```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import zscore
from sklearn.preprocessing import Imputer
from sklearn.metrics import accuracy_score
import seaborn as sns
import os
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib as mp
import seaborn as sns
%matplotlib inline
sns.set(style="ticks")

from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import zscore
from sklearn.preprocessing import Imputer
from sklearn.metrics import accuracy_score
import seaborn as sns
import os
%matplotlib inline
from sklearn import metrics
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473

Enter your authorization code:
··········
Mounted at /content/drive
```

```
data = pd.read_csv('/content/drive/My Drive/22sept/parkinsons1.csv')
```

```
data
```

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:J
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	
5	phon_R01_S01_6	120.552	131.162	113.787	0.00968	
6	phon_R01_S02_1	120.267	137.244	114.820	0.00333	
7	phon_R01_S02_2	107.332	113.840	104.315	0.00290	
8	phon_R01_S02_3	95.730	132.068	91.754	0.00551	
9	phon_R01_S02_4	95.056	120.103	91.226	0.00532	
10	phon_R01_S02_5	88.333	112.240	84.072	0.00505	
11	phon_R01_S02_6	91.904	115.871	86.292	0.00540	
12	phon_R01_S04_1	136.926	159.866	131.276	0.00293	
13	phon_R01_S04_2	139.173	179.139	76.556	0.00390	
14	phon_R01_S04_3	152.845	163.305	75.836	0.00294	
15	phon_R01_S04_4	142.167	217.455	83.159	0.00369	
16	phon_R01_S04_5	144.188	349.259	82.764	0.00544	
17	phon_R01_S04_6	168.778	232.181	75.603	0.00718	
18	phon_R01_S05_1	153.046	175.829	68.623	0.00742	
19	phon_R01_S05_2	156.405	189.398	142.822	0.00768	
20	phon_R01_S05_3	153.848	165.738	65.782	0.00840	
21	phon_R01_S05_4	153.880	172.860	78.128	0.00480	
22	phon_R01_S05_5	167.930	193.221	79.068	0.00442	
23	phon_R01_S05_6	173.917	192.735	86.180	0.00476	
24	phon_R01_S06_1	163.656	200.841	76.779	0.00742	
25	phon_R01_S06_2	104.400	206.002	77.968	0.00633	
26	phon_R01_S06_3	171.041	208.313	75.501	0.00455	
27	phon_R01_S06_4	146.845	208.701	81.737	0.00496	
28	phon_R01_S06_5	155.358	227.383	80.055	0.00310	
29	phon_R01_S06_6	162.568	198.346	77.630	0.00502	
...	...	...	...	...	...	
165	phon_R01_S42_1	236.200	244.663	102.137	0.00277	
166	phon_R01_S42_2	237.323	243.709	229.256	0.00303	

166	phon_R01_S42_2	237.323	243.709	229.256	0.00303
167	phon_R01_S42_3	260.105	264.919	237.303	0.00339
168	phon_R01_S42_4	197.569	217.627	90.794	0.00803
169	phon_R01_S42_5	240.301	245.135	219.783	0.00517
170	phon_R01_S42_6	244.990	272.210	239.170	0.00451
171	phon_R01_S43_1	112.547	133.374	105.715	0.00355
172	phon_R01_S43_2	110.739	113.597	100.139	0.00356
173	phon_R01_S43_3	113.715	116.443	96.913	0.00349
174	phon_R01_S43_4	117.004	144.466	99.923	0.00353
175	phon_R01_S43_5	115.380	123.109	108.634	0.00332
176	phon_R01_S43_6	116.388	129.038	108.970	0.00346
177	phon_R01_S44_1	151.737	190.204	129.859	0.00314
178	phon_R01_S44_2	148.790	158.359	138.990	0.00309
179	phon_R01_S44_3	148.143	155.982	135.041	0.00392
180	phon_R01_S44_4	150.440	163.441	144.736	0.00396
181	phon_R01_S44_5	148.462	161.078	141.998	0.00397
182	phon_R01_S44_6	149.818	163.417	144.786	0.00336
183	phon_R01_S49_1	117.226	123.925	106.656	0.00417
184	phon_R01_S49_2	116.848	217.552	99.503	0.00531
185	phon_R01_S49_3	116.286	177.291	96.983	0.00314
186	phon_R01_S49_4	116.556	592.030	86.228	0.00496
187	phon_R01_S49_5	116.342	581.289	94.246	0.00267
188	phon_R01_S49_6	114.563	119.167	86.647	0.00327
189	phon_R01_S50_1	201.774	262.707	78.228	0.00694
190	phon_R01_S50_2	174.188	230.978	94.261	0.00459
191	phon_R01_S50_3	209.516	253.017	89.488	0.00564
192	phon_R01_S50_4	174.688	240.005	74.287	0.01360
193	phon_R01_S50_5	198.764	396.961	74.904	0.00740

```
data.describe().transpose()
```

```
data.describe(include=[np.number]).transpose()
```

```
data.info()
```

```
for feature in data.columns: # Loop through all columns in the dataframe
    if data[feature].dtype == 'object': # Only apply for columns with categorical strings
        data[feature] = pd.Categorical(data[feature]).codes # Replace strings with an integer
```

```
data.head(10)
```

```
#Using univariate &amp;amp; bivariate analysis to check the individual attributes for their basic sta

sns.pairplot(data, hue = "status")
```

```
plt = sns.pairplot(data[['MDVP:Fo(Hz)','MDVP:Fhi(Hz)','MDVP:Flo(Hz)','MDVP:Jitter(%)','MDVP:Jitte
```

```
plt = sns.boxplot(data[['MDVP:Fo(Hz)']])
```

```
plt = sns.boxplot(data[['MDVP:Fhi(Hz)']])
```

```
plt = sns.boxplot(data[['MDVP:Flo(Hz)']])
```

```
plt = sns.boxplot(data[['MDVP:Jitter(%)']])
```

```
plt = sns.boxplot(data[['MDVP:Jitter(Abs)']])
```

```
plt = sns.boxplot(data[['MDVP:RAP']])
```

```
plt = sns.boxplot(data[['MDVP:PPQ']])
```

```
plt = sns.boxplot(data[['Jitter:DDP']])
```

```
plt = sns.boxplot(data[['MDVP:Shimmer']])
```

```
plt = sns.boxplot(data[['MDVP:Shimmer(dB)']])
```

```
plt = sns.boxplot(data[['Shimmer:APQ3']])
```

```
plt = sns.boxplot(data[['Shimmer:APQ5']])
```

```
plt = sns.boxplot(data[['MDVP:APQ']])
```

```
plt = sns.boxplot(data[['Shimmer:DDA']])
```

```
plt = sns.boxplot(data[['NHR']])
```

```
plt = sns.boxplot(data[['RPDE']])
```

```
plt = sns.boxplot(data[['DFA']])
```

```
plt = sns.boxplot(data[['spread1']])
```

```
plt = sns.boxplot(data[['spread2']])
```

```
plt = sns.boxplot(data[['D2']])
```

```
plt = sns.boxplot(data[['PPE']])
```

```
from matplotlib import pyplot as plt
plt.figure(figsize=(25, 25))
ax = sns.heatmap(data.corr(), vmax=.8, square=True, fmt='.2f', annot=True, linecolor='white', lin
plt.title('Correlation')
```

```
plt.show()
```

```
list(data.columns)
```

```
col_labels = list(data.columns)
```

```
col_labels
```

```
 #Split the dataset into training and test set in the ratio of 70:30
```

```
train_char_label = ['No', 'Yes']

# capture the target column ("status") into separate vectors for training set and test set

X = data.drop("status" , axis=1)

y = data.pop("status")

from sklearn.model_selection import train_test_split

X_train, X_test, train_labels, test_labels = train_test_split(X, y, test_size=.30, random_state=1

from sklearn.tree import DecisionTreeClassifier

#Create the model using "entropy" method of reducing the entropy and fit it to training data. (

dt_model = DecisionTreeClassifier(criterion = 'entropy' )

dt_model.fit(X_train, train_labels)
```

```
from IPython.display import Image
#import pydotplus as pydot
from sklearn import tree
from os import system

parkinsons_Tree_File = open('/content/drive/My Drive/22sept/parkinsons_tree.dot','w')
dot_data = tree.export_graphviz(dt_model, out_file=parkinsons_Tree_File, feature_names = list(X_t

parkinsons_Tree_File.close()

print (pd.DataFrame(dt_model.feature_importances_, columns = ["Imp"], index = X_train.columns))
```

```
system("dot -Tpng parkinsons_tree.dot -o parkinsons_tree.png")
Image("parkinsons_tree.png", width=100, height = 100)
```

```
y_predict = dt_model.predict(X_test)
```

```
print(dt_model.score(X_train , train_labels))
print(dt_model.score(X_test , test_labels))
```

```
print(metrics.confusion_matrix(test_labels, y_predict))
```

```
dtc = DecisionTreeClassifier()
dtc.fit(X_train , train_labels)
```

```
print("Decision tree classifier, got {}% accuracy on the test set.".format(accuracy_score(test_la
```

```
# Use regularization parameters of max_depth, min_sample_leaf to recreate the model
```

Use regularization parameters of max_depth, min_sample_leaf to recreate the model

```
dt_model = DecisionTreeClassifier(criterion = 'entropy' , min_samples_leaf=1,max_depth=5, random_
```

```
dt_model.fit(X_train, train_labels)
```

```
from IPython.display import Image
#import pydotplus as pydot
from sklearn import tree
from os import system
```

```
parkinsons_Tree_File = open('/content/drive/My Drive/22sept/parkinsons_tree.dot','w')
dot_data = tree.export_graphviz(dt_model, out_file=parkinsons_Tree_File, feature_names = list(X_t

parkinsons_Tree_File.close()
```

```
print (pd.DataFrame(dt_model.feature_importances_, columns = ["Imp"], index = X_train.columns))
```

```
system("dot -Tpng parkinsons_tree.dot -o parkinsons_tree.png")
Image("parkinsons_tree.png", width=100, height = 100)
```

```
y_predict = dt_model.predict(X_test)
```

```
print(dt_model.score(X_train , train_labels))
print(dt_model.score(X_test , test_labels))
```

```
print(metrics.confusion_matrix(test_labels, y_predict))
```

Double-click (or enter) to edit

```
dtc = DecisionTreeClassifier()
dtc.fit(X_train , train_labels)
```

```
print("Decision tree classifier, got {}% accuracy on the test set.".format(accuracy_score(test_la
```

```
# No significant change in accuracy by changing the paramaters of max_depth, min_sample_leaf
```

Ensemble RandomForest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rfcl = RandomForestClassifier(n_estimators = 50)
rfcl = rfcl.fit(X_train, train_labels)
```

```
y_predict = rfcl.predict(X_test)
print(rfcl.score(X_test , test_labels))
print(metrics.confusion_matrix(test_labels, y_predict))
```