```
#importing the libraries
import pandas as pd
import numpy as np
import seaborn as sns
import os, sys
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn import model_selection

from xgboost import XGBClassifier


# reading the file data
df=pd.read_csv('/content/parkinsons.data')
df.head()
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	M
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	

```
# the number of rows and columns
print(f"The shape of the DatFrame is: {df.shape}, which means there are {df.sha
```

```
The shape of the DatFrame is: (195, 24), which means there are 195 rows and
```

```
df.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
```

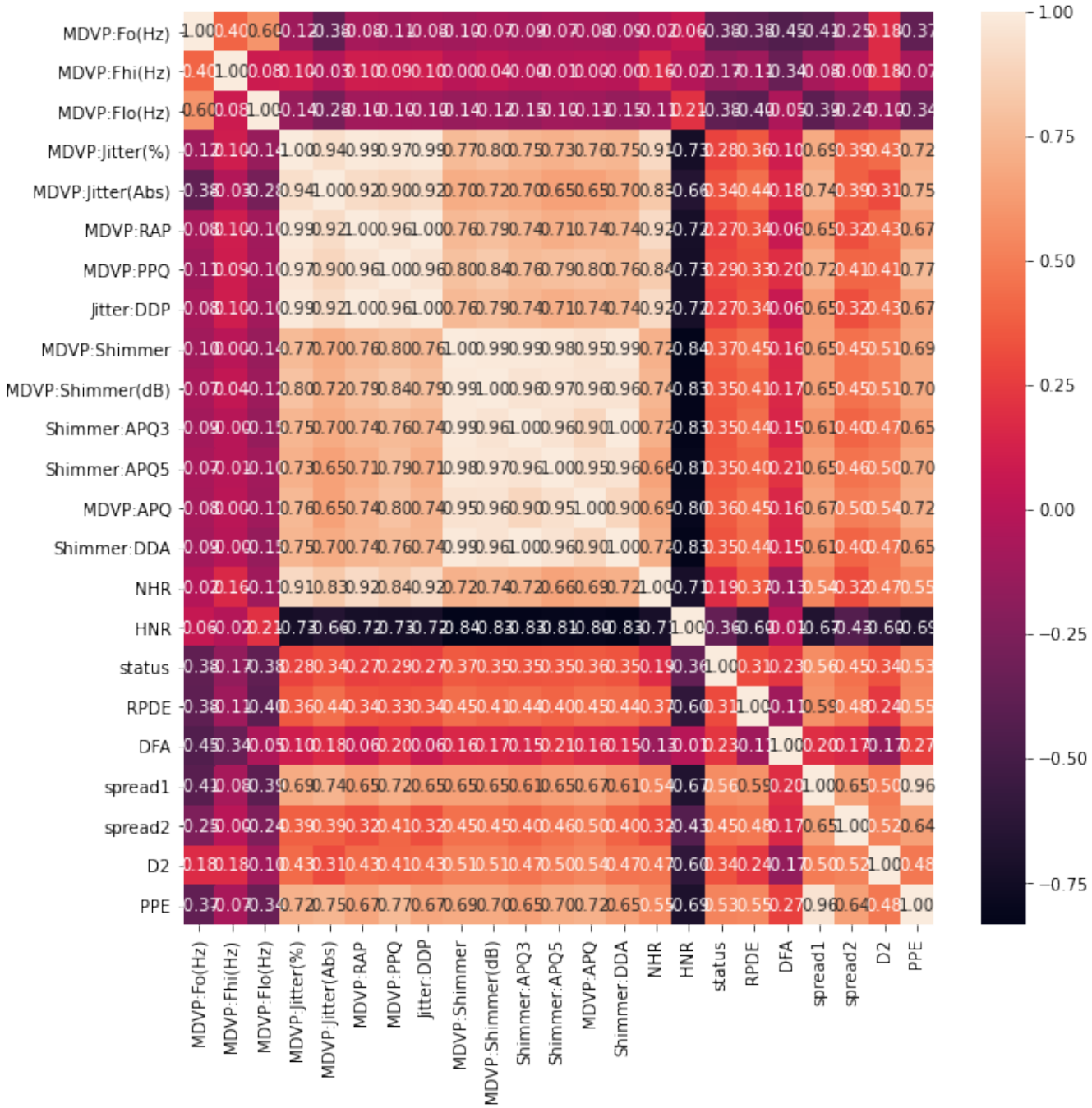
```
# to check if any of the columns have null values.
print(df.isnull().sum())
```

```
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
```

```
df_summary = df.describe()
df_summary
```

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(A
count	195.000000	195.000000	195.000000	195.000000	195.00
mean	154.228641	197.104918	116.324631	0.006220	0.00
std	41.390065	91.491548	43.521413	0.004848	0.00
min	88.333000	102.145000	65.476000	0.001680	0.00
25%	117.572000	134.862500	84.291000	0.003460	0.00
50%	148.790000	175.829000	104.315000	0.004940	0.00
75%	182.769000	224.205500	140.018500	0.007365	0.00
max	260.105000	592.030000	239.170000	0.033160	0.00

```
# heat map using correlation
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(), fmt='.2f', annot=True);
```



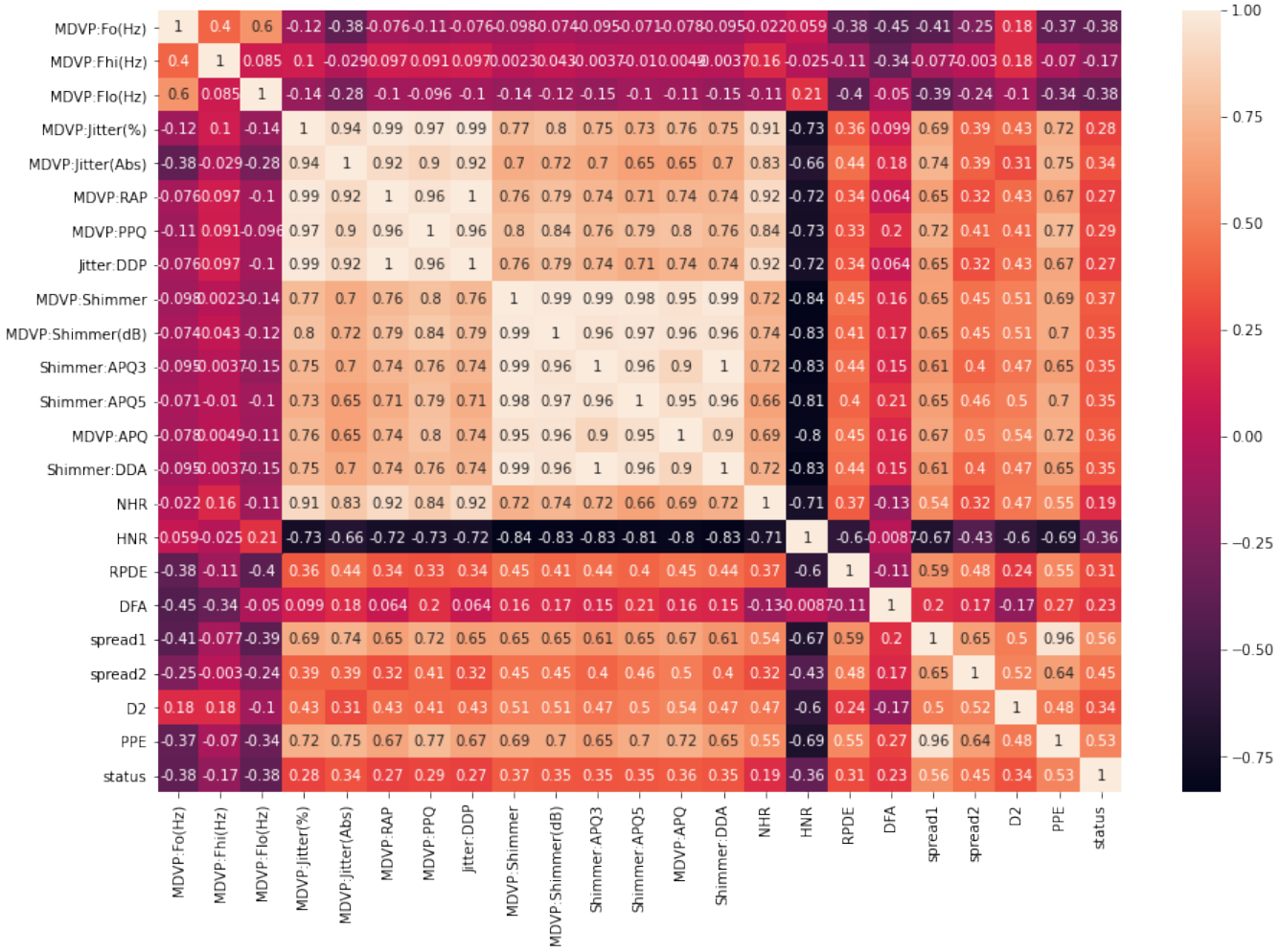
```
df[["MDVP:Fo(Hz)", "MDVP:Jitter(Abs)", "status"]]
```

	MDVP:Fo(Hz)	MDVP:Jitter(Abs)	status
0	119.992	0.00007	1
1	122.400	0.00008	1
2	116.682	0.00009	1
3	116.676	0.00009	1
4	116.014	0.00011	1
...	...	...	...
190	174.188	0.00003	0
191	209.516	0.00003	0
192	174.688	0.00008	0
193	198.764	0.00004	0
194	214.289	0.00003	0

195 rows × 3 columns

```
df_train = df.copy().drop(columns=["name"])
col_names = df_train.columns.tolist()
target_col = ["status"]
col_names.remove(target_col[0])
df_train = df_train[col_names + target_col]
```

```
plt.figure(figsize=(15,10))
corr = df_train.corr()
sns.heatmap(corr, annot=True, fmt='.2g');
```




```
# Get the features and labels
features = df.loc[:,df.columns != 'status'].values[:,1:]
labels=df.loc[:,'status'].values
```

```
# Get the label of each label (0 and 1) in labels
print(labels[labels==1].shape[0], labels[labels==0].shape[0])
```

```
147 48
```

Using the XGBoost Algorithm

```
# Scale the features to between -1 and 1
scaler=MinMaxScaler((-1,1))
x=scaler.fit_transform(features)
y=labels
```

```
# Spliting the dataset into train and test
x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.2, random_stat
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

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

```
#Training the model for XGboost
model=XGBClassifier()
model.fit(x_train,y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

```
#Calculating the accuracy
y_pred=model.predict(x_test)
print(accuracy_score(y_test, y_pred))
```

```
0.9487179487179487
```

```
# confusion matrix
from sklearn.metrics import confusion_matrix
pd.DataFrame(
    confusion_matrix(y_test, y_pred),
    columns=['Predicted Healthy', 'Predicted Parkinsons'],
    index=['True Healthy', 'True Parkinsons']
)
```

	Predicted Healthy	Predicted Parkinsons
True Healthy	5	2
True Parkinsons	0	32

Using KNN Algorithm

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
x_train = pca.fit_transform(x_train)
x_test = pca.transform(x_test)
variance = pca.explained_variance_ratio_
classifi = KNeighborsClassifier(n_neighbors = 8,p=2,metric ='minkowski')
classifi.fit(x_train,y_train)
y_pred = classifi.predict(x_test)
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
accuracy_score(y_test,y_pred)
```

```
0.8974358974358975
```

```
from sklearn.metrics import confusion_matrix
pd.DataFrame(
confusion_matrix(y_test, y_pred),
columns=['Predicted Healthy', 'Predicted Parkinsons'],
index=['True Healthy', 'True Parkinsons']
)
```

	Predicted Healthy	Predicted Parkinsons
True Healthy	4	3
True Parkinsons	1	31

Using Random Forest Algorithm

```
X = df.drop('status', axis=1)
X = X.drop('name', axis=1)
y = df['status']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=30, max_depth=10, random_st
```

```
random_forest.fit(X_train, y_train)
from sklearn.metrics import accuracy_score
y_predict = random_forest.predict(X_test)
accuracy_score(y_test, y_predict)
```

```
    0.9387755102040817
```

```
from sklearn.metrics import confusion_matrix
pd.DataFrame(
confusion_matrix(y_test, y_predict),
columns=['Predicted Healthy', 'Predicted Parkinsons'],
index=['True Healthy', 'True Parkinsons']
)
```

	Predicted Healthy	Predicted Parkinsons
True Healthy	11	1
True Parkinsons	2	35

Conclusion

Parkinsons Disease affect the Central Nervous System of Brain, and unless it detected early similarly treatement and precautions can be taken. For early detection we used the machine learning algorithms such as XGBoost, KNN, Random Forest. We also further check the accuracy which XGBoost gives us the best score and enable to detect the disease early and treatement can be started.

✓ 0s completed at 9:06 AM