14_Parkinsons_Disease_Detection_Using_SVM_A_Machine_Learning_Ap

July 10, 2024

0.1 Parkinson's Disease Detection Using SVM: A Machine Learning Approach-Vignesh Prabhu

Explore how Support Vector Machines (SVM) enhance the accuracy of Parkinson's disease detection. This project leverages SVM, a powerful machine learning algorithm, to analyze clinical data and predict the presence of Parkinson's disease with high precision. Discover the intersection of machine learning and healthcare in advancing diagnostic capabilities.

Import Dependencies

[1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Data Collection and PreProcessing

[2]:

```python
#Load the data To DataFrame
parkinsons_data = pd.read_csv('/content/parkinsons.csv')
```

[3]:

```python
#To Print First 5 data's in Dataset
parkinsons_data.head()
```

[3]:

```
             name  MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0  phon_R01_S01_1      119.992       157.302        74.997         0.00784
1  phon_R01_S01_2      122.400       148.650       113.819         0.00968
2  phon_R01_S01_3      116.682       131.111       111.555         0.01050
3  phon_R01_S01_4      116.676       137.871       111.366         0.00997
4  phon_R01_S01_5      116.014       141.781       110.655         0.01284

   MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  …  \
0           0.00007   0.00370   0.00554     0.01109       0.04374  …
1           0.00008   0.00465   0.00696     0.01394       0.06134  …
2           0.00009   0.00544   0.00781     0.01633       0.05233  …
```

```
3           0.00009    0.00502    0.00698     0.01505       0.05492  …
4           0.00011    0.00655    0.00908     0.01966       0.06425  …

   Shimmer:DDA      NHR     HNR  status      RPDE       DFA   spread1  \
0      0.06545  0.02211  21.033       1  0.414783  0.815285 -4.813031
1      0.09403  0.01929  19.085       1  0.458359  0.819521 -4.075192
2      0.08270  0.01309  20.651       1  0.429895  0.825288 -4.443179
3      0.08771  0.01353  20.644       1  0.434969  0.819235 -4.117501
4      0.10470  0.01767  19.649       1  0.417356  0.823484 -3.747787

    spread2        D2       PPE
0  0.266482  2.301442  0.284654
1  0.335590  2.486855  0.368674
2  0.311173  2.342259  0.332634
3  0.334147  2.405554  0.368975
4  0.234513  2.332180  0.410335

[5 rows x 24 columns]
```

[4]:
```
#To check Number of Rows and Columns
parkinsons_data.shape
```

[4]:
```
(195, 24)
```

[5]:
```
# To Check Null values
parkinsons_data.isnull().sum()
```

[5]:
```
name                0
MDVP:Fo(Hz)         0
MDVP:Fhi(Hz)        0
MDVP:Flo(Hz)        0
MDVP:Jitter(%)      0
MDVP:Jitter(Abs)    0
MDVP:RAP            0
MDVP:PPQ            0
Jitter:DDP          0
MDVP:Shimmer        0
MDVP:Shimmer(dB)    0
Shimmer:APQ3        0
Shimmer:APQ5        0
MDVP:APQ            0
Shimmer:DDA         0
NHR                 0
HNR                 0
status              0
RPDE                0
DFA                 0
```

```
spread1             0
spread2             0
D2                  0
PPE                 0
dtype: int64
```

[6]:

```
#To check Complete Information
parkinsons_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   name              195 non-null    object
 1   MDVP:Fo(Hz)       195 non-null    float64
 2   MDVP:Fhi(Hz)      195 non-null    float64
 3   MDVP:Flo(Hz)      195 non-null    float64
 4   MDVP:Jitter(%)    195 non-null    float64
 5   MDVP:Jitter(Abs)  195 non-null    float64
 6   MDVP:RAP          195 non-null    float64
 7   MDVP:PPQ          195 non-null    float64
 8   Jitter:DDP        195 non-null    float64
 9   MDVP:Shimmer      195 non-null    float64
 10  MDVP:Shimmer(dB)  195 non-null    float64
 11  Shimmer:APQ3      195 non-null    float64
 12  Shimmer:APQ5      195 non-null    float64
 13  MDVP:APQ          195 non-null    float64
 14  Shimmer:DDA       195 non-null    float64
 15  NHR               195 non-null    float64
 16  HNR               195 non-null    float64
 17  status            195 non-null    int64
 18  RPDE              195 non-null    float64
 19  DFA               195 non-null    float64
 20  spread1           195 non-null    float64
 21  spread2           195 non-null    float64
 22  D2                195 non-null    float64
 23  PPE               195 non-null    float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.7+ KB
```

[7]:

```
#statistical measures
parkinsons_data.describe()
```

[7]:

```
       MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
count   195.000000    195.000000    195.000000      195.000000
mean    154.228641    197.104918    116.324631        0.006220
```

```
std       41.390065    91.491548    43.521413        0.004848
min       88.333000   102.145000    65.476000        0.001680
25%      117.572000   134.862500    84.291000        0.003460
50%      148.790000   175.829000   104.315000        0.004940
75%      182.769000   224.205500   140.018500        0.007365
max      260.105000   592.030000   239.170000        0.033160

       MDVP:Jitter(Abs)    MDVP:RAP    MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  \
count        195.000000  195.000000  195.000000  195.000000    195.000000
mean           0.000044    0.003306    0.003446    0.009920      0.029709
std            0.000035    0.002968    0.002759    0.008903      0.018857
min            0.000007    0.000680    0.000920    0.002040      0.009540
25%            0.000020    0.001660    0.001860    0.004985      0.016505
50%            0.000030    0.002500    0.002690    0.007490      0.022970
75%            0.000060    0.003835    0.003955    0.011505      0.037885
max            0.000260    0.021440    0.019580    0.064330      0.119080

       MDVP:Shimmer(dB)  …  Shimmer:DDA         NHR         HNR      status  \
count        195.000000  …   195.000000  195.000000  195.000000  195.000000
mean           0.282251  …     0.046993    0.024847   21.885974    0.753846
std            0.194877  …     0.030459    0.040418    4.425764    0.431878
min            0.085000  …     0.013640    0.000650    8.441000    0.000000
25%            0.148500  …     0.024735    0.005925   19.198000    1.000000
50%            0.221000  …     0.038360    0.011660   22.085000    1.000000
75%            0.350000  …     0.060795    0.025640   25.075500    1.000000
max            1.302000  …     0.169420    0.314820   33.047000    1.000000

             RPDE         DFA     spread1     spread2          D2         PPE
count  195.000000  195.000000  195.000000  195.000000  195.000000  195.000000
mean     0.498536    0.718099   -5.684397    0.226510    2.381826    0.206552
std      0.103942    0.055336    1.090208    0.083406    0.382799    0.090119
min      0.256570    0.574282   -7.964984    0.006274    1.423287    0.044539
25%      0.421306    0.674758   -6.450096    0.174351    2.099125    0.137451
50%      0.495954    0.722254   -5.720868    0.218885    2.361532    0.194052
75%      0.587562    0.761881   -5.046192    0.279234    2.636456    0.252980
max      0.685151    0.825288   -2.434031    0.450493    3.671155    0.527367

[8 rows x 23 columns]
```

[8]:

```
#Distribution Of Target
parkinsons_data['status'].value_counts() #1 for parkinsons , 0 - Without␣
 ↪Parkinsons
```

[8]:

```
status
1    147
0     48
Name: count, dtype: int64
```

Spliting data Into Feature and Target

[9]:
```
X= parkinsons_data.drop(columns=['name','status'],axis=1)
Y= parkinsons_data['status']
```

[10]:
```
print(X)
```

```
     MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0        119.992       157.302        74.997         0.00784
1        122.400       148.650       113.819         0.00968
2        116.682       131.111       111.555         0.01050
3        116.676       137.871       111.366         0.00997
4        116.014       141.781       110.655         0.01284
..           …            …           …             …
190      174.188       230.978        94.261         0.00459
191      209.516       253.017        89.488         0.00564
192      174.688       240.005        74.287         0.01360
193      198.764       396.961        74.904         0.00740
194      214.289       260.277        77.973         0.00567

     MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  \
0             0.00007   0.00370   0.00554     0.01109       0.04374
1             0.00008   0.00465   0.00696     0.01394       0.06134
2             0.00009   0.00544   0.00781     0.01633       0.05233
3             0.00009   0.00502   0.00698     0.01505       0.05492
4             0.00011   0.00655   0.00908     0.01966       0.06425
..                …        …        …          …           …
190           0.00003   0.00263   0.00259     0.00790       0.04087
191           0.00003   0.00331   0.00292     0.00994       0.02751
192           0.00008   0.00624   0.00564     0.01873       0.02308
193           0.00004   0.00370   0.00390     0.01109       0.02296
194           0.00003   0.00295   0.00317     0.00885       0.01884

     MDVP:Shimmer(dB)  …  MDVP:APQ  Shimmer:DDA      NHR     HNR      RPDE  \
0               0.426  …   0.02971      0.06545  0.02211  21.033  0.414783
1               0.626  …   0.04368      0.09403  0.01929  19.085  0.458359
2               0.482  …   0.03590      0.08270  0.01309  20.651  0.429895
3               0.517  …   0.03772      0.08771  0.01353  20.644  0.434969
4               0.584  …   0.04465      0.10470  0.01767  19.649  0.417356
..                …   …       …            …        …       …        …
190             0.405  …   0.02745      0.07008  0.02764  19.517  0.448439
191             0.263  …   0.01879      0.04812  0.01810  19.147  0.431674
192             0.256  …   0.01667      0.03804  0.10715  17.883  0.407567
193             0.241  …   0.01588      0.03794  0.07223  19.020  0.451221
194             0.190  …   0.01373      0.03078  0.04398  21.209  0.462803

          DFA   spread1   spread2        D2       PPE
0    0.815285 -4.813031  0.266482  2.301442  0.284654
```

```
1    0.819521 -4.075192  0.335590  2.486855  0.368674
2    0.825288 -4.443179  0.311173  2.342259  0.332634
3    0.819235 -4.117501  0.334147  2.405554  0.368975
4    0.823484 -3.747787  0.234513  2.332180  0.410335
..        …         …         …         …         …
190  0.657899 -6.538586  0.121952  2.657476  0.133050
191  0.683244 -6.195325  0.129303  2.784312  0.168895
192  0.655683 -6.787197  0.158453  2.679772  0.131728
193  0.643956 -6.744577  0.207454  2.138608  0.123306
194  0.664357 -5.724056  0.190667  2.555477  0.148569

[195 rows x 22 columns]
```

[11]:

```
print(Y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
190    0
191    0
192    0
193    0
194    0
Name: status, Length: 195, dtype: int64
```

Spilit Data Into Training and Testing

[12]:

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.
  ↪2,random_state=2)
```

[13]:

```
print(X.shape,X_train.shape,X_test.shape)
```

```
(195, 22) (156, 22) (39, 22)
```

Data Standardization

[14]:

```
scaler = StandardScaler()
```

[15]:

```
scaler.fit(X_train)
```

[15]:

```
StandardScaler()
```

[16]:

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

[17]:

```
print(X_train)
```

```
[[ 0.63239631 -0.02731081 -0.87985049 … -0.97586547 -0.55160318
   0.07769494]
 [-1.05512719 -0.83337041 -0.9284778  …  0.3981808  -0.61014073
   0.39291782]
 [ 0.02996187 -0.29531068 -1.12211107 … -0.43937044 -0.62849605
  -0.50948408]
 …
 [-0.9096785  -0.6637302  -0.160638   …  1.22001022 -0.47404629
  -0.2159482 ]
 [-0.35977689  0.19731822 -0.79063679 … -0.17896029 -0.47272835
   0.28181221]
 [ 1.01957066  0.19922317 -0.61914972 … -0.716232    1.23632066
  -0.05829386]]
```

Model Training

[18]:

```
svm_model = svm.SVC(kernel='linear')
```

[19]:

```
#Training The SVM model with training data
svm_model.fit(X_train,Y_train)
```

[19]:

```
SVC(kernel='linear')
```

Model Evaluation

[20]:

```
#accuracy Score on Training data
X_train_prediction = svm_model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train,X_train_prediction)
```

[21]:

```
print('Accuracy Score of Training Data : ',training_data_accuracy)
```

```
Accuracy Score of Training Data :  0.8846153846153846
```

[22]:

```
#accuracy Score on Test data
X_test_prediction = svm_model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test,X_test_prediction)
```

[23]:

```
print('Accuracy Score of Test Data : ',test_data_accuracy)
```

```
Accuracy Score of Test Data :  0.8717948717948718
```

Building a Predictive System

[24]:

```
#input_data = (95.730,132.068,91.754,0.00551,0.00006,0.00293,0.00332,0.00880,0.
 ↪02093,0.191,0.01073,0.01277,0.01717,0.03218,0.01070,21.812,0.615551,0.
 ↪773587,-5.498678,0.327769,2.322511,0.231571)
input_data = (197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.
 ↪00498,0.01098,0.09700,0.00563,0.00680,0.00802,0.01689,0.00339,26.77500,0.
 ↪422229,0.741367,-7.348300,0.177551,1.743867,0.085569)
```

```python
#Changing input data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

#Reshape the numpy array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

#Standardize the data
std_data = scaler.transform(input_data_reshaped)

prediction = svm_model.predict(std_data)
print(prediction)
if (prediction[0] == 0):
  print("The Person does not have Parkinsons Disease")

else:
  print("The Person has Parkinsons")
```

```
[0]
The Person does not have Parkinsons Disease
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

Our SVM model effectively detects Parkinson's disease, showcasing the potential of machine learning for early diagnosis and better patient outcomes. Proper data preprocessing and feature scaling were crucial for its success.

Thank You!