Telemonitoring of Parkinson's disease

progression by non-invasive speech tests

UPRDS

Unified Parkinson’s Disease Rating Scale

The goal of this project is to improve prediction of Parkinson's disease (PD) progression, which is needed to support clinical decision-making and to accelerate research trials.

About the dataset

Data of each patient

motor indicators UPDRS and total UPDRS

medical sound measurements

```
parkinsons_data.head()
```

	age	sex	test_time	motor_UPDRS	total_UPDRS	Jitter(%)	Jitter(Abs)	Jitter:RAP	Jitter:PPQ5	Jitter:DDP	...	Shimmer(dB)	Shimmer:APQ3	Shimmer:APQ5
0	72	0	5.6431	28.199	34.398	0.00662	0.000034	0.00401	0.00317	0.01204	...	0.230	0.01438	0.01309
1	72	0	12.6660	28.447	34.894	0.00300	0.000017	0.00132	0.00150	0.00395	...	0.179	0.00994	0.01072
2	72	0	19.6810	28.695	35.389	0.00481	0.000025	0.00205	0.00208	0.00616	...	0.181	0.00734	0.00844
3	72	0	25.6470	28.905	35.810	0.00528	0.000027	0.00191	0.00264	0.00573	...	0.327	0.01106	0.01265
4	72	0	33.6420	29.187	36.375	0.00335	0.000020	0.00093	0.00130	0.00278	...	0.176	0.00679	0.00929

5 rows × 21 columns

```
print(parkinsons_data.shape)
```

```
(5875, 21)
```

Great!

In our case fortunately there are no missing values at all. we will not worry about this problem

```
parkinsons_data.isnull().sum()
```

```
age              0
sex              0
test_time        0
motor_UPDRS      0
total_UPDRS      0
Jitter(%)        0
Jitter(Abs)      0
Jitter:RAP       0
Jitter:PPQ5      0
Jitter:DDP       0
Shimmer          0
Shimmer(dB)      0
Shimmer:APQ3     0
Shimmer:APQ5     0
Shimmer:APQ11    0
Shimmer:DDA      0
NHR              0
HNR              0
RPDE             0
DFA              0
PPE              0
dtype: int64
```

split our data into train and test

```
arr = parkinsons_data.values
X1 = arr[:,0:4]
X2 = arr[:,6:]
X = np.hstack((X1,X2))
Y = arr[:,4:6]
```

```
X.shape
```

```
(5875, 19)
```

```
Y.shape
```

```
(5875, 2)
```

We divide the table into input and output elements
X : features
Y : target (motor UPDRS and total UPDRS)

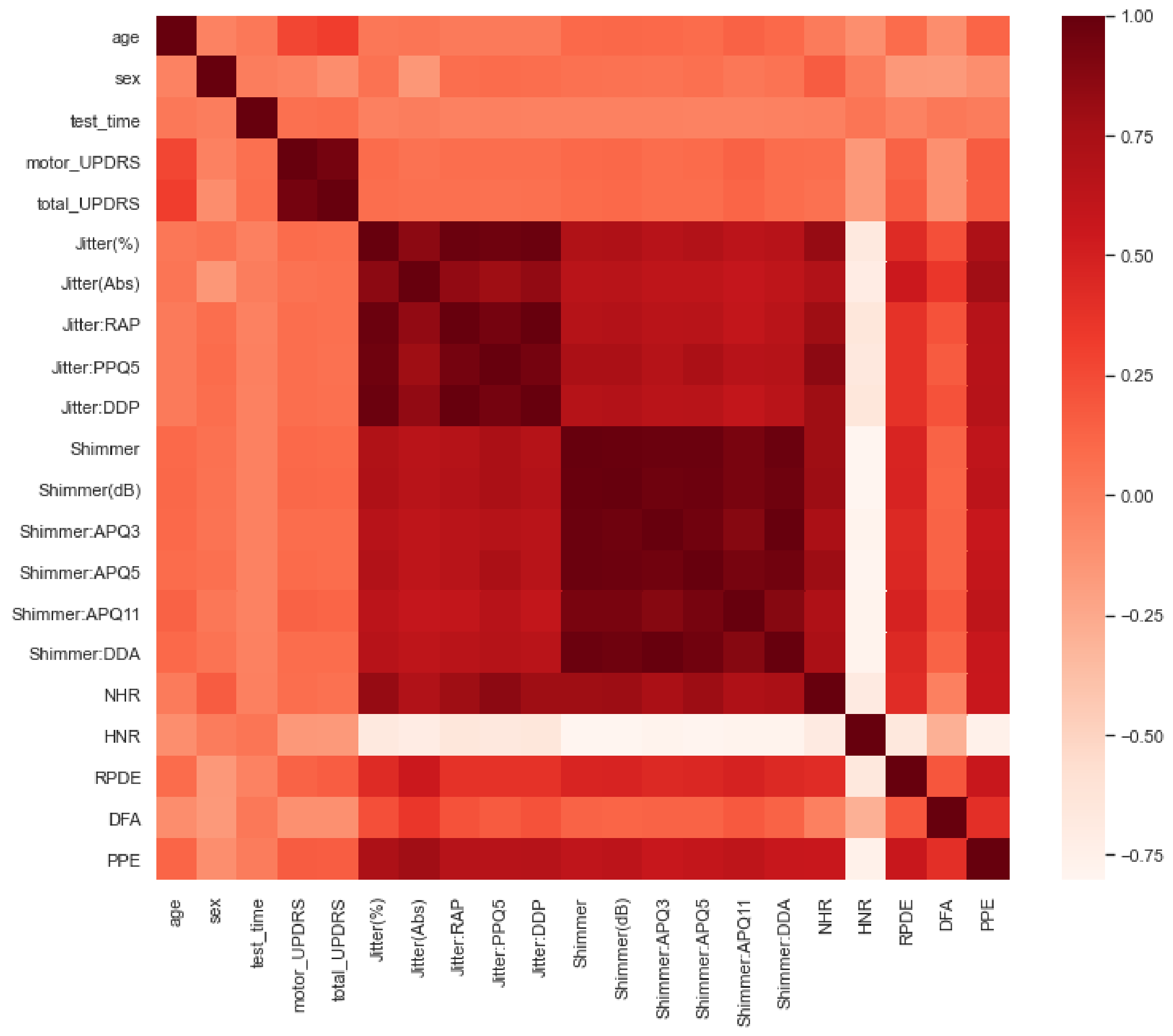
```
# Load and summarize the dataset
from sklearn.model_selection import train_test_split

# split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=1)

print('Train', X_train.shape, y_train.shape)
print('Test', X_test.shape, y_test.shape)
```

```
Train (4112, 19) (4112, 2)
Test (1763, 19) (1763, 2)
```

Feature selection

- Filter Method
- Wrapper Method
- Embedded Method

01 Filter Method



Motor UPDRS

```
#Correlation with output variable
cor_target = abs(cor['motor_UPDRS'])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.1]
relevant_features
```

```
subject#        0.252919
age             0.273665
motor_UPDRS     1.000000
total_UPDRS     0.947231
Shimmer         0.102349
Shimmer(dB)     0.110076
Shimmer:APQ11   0.136560
HNR             0.157029
RPDE            0.128607
DFA             0.116242
PPE             0.162433
Name: motor_UPDRS, dtype: float64
```

Total UPDRS

```
#Correlation with output variable
cor_target = abs(cor['total_UPDRS'])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.1]
relevant_features
```

```
subject#        0.253643
age             0.310290
motor_UPDRS     0.947231
total_UPDRS     1.000000
Shimmer:APQ11   0.120838
HNR             0.162117
RPDE            0.156897
DFA             0.113475
PPE             0.156195
Name: total_UPDRS, dtype: float64
```

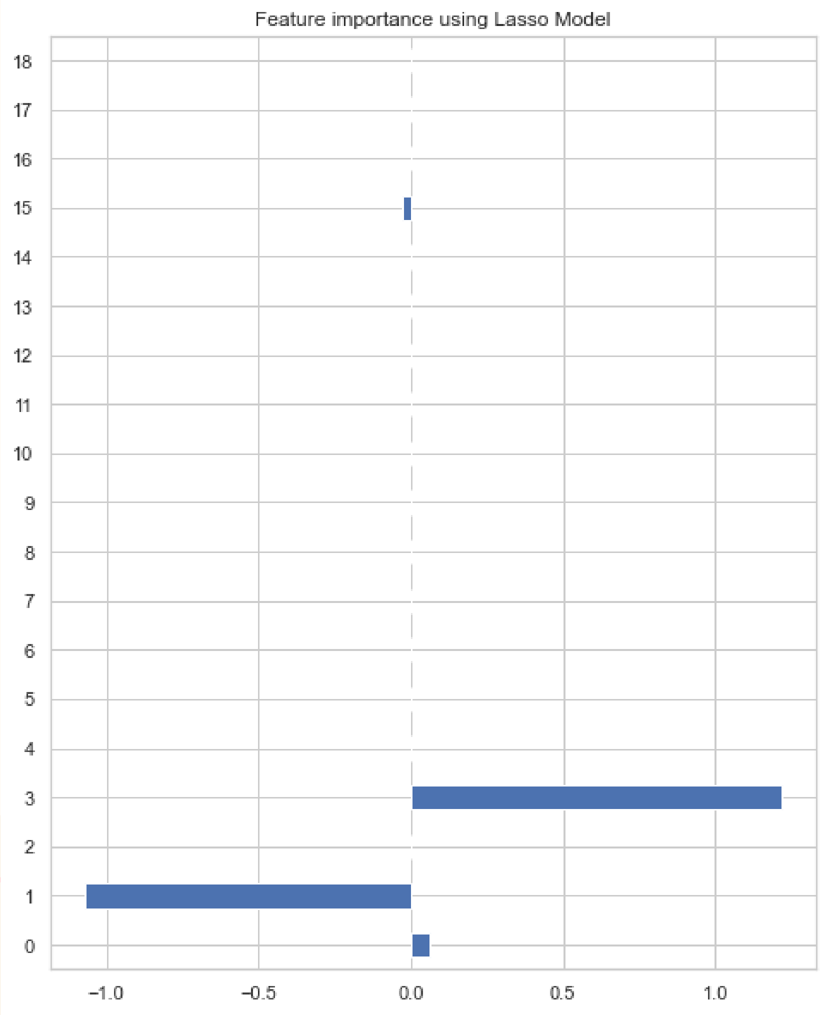
02 Wrapper Method

Backward Elimination

```
#Backward Elimination
cols = list(pd.DataFrame(X_train).columns)
pmax = 1
while (len(cols)>0):
    p= []
    X_1 = pd.DataFrame(X_train)[cols]
    X_1 = sm.add_constant(X_1)
    model = sm.OLS(pd.DataFrame(y_train[:,0]),X_1).fit()
    p = pd.Series(model.pvalues.values[1:],index = cols)
    pmax = max(p)
    feature_with_p_max = p.idxmax()
    if(pmax>0.05):
        cols.remove(feature_with_p_max)
    else:
        break
selected_features_BE = cols
print(selected_features_BE)
```

```
[0, 1, 2, 3, 4, 6, 9, 11, 12, 15, 16, 17, 18]
```

03 Embedded Method



Awesome!

I choose Backward Elimination

```
X_train = pd.DataFrame(X_train)
X_train = X_train[[0, 1, 2, 3, 4, 6, 9, 11, 12, 15, 16, 17, 18]]
X_train = X_train.values

X_test = pd.DataFrame(X_test)
X_test = X_test[[0, 1, 2, 3, 4, 6, 9, 11, 12, 15, 16, 17, 18]]
X_test = X_test.values
```

Conclusions

We saw how to select features using multiple methods for Numeric Data and compared their results. Now there arises a confusion of which method to choose in what situation. I think :
Filter method is less accurate.
Wrapper and Embedded methods give more accurate results but as they are computationally expensive, these method are suited when you have lesser features (~20).