Parkinson's Disease Detection Using Machine Learning

June 1, 2024

0.0.1 Project Introduction:

This project aims to develop a machine learning model to predict whether a person has Parkinson's Disease based on vocal features extracted from their voice recordings. Utilizing the Parkinson's Disease Dataset from the UCI Machine Learning Repository, we will preprocess the data, engineer features, and train a Random Forest classifier. The model's performance will be evaluated using various metrics to ensure its accuracy and reliability. By providing a scalable and efficient screening tool, this project aspires to aid in the early diagnosis and monitoring of Parkinson's Disease, potentially improving patient outcomes and reducing the burden on healthcare systems.

[1]:
```
#Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
```

[9]:
```
# Loading the dataset
data = pd.read_csv('Parkinsson disease.csv')
```

[10]:
```
print(data.head())
```

```
             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]
```

[14]:
```
#missing values
print(data.isnull().sum())
```

```
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
```

[15]:
```
print(data.describe())
```

```
       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]
```

[16]:
```
#Extracting features and lebels:
x = data.drop(['name', 'status'], axis =1)
y = data['status']
```

[17]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2 ,␣
↪random_state = 42)
```

[20]:
```
# Scaling the features
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

[21]:
```
# Training a Random Forest Classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(x_train, y_train)
```

[21]:
```
RandomForestClassifier(random_state=42)
```

[23]:
```
# Making predictions
y_pred = clf.predict(x_test)
```

[24]:
```
# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

[26]:
```
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.9487179487179487
```

[27]:
```
print('Confusion Matrix:')
print(conf_matrix)
```

```
Confusion Matrix:
[[ 5  2]
 [ 0 32]]
```

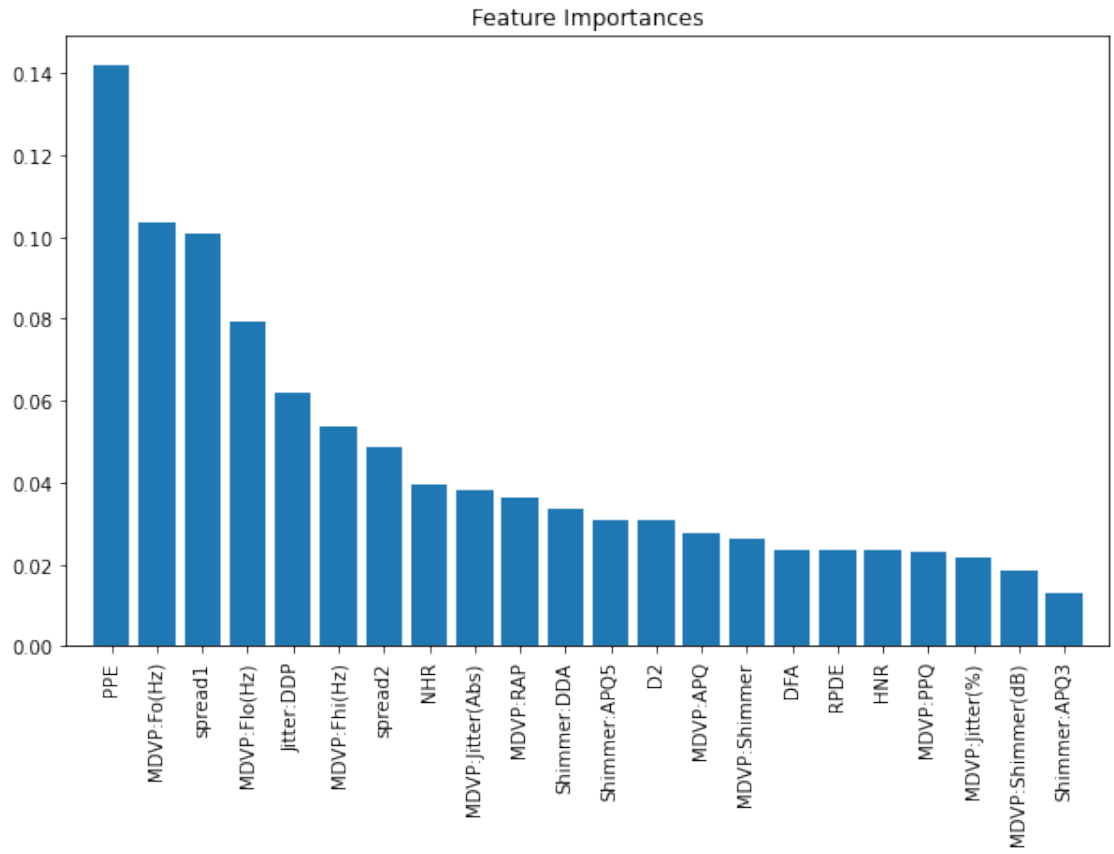
[28]:
```
print('Classification Report:')
print(class_report)
```

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

           0       1.00      0.71      0.83         7
           1       0.94      1.00      0.97        32

    accuracy                           0.95        39
   macro avg       0.97      0.86      0.90        39
weighted avg       0.95      0.95      0.95        39
```

0.0.2 In the next two notebooks I am plotting Feature importance to help us understand which features (input variables) contribute the most to the prediction accuracy of the model

[30]:
```
# Plot feature importances
importances = clf.feature_importances_
indices = np.argsort(importances)[::-1]
features = x.columns
```

[32]:
```
plt.figure(figsize=(10, 6))
plt.title("Feature Importances")
plt.bar(range(x.shape[1]), importances[indices], align="center")
plt.xticks(range(x.shape[1]), features[indices], rotation=90)
plt.xlim([-1, x.shape[1]])
plt.show()
```



Author: Bahraleloom Abdalrahem **Email:** bahraleloom@gmail.com **GitHub:** https://github.com/Bahraleloom **Kaggle:** https://www.kaggle.com/bahraleloom **Date:** 01/06/2024