**Random Forests in python using scikit-learn**

In this post we’ll be using the Parkinson’s data set available from UCI [here](https://archive.ics.uci.edu/ml/datasets/parkinsons) to predict Parkinson’s status from potential predictors using Random Forests.

[Decision trees](http://benalexkeen.com/decision-tree-classifier-in-python-using-scikit-learn/) are a great tool but they can often overfit the training set of data unless pruned effectively, hindering their predictive capabilities.

Random forests are an ensemble model of many decision trees, in which each tree will specialise its focus on a particular feature, while maintaining an overview of all features.

Each tree in the random forest will do its own random train/test split of the data, known as bootstrap aggregation and the samples not included are known as the ‘out-of-bag’ samples. Additionally each tree will do feature bagging at each node-branch split to lessen the effects of a feature that is highly correlated with the response.

While an individual tree might be sensitive to outliers, the ensemble model will likely not be.

The model predicts new labels by taking a majority vote from each of its trees given a new observation.

We’ll start by loading up our data

In [1]:

**import** **pandas** **as** **pd**

df = pd.read\_csv('data/parkinsons.data')

**print**(df.head())

name MDVP:Fo(Hz) MDVP:Fhi(Hz) MDVP:Flo(Hz) MDVP:Jitter(%) \

0 phon\_R01\_S01\_1 119.992 157.302 74.997 0.00784

1 phon\_R01\_S01\_2 122.400 148.650 113.819 0.00968

2 phon\_R01\_S01\_3 116.682 131.111 111.555 0.01050

3 phon\_R01\_S01\_4 116.676 137.871 111.366 0.00997

4 phon\_R01\_S01\_5 116.014 141.781 110.655 0.01284

MDVP:Jitter(Abs) MDVP:RAP MDVP:PPQ Jitter:DDP MDVP:Shimmer ... \

0 0.00007 0.00370 0.00554 0.01109 0.04374 ...

1 0.00008 0.00465 0.00696 0.01394 0.06134 ...

2 0.00009 0.00544 0.00781 0.01633 0.05233 ...

3 0.00009 0.00502 0.00698 0.01505 0.05492 ...

4 0.00011 0.00655 0.00908 0.01966 0.06425 ...

Shimmer:DDA NHR HNR status RPDE DFA spread1 \

0 0.06545 0.02211 21.033 1 0.414783 0.815285 -4.813031

1 0.09403 0.01929 19.085 1 0.458359 0.819521 -4.075192

2 0.08270 0.01309 20.651 1 0.429895 0.825288 -4.443179

3 0.08771 0.01353 20.644 1 0.434969 0.819235 -4.117501

4 0.10470 0.01767 19.649 1 0.417356 0.823484 -3.747787

spread2 D2 PPE

0 0.266482 2.301442 0.284654

1 0.335590 2.486855 0.368674

2 0.311173 2.342259 0.332634

3 0.334147 2.405554 0.368975

4 0.234513 2.332180 0.410335

[5 rows x 24 columns]

We don’t want the name feature in our DataFrame so we’ll drop this and split our data into features and labels.

In [2]:

X = df.drop('status', axis=1)

X = X.drop('name', axis=1)

y = df['status']

Now we can split the data into a training and test set of data.

In [3]:

**from** **sklearn.model\_selection** **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=1)

We now create and train our model. The number of estimators (n\_estimators) determines how dense our decision forest is and the random\_state is given for reproducibility.

In [4]:

**from** **sklearn.ensemble** **import** RandomForestClassifier

random\_forest = RandomForestClassifier(n\_estimators=30, max\_depth=10, random\_state=1)

In [5]:

random\_forest.fit(X\_train, y\_train)

Out[5]:

RandomForestClassifier(bootstrap=True, class\_weight=None, criterion='gini',

max\_depth=10, max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_split=1e-07, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators=30, n\_jobs=1, oob\_score=False, random\_state=1,

verbose=0, warm\_start=False)

Now we evaluate our model on our test set.

In [6]:

**from** **sklearn.metrics** **import** accuracy\_score

y\_predict = random\_forest.predict(X\_test)

accuracy\_score(y\_test, y\_predict)

Out[6]:

0.93877551020408168

In [7]:

**from** **sklearn.metrics** **import** confusion\_matrix

pd.DataFrame(

confusion\_matrix(y\_test, y\_predict),

columns=['Predicted Healthy', 'Predicted Parkinsons'],

index=['True Healthy', 'True Parkinsons']

)

Out[7]:

|  | **Predicted Healthy** | **Predicted Parkinsons** |
| --- | --- | --- |
| **True Healthy** | **11** | **1** |
| **True Parkinsons** | **2** | **35** |

Our Decision Forest performs well on this limited set of data. We would need more data and more domain knowledge to effectively evaluate this model.