Lesson 9 Assignment

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
```

In [2]:
```
os.getcwd()
```

Out[2]:
```
'C:\\Users\\HARIbol\\Downloads\\AI-ML\\PSU\\DAAN-862\\Lesson 9'
```

In [3]:
```
data = pd.read_csv("parkinsons+telemonitoring\\parkinsons_updrs.data")
data
```

Out[3]:

	subject#	age	sex	test_time	motor_UPDRS	total_UPDRS	Jitter(%)	Jitter(Abs)	Jitter:RAP	Jitter:PPQ5	...	Shimmer(dB
0	1	72	0	5.6431	28.199	34.398	0.00662	0.000034	0.00401	0.00317	...	0.23(
1	1	72	0	12.6660	28.447	34.894	0.00300	0.000017	0.00132	0.00150	...	0.17
2	1	72	0	19.6810	28.695	35.389	0.00481	0.000025	0.00205	0.00208	...	0.18
3	1	72	0	25.6470	28.905	35.810	0.00528	0.000027	0.00191	0.00264	...	0.32
4	1	72	0	33.6420	29.187	36.375	0.00335	0.000020	0.00093	0.00130	...	0.17(
...	...	...	...	...	...	...	...	...	...	...	...	.
5870	42	61	0	142.7900	22.485	33.485	0.00406	0.000031	0.00167	0.00168	...	0.16(
5871	42	61	0	149.8400	21.988	32.988	0.00297	0.000025	0.00119	0.00147	...	0.21
5872	42	61	0	156.8200	21.495	32.495	0.00349	0.000025	0.00152	0.00187	...	0.24
5873	42	61	0	163.7300	21.007	32.007	0.00281	0.000020	0.00128	0.00151	...	0.13
5874	42	61	0	170.7300	20.513	31.513	0.00282	0.000021	0.00135	0.00166	...	0.17

5875 rows × 22 columns

1. Perform exploratory analysis on the Parkinsons data and remove motor_UPDRS column

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
subject#         0
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
```

In [5]:
```
data.duplicated().sum()
```

Out[5]:
```
0
```

In [6]:
```
data.describe()
```

Out[6]:

	subject#	age	sex	test_time	motor_UPDRS	total_UPDRS	Jitter(%)	Jitter(Abs)	Jitter:RAP
count	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000	5875.000000
mean	21.494128	64.804936	0.317787	92.863722	21.296229	29.018942	0.006154	0.000044	0.002987
std	12.372279	8.821524	0.465656	53.445602	8.129282	10.700283	0.005624	0.000036	0.003124
min	1.000000	36.000000	0.000000	-4.262500	5.037700	7.000000	0.000830	0.000002	0.000330
25%	10.000000	58.000000	0.000000	46.847500	15.000000	21.371000	0.003580	0.000022	0.001580
50%	22.000000	65.000000	0.000000	91.523000	20.871000	27.576000	0.004900	0.000035	0.002250
75%	33.000000	72.000000	1.000000	138.445000	27.596500	36.399000	0.006800	0.000053	0.003290
max	42.000000	85.000000	1.000000	215.490000	39.511000	54.992000	0.099990	0.000446	0.057540

8 rows × 22 columns

In [7]:
```
data.drop(columns = ['motor_UPDRS'], inplace = True)
data
```

Out[7]:

	subject#	age	sex	test_time	total_UPDRS	Jitter(%)	Jitter(Abs)	Jitter:RAP	Jitter:PPQ5	Jitter:DDP	...	Shimmer(dB)	S
0	1	72	0	5.6431	34.398	0.00662	0.000034	0.00401	0.00317	0.01204	...	0.230	
1	1	72	0	12.6660	34.894	0.00300	0.000017	0.00132	0.00150	0.00395	...	0.179	
2	1	72	0	19.6810	35.389	0.00481	0.000025	0.00205	0.00208	0.00616	...	0.181	
3	1	72	0	25.6470	35.810	0.00528	0.000027	0.00191	0.00264	0.00573	...	0.327	
4	1	72	0	33.6420	36.375	0.00335	0.000020	0.00093	0.00130	0.00278	...	0.176	
...	...	...	...	...	...	...	...	...	...	...	...	...	
5870	42	61	0	142.7900	33.485	0.00406	0.000031	0.00167	0.00168	0.00500	...	0.160	
5871	42	61	0	149.8400	32.988	0.00297	0.000025	0.00119	0.00147	0.00358	...	0.215	
5872	42	61	0	156.8200	32.495	0.00349	0.000025	0.00152	0.00187	0.00456	...	0.244	
5873	42	61	0	163.7300	32.007	0.00281	0.000020	0.00128	0.00151	0.00383	...	0.131	
5874	42	61	0	170.7300	31.513	0.00282	0.000021	0.00135	0.00166	0.00406	...	0.171	

5875 rows × 21 columns

Importing additional libraries and splitting the data into Training and Test data

In [8]:
```
from sklearn import linear_model, tree, neural_network
from sklearn.model_selection import cross_validate, train_test_split
from sklearn import metrics
```

In [9]:
```
X, y = data.iloc[:,5:], data['total_UPDRS']
X.shape, y.shape
```

Out[9]:
```
((5875, 16), (5875,))
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

2. Use cross-validation to build a linear regression model to predict total_UPDRS

We will be using cross_validate to use multiple scoring metrics (MAE, R2 score)

In [11]:
```
lr = linear_model.LinearRegression()
scoring = ['neg_mean_absolute_error', 'r2']
```

In [12]:
```
scores = cross_validate(lr, X_train, y_train, scoring = scoring, cv = 10, return_estimator = True)
lr_scores = pd.DataFrame(scores)
lr_scores
```

Out[12]:

	fit_time	score_time	estimator	test_neg_mean_absolute_error	test_r2
0	0.010230	0.001027	LinearRegression()	-8.395264	0.103388
1	0.003041	0.001019	LinearRegression()	-7.759161	0.157131
2	0.003032	0.001021	LinearRegression()	-8.471401	0.054159
3	0.007155	0.002036	LinearRegression()	-8.411046	0.087695
4	0.003032	0.001013	LinearRegression()	-8.608561	0.087858
5	0.003778	0.002497	LinearRegression()	-8.211038	0.093764
6	0.003997	0.001505	LinearRegression()	-8.656922	0.059912
7	0.003002	0.001500	LinearRegression()	-8.295720	0.112874
8	0.003000	0.002008	LinearRegression()	-8.272678	0.103636
9	0.004521	0.001000	LinearRegression()	-8.294102	0.115632

In [13]:
```
lr_scores.drop(columns = ['estimator']).mean()
```

Out[13]:
```
fit_time                      0.004479
score_time                    0.001462
test_neg_mean_absolute_error -8.337589
test_r2                       0.097605
dtype: float64
```

In [14]:
```
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

In [15]:
```
metrics.mean_absolute_error(y_test, y_pred)
```

Out[15]:
```
8.29559086174014
```

In [16]:
```
metrics.r2_score(y_test, y_pred)
```

Out[16]:
```
0.0800522051956698
```

3. Use cross-validation to build a regression tree model to predict total_UPDRS.

In [17]:
```
tree_model = tree.DecisionTreeRegressor(min_samples_leaf = 5, random_state = 42)
scores = cross_validate(tree_model, X_train, y_train, scoring = scoring, cv = 10, return_estimator = True)
tree_scores = pd.DataFrame(scores)
tree_scores
```

Out[17]:

	fit_time	score_time	estimator	test_neg_mean_absolute_error	test_r2
0	0.049703	0.002003	DecisionTreeRegressor(min_samples_leaf=5, rand...	-8.438403	-0.125665
1	0.050161	0.001500	DecisionTreeRegressor(min_samples_leaf=5, rand...	-7.237230	0.096395
2	0.042536	0.002003	DecisionTreeRegressor(min_samples_leaf=5, rand...	-7.934703	-0.029780
3	0.046826	0.001992	DecisionTreeRegressor(min_samples_leaf=5, rand...	-8.519514	-0.133834
4	0.040304	0.001548	DecisionTreeRegressor(min_samples_leaf=5, rand...	-8.196098	-0.000899
5	0.043653	0.001501	DecisionTreeRegressor(min_samples_leaf=5, rand...	-7.927254	0.007910
6	0.041659	0.000985	DecisionTreeRegressor(min_samples_leaf=5, rand...	-8.141796	-0.062739
7	0.039202	0.001535	DecisionTreeRegressor(min_samples_leaf=5, rand...	-7.910910	-0.032490
8	0.040684	0.001525	DecisionTreeRegressor(min_samples_leaf=5, rand...	-7.779899	0.043234
9	0.043107	0.001486	DecisionTreeRegressor(min_samples_leaf=5, rand...	-8.314968	-0.088842

In [18]:
```
tree_scores.drop('estimator', axis = 1).mean()
```

Out[18]:
```
fit_time                       0.043784
score_time                     0.001608
test_neg_mean_absolute_error   -8.040078
test_r2                        -0.032671
dtype: float64
```

In [19]:
```
tree_model.fit(X_train, y_train)
y_pred = tree_model.predict(X_test)
```

In [20]:
```
metrics.mean_absolute_error(y_test, y_pred)
```

Out[20]:
```
7.987609734110097
```

In [21]:
```
metrics.r2_score(y_test, y_pred)
```

Out[21]:
```
-0.09123134797838461
```

4. Use cross-validation to build a neural network model to predict total_UPDRS

In [22]:
```
nn_model = neural_network.MLPRegressor(20, activation = 'logistic', solver = 'adam', max_iter = 1000, random_state =
scores = cross_validate(nn_model, X_train, y_train, scoring = scoring, cv = 10, return_estimator = True)
nn_scores = pd.DataFrame(scores)
nn_scores
```

Out[22]:

	fit_time	score_time	estimator	test_neg_mean_absolute_error	test_r2
0	0.769779	0.001501	MLPRegressor(activation='logistic', hidden_lay...	-8.734019	-0.008105
1	0.777598	0.002160	MLPRegressor(activation='logistic', hidden_lay...	-8.282964	-0.003067
2	1.114762	0.002509	MLPRegressor(activation='logistic', hidden_lay...	-8.629830	-0.000139
3	0.727851	0.001001	MLPRegressor(activation='logistic', hidden_lay...	-8.740605	-0.004957
4	1.000099	0.001001	MLPRegressor(activation='logistic', hidden_lay...	-9.040304	-0.001021
5	0.770910	0.001003	MLPRegressor(activation='logistic', hidden_lay...	-8.521242	-0.001739
6	0.832389	0.002983	MLPRegressor(activation='logistic', hidden_lay...	-8.809091	-0.001535
7	0.744529	0.002009	MLPRegressor(activation='logistic', hidden_lay...	-8.833129	-0.002048
8	0.720045	0.002000	MLPRegressor(activation='logistic', hidden_lay...	-8.626611	-0.001145
9	0.736465	0.001995	MLPRegressor(activation='logistic', hidden_lay...	-8.736605	-0.001551

In [23]:
```
nn_scores.drop('estimator', axis = 1).mean()
```

Out[23]:
```
fit_time                      0.819443
score_time                    0.001816
test_neg_mean_absolute_error -8.695440
test_r2                      -0.002531
dtype: float64
```

In [24]:
```
nn_model.fit(X_train, y_train)
y_pred = nn_model.predict(X_test)
```

In [25]:
```
metrics.mean_absolute_error(y_test, y_pred)
```

Out[25]:
```
8.547457650272142
```

In [26]:
```
metrics.r2_score(y_test, y_pred)
```

Out[26]:
```
-0.0008327635219778085
```

5. Compare their performance with MAE, which model has better performance? Is there any way to improve the model?

Answer ➡️

If we compare all the three models in terms of MAE, Decision Tree Regressor is a better performer.
We can improve the model by reducing the "min_samples_leaf" value but reduce it while avoiding overfitting. Lowering 'min_samples_leaf' will create complex trees and capture more detail with less bias but has a higher risk of overfitting. But as we are using cross validation, probably we will avoid overfitting it. We can explore that option using GriSearchCV and find the best min_samples_leaf value without overfitting.

6. Try to optimize the tree model or neural network model (Choose one)

We will try to optimize the Decision Tree Regressor model using GridSearchCV.

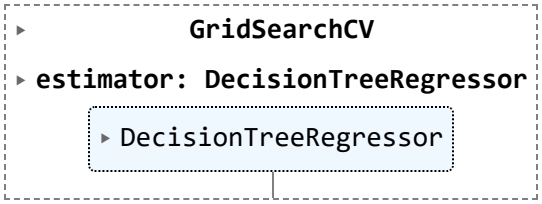
In [27]:
```
from sklearn.model_selection import GridSearchCV
```

In [28]:
```
tuned_params = {"criterion":['squared_error', 'friedman_mse', 'poisson'],
                "min_samples_leaf": [5,3,1],
                "max_depth" : [4,8,12],
                "min_samples_split":[2,5,10]}
```

In [29]:
```
grid_search = GridSearchCV(tree_model, tuned_params, cv = 10, scoring = 'neg_mean_absolute_error')
```

In [30]:
```
grid_search.fit(X_train, y_train)
```

Out[30]:



In [31]:
```
grid_search.best_params_
```

Out[31]:
```
{'criterion': 'friedman_mse',
 'max_depth': 8,
 'min_samples_leaf': 3,
 'min_samples_split': 10}
```

In [32]:
```
best_tree_model = grid_search.best_estimator_
```

In [33]:
```
y_pred = best_tree_model.predict(X_test)
```

In [34]:
```
metrics.mean_absolute_error(y_test, y_pred)
```

Out[34]:
```
7.485179211902424
```

In [35]:
```
metrics.r2_score(y_test, y_pred)
```

Out[35]:
```
0.15706246863133633
```

Observation ➡️ Using GridSearchCV improved the model as the MAE reduced.

End of assignment