In [50]:
```
import os
import pandas as pd
import sklearn.metrics
from sklearn import linear_model, tree, neural_network
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_absolute_error
import matplotlib.pyplot as plt

# Question 1 - Import and Exploratory
path = 'D:\\Documents\\DAAN862'
os.chdir(path)
pkdis = pd.read_csv('Parkinsons-Disease-Prediction-raw_dataset.csv')

pkdis.describe()
```

Out[50]:

	subject#	age	sex	test_time	motor_UPDRS	total_UPDRS	Jitter(%)	Jitter(Abs)	Jitter:RAP	Jitter:PPQ5	...	Shimmer(dB)	Shimmer:APQ3	Shimmer:APQ5	Shimmer:
count	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000	...	5875.000000	5875.000000	5875.000000	5875.
mean	21.494128	64.804936	0.317787	92.863722	21.296229	29.018942	0.006154	0.000044	0.002987	0.003277	...	0.310960	0.017156	0.020144	0.
std	12.372279	8.821524	0.465656	53.445602	8.129282	10.700283	0.005624	0.000036	0.003124	0.003732	...	0.230254	0.013237	0.016664	0.
min	1.000000	36.000000	0.000000	-4.262500	5.037700	7.000000	0.000830	0.000002	0.000330	0.000430	...	0.026000	0.001610	0.001940	0.
25%	10.000000	58.000000	0.000000	46.847500	15.000000	21.371000	0.003580	0.000022	0.001580	0.001820	...	0.175000	0.009280	0.010790	0.
50%	22.000000	65.000000	0.000000	91.523000	20.871000	27.576000	0.004900	0.000035	0.002250	0.002490	...	0.253000	0.013700	0.015940	0.
75%	33.000000	72.000000	1.000000	138.445000	27.596500	36.399000	0.006800	0.000053	0.003290	0.003460	...	0.365000	0.020575	0.023755	0.
max	42.000000	85.000000	1.000000	215.490000	39.511000	54.992000	0.099990	0.000446	0.057540	0.069560	...	2.107000	0.162670	0.167020	0.

8 rows × 22 columns

In [51]:
```
# Question 1 Cont - Drop 'motor_UPDRS'
# Commenting out just to not repeat when not needed
#pkdis =  pkdis.drop(['motor_UPDRS'], axis=1)

# Determine cor values in order o start linear regression while minimizing colinearity
pk_cor = pkdis.corrwith(pkdis.total_UPDRS)
print(pk_cor)
```

```
subject#         0.253643
age              0.310290
sex             -0.096559
test_time        0.075263
total_UPDRS      1.000000
Jitter(%)        0.074247
Jitter(Abs)      0.066927
Jitter:RAP       0.064015
Jitter:PPQ5      0.063352
Jitter:DDP       0.064027
Shimmer          0.092141
Shimmer(dB)      0.098790
Shimmer:APQ3     0.079363
Shimmer:APQ5     0.083467
Shimmer:APQ11    0.120838
Shimmer:DDA      0.079363
NHR              0.060952
HNR             -0.162117
RPDE             0.156897
DFA             -0.113475
PPE              0.156195
dtype: float64
```

In [5]:
```
# Question 2 - Build Linear Regression and use Cross Validate
X = pkdis[['subject#', 'age', 'test_time', 'Jitter(%)',
           'Shimmer:APQ11', 'NHR', 'HNR', 'RPDE', 'DFA', 'PPE']]
y = pkdis['total_UPDRS']

#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=123)

lr = linear_model.LinearRegression()
#lr.fit(X_train, y_train)
accuracies = cross_val_score(lr, X, y, cv=10, scoring='neg_mean_absolute_error')

# Display accuracies and get average performance for LR
display(accuracies)
display(pd.Series(accuracies).describe())
```

```
array([-10.20027154, -10.11509346,  -4.06489379,  -7.14983745,
        -9.48070711,  -9.27846107,  -7.61314438,  -7.82232471,
       -13.06497054,  -8.12757482])
count    10.000000
mean     -8.691728
std       2.366811
min     -13.064971
25%      -9.956497
50%      -8.703018
75%      -7.665439
max      -4.064894
dtype: float64
```

In [6]:
```
# Question 3 - Build Decision Tree Model and use Cross Validate
X_full = pkdis.drop(['total_UPDRS'], axis=1)

dt = tree.DecisionTreeRegressor()
accuracies = cross_val_score(dt, X_full, y, cv=10, scoring='neg_mean_absolute_error')

# Get k-fold scores for overall performance
display(accuracies)
display(pd.Series(accuracies).describe())
```

```
array([-12.90845408, -11.5512415 , -12.94040986,  -7.22597789,
        -9.52775714,  -6.61619659, -10.40549915, -13.38689421,
       -11.06825383, -12.37578535])
```

```
count    10.000000
mean    -10.800647
std       2.377918
min     -13.386894
25%     -12.775287
50%     -11.309748
75%      -9.747193
max      -6.616197
dtype: float64
```

In [56]:
```
# Question 4 - Build Neural Network Model with Cross Validate
from sklearn.preprocessing import MinMaxScaler
import numpy as np

scaler = MinMaxScaler()
scaler.fit(pkdis)
pkdis_scaled = scaler.transform(pkdis)
y_scaled = pkdis_scaled[:, 4]
X_full_scaled = np.delete(pkdis_scaled, 4, axis=1)

nn = neural_network.MLPRegressor(max_iter=1000)

accuracies = cross_val_score(nn, X_full_scaled, y_scaled, cv=10, scoring='neg_mean_absolute_error')

# Get accuracies for folds
display(accuracies)
display(pd.Series(accuracies).describe())
```

```
array([-0.31054856, -0.22722455, -0.07878046, -0.07738875, -0.16652134,
       -0.2345488 , -0.18786882, -0.20063932, -0.2791131 , -0.25904509])
count    10.000000
mean     -0.202168
std       0.078067
min      -0.310549
25%      -0.252921
50%      -0.213932
75%      -0.171858
max      -0.077389
dtype: float64
```

In [57]:
```
# Question 5 - Which model performed best?
# From the above results with neg MAE, we can see the Neural Network
# performed the best (higher negative MAE is better)
# In order to improve the model, we could reduce the number of attributes used in
# the model and/or increase the k-fold numbers.

accuracies = cross_val_score(nn, X_full_scaled, y_scaled, cv=20, scoring='neg_mean_absolute_error')

# Display accuracies and get average performance for LR
display(accuracies)
display(pd.Series(accuracies).describe())

# As shown, the improvement in the neg MAE metric was minimal but does allow
# for some room or improvement, further analysis could be used to remove more unrelated attributes
```

```
array([-0.29445497, -0.20668533, -0.30329054, -0.20793764, -0.09180472,
       -0.09679195, -0.08631599, -0.08813779, -0.19461427, -0.13520725,
       -0.11594261, -0.27622115, -0.12439786, -0.18316096, -0.25669235,
       -0.14377325, -0.31582845, -0.29715004, -0.40446659, -0.14387837])
count    20.000000
mean     -0.198338
std       0.093112
min      -0.404467
25%      -0.280780
50%      -0.188888
75%      -0.122284
max      -0.086316
dtype: float64
```

In [69]:
```
# Question 6 - Improve Neural Network or Decision Tree model
# We will implement the model optimization of Lesson 9 for Decision Tree

# We can also scale the values for the x and y for the decision tree too, which would
# have a great impact on optimization

tree_opt_1 = tree.DecisionTreeRegressor(random_state=39)
accuracies = cross_val_score(tree_opt_1, X_full_scaled, y_scaled, cv=10, scoring='neg_mean_absolute_error')

# Get k-fold scores for overall performance
display(accuracies)
display(pd.Series(accuracies).describe())

# As shown, scaling values for this example helped a lot too and increased the negative MAE significantly
```

```
array([-0.26895451, -0.22059859, -0.31113023, -0.11531042, -0.20897136,
       -0.16994465, -0.1260313 , -0.30020651, -0.23053278, -0.15243185])
count    10.000000
mean     -0.210411
std       0.069369
min      -0.311130
25%      -0.259349
50%      -0.214785
75%      -0.156810
max      -0.115310
dtype: float64
```

In [ ]: