PARKINSON'S PROJECT

Objective:About parkinson's project using python and machine learning.

We build a project using machine learning by downloading dataset(i.e par.csv) from kaggle.

IMPORTING LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
from pandas import Series,DataFrame
import random
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import learning_curve, validation_curve
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from sklearn.metrics import make_scorer, accuracy_score
from sklearn.metrics import roc_curve, roc_auc_score ,auc, plot_roc_curve
from sklearn import svm
import sklearn.metrics
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
```

EDA

READ THE DATASET

```
par=pd.read_csv('par.csv')
par.head(8)
```

	id	gender	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses
0	0	1	0.57227	240	239	0.008064
1	0	1	0.53966	234	233	0.008258
2	0	1	0.58982	232	231	0.008340
3	1	0	0.59257	178	177	0.010858
4	1	0	0.53028	236	235	0.008162
5	1	0	0.65451	226	221	0.007631
6	2	1	0.54543	322	321	0.005991

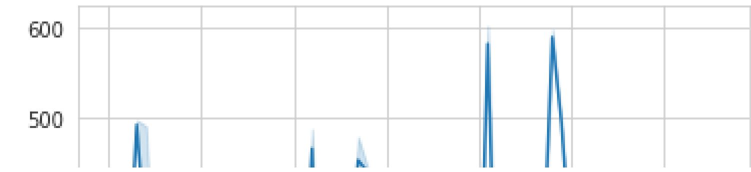
```
par.isnull()
```

	id	gender	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
194	False	False	False	False	False	False
195	False	False	False	False	False	False
196	False	False	False	False	False	False
197	False	False	False	False	False	False
198	False	False	False	False	False	False

199 rows × 6 columns

LINE PLOT

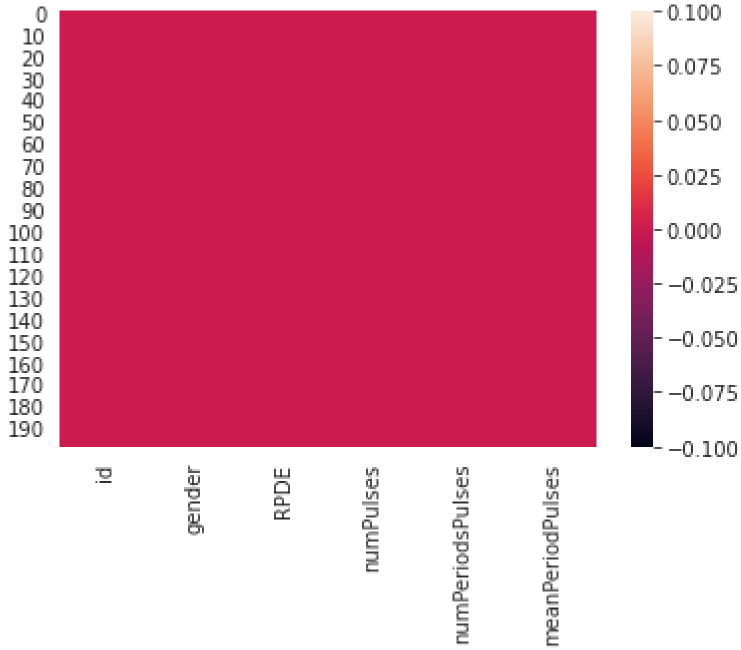
```
sns.lineplot(data=par,x="id",y="numPulses")
plt.show()
```



HEAT MAP

```
sns.heatmap(par.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb590068910>
```



BOX PLOT

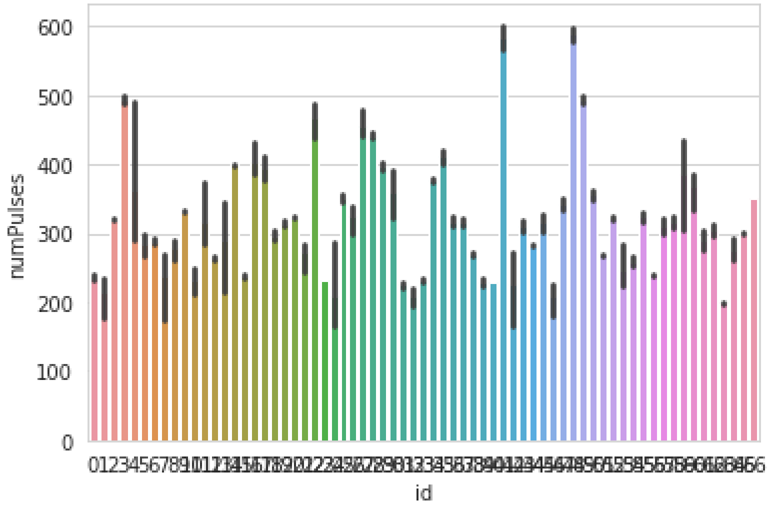
```
plt.figure(figsize=(10,8))
sns.boxplot(x='id',y='numPulses',data=par,palette='Dark2_r')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb58735aa90>
```

BAR PLOT

```
sns.barplot(data=par,x='id',y='numPulses')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb586d2e950>
```



SCATTER PLOT

```
plt.figure(figsize=(6,6))
sns.scatterplot(x='id',y='numPulses',data=par,palette='Dark2_r')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5870a4210>
```

CLEANING DATASET

```
par.drop('numPeriodsPulses',axis=1,inplace=True)
```

```
mall.head(50)
```

	number	Age	Spending Score (1-100)
0	1	19	39
1	2	21	81
2	3	20	6
3	4	23	77

```
par.head()
```

	id	gender	RPDE	numPulses	meanPeriodPulses
0	0	1	0.57227	240	0.008064
1	0	1	0.53966	234	0.008258
2	0	1	0.58982	232	0.008340
3	1	0	0.59257	178	0.010858
4	1	0	0.53028	236	0.008162

SPLITING THE *DATA*

```
x = par.iloc[:, :-1].values
y = par.iloc[:, :4].values
z=pd.DataFrame(x)
w=pd.DataFrame(y)
from sklearn.preprocessing import LabelEncoder
labelencoder_x=LabelEncoder()
labelencoder_y=LabelEncoder()

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
z_train=pd.DataFrame(x_train)
z_test=pd.DataFrame(x_test)
w_train=pd.DataFrame(y_train)
w_test=pd.DataFrame(y_test)
```

23 24 31 73

DATA CLEANING

```
par.head()
```

id	gender	RPDE	numPulses	meanPeriodPulses

```
par.tail()
```

	id	gender	RPDE	numPulses	meanPeriodPulses
194	64	1	0.56929	262	0.007351
195	65	1	0.32571	299	0.006454
196	65	1	0.33804	301	0.006422
197	65	1	0.29659	300	0.006450
198	66	0	0.44355	351	0.005503

```
par.isna()
```

	id	gender	RPDE	numPulses	meanPeriodPulses
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
194	False	False	False	False	False
195	False	False	False	False	False
196	False	False	False	False	False
197	False	False	False	False	False
198	False	False	False	False	False

199 rows × 5 columns

```
par.isna().any()
```

```
id                  False
gender              False
RPDE                False
numPulses           False
meanPeriodPulses    False
dtype: bool
```

```
par.isna().sum()
```

```
id                  0
gender              0
RPDE                0
numPulses           0
```

```
meanPeriodPulses    0
dtype: int64
```

```
par.isna().any().sum()
```

```
0
```

```
par.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
194    False
195    False
196    False
197    False
198    False
Length: 199, dtype: bool
```

MACHINE LEARNING MODELS

1)*DECISION TREE CLASSIFIER*

```
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import plot_tree
```

```
par = sns.load_dataset('iris')
par.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
par.shape
```

```
(150, 5)
```

```
par.isnull().any()
```

```
sepal_length    False
sepal_width     False
petal_length    False
petal_width     False
species         False
dtype: bool
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
print('Training split input-',x_train.shape)
print('Testing split input-',x_test.shape)
```

```
Training split input- (159, 4)
Testing split input- (40, 4)
```

```
from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier()
print('Decision Tree Classifier Created')
```

```
Decision Tree Classifier Created
```

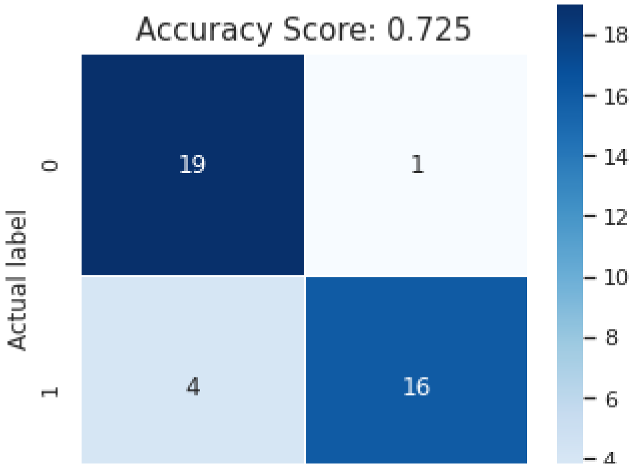
FITTING,PREDICTION AND ACCURACY

```
y_pred = dtree.predict(x_test)
print("Classification report - \n", classification_report(y_test,y_pred))
```

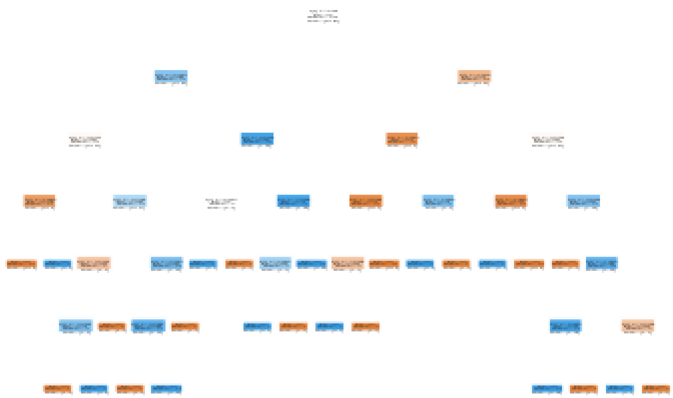
```
Classification report -
               precision    recall  f1-score   support

           0       0.74      0.70      0.72        20
           1       0.71      0.75      0.73        20

    accuracy                           0.73        40
   macro avg       0.73      0.72      0.72        40
weighted avg       0.73      0.72      0.72        40
```

```
import seaborn as sns
dtree.fit(x_train,y_train)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(dtree.score(x_test, y_test))
plt.title(all_sample_title, size = 15)
```

```
Text(0.5, 1.0, 'Accuracy Score: 0.725')
```




```
dec_tree = plot_tree(decision_tree=dtree,filled=True,precision=4,rounded=True)
```



2)**K-MAPS**

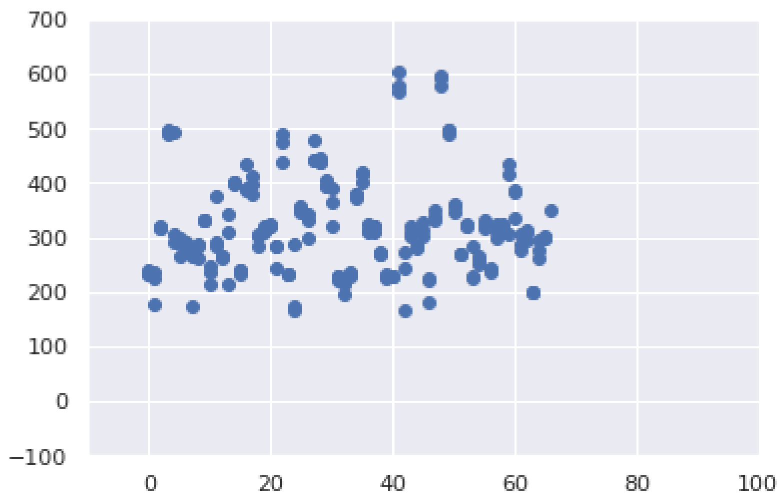
```
import statsmodels.api as sm
sns.set()
from sklearn.cluster import KMeans
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnin
  import pandas.util.testing as tm
```

```
par= pd.read_csv('par.csv')
par
```

	id	gender	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses
0	0	1	0.57227	240	239	0.008064
1	0	1	0.53966	234	233	0.008258
2	0	1	0.58982	232	231	0.008340
3	1	0	0.59257	178	177	0.010858
4	1	0	0.53028	236	235	0.008162
...	...	...	...	...	...	...

```
plt.scatter(par['id'],par['numPulses'])
plt.xlim(-10,100)
plt.ylim(-100,700)
plt.show()
```




```
x = np.array(par.drop(['id'], 1).astype(float))
```

```
y = np.array(par['id'])
```

```
par.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                199 non-null    int64
 1   gender            199 non-null    int64
 2   RPDE              199 non-null    float64
 3   numPulses         199 non-null    int64
 4   numPeriodsPulses  199 non-null    int64
 5   meanPeriodPulses  199 non-null    float64
dtypes: float64(2), int64(4)
memory usage: 9.5 KB
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
par_scaled = scaler.fit_transform(par)
pd.DataFrame(par_scaled).describe()
```

	0	1	2	3	4	
count	1.990000e+02	1.990000e+02	1.990000e+02	1.990000e+02	1.990000e+02	1.990(
mean	-1.004222e-17	1.863389e-16	4.463208e-17	1.743441e-16	-1.263646e-16	6.890
std	1.002522e+00	1.002522e+00	1.002522e+00	1.002522e+00	1.002522e+00	1.002!
min	-1.705961e+00	-1.194500e+00	-2.360945e+00	-1.721502e+00	-1.715797e+00	-2.007!
25%	-8.704314e-01	-1.194500e+00	-7.798465e-01	-6.721430e-01	-6.687016e-01	-6.373
50%	1.731941e-02	8.371707e-01	1.196003e-01	-1.332828e-01	-1.310042e-01	-1.118
75%	8.528495e-01	8.371707e-01	7.166378e-01	4.282664e-01	4.293331e-01	6.151
max	1.740600e+00	8.371707e-01	2.161323e+00	3.224668e+00	3.219700e+00	3.174!

```
kmeans = KMeans(n_clusters=2)
kmeans.fit(x)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```
correct = 0
for i in range(len(x)):
    predict_me = np.array(x[i].astype(float))
    predict_me = predict_me.reshape(-1, len(predict_me))
    prediction = kmeans.predict(predict_me)
    if prediction[0] == y[i]:
        correct += 1

print(correct/len(x))
```

```
0.01507537688442211
```

FITTING,PREDICTING AND ACCURACY

```
kmeans = KMeans(n_clusters=2, init='k-means++')
kmeans.fit(par_scaled)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```
kmeans = KMeans(n_jobs= -1, n_clusters = 5, init='k-means++')
kmeans.fit(par_scaled)
pred = kmeans.predict(par_scaled)
print(pred)
```

```
[0 0 0 0 0 0 2 2 2 3 3 3 3 1 1 2 2 2 2 2 2 0 0 0 2 2 2 2 2 2 0 0 0 2 2 2 2
 2 2 2 0 2 3 3 3 0 0 0 3 1 1 3 3 1 1 1 1 2 2 2 1 1 1 2 2 0 3 3 3 0 0 0 0 0
 1 1 1 1 2 2 2 3 3 3 3 3 3 3 3 1 1 3 0 0 0 0 0 0 0 0 0 1 3 3 3 3 3 4 4 4
 1 1 1 4 4 4 0 0 0 0 0 0 3 3 3 0 0 4 4 4 4 4 4 4 1 1 1 0 0 0 4 4 4 3 3 3 3
 3 3 1 1 1 4 4 4 4 4 4 1 0 0 4 4 4 4 4 4 0 0 0 1 1 1 4 4 4 1 3 3 1 1 1 4 4
 4 1 1 1 0 0 0 4 4 4 4 4 4 1]
```

```
kmeans= KMeans(n_clusters=120)
kmeans.fit(x_train,y_train)
y_pred= kmeans.predict(x_test)
```

3)**K-NEAREST NEIGHBOURS**

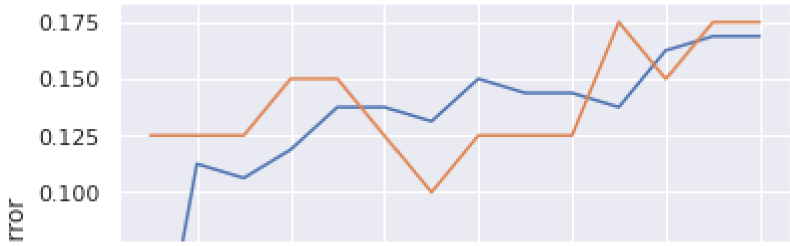
```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

```
x,y=make_classification(n_samples=200 ,n_features=8,n_informative=8,n_redundant=0,n_repeat
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2,random_state=32)
sc= StandardScaler()
sc.fit(x_train)
x_train= sc.transform(x_train)
sc.fit(x_test)
x_test= sc.transform(x_test)
x.shape
```

```
(200, 8)
```

```
error1= []
error2= []
for k in range(1,15):
    knn= KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train,y_train)
    y_pred1= knn.predict(x_train)
    error1.append(np.mean(y_train!= y_pred1))
    y_pred2= knn.predict(x_test)
    error2.append(np.mean(y_test!= y_pred2))
plt.plot(range(1,15),error1,label="train")
plt.plot(range(1,15),error2,label="test")
plt.xlabel('k Value')
plt.ylabel('Error')
```

```
Text(0, 0.5, 'Error')
```



FITTING,PREDICTING AND ACCURACY

Double-click (or enter) to edit

```
knn= KNeighborsClassifier(n_neighbors=6)
knn.fit(x_train,y_train)
y_pred= knn.predict(x_test)
metrics.accuracy_score(y_test,y_pred)
```

```
0.875
```

SUMMARY

Parkinson's disease is a disorder of the central nervous system that affects movement, often including tremors.We do parkinson's project using python and machine learning.

1)EXPLORATORY DATA ANALYSIS Exploratory data analysis is the analysis of the data and brings out the insights. EDA is an approach to analyse the data with the help of various tools and graphical techniques like barplot, histogram etc.There are many libraries available in python like pandas, NumPy, matplotlib, seaborn etc. with the help of those we can do the analysis of the data.

2)DATA CLEANING Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset.

3)SPLIT YOUR DATA WITH 80-20%% The most common split ratio is 80:20. That is 80% of the dataset goes into the training set and 20% of the dataset goes into the testing set. Before splitting the data, make sure that the dataset is large enough. Train/Test split works well with large datasets.The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem.

4)MACHINE LEARNING MODELS *Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values. -.

->> DECISION TREE: A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning.

->> K-MEANS: Kmeans Algorithm is an Iterative algorithm that divides a group of n datasets into k subgroups /clusters based on the similarity and their mean distance from the centroid of that particular subgroup/ formed. K, here is the pre-defined number of clusters to be formed by the Algorithm. If K=3, It means the number of clusters to be formed from the dataset is 3.

->> K-NEAREST NEIGHBOUR: K-Nearest Neighbors, or KNN for short, is one of the simplest machine learning algorithms and is used in a wide array of institutions. KNN is a non-parametric, lazy learning algorithm. When we say a technique is non-parametric, it means that it does not make any assumptions about the underlying data. In other words, it makes its selection based off of the proximity to other data points regardless of what feature the numerical values represent. Being a lazy learning algorithm implies that there is little to no training phase.