affected patients. If diagnosed early, patients can improve their quality of life even if the disease progresses. As PD patients show unique and characteristic vocal features, their voice data would be an invaluable procedure for diagnosis. To this end, various machine learning techniques are used to assist patients in getting early medication use and are applied to the voice data of patients. With the use of machine learning techniques, mathematical models, and statistical expertise, the entire process of data processing can be automated and made more efficient. Depending on the task at hand and the machine's capabilities, graphs, charts, tables, photos, and a variety of other formats can be generated.

Literature Survey

Rahul R. Zaveri and Pramila M. Chawan are the authors in [1]. The datamining techniques used in this paper are Decision Tree, SVM, Naive Bayes, K-NN, and Logistic Regression. All the classifiers are compared with each other to get the highest accuracy. Using Decision Tree classifier, we achieve highest accuracy of 94%.

Raunak Sulekh and Anupam Bhatia are the researchers in [2]. The Naive Bayesian classification model was used in the current paper. A recorded speech signals dataset is analyzed in this paper whose features are voice measures. This model classifies a person with PD as 1 and healthy as 0. In this paper, the authors used the Rapid miner tool to analyze the data. An accuracy of 98.5% is recorded.

Marianna Amboni et al. proposed the paper "Using Gait Analysis' Parameters to Classify Parkinsonism: A Data Mining Approach" [3]. In this research, the authors compared the performance of two algorithms: Random Forest and Gradient Boosted Trees. Using a data mining approach, PD patients at various stages were taken into consideration and identified as typical or atypical based on gait analysis. Random Forest obtained the highest accuracy of 86.4%.

Marimuthu et.al. proposed [4] to detect PD. The proposed model used XGBClassifier as the traditional machine learning techniques failed to detect PD accurately because of its complex features. The dataset Parkinson's disease is taken from the UCI ML repository to implement the model. XGBClassifier is compared with other classifiers such as Logistic Regression, Decision Tree, and SVC. The XGBClassifier achieved an accuracy of 94.87% proving that it is better than traditional algorithms. [6-14]

Methodology

Computer systems now have the potential to automatically learn without being explicitly programmed thanks to machine learning. The high-level overview of key system components and crucial interrelationships is described in the Figure 1. The following steps make up the execution flow represented by this diagram.

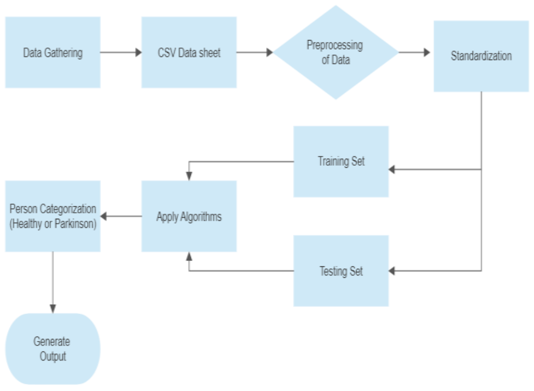


Figure 1 : Architecture

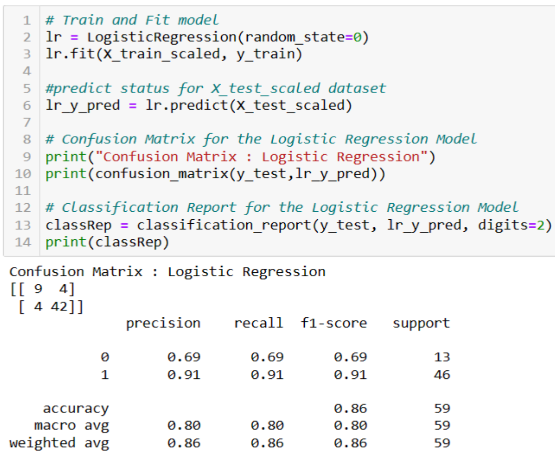
The process flow that is utilised to forecast Parkinson's disease is defined in the architectural diagram. Data collection and preparation into a comprehensible format constitute the first phase. Data that has undergone pre-processing is standardised. The dataset must then be divided into train data and test data before we can train it. Each machine learning method is trained using the Parkinson's data. We must test using the same methods after training the data with them. Finally, classification accuracy is used to compare the algorithms' output.

Implementation

The classification algorithms used in our project are Logistic Regression, K-Nearest Neighbour, Support Vector Machine, Random Forest, Stacking, AdaBoost, and XGBoost.

Logistic Regression

Logistic regression is a statistical method that models the relationship between a binary or categorical outcome variable and one or more predictor variables. This algorithm uses a function which has an S-shaped curve, which allows for the prediction of probabilities of the outcome variable being in a certain category. The logistic regression model estimates the coefficients of the predictor variables by minimizing the log-likelihood function, which measures the goodness of fit of the model. This process involves iteratively adjusting the coefficients until the model reaches convergence.



```
# Train and Fit model
lr = LogisticRegression(random_state=0)
lr.fit(X_train_scaled, y_train)

#predict status for X_test_scaled dataset
lr_y_pred = lr.predict(X_test_scaled)

# Confusion Matrix for the Logistic Regression Model
print("Confusion Matrix : Logistic Regression")
print(confusion_matrix(y_test,lr_y_pred))

# Classification Report for the Logistic Regression Model
classRep = classification_report(y_test, lr_y_pred, digits=2)
print(classRep)
```

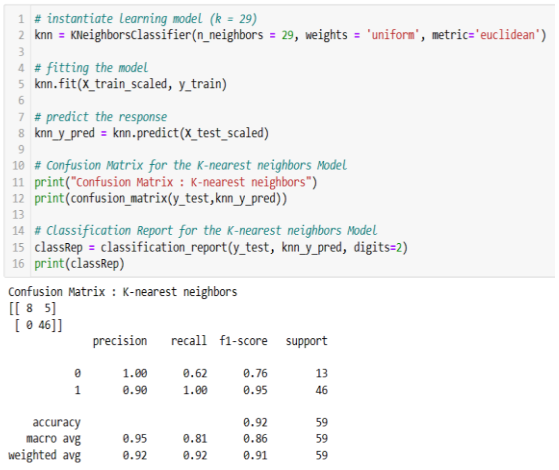
```
Confusion Matrix : Logistic Regression
[[ 9  4]
 [ 4 42]]
              precision    recall  f1-score   support

           0       0.69      0.69      0.69        13
           1       0.91      0.91      0.91        46

    accuracy                           0.86        59
   macro avg       0.80      0.80      0.80        59
weighted avg       0.86      0.86      0.86        59
```

Figure 2 : Building Logistic Regression Model

Figure 2 shows the steps involved in training and fitting the model.

K-Nearest Neighbor

K nearest neighbor (KNN) is a popular algorithm in machine learning techniques used for classification and regression tasks. In KNN, the algorithm classifies or predicts new data points by comparing them with their nearest neighbors in the training dataset. In this model, we will find the distance between new and available data points. Once the distances are calculated, the algorithm selects the K nearest neighbors and assigns the new data point to the most common class among these neighbors (for classification) or calculates the average value of the nearest neighbors (for regression). The below Figure shows the steps involved in training and fitting the model.



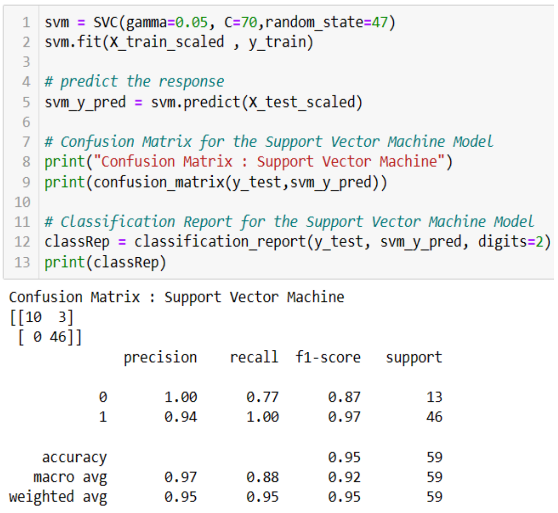
```
# instantiate learning model (k = 29)
knn = KNeighborsClassifier(n_neighbors = 29, weights = 'uniform', metric='euclidean')

# fitting the model
knn.fit(X_train_scaled, y_train)

# predict the response
knn_y_pred = knn.predict(X_test_scaled)

# Confusion Matrix for the K-nearest neighbors Model
print("Confusion Matrix : K-nearest neighbors")
print(confusion_matrix(y_test,knn_y_pred))

# Classification Report for the K-nearest neighbors Model
classRep = classification_report(y_test, knn_y_pred, digits=2)
print(classRep)
```

```
Confusion Matrix : K-nearest neighbors
[[ 8  5]
 [ 0 46]]
              precision    recall  f1-score   support

           0       1.00      0.62      0.76        13
           1       0.90      1.00      0.95        46

    accuracy                           0.92        59
   macro avg       0.95      0.81      0.86        59
weighted avg       0.92      0.92      0.91        59
```

Figure 3 : Building KNN Model

Support Vector Machine

Support Vector Machine (SVM) is a popular algorithm in machine learning techniques used for classification and regression tasks. It works by finding the optimal hyper plane that separates the different classes or predicts the value of the target variable. In SVM, the algorithm constructs a hyperplane that maximizes the margin between the different classes or the predicted values. The SVM algorithm tries to maximize this margin while also minimizing the classification error. SVM has several advantages, such as the ability to handle high-dimensional data, the ability to find the optimal solution, and the ability to handle non-linearly separable data through kernel functions. However, it can be computationally expensive and sensitive to the choice of the kernel function and its parameters.



```
svm = SVC(gamma=0.05, C=70,random_state=47)
svm.fit(X_train_scaled , y_train)

# predict the response
svm_y_pred = svm.predict(X_test_scaled)

# Confusion Matrix for the Support Vector Machine Model
print("Confusion Matrix : Support Vector Machine")
print(confusion_matrix(y_test,svm_y_pred))

# Classification Report for the Support Vector Machine Model
classRep = classification_report(y_test, svm_y_pred, digits=2)
print(classRep)
```

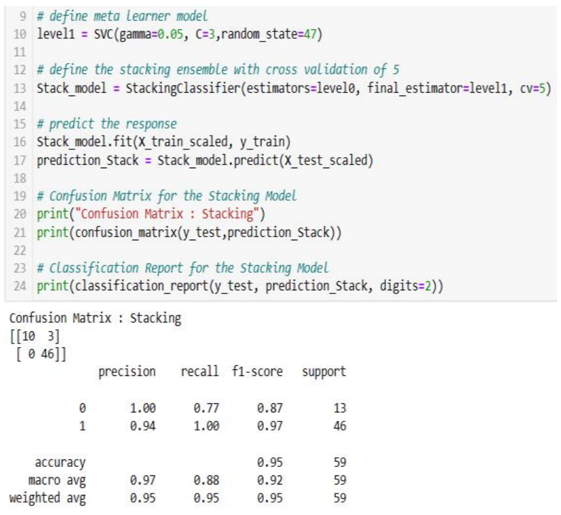
```
Confusion Matrix : Support Vector Machine
[[10  3]
 [ 0 46]]
              precision    recall  f1-score   support

           0       1.00      0.77      0.87        13
           1       0.94      1.00      0.97        46

    accuracy                           0.95        59
   macro avg       0.97      0.88      0.92        59
weighted avg       0.95      0.95      0.95        59
```

Figure 4 : Building SVM Model

Figure 4 shows the steps involved in training and fitting the model.

Stacking

Stacking is a machine learning technique that is used to integrate the outcomes of many models in order to increase the effectiveness of the final prediction. In stacking, the predictions of several base models are combined by training a meta-model, also known as a stacking model, on their outputs. The stacking model takes the output predictions of the base models as input features and learns to make the final prediction based on this input. The meta-model can be any type of machine learning algorithm, such as logistic regression, decision tree, or neural network. In this model, training data is divided into multiple folds and given to the base models. Some parts of the data are used for training purposes, and the other parts are used for prediction. The predictions of the base models are then combined and used as input features for the stacking model, which is trained on the remaining data to make the final prediction. After training, the stacking model can be used to predict the target variable for new data by passing the output of the base models through the meta-model. Stacking can improve the performance of the base models by combining their strengths and compensating for their weaknesses. It can also reduce the risk of over fitting by using cross-validation to train the stacking model.



```
# define meta learner model
level1 = SVC(gamma=0.05, C=3,random_state=47)

# define the stacking ensemble with cross validation of 5
Stack_model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)

# predict the response
Stack_model.fit(X_train_scaled, y_train)
prediction_Stack = Stack_model.predict(X_test_scaled)

# Confusion Matrix for the Stacking Model
print("Confusion Matrix : Stacking")
print(confusion_matrix(y_test,prediction_Stack))

# Classification Report for the Stacking Model
print(classification_report(y_test, prediction_Stack, digits=2))
```

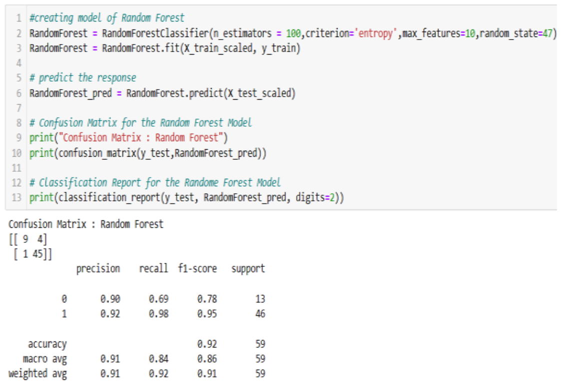
```
Confusion Matrix : Stacking
[[10  3]
 [ 0 46]]
              precision    recall  f1-score   support

           0       1.00      0.77      0.87        13
           1       0.94      1.00      0.97        46

    accuracy                           0.95        59
   macro avg       0.97      0.88      0.92        59
weighted avg       0.95      0.95      0.95        59
```

Figure 5 : Building Stacking Model

Figure 5 shows the steps involved in training and fitting the model.

Random Forest

Random forest is a popular algorithm in advanced machine learning techniques used for classification and regression tasks. It works by constructing multiple decision trees and combining their predictions to improve the accuracy of the final prediction. Every decision tree is constructed based on different subsets of data and features. This helps to reduce over fitting and improve the diversity of the trees. The decision trees are trained using a recursive binary splitting algorithm that selects the best split point based on a measure of node impurity, such as Gini impurity or entropy. To make a prediction for new data, the random forest algorithm passes the data through all the decision trees and aggregates their predictions. The most common prediction among the trees is used as the final prediction. Figure 6 shows the steps involved in training and fitting the model.



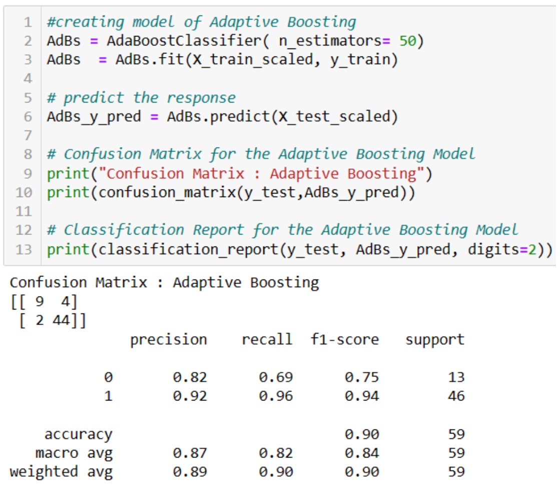
```
#creating model of Random Forest
RandomForest = RandomForestClassifier(n_estimators = 100,criterion='entropy',max_features=10,random_state=47)
RandomForest = RandomForest.fit(X_train_scaled, y_train)

# predict the response
RandomForest_pred = RandomForest.predict(X_test_scaled)

# Confusion Matrix for the Random Forest Model
print("Confusion Matrix : Random Forest")
print(confusion_matrix(y_test,RandomForest_pred))

# Classification Report for the Randome Forest Model
print(classification_report(y_test, RandomForest_pred, digits=2))
```

```
Confusion Matrix : Random Forest
[[ 9  4]
 [ 1 45]]
              precision    recall  f1-score   support

           0       0.90      0.69      0.78        13
           1       0.92      0.98      0.95        46

    accuracy                           0.92        59
   macro avg       0.91      0.84      0.86        59
weighted avg       0.91      0.92      0.91        59
```

Figure 6 : Building Random Forest Model

AdaBoost

AdaBoost is a popular algorithm in advanced machine learning techniques used for classification and regression tasks. It works by combining multiple weak classifiers into a strong classifier. In AdaBoost, the weak classifiers are trained sequentially on the training data, with each classifier focusing on the data points that were misclassified by the previous classifiers. The algorithm assigns weights to the data points based on their classification error, so that the next classifier focuses more on the misclassified data points. After training, the weak classifiers are combined using a weighted sum to make the final prediction. The weights are determined based on the performance of each classifier on the training data.



```
#creating model of Adaptive Boosting
AdBs = AdaBoostClassifier( n_estimators= 50)
AdBs  = AdBs.fit(X_train_scaled, y_train)

# predict the response
AdBs_y_pred = AdBs.predict(X_test_scaled)

# Confusion Matrix for the Adaptive Boosting Model
print("Confusion Matrix : Adaptive Boosting")
print(confusion_matrix(y_test,AdBs_y_pred))

# Classification Report for the Adaptive Boosting Model
print(classification_report(y_test, AdBs_y_pred, digits=2))
```

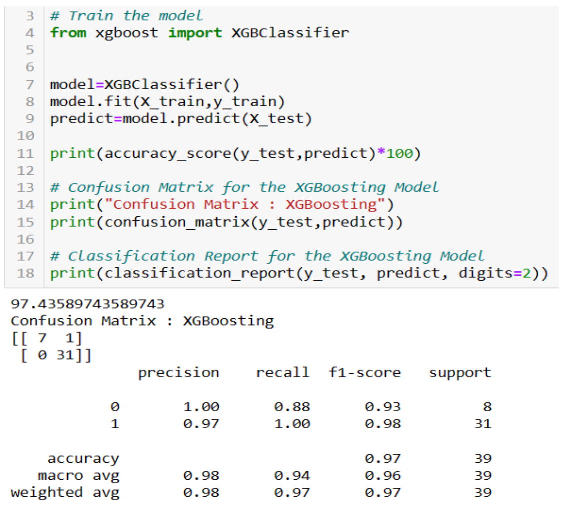
```
Confusion Matrix : Adaptive Boosting
[[ 9  4]
 [ 2 44]]
              precision    recall  f1-score   support

           0       0.82      0.69      0.75        13
           1       0.92      0.96      0.94        46

    accuracy                           0.90        59
   macro avg       0.87      0.82      0.84        59
weighted avg       0.89      0.90      0.90        59
```

Figure 7 : Building AdaBoost Model

Figure 7 shows the steps involved in training and fitting the model.

XGBoost

XGBoost is a popular algorithm in advanced machine learning techniques used for classification and regression tasks. It is an extension of the traditional gradient boosting algorithm that uses a more efficient and scalable implementation. XGBoost works by constructing multiple decision trees sequentially and combining their predictions to improve the accuracy of the final prediction. Each Every tree is developed to decrease the loss function, which calculates the gap between the predicted and actual values. XGBoost uses gradient boosting to optimize the loss function, which involves calculating the gradient and Hessian of the loss function with respect to the predicted values. In order to increase the effectiveness of the algorithm, this model uses different regularisation methods like L1 and L2 regularisation, dropout, and early stopping. These techniques help to prevent overfitting, reduce the complexity of the model, and improve the generalization performance. Figure 8 shows the steps involved in training and fitting the model.



```
# Train the model
from xgboost import XGBClassifier


model=XGBClassifier()
model.fit(X_train,y_train)
predict=model.predict(X_test)

print(accuracy_score(y_test,predict)*100)

# Confusion Matrix for the XGBoosting Model
print("Confusion Matrix : XGBoosting")
print(confusion_matrix(y_test,predict))

# Classification Report for the XGBoosting Model
print(classification_report(y_test, predict, digits=2))
```

```
97.43589743589743
Confusion Matrix : XGBoosting
[[ 7  1]
 [ 0 31]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       0.97      1.00      0.98        31

    accuracy                           0.97        39
   macro avg       0.98      0.94      0.96        39
weighted avg       0.98      0.97      0.97        39
```

Figure 8 : Building XGBoost Model

Results

The results of Parkinson's disease prediction using machine learning are typically evaluated using performance metrics such as accuracy, precision, recall, F1 score. A high accuracy score indicates that the model is able to correctly classify Parkinson's disease patients and healthy individuals. Precision measures the percentage of true Parkinson's disease patients among all the predicted Parkinson's disease patients. Recall measures the percentage of true Parkinson's disease patients that were correctly identified by the model. F1 score is the harmonic mean of precision and recall. Table 1 shows the performance metrics of the models used.

Table 1 : Performance metrics of models

Algorithm	**Precision**	**Recall Score**	**F1 Score**	**Accuracy**
Logistic Regression	0.86	0.86	0.86	86%
K-Nearest Neighbor	0.92	0.92	0.91	92%
Support Vector Machine	0.95	0.95	0.95	95%
Random Forest	0.91	0.92	0.91	92%
Stacking	0.95	0.95	0.95	95%
AdaBoost	0.89	0.90	0.90	90%
XGBoost	0.98	0.97	0.97	97%

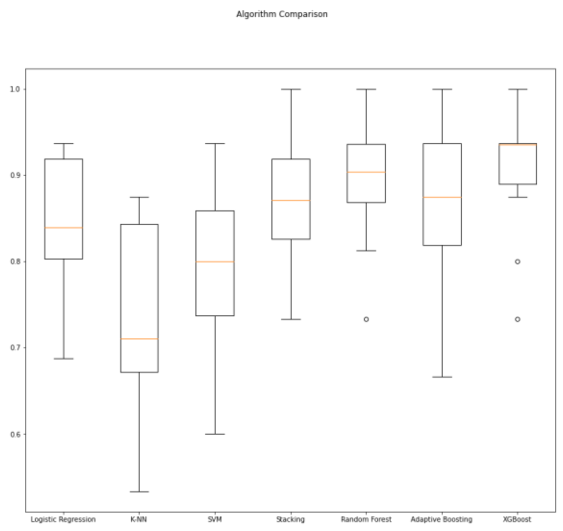


Figure 9 : Models Comparision

After analyzing Table 1 and Figure 9, XGBoost model has the greatest accuracy of 97% when comparing the metrics of several machine learning models.

Conclusion

In conclusion, Parkinson's disease is a neurodegenerative disorder that affects the motor system of the human body. It is characterized by symptoms such as tremors, stiffness, and slowness of movement. While there is no cure for Parkinson's disease, early detection and management can help to improve the patient's quality of life. However, the development of machine learning models for Parkinson's disease detection requires careful data collection, preprocessing, feature selection, and model selection. Moreover, the models need to be validated and fine-tuned to ensure their accuracy and reliability. Overall, machine learning has the potential to revolutionize the early detection and management of Parkinson's disease, leading to better patient outcomes and quality of life.

References

[1] Rahul R. Zaveri and Pramila M. Chawan, “Prediction of Parkinson's Disease using Data Mining: A Survey”, 2021/02/21.

[2] Dr. Anupam Bhatia and Raunak Sulekh, “Predictive Model for Parkinson’s Disease through Naive Bayes Classification”.

[3] Marianna Amboni, et al, “Using gait analysis’ parameters to classify Parkinsonism: A data mining approach” Computer Methods and Programs in Biomedicine vol. 180, Oct. 2019.

[4] Marimuthu M., Vidhya G., Dhaynithi J., Mohanraj G., Basker N., Theetchenya S., Vidyabharathi D. (2021). Detection of Parkinson’s disease using Machine Learning Approach. Annals of the Romanian Society for Cell Biology, 2544–2550.

[5] Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection', Little MA, McSharry PE, Roberts SJ, Costello DAE, Moroz IM. BioMedical Engineering OnLine 2007.

[6] Sri Hari Nallamala, et al., "A Literature Survey on Data Mining Approach to Effectively Handle Cancer Treatment", (IJET) (UAE), ISSN: 2227 – 524X, Vol. 7, No 2.7, SI 7, Page No: 729 – 732, March 2018.

[7] Sri Hari Nallamala, et.al., "An Appraisal on Recurrent Pattern Analysis Algorithm from the Net Monitor Records", (IJET) (UAE), ISSN: 2227 – 524X, Vol. 7, No 2.7, SI 7, Page No: 542 – 545, March 2018.

[8] Sri Hari Nallamala, et.al, "Qualitative Metrics on Breast Cancer Diagnosis with Neuro Fuzzy Inference Systems", International Journal of Advanced Trends in Computer Science and Engineering, (IJATCSE), ISSN (ONLINE): 2278 – 3091, Vol. 8 No. 2, Page No: 259 – 264, March / April 2019.

[9] Sri Hari Nallamala, et.al, "Breast Cancer Detection using Machine Learning Way", International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277-3878, Volume-8, Issue-2S3, Page No: 1402 – 1405, July 2019.

[10] Sri Hari Nallamala, et.al, "Pedagogy and Reduction of K-nn Algorithm for Filtering Samples in the Breast Cancer Treatment", International Journal of Scientific and Technology Research, (IJSTR), ISSN: 2277-8616, Vol. 8, Issue 11, Page No: 2168 – 2173, November 2019.

[11] Kolla Bhanu Prakash, Sri Hari Nallamala, et al., "Accurate Hand Gesture Recognition using CNN and RNN Approaches" International Journal of Advanced Trends in Computer Science and Engineering, 9(3), May – June 2020, 3216 – 3222.

[12] Sri Hari Nallamala, et al., "A Review on 'Applications, Early Successes & Challenges of Big Data in Modern Healthcare Management'", Vol.83, May - June 2020 ISSN: 0193-4120 Page No. 11117 – 11121.

[13] Nallamala, S.H., et al., "A Brief Analysis of Collaborative and Content Based Filtering Algorithms used in Recommender Systems", IOP Conference Series: Materials Science and Engineering, 2020, 981(2), 022008.

[14] Nallamala, S.H., Mishra, P., Koneru, S.V., "Breast cancer detection using machine learning approaches", International Journal of Recent Technology and Engineering, 2019, 7(5), pp. 478–481.