Part 2: Parkinson's Disease Classification Data Set

Attribute Information:

Various speech signal processing algorithms including Time Frequency Features, Mel Frequency Cepstral Coefficients (MFCCs), Wavelet Transform based Features, Vocal Fold Features and TWQT features have been applied to the speech recordings of Parkinson's Disease (PD) patients to extract clinically useful information for PD assessment.

2.1. Load

Below are some basic information about the dataframe.

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	...
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	...
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	...
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	...
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	...
...	...	...	...	...	...	...	...	...	...	...	...
751	250	0	0.80903	0.56355	0.28385	417	416	0.004627	0.000052	0.00064	...
752	250	0	0.16084	0.56499	0.59194	415	413	0.004550	0.000220	0.00143	...
753	251	0	0.88389	0.72335	0.46815	381	380	0.005069	0.000103	0.00076	...
754	251	0	0.83782	0.74890	0.49823	340	339	0.005679	0.000055	0.00092	...
755	251	0	0.81304	0.76471	0.46374	340	339	0.005676	0.000037	0.00078	...

756 rows × 755 columns

Figure 1: Dataframe

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...
count	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	...
mean	125.500000	0.515873	0.746284	0.700414	0.489058	323.972222	322.678571	0.006360	0.000383	0.002324	...
std	72.793721	0.500079	0.169294	0.069718	0.137442	99.219059	99.402499	0.001826	0.000728	0.002628	...
min	0.000000	0.000000	0.041551	0.543500	0.154300	2.000000	1.000000	0.002107	0.000011	0.000210	...
25%	62.750000	0.000000	0.762833	0.647053	0.386537	251.000000	250.000000	0.005003	0.000049	0.000970	...
50%	125.500000	1.000000	0.809655	0.700525	0.484355	317.000000	316.000000	0.006048	0.000077	0.001495	...
75%	188.250000	1.000000	0.834315	0.754985	0.586515	384.250000	383.250000	0.007528	0.000171	0.002520	...
max	251.000000	1.000000	0.907660	0.852640	0.871230	907.000000	905.000000	0.012966	0.003483	0.027750	...

8 rows × 755 columns

Figure 2: Summary Table

```
RangeIndex: 756 entries, 0 to 755
Columns: 755 entries, id to class
dtypes: float64(749), int64(6)
memory usage: 4.4 MB
```

Figure 3: Information

2.3. Data Preprocessing

2.3.1. Missing Values

```
id                            0
gender                        0
PPE                           0
DFA                           0
RPDE                          0
                             ..
tqwt_kurtosisValue_dec_33     0
tqwt_kurtosisValue_dec_34     0
tqwt_kurtosisValue_dec_35     0
tqwt_kurtosisValue_dec_36     0
class                         0
Length: 755, dtype: int64
```

Figure 4: No missing values

2.3.2. Duplicated Values

```
0      False
1      False
2      False
3      False
4      False
       ...
751    False
752    False
753    False
754    False
755    False
Length: 756, dtype: bool
```

Figure 5: duplicated values

```
True
```

Figure 6: We have some duplicated values

If an element is duplicated in the training data, it is effectively the same as having its 'weight' doubled. That element becomes twice as important when the classifier is fitting our data, and the classifier becomes biased towards correctly classifying that particular scenario over others so we drop the duplicated values.

2.3.3. Redundant Columns

We also drop the redundant *id* and *gender* column.

2.4. Shuffle data and split train/test data

I used *train_ test_ split* in *sklearn.model_ selection* to shuffle and split data into 30/70 portions.
Also I used StandardScaler to help you standardize the dataset's features onto unit scale (mean = 0 and variance = 1) which is a requirement for the optimal performance of many machine learning algorithms and PCA.

2.5. Model with all 1's as outcomes

```
number of 1 in y_train:  392
y_train shape:  528
number of 1 in y_test:  171
y_test shape:  227
train_acc:  0.7424242424242424
test_acc:  0.7533039647577092
```

Figure 7: Train/test accuracy

```
[[   0  56]
 [   0 171]]
```

Figure 8: confusion matrix

We can see that the accuracy metric alone fails to determine if a model has predictive power! The accuracy of 75% on the test dataset is not a bad accuracy and it's definitely better than a random guess. So if we only look at this metric we might think we have built a good model that can make predictions with 75% accuracy. However, we know that in reality we did not predict any class 0 instances and we predicted all examples in this class wrong! Therefore, in order to prevent such mistakes, we need to look at other metrics, more specifically we need to look at the percentages of correct predictions in each class separately.

2.6. Classification - Before Feature Extraction Using PCA

In this section we have the

- Classification Model
  - Confusion Matrix
  - Train/Test Accuracy
  - Classification Report

For both before and after parameter tuning using cross validation.

2.6.1. Decision Tree

```
decision_tree confusion_matrix :
 [[ 39  17]
 [ 18 153]]
Accuracy: 84.58 %
Train accuracy: 100 %
Test accuracy: 84.58 %
classification_report:
              precision    recall  f1-score   support

           0       0.68      0.70      0.69        56
           1       0.90      0.89      0.90       171

    accuracy                           0.85       227
   macro avg       0.79      0.80      0.79       227
weighted avg       0.85      0.85      0.85       227
```

Figure 9: Before Parameter Tuning

```
decision_tree confusion_matrix :
 [[ 25  31]
 [  7 164]]
Accuracy: 83.26 %
Train accuracy: 87.12 %
Test accuracy: 83.26 %
classification_report:
              precision    recall  f1-score   support

           0       0.78      0.45      0.57        56
           1       0.84      0.96      0.90       171

    accuracy                           0.83       227
   macro avg       0.81      0.70      0.73       227
weighted avg       0.83      0.83      0.82       227
```

Figure 10: After Parameter Tuning

```
{'ccp_alpha': 0.01,
 'criterion': 'gini',
 'max_depth': 6,
 'max_features': 'auto'}
```

Figure 11: Best Parameters Using Cross Validation

2.6.2. KNN

```
knn confusion_matrix :
 [[ 39  17]
 [  4 167]]
Accuracy: 90.75 %
Train accuracy: 95.08 %
Test accuracy: 90.75 %
classification_report:
              precision    recall  f1-score   support

           0       0.91      0.70      0.79        56
           1       0.91      0.98      0.94       171

    accuracy                           0.91       227
   macro avg       0.91      0.84      0.86       227
weighted avg       0.91      0.91      0.90       227
```

Figure 12: Before Parameter Tuning

```
knn confusion_matrix :
 [[ 45  11]
 [  3 168]]
Accuracy: 93.83 %
Train accuracy: 100.0 %
Test accuracy: 93.83 %
classification_report:
              precision    recall  f1-score   support

           0       0.94      0.80      0.87        56
           1       0.94      0.98      0.96       171

    accuracy                           0.94       227
   macro avg       0.94      0.89      0.91       227
weighted avg       0.94      0.94      0.94       227
```

Figure 13: After Parameter Tuning

```
{'leaf_size': 20,
 'metric': 'minkowski',
 'n_neighbors': 1,
 'p': 1,
 'weights': 'uniform'}
```

Figure 14: Best Parameters Using Cross Validation

2.6.3. SVM

```
svm confusion_matrix :
 [[ 25  31]
 [  0 171]]
Accuracy: 86.34 %
Train accuracy: 90.15 %
Test accuracy: 86.34 %
classification_report:
              precision    recall  f1-score   support

           0       1.00      0.45      0.62        56
           1       0.85      1.00      0.92       171

    accuracy                           0.86       227
   macro avg       0.92      0.72      0.77       227
weighted avg       0.88      0.86      0.84       227
```

Figure 15: Before Parameter Tuning

```
svm confusion_matrix :
 [[ 35  21]
 [  1 170]]
Accuracy: 90.31 %
Train accuracy: 100.0 %
Test accuracy: 90.31 %
classification_report:
              precision    recall  f1-score   support

           0       0.97      0.62      0.76        56
           1       0.89      0.99      0.94       171

    accuracy                           0.90       227
   macro avg       0.93      0.81      0.85       227
weighted avg       0.91      0.90      0.90       227
```

Figure 16: After Parameter Tuning

```
{'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
```

Figure 17: Best Parameters Using Cross Validation

2.6.4. Random Forest

```
random_forest confusion_matrix :
 [[ 35  21]
 [  0 171]]
Accuracy: 90.75 %
Train accuracy: 100.0 %
Test accuracy: 90.75 %
classification_report:
              precision    recall  f1-score   support

           0       1.00      0.62      0.77        56
           1       0.89      1.00      0.94       171

    accuracy                           0.91       227
   macro avg       0.95      0.81      0.86       227
weighted avg       0.92      0.91      0.90       227
```

Figure 18: Before Parameter Tuning

```
random_forest confusion_matrix :
 [[ 32  24]
 [  0 171]]
Accuracy: 89.43 %
Train accuracy: 100.0 %
Test accuracy: 89.43 %
classification_report:
              precision    recall  f1-score   support

           0       1.00      0.57      0.73        56
           1       0.88      1.00      0.93       171

    accuracy                           0.89       227
   macro avg       0.94      0.79      0.83       227
weighted avg       0.91      0.89      0.88       227
```

Figure 19: After Parameter Tuning

```
{'max_depth': 20,
 'max_features': 'auto',
 'min_samples_leaf': 1,
 'min_samples_split': 5,
 'n_estimators': 500}
```

Figure 20: Best Parameters Using Cross Validation

2.6. Feature Extraction Using PCA

We perform PCA, setting the "number of components" to different values of 2, 4, 6, 8 and then compare the accuracy of models.

2.7. Classification - After Feature Extraction Using PCA

To keep things simple, the final accuracy results of all the models for each "number of components" is written here. This helps us to better compare them. The full results are available in the .ipynb file.

	Classification	ParameterTuning	Accuracy	Train accuracy	Test accuracy
3	KNN	after	93.83	100.00	93.83
2	KNN	before	90.75	95.08	90.75
6	Random Forest	before	90.75	100.00	90.75
5	SVM	after	90.31	100.00	90.31
7	Random Forest	after	89.43	100.00	89.43
4	SVM	before	86.34	90.15	86.34
0	Decision Tree	before	84.58	100.00	84.58
1	Decision Tree	after	83.26	87.12	83.26

Figure 21: Before PCA

- Without cross validation:

	Classification	ParameterTuning	Accuracy	Train accuracy	Test accuracy
4	SVM	before	81.50	81.44	81.50
6	Random Forest	before	78.85	100.00	78.85
2	KNN	before	75.33	89.02	75.33
0	Decision Tree	before	72.25	100.00	72.25
1	Decision Tree	after	0.00	0.00	0.00
3	KNN	after	0.00	0.00	0.00
5	SVM	after	0.00	0.00	0.00
7	Random Forest	after	0.00	0.00	0.00

Figure 22: After PCA with 2 components

	Classification	ParameterTuning	Accuracy	Train accuracy	Test accuracy
2	KNN	before	80.62	90.72	80.62
6	Random Forest	before	80.62	100.00	80.62
4	SVM	before	80.18	81.44	80.18
0	Decision Tree	before	76.65	100.00	76.65
1	Decision Tree	after	0.00	0.00	0.00
3	KNN	after	0.00	0.00	0.00
5	SVM	after	0.00	0.00	0.00
7	Random Forest	after	0.00	0.00	0.00

Figure 23: After PCA with 4 components

	Classification	ParameterTuning	Accuracy	Train accuracy	Test accuracy
6	Random Forest	before	83.26	100.00	83.26
2	KNN	before	82.38	92.99	82.38
4	SVM	before	81.94	83.33	81.94
0	Decision Tree	before	81.50	100.00	81.50
1	Decision Tree	after	0.00	0.00	0.00
3	KNN	after	0.00	0.00	0.00
5	SVM	after	0.00	0.00	0.00
7	Random Forest	after	0.00	0.00	0.00

Figure 24: After PCA with 6 components

	Classification	ParameterTuning	Accuracy	Train accuracy	Test accuracy
6	Random Forest	before	86.78	100.00	86.78
4	SVM	before	83.70	85.61	83.70
2	KNN	before	79.74	94.32	79.74
0	Decision Tree	before	77.53	100.00	77.53
1	Decision Tree	after	0.00	0.00	0.00
3	KNN	after	0.00	0.00	0.00
5	SVM	after	0.00	0.00	0.00
7	Random Forest	after	0.00	0.00	0.00

Figure 25: After PCA with 8 components

As we can see here, by increasing the number of components from 2 to 6 the accuracy increases but 8 components would decrease the accuracy. Thus the best number for components would be 6. This number is small enough to save time and complexity but also preserves the data such that the accuracy is closer to before we performed PCA.

2.8. Analysis

Gathering the model accuracy and test/train accuracy for both before and after parameter tuning and PCA and using barplot to visually compare them.

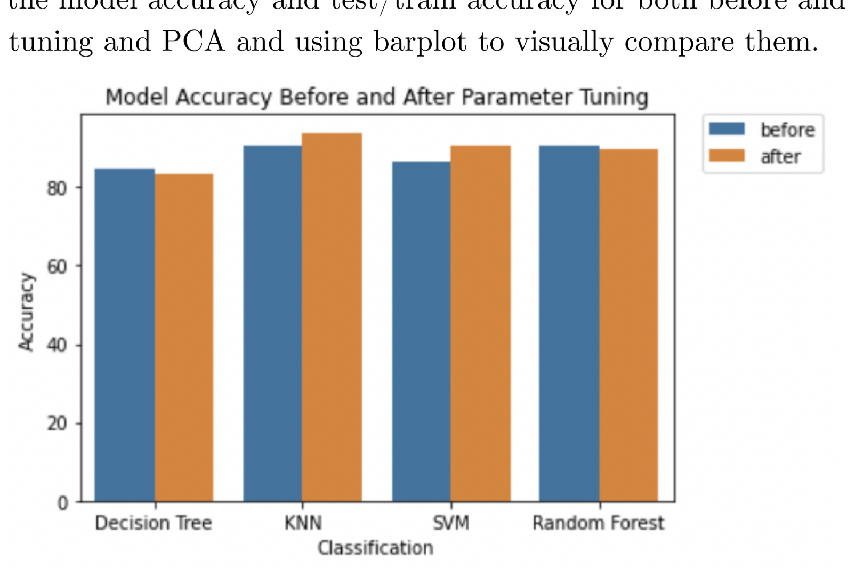

Figure 26: Before PCA

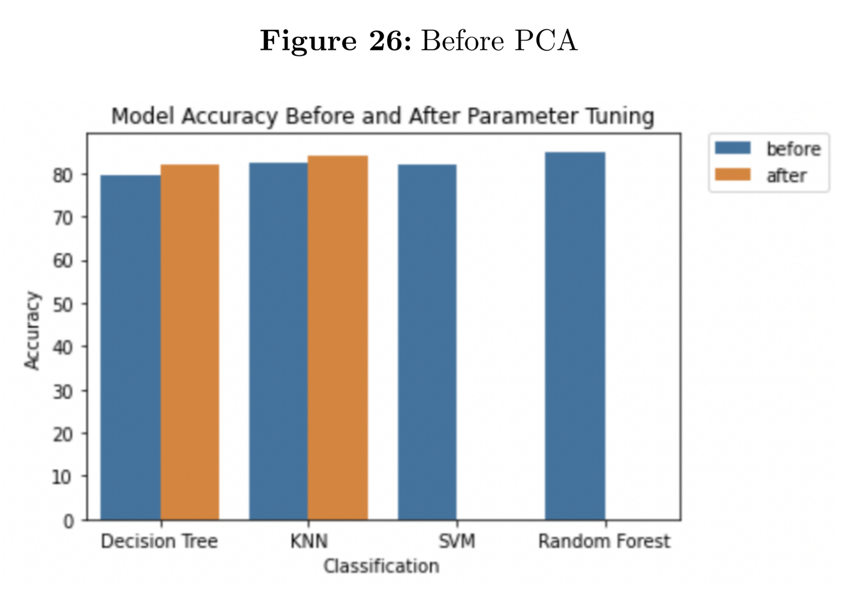

Figure 27: After PCA with 6 number of components.

2.9. Conclusions

Notes from the plots above:

- Before PCA, **KNN** has the **highest accuracy** both before and after parameter tuning.
- In order of accuracy we have from high to low:
  - KNN (after cross validation)
  - KNN (before cross validation)
  - Random Forest (before cross validation)
  - SVM (after cross validation)

- After PCA, **Random Forest** has the **highest accuracy** without parameter tuning.
- In order of accuracy we have from high to low:
  - Random Forest
  - KNN
  - SVM
  - Decision Tree
  -
- Overall PCA would decrease the accuracy due to loss of data.

- **Parameter tuning** using cross validation has improved the accuracy for some models but decreased it for others. This is because the default parameters where better than those that we searched through.