

## ## Recursive Feature Elimination (RFE)

Scikit Learn does most of the heavy lifting just import RFE from `sklearn.feature_selection` and pass any classifier model to the `RFE()` method with the number of features to select. Using familiar Scikit Learn syntax, the `.fit()` method must then be called.

```
In [8]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
In [9]: from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import accuracy_score
```

```
In [10]: data = pd.read_csv('0.9_5subjectslabelled_data.csv', nrows = 31437)
data.head()
```

Out[10]:

	Time Snap	AccX	AccY	AccZ	Gyro_X	Knee Angles	Gait Cycle Phase
0	120.775	-0.181472	-0.088708	-0.665352	0.087145	67.223821	5
1	120.780	-0.181443	-0.088745	-0.659870	0.103506	67.217858	5
2	120.785	-0.183826	-0.089735	-0.654215	0.117635	67.154903	5
3	120.790	-0.188545	-0.091706	-0.648504	0.129259	67.011479	5
4	120.795	-0.195535	-0.094645	-0.642857	0.138193	66.799616	5

```
In [11]: X = data.drop('Gait Cycle Phase', axis = 1)
y = data['Gait Cycle Phase']

X.shape, y.shape
```

Out[11]: ((31436, 6), (31436,))

```
In [12]: X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state=42)
X_train.shape, X_test.shape
```

Out[12]: ((25148, 6), (6288, 6))

## Feature selection by feature importance of random forest classifier

```
In [13]: sel = SelectFromModel(RandomForestClassifier(n_estimators=100, random_state=0,  
sel.fit(X_train, y_train)  
sel.get_support()
```

```
Out[13]: array([False,  True, False, False, False,  True])
```

```
In [14]: X_train.columns
```

```
Out[14]: Index(['Time Snap', 'AccX', 'AccY', 'AccZ', 'Gyro_X', 'Knee Angles'], dtype  
='object')
```

```
In [15]: features = X_train.columns[sel.get_support()]
```

```
In [16]: features
```

```
Out[16]: Index(['AccX', 'Knee Angles'], dtype='object')
```

```
In [17]: len(features)
```

```
Out[17]: 2
```

```
In [18]: np.mean(sel.estimator_.feature_importances_)
```

```
Out[18]: 0.16666666666666666
```

```
In [19]: sel.estimator_.feature_importances_
```

```
Out[19]: array([0.05654607, 0.21871844, 0.13795007, 0.16501932, 0.11791425,  
0.30385185])
```

```
In [20]: X_train_rfc = sel.transform(X_train)  
X_test_rfc = sel.transform(X_test)
```

```
In [21]: def run_randomForest(X_train, X_test, y_train, y_test):  
    clf = RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1)  
    clf.fit(X_train, y_train)  
    y_pred = clf.predict(X_test)  
    print('Accuracy: ', accuracy_score(y_test, y_pred))
```

```
In [22]: %%time
run_randomForest(X_train_rfc, X_test_rfc, y_train, y_test)
```

Accuracy: 0.6401081424936387  
 CPU times: total: 23 s  
 Wall time: 5.64 s

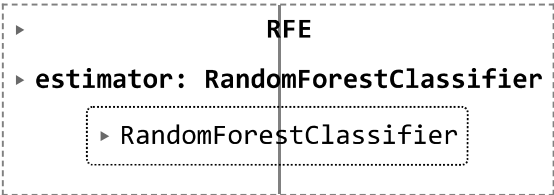
```
In [23]: %%time
run_randomForest(X_train, X_test, y_train, y_test)
```

Accuracy: 0.9209605597964376  
 CPU times: total: 32.4 s  
 Wall time: 7.11 s

## Recursive Feature Elimination (RFE)

```
In [24]: from sklearn.feature_selection import RFE
sel = RFE(RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1),
sel.fit(X_train, y_train)
```

```
Out[24]:
```



```
In [25]: sel.get_support()
```

```
Out[25]: array([ True,  True,  True,  True,  True,  True])
```

```
In [26]: features = X_train.columns[sel.get_support()]
```

```
In [27]: features
```

```
Out[27]: Index(['Time Snap', 'AccX', 'AccY', 'AccZ', 'Gyro_X', 'Knee Angles'], dtype
='object')
```

```
In [28]: len(features)
```

```
Out[28]: 6
```

```
In [31]: X_train_rfe = sel.transform(X_train)
X_test_rfe = sel.transform(X_test)
```

In [ ]:

```
In [32]: %%time
run_randomForest(X_train_rfe, X_test_rfe, y_train, y_test)
```

Accuracy: 0.9209605597964376

CPU times: total: 32.5 s

Wall time: 6.96 s

```
In [33]: %%time
run_randomForest(X_train, X_test, y_train, y_test)
```

Accuracy: 0.9209605597964376

CPU times: total: 32 s

Wall time: 6.58 s

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: