

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
```

```
# loading the data from csv file to a Pandas DataFrame
parkinsons_data = pd.read_csv('C:\Sunaina\Desktop\Diseas Prediction app\dataset\parkinsons.csv')
```

```
<>:2: SyntaxWarning: invalid escape sequence '\S'
<>:2: SyntaxWarning: invalid escape sequence '\S'
C:\Users\sairam\AppData\Local\Temp\ipykernel_3412\2933840527.py:2: SyntaxWarning: invalid escape sequence '\S'
parkinsons_data = pd.read_csv('C:\Sunaina\Desktop\Diseas Prediction app\dataset\parkinsons.csv')
```

```
# printing the first 5 rows of the dataframe
parkinsons_data.head()
```

```

      name  MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer
0  phon_R01_S01_1      119.992      157.302       74.997        0.00784         0.00007      0.00370      0.00554      0.01109         0.043
1  phon_R01_S01_2      122.400      148.650      113.819        0.00968         0.00008      0.00465      0.00696      0.01394         0.061
2  phon_R01_S01_3      116.682      131.111      111.555        0.01050         0.00009      0.00544      0.00781      0.01633         0.052
3  phon_R01_S01_4      116.676      137.871      111.366        0.00997         0.00009      0.00502      0.00698      0.01505         0.054
4  phon_R01_S01_5      116.014      141.781      110.655        0.01284         0.00011      0.00655      0.00908      0.01966         0.064
5 rows x 24 columns
```

```
# number of rows and columns in the dataframe
parkinsons_data.shape
```

```
(195, 24)
```

```
# getting more information about the dataset
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
```

```
# checking for missing values in each column
parkinsons_data.isnull().sum()
```

```
name      0
MDVP:F0(Hz)  0
MDVP:F1(Hz)  0
MDVP:F1o(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
```

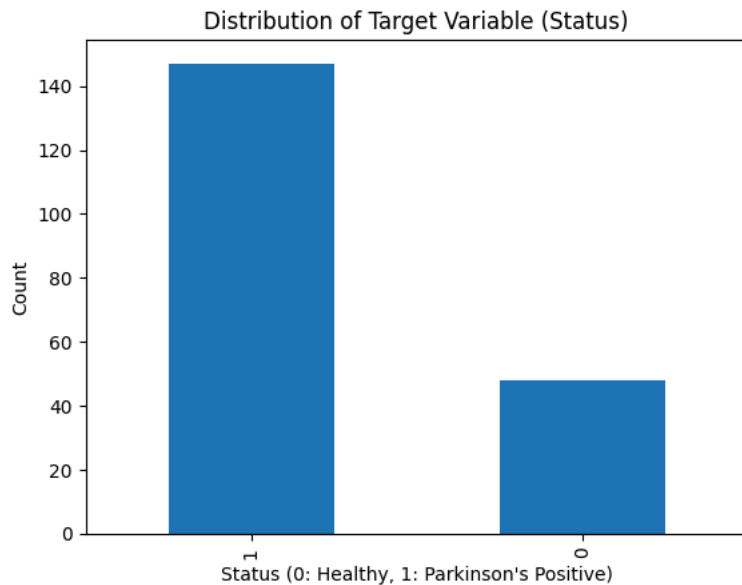
```
# getting some statistical measures about the data
parkinsons_data.describe()
```

```
count    195.000000  195.000000  195.000000  195.000000  195.000000  195.000000  195.000000  195.000000  195.000000  195.000000
mean     154.228641  197.104918  116.324631   0.006220   0.000044   0.003306   0.003446   0.009920   0.029709
std       41.390065   91.491548   43.521413   0.004848   0.000035   0.002968   0.002759   0.008903   0.018857
min       88.333000  102.145000   65.476000   0.001680   0.000007   0.000680   0.000920   0.002040   0.009540
25%      117.572000  134.862500   84.291000   0.003460   0.000020   0.001660   0.001860   0.004985   0.016505
50%      148.790000  175.829000  104.315000   0.004940   0.000030   0.002500   0.002690   0.007490   0.022970
75%      182.769000  224.205500  140.018500   0.007365   0.000060   0.003835   0.003955   0.011505   0.037885
max      260.105000  592.030000  239.170000   0.033160   0.000260   0.021440   0.019580   0.064330   0.119080
```

8 rows × 23 columns

```
# distribution of target Variable
parkinsons_data['status'].value_counts()
```

```
parkinsons_data['status'].value_counts().plot(kind='bar')
plt.title('Distribution of Target Variable (Status)')
plt.xlabel('Status (0: Healthy, 1: Parkinson\'s Positive)')
plt.ylabel('Count')
plt.show()
```



```
# grouping the data based on the target variable
# Grouping by 'status' and calculating the mean only for numeric columns
numeric_data = parkinsons_data.select_dtypes(include=[np.number])
mean_values_by_status = numeric_data.groupby(parkinsons_data['status']).mean()
print(mean_values_by_status)
```



```
MDVP:F0(Hz) MDVP:F1(Hz) MDVP:F2(Hz) MDVP:Jitter(%) \
status
0 181.937771 223.636750 145.207292 0.003866
1 145.180762 188.441463 106.893558 0.006989

MDVP:Jitter(Abs) MDVP:RAP MDVP:PPQ Jitter:DDP MDVP:Shimmer \
status
0 0.000023 0.001925 0.002056 0.005776 0.017615
1 0.000051 0.003757 0.003900 0.011273 0.033658

MDVP:Shimmer(dB) ... Shimmer:DDA NHR HNR status \
status
0 0.162958 ... 0.028511 0.011483 24.678750 0.0
1 0.321204 ... 0.053027 0.029211 20.974048 1.0

RPDE DFA spread1 spread2 D2 PPE
status
0 0.442552 0.695716 -6.759264 0.160292 2.154491 0.123017
1 0.516816 0.725408 -5.333420 0.248133 2.456058 0.233828

[2 rows x 23 columns]
```

Data Pre-Processing

Separating the features & Target

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

```
print(X)
```



```
MDVP:F0(Hz) MDVP:F1(Hz) MDVP:F2(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]

```
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
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

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

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

Model Training

Support Vector Machine Model

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

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

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

Model Evaluation

Accuracy Score

```
# 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)

print('Accuracy score of test data : ', test_data_accuracy)
```

```
→ Accuracy score of test data : 0.8717948717948718
```

```
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,

# changing input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

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

prediction = model.predict(input_data_resaped)
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
C:\Users\sairam\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, bu
warnings.warn(
```

```
import pickle
```

```
filename = 'parkinsons_model.pkl'
pickle.dump(model, open(filename, 'wb'))
```

```
# loading the saved model
loaded_model = pickle.load(open('parkinsons_model.sav', 'rb'))
```

```
for column in X.columns:
    print(column)
```

```
→ MDVP:F0(Hz)
MDVP:Fhi(Hz)
MDVP:Flo(Hz)
MDVP:Jitter(%)
MDVP:Jitter(Abs)
MDVP:RAP
MDVP:PPQ
Jitter:DDP
MDVP:Shimmer
MDVP:Shimmer(dB)
Shimmer:APQ3
Shimmer:APQ5
MDVP:APQ
Shimmer:DDA
NHR
HNR
RPDE
```

DFA
spread1
spread2
D2
PPE

```
from sklearn.ensemble import RandomForestClassifier
```

```
rfc = RandomForestClassifier(criterion = 'gini', max_depth = 7, max_features = 'sqrt', min_samples_leaf = 2, min_samples_split = 4, n_estimators = 180)
rfc.fit(X_train, Y_train)
```

```
y_pred5 = rfc.predict(X_test)
```

```
acc5=accuracy_score(Y_test,Y_pred5)
accuracies['RF']=acc5*100
```

```
print("Training accuracy score of the model is:",accuracy_score(Y_train, rfc.predict(X_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(Y_test,Y_pred5)*100,"%")
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[4], line 4
      1 from sklearn.ensemble import RandomForestClassifier
      3 rfc = RandomForestClassifier(criterion = 'gini', max_depth = 7, max_features = 'sqrt', min_samples_leaf = 2, min_samples_split
= 4, n_estimators = 180)
----> 4 rfc.fit(X_train, Y_train)
      6 y_pred5 = rfc.predict(X_test)
      8 acc5=accuracy_score(Y_test,Y_pred5)

NameError: name 'X_train' is not defined
```

```
import pickle
model = svc
pickle.dump(model, open("parkinsons.pkl", 'wb'))
```