Parkinson's Disease Detection Using Different Machine Learning Algorithms

Chirag Mittal, Amritanshu Sharma

Electrical Engineering, Punjab Engineering College (Deemed to be University), Chandigarh

DOI: 10.29322/IJSRP.12.02.2022.p12205
http://dx.doi.org/10.29322/IJSRP.12.02.2022.p12205

***Abstract*–** Introduction:-Parkinson's Disease is a widespread degenerative syndrome that affects the nervous system. Its early symptoms include tremor, rigidity, and vocal impairment (dysphonia). This paper proposes performance of Machine Learning Methods in Diagnosing Parkinson's Disease. Several machine-learning techniques were considered and trained with the same data set to classify healthy and Parkinson's Disease patients.

Methods:-We have used various machine learning-based techniques for Parkinson's disease (PD) diagnosis. These machine learning techniques includes K Nearest Neighbors (k-NN), Naïve Bayes (NB), Decision Tree (DT), Support Vector Machine (SVM), Stochastic Gradient Descent (SGD), Random Forest and XGBoost. We have used these algorithms to check the best algorithm for detection of Parkinson's Dis- ease.

Result:-After applying these algorithms we found our accuracies of these algorithms are as follows: - Naive Bayes(71.79%), SGD(84.61%), Decision Tree(84.61%), KNN(97.43%), Random Forest(89.74%), SVM(87.18%) and XGBoost(94.87%).

Conclusion:-After considering all algorithms and analyzing their accuracies we found out that KNN is the best of all the algorithms used by us for detection of Parkinson Disease with accuracy of 97.43 percent.

I. INTRODUCTION

Parkinson's disease is a nervous system disease that affects your ability to control movement. The disease usually starts out slowly and worsens over time. If you have Parkinson's disease, you may shake, have muscle stiffness, and have trouble walking and maintaining your balance and coordination. As the disease worsens, you may have trouble talking, sleeping, have mental and memory problems, experience behavioral changes and have other symptoms.

About 50% more men than women get Parkinson's disease. It is most commonly seen in persons 60 years of age and older. However, up to 10% of patients are diagnosed before age 50.

About 60,000 new cases of Parkinson's disease are diagnosed in the United States each year.

Parkinson's disease occurs when nerve cells (neurons) in an area of the brain called the substantia nigra become impaired or die. These cells normally produce dopamine, a chemical (neurotransmitter) that helps the cells of the brain communicate (transmits signals, "messages," between areas in the brain). When these nerve cells become impaired or die, they produce less dopamine. Dopamine is especially important for the operation of another area of the brain called the basal ganglia. This area of the brain is responsible for organizing the brain's commands for body movement. The loss of dopamine causes the movement symptoms seen in people with Parkinson's disease.

People with Parkinson's disease also lose another neurotransmitter called norepinephrine. This chemical is needed for proper functioning of the sympathetic nervous system. This system controls some of the body's autonomic functions such as digestion, heart rate, blood pressure and breathing. Loss of norepinephrine causes some of the non-movement-related symptoms of Parkinson's disease.

Symptoms of PD include Tremor, bradykinesia, rigid muscles, speech changes, etc.

A. DATASET USED

1) Source: The dataset was created by Max Little of the University of Oxford, in collaboration with the National Centre for Voice and Speech, Denver, Colorado, who recorded the speech signals. The original study published the feature extraction methods for general voice disorders.

2) Data Set Information: This dataset is composed of a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD). Each column in the table is a particular voice measure, and each row corresponds one of 195 voice recording from these individuals ("name" column). The main aim of the data is to discriminate healthy people from those with PD, according to "status" column which is set to 0 for healthy and 1 for PD. The data is in ASCII CSV format. The rows of the CSV file contain an instance corresponding to one voice recording. There are around six recordings per patient, the name of the patient is identified in the first column.

3) Attribute Information: Matrix column entries (attributes/features):

name - ASCII subject name and recording number

MDVP:Fo(Hz) - Average vocal fundamental frequency
MDVP:Fhi(Hz) - Maximum vocal fundamental frequency
MDVP:Flo(Hz) - Minimum vocal fundamental frequency
MDVP:Jitter(%),MDVP:Jitter(Abs),MDVP:RAP,MDVP:PPQ Jitter:DDP - Several measures of variation in fundamental frequency
MDVP:Shimmer,MDVP:Shimmer(dB),Shimmer:APQ3 Shimmer:APQ5,MDVP:APQ,Shimmer:DDA - Several measures of variation in amplitude
NHR,HNR - Two measures of ratio of noise to tonal components in the voicestatus - Health status of the subject (one) - Parkinson's, (zero) - healthy
RPDE,D2 - Two nonlinear dynamical complexity measures
DFA - Signal fractal scaling exponentspread1,spread2,PPE - Three nonlinear measures of fundamental frequency variation
Link of the dataset used can be found below: -
https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	...	D2	PPE
count	195.000000	195.000000	...	195.000000	195.000000
mean	154.228641	197.104918	...	2.381826	0.206552
std	41.390065	91.491548	...	0.382799	0.090119
min	88.333000	102.145000	...	1.423287	0.044539
25%	117.572000	134.862500	...	2.099125	0.137451
50%	148.790000	175.829000	...	2.361532	0.194052
75%	182.769000	224.205500	...	2.636456	0.252980
max	260.105000	592.030000	...	3.671155	0.527367

Figure 1: **Description of dataset used which includes count of data points , mean, standard deviation etc.**

II. METHODS

A. DATA PREPROCESSING

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

In Figure 2 we have used SimpleImputer to replace the missing values with mean of whole column. This is used to deal with values which are not available to us. We can also use standard deviation, median or mode to replace the missing values.

In Figure 3 we have used MinMaxScaler for feature scaling , it helps in normalization of data. Normalization scales each input variable separately to the range 0-1, which is the range for floating-point values where we have the most precision.

A value is normalized as follows:

$y = (x – min) / (max – min)$

Where the minimum and maximum values pertain to the value x being normalized.

For example, for a dataset, we could guesstimate the min and max observable values as 30 and -10. We can then normalize any value, like 18.8, as follows:

$y = (x – min) / (max – min)y = (18.8 – (-10)) / (30 – (-10))y = 28.8 / 40y = 0.72$

```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer=imputer.fit(x[:,:])
x[:,:]=imputer.transform(x[:,:])
print(x)
```

```
[[-0.63138346 -0.77481654 -0.89037042 ...  0.17153026 -0.21867743
  -0.0053808 ]
 [-0.6033463  -0.81013911 -0.4433544  ...  0.48267409 -0.05370956
   0.34265204]
 [-0.66992292 -0.88174367 -0.46942324 ...  0.37274182 -0.18236124
   0.19336492]
 ...
 [ 0.00546073 -0.43717403 -0.89854572 ... -0.31484696  0.11793486
  -0.63884033]
 [ 0.28578581  0.20361309 -0.89144127 ... -0.09423055 -0.36355605
  -0.67372646]
 [ 0.46654868 -0.35441175 -0.85610326 ... -0.16981039  0.00734563
  -0.5690805 ]]
```

Figure 2: **Identify and replace missing values in each column of our dataset. The imputer is fit on the dataset to calculate the statistic for each column.**

```
scaler = MinMaxScaler()
scaled = scaler.fit_transform(x)
print(scaled)
```

```
[[0.18430827 0.11259173 0.05481479 ... 0.58576513 0.39066128 0.4973096 ]
 [0.19832685 0.09493044 0.2783228  ... 0.74133704 0.47314522 0.67132602]
 [0.16503854 0.05912816 0.26528838 ... 0.68637091 0.40881938 0.59668246]
 ...
 [0.50273036 0.28141298 0.05072714 ... 0.34257652 0.55896743 0.18057983]
 [0.6428929  0.60180655 0.05427936 ... 0.45288473 0.31822198 0.16313677]
 [0.73327434 0.32279413 0.07194837 ... 0.41509481 0.50367281 0.21545975]]
```

Figure 3: **MinMaxScaler scales and normalizes each feature individually such that it is in the given range on the training set between 0 and 1.**

B. SGD Algorithm

SGD, we find out the gradient of the cost function of a single example at each iteration instead of the sum of the gradient of the cost function of all the examples. In SGD, since only one sample from the dataset is chosen at random for each iteration, the path taken by the algorithm to reach the minima is usually noisier than your typical Gradient Descent algorithm. But that doesn't matter all that much because the path taken by the algorithm does not matter, as long as we reach the minima and with a significantly shorter training time.

One thing to be noted is that, as SGD is generally noisier than typical Gradient Descent, it usually took a higher number of iterations to reach the minima, because of its randomness in its descent. Even though it requires a higher number of iterations to reach the minima than typical Gradient De- scent, it is still computationally much less expensive than typical Gradient Descent. Hence, in most scenarios, SGD is preferred over

Batch Gradient Descent for optimizing a learning algorithm.

Figure 4 showing the code to train our model using SGD algorithm.

```
clf = SGDClassifier(loss="hinge", penalty="l2", max_iter=50)
clf.fit(x_train, y_train)
y_pred= clf.predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
cm= confusion_matrix(y_test, y_pred)
print(cm)
```

```
84.61538461538461
[[ 1  6]
 [ 0 32]]
```

Figure 4: **SGDClassifier implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties for classification and accuracy that we get is 84.61%.**

C. NAIVE BAYES

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features. For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting.

Figure 5 showing the code to train our model using Naïve Bayes algorithm.

```
gnb = GaussianNB()
y_pred = gnb.fit(x_train, y_train).predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
cm= confusion_matrix(y_test, y_pred)
print(cm)
```

```
71.7948717948718
[[ 5  2]
 [ 9 23]]
```

Figure 5: **In the above code, we have used the GaussianNB classifier to fit it to the training dataset. We can also use other classifiers as per our requirement and accuracy we get is 71.79%.**

D. K-NEAREST NEIGHBORS

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called train- ing data), which classifies coordinates into groups identified by an attribute.

Figure 6 showing the code to train our model using K-Nearest Neighbors algorithm.

```
classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
classifier.fit(x_train, y_train)
y_pred= classifier.predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
cm= confusion_matrix(y_test, y_pred)
print(cm)
```

```
97.43589743589743
[[ 6  1]
 [ 0 32]]
```

Figure 6: **In the above code, we have used the KNeighborsClassifier classifier to fit it to the training dataset and accuracy we get is 97.43%.**

E. DECISION TREE

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Construction of Decision Tree: A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Figure 7 showing the code to train our model using Decision Tree algorithm.

```
clf = tree.DecisionTreeClassifier()
clf.fit(x_train, y_train)
y_pred= clf.predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
cm= confusion_matrix(y_test, y_pred)
print(cm)
```

```
84.61538461538461
[[ 6  1]
 [ 5 27]]
```

Figure 7: **In the above code, we have used the DecisionTreeClassifier to fit it to the training dataset and accuracy we get is 84.61%.**

F. SUPPORT VECTOR MACHINES (SVM)

Support Vector Machine (SVM) is a relatively simple Su-

pervised Machine Learning Algorithm used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line.

In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyperplane to separate the data. So by this, you must have understood that inherently, SVM can only perform binary classification (i.e., choose between two classes). However, there are various techniques to use for multi-class problems.

Support Vector Machine for Multi-Class Problems

To perform SVM on multi-class problems, we can create a binary classifier for each class of the data. The two results of each classifier will be:

- The data point belongs to that class OR
- The data point does not belong to that class.

Figure 8 showing the code to train our model using Support Vector Machines algorithm.

```
clf = svm.SVC()
clf.fit(x_train, y_train)
y_pred= clf.predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
cm= confusion_matrix(y_test, y_pred)
print(cm)
```

```
87.17948717948718
[[ 2  5]
 [ 0 32]]
```

Figure 8: **In the above code, the training set will be fitted to the SVM classifier and accuracy we get is 87.18%.**

G. Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

Figure 9 showing the code to train our model using Random Forest algorithm.

```
clf = RandomForestClassifier(n_estimators = 100)
clf.fit(x_train, y_train)
y_pred= clf.predict(x_test)
print(accuracy_score(y_test, y_pred)*100)
cm= confusion_matrix(y_test, y_pred)
print(cm)
```

```
89.74358974358975
[[ 4  3]
 [ 1 31]]
```

Figure 9: **In the above code, the training set will be fitted to the RandomForestClassifier and accuracy we get is 89.74%.**

H. XGBoost

XGBoost is an algorithm. That has recently been dominating applied gadget learning. XGBoost set of rules is an implementation of gradient boosted choice timber. That changed into the design for pace and overall performance.

Figure 10 showing the code to train our model XGBoost algorithm.

```
model=XGBClassifier(eval_metric='mlogloss')

model.fit(x_train,y_train)
y_pred=model.predict(x_test)

print(accuracy_score(y_test, y_pred)*100)
cm= confusion_matrix(y_test, y_pred)
print(cm)
```

```
94.87179487179486
[[ 5  2]
 [ 0 32]]
```

Figure 10: **In the above code, the training set will be fitted to the XGBClassifier and accuracy we get is 94.87%.**

III. Result

We used various classification algorithms to detect whether a person is suffering from Parkinson's disease or not and there accuracy are as follows:- Naive Bayes(71.79%), SGD(84.61%), Decision Tree(84.61%), KNN(97.43%), Random Forest(89.74%), SVM(87.18%) and XGBoost(94.87%).

Table I: **Accuracy of different algorithms used for detection of Parkinson's Disease.**

S.no	Algorithm	Accuracy
1)	Naive bayes	71.79
2)	SGD	84.61
3)	Decision Tree	84.61
4)	SVM	87.18
5)	Random Forest	89.74

6)	XGBoost	94.87
7)	KNN	97.43

97.43 % to predict the onset of the disease which will enable early treatment and save a life. Figure 11 shows plot of testing values and Figure 12 shows plot of predicted values while using KNN algorithm to predict Parkinson's disease.

IV. Conclusion

Parkinson's disease affects the CNS of the brain and has yet no treatment unless it's detected early. Late detection leads to no treatment and loss of life. Thus, its early detection is significant. For early detection of the disease, we utilized various machine learning algorithms to detect Parkinson's disease. We checked our Parkinson disease data and found out that K-Nearest Neighbors is the best Algorithm with accuracy of 97.43% to predict the onset of the disease which will enable early treatment and save a life. Figure 11 shows plot of testing values and Figure 12 shows plot of predicted values while using KNN algorithm to predict Parkinson's disease.

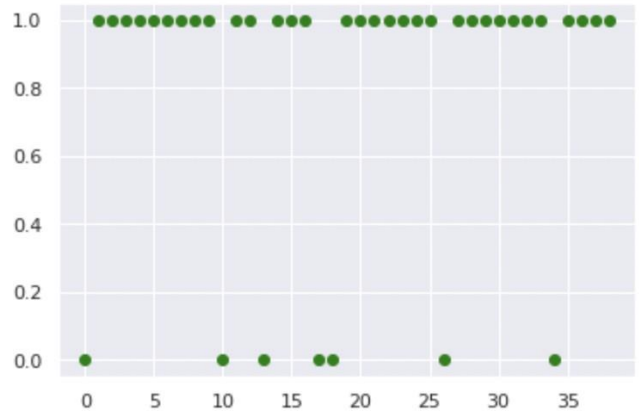

Figure 11: **Plot showing testing values while applying KNN algorithm.**

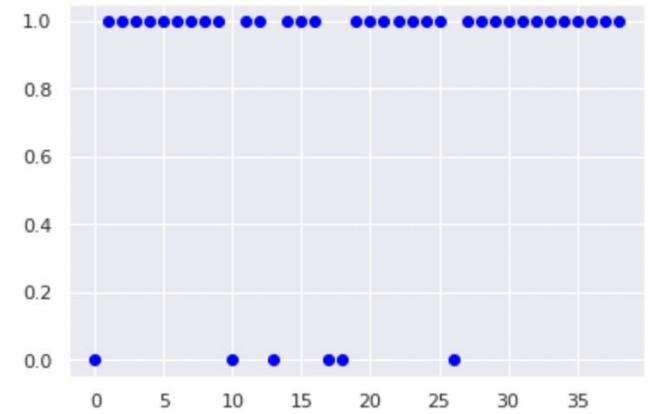

Figure 12: **Plot showing predicted values while applying KNN algorithm.**

V. References

1. https://www.nia.nih.gov/health/parkinsons-disease#:~:text=Parkinson's%20disease%20is%20a%20brain,have%20difficulty%20walking%20and%20talking. - Parkinson's Disease | National Institute on Aging
2. https://www.javatpoint.com/classification-algorithm-in- machine-learning - Classification Algorithm in Machine Learning – Javatpoint
3. https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/ - ML | Data Preprocessing in Python – GeeksforGeeks
4. cten Z, Burns SM, Menzin JA. Predictors of Parkinson disease in a Medicare population— An application of machine learning in early disease detection. Presented at: ISPOR 2020; May 18-20, 2020; Abstract AI2. https://bit.ly/3go1dh8
5. https://www.parkinson.org/understanding-parkinsons/10-early-warning-signs - 10 Early Signs of Parkinson's Disease
6. Little, M. A., McSharry, P. E., Roberts, S. J., Costello, D. A. E., & Moroz, I. M. (2007). Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. BioMedical Engineering OnLine, 6, Article: 23.

AUTHORS

First Author – Chirag Mittal, B.Tech., Electrical Engineering, Punjab Engineering College, Chandigarh and chi15.mittal@gmail.com

Second Author – Amritanshu Sharma, B.Tech., Electrical Engineering, Punjab Engineering College, Chandigarh and amritanshusharma25@gmail.com