In this lab, you'll explore the breast cancer dataset and try to train the model to predict if the person is having breast cancer or not. We will start off with a weak learner, a decision tree with maximum depth = 2.

We will then build an adaboost ensemble with 50 trees with a step of 3 and compare the performance with the weak learner.

Let's get started by loading the libraries.

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

from sklearn.cross_validation import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.datasets import load_digits
from sklearn import metrics
%matplotlib inline

import os
import warnings
warnings.filterwarnings('ignore')
```

```
C:\Users\Kaustabh.Singh\Anaconda3\lib\site-packages\sklearn\cross_validation.
py:41: DeprecationWarning: This module was deprecated in version 0.18 in favo
r of the model_selection module into which all the refactored classes and fun
ctions are moved. Also note that the interface of the new CV iterators are di
fferent from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

We will use the breast cancer dataset in which the target variable has 1 if the person has cancer and 0 otherwise. Let's load the data.

In [2]:
```
cancer = load_breast_cancer()
digits = load_digits()

data = cancer
```

In [3]:
```
df = pd.DataFrame(data= np.c_[data['data'], data['target']],
                  columns= list(data['feature_names']) + ['target'])
df['target'] = df['target'].astype('uint16')
```

In [4]:
```
df
```

Out[4]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points |
|---|---|---|---|---|---|---|---|---|
| **0** | 17.990 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.300100 | 0.147100 |
| **1** | 20.570 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.086900 | 0.070170 |
| **2** | 19.690 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.197400 | 0.127900 |
| **3** | 11.420 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.241400 | 0.105200 |
| **4** | 20.290 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.198000 | 0.104300 |
| **5** | 12.450 | 15.70 | 82.57 | 477.1 | 0.12780 | 0.17000 | 0.157800 | 0.080890 |
| **6** | 18.250 | 19.98 | 119.60 | 1040.0 | 0.09463 | 0.10900 | 0.112700 | 0.074000 |
| **7** | 13.710 | 20.83 | 90.20 | 577.9 | 0.11890 | 0.16450 | 0.093660 | 0.059850 |
| **8** | 13.000 | 21.82 | 87.50 | 519.8 | 0.12730 | 0.19320 | 0.185900 | 0.093530 |
| **9** | 12.460 | 24.04 | 83.97 | 475.9 | 0.11860 | 0.23960 | 0.227300 | 0.085430 |
| **10** | 16.020 | 23.24 | 102.70 | 797.8 | 0.08206 | 0.06669 | 0.032990 | 0.033230 |
| **11** | 15.780 | 17.89 | 103.60 | 781.0 | 0.09710 | 0.12920 | 0.099540 | 0.066060 |
| **12** | 19.170 | 24.80 | 132.40 | 1123.0 | 0.09740 | 0.24580 | 0.206500 | 0.111800 |
| **13** | 15.850 | 23.95 | 103.70 | 782.7 | 0.08401 | 0.10020 | 0.099380 | 0.053640 |
| **14** | 13.730 | 22.61 | 93.60 | 578.3 | 0.11310 | 0.22930 | 0.212800 | 0.080250 |
| **15** | 14.540 | 27.54 | 96.73 | 658.8 | 0.11390 | 0.15950 | 0.163900 | 0.073640 |
| **16** | 14.680 | 20.13 | 94.74 | 684.5 | 0.09867 | 0.07200 | 0.073950 | 0.052590 |
| **17** | 16.130 | 20.68 | 108.10 | 798.8 | 0.11700 | 0.20220 | 0.172200 | 0.102800 |
| **18** | 19.810 | 22.15 | 130.00 | 1260.0 | 0.09831 | 0.10270 | 0.147900 | 0.094980 |
| **19** | 13.540 | 14.36 | 87.46 | 566.3 | 0.09779 | 0.08129 | 0.066640 | 0.047810 |
| **20** | 13.080 | 15.71 | 85.63 | 520.0 | 0.10750 | 0.12700 | 0.045680 | 0.031100 |
| **21** | 9.504 | 12.44 | 60.34 | 273.9 | 0.10240 | 0.06492 | 0.029560 | 0.020760 |
| **22** | 15.340 | 14.26 | 102.50 | 704.4 | 0.10730 | 0.21350 | 0.207700 | 0.097560 |
| **23** | 21.160 | 23.04 | 137.20 | 1404.0 | 0.09428 | 0.10220 | 0.109700 | 0.086320 |
| **24** | 16.650 | 21.38 | 110.00 | 904.6 | 0.11210 | 0.14570 | 0.152500 | 0.091700 |
| **25** | 17.140 | 16.40 | 116.00 | 912.7 | 0.11860 | 0.22760 | 0.222900 | 0.140100 |
| **26** | 14.580 | 21.53 | 97.41 | 644.8 | 0.10540 | 0.18680 | 0.142500 | 0.087830 |
| **27** | 18.610 | 20.25 | 122.10 | 1094.0 | 0.09440 | 0.10660 | 0.149000 | 0.077310 |
| **28** | 15.300 | 25.27 | 102.40 | 732.4 | 0.10820 | 0.16970 | 0.168300 | 0.087510 |
| **29** | 17.570 | 15.05 | 115.00 | 955.1 | 0.09847 | 0.11570 | 0.098750 | 0.079530 |

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points |
|---|---|---|---|---|---|---|---|---|
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **539** | 7.691 | 25.44 | 48.34 | 170.4 | 0.08668 | 0.11990 | 0.092520 | 0.013640 |
| **540** | 11.540 | 14.44 | 74.65 | 402.9 | 0.09984 | 0.11200 | 0.067370 | 0.025940 |
| **541** | 14.470 | 24.99 | 95.81 | 656.4 | 0.08837 | 0.12300 | 0.100900 | 0.038900 |
| **542** | 14.740 | 25.42 | 94.70 | 668.6 | 0.08275 | 0.07214 | 0.041050 | 0.030270 |
| **543** | 13.210 | 28.06 | 84.88 | 538.4 | 0.08671 | 0.06877 | 0.029870 | 0.032750 |
| **544** | 13.870 | 20.70 | 89.77 | 584.8 | 0.09578 | 0.10180 | 0.036880 | 0.023690 |
| **545** | 13.620 | 23.23 | 87.19 | 573.2 | 0.09246 | 0.06747 | 0.029740 | 0.024430 |
| **546** | 10.320 | 16.35 | 65.31 | 324.9 | 0.09434 | 0.04994 | 0.010120 | 0.005495 |
| **547** | 10.260 | 16.58 | 65.85 | 320.8 | 0.08877 | 0.08066 | 0.043580 | 0.024380 |
| **548** | 9.683 | 19.34 | 61.05 | 285.7 | 0.08491 | 0.05030 | 0.023370 | 0.009615 |
| **549** | 10.820 | 24.21 | 68.89 | 361.6 | 0.08192 | 0.06602 | 0.015480 | 0.008160 |
| **550** | 10.860 | 21.48 | 68.51 | 360.5 | 0.07431 | 0.04227 | 0.000000 | 0.000000 |
| **551** | 11.130 | 22.44 | 71.49 | 378.4 | 0.09566 | 0.08194 | 0.048240 | 0.022570 |
| **552** | 12.770 | 29.43 | 81.35 | 507.9 | 0.08276 | 0.04234 | 0.019970 | 0.014990 |
| **553** | 9.333 | 21.94 | 59.01 | 264.0 | 0.09240 | 0.05605 | 0.039960 | 0.012820 |
| **554** | 12.880 | 28.92 | 82.50 | 514.3 | 0.08123 | 0.05824 | 0.061950 | 0.023430 |
| **555** | 10.290 | 27.61 | 65.67 | 321.4 | 0.09030 | 0.07658 | 0.059990 | 0.027380 |
| **556** | 10.160 | 19.59 | 64.73 | 311.7 | 0.10030 | 0.07504 | 0.005025 | 0.011160 |
| **557** | 9.423 | 27.88 | 59.26 | 271.3 | 0.08123 | 0.04971 | 0.000000 | 0.000000 |
| **558** | 14.590 | 22.68 | 96.39 | 657.1 | 0.08473 | 0.13300 | 0.102900 | 0.037360 |
| **559** | 11.510 | 23.93 | 74.52 | 403.5 | 0.09261 | 0.10210 | 0.111200 | 0.041050 |
| **560** | 14.050 | 27.15 | 91.38 | 600.4 | 0.09929 | 0.11260 | 0.044620 | 0.043040 |
| **561** | 11.200 | 29.37 | 70.67 | 386.0 | 0.07449 | 0.03558 | 0.000000 | 0.000000 |
| **562** | 15.220 | 30.62 | 103.40 | 716.9 | 0.10480 | 0.20870 | 0.255000 | 0.094290 |
| **563** | 20.920 | 25.09 | 143.00 | 1347.0 | 0.10990 | 0.22360 | 0.317400 | 0.147400 |
| **564** | 21.560 | 22.39 | 142.00 | 1479.0 | 0.11100 | 0.11590 | 0.243900 | 0.138900 |
| **565** | 20.130 | 28.25 | 131.20 | 1261.0 | 0.09780 | 0.10340 | 0.144000 | 0.097910 |
| **566** | 16.600 | 28.08 | 108.30 | 858.1 | 0.08455 | 0.10230 | 0.092510 | 0.053020 |
| **567** | 20.600 | 29.33 | 140.10 | 1265.0 | 0.11780 | 0.27700 | 0.351400 | 0.152000 |
| **568** | 7.760 | 24.54 | 47.92 | 181.0 | 0.05263 | 0.04362 | 0.000000 | 0.000000 |

569 rows × 31 columns

In [5]:
```
df.head()
```

Out[5]:

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | sy |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.3001 | 0.14710 | 0.2 |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.0869 | 0.07017 | 0.1 |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.1974 | 0.12790 | 0.2 |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.2414 | 0.10520 | 0.2 |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.1980 | 0.10430 | 0.1 |

5 rows × 31 columns

In [6]:
```
# adaboost experiments
# create x and y train
X = df.drop('target', axis=1)
y = df[['target']]

# split data into train and test/validation sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, rand
om_state=101)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(455, 30)
(455, 1)
(114, 30)
(114, 1)
```

In [7]:
```
# check the average cancer occurence rates in train and test data, should be c
omparable
print(y_train.mean())
print(y_test.mean())
```

```
target    0.626374
dtype: float64
target    0.631579
dtype: float64
```

In [8]:
```
# base estimator: a weak learner with max_depth=2
shallow_tree = DecisionTreeClassifier(max_depth=2, random_state = 100)
```

In [9]:
```
# fit the shallow decision tree
shallow_tree.fit(X_train, y_train)

# test error
y_pred = shallow_tree.predict(X_test)
score = metrics.accuracy_score(y_test, y_pred)
score
```

Out[9]: 0.9385964912280702

Now, we will see the accuracy using the AdaBoost algorithm. In this following code, we will write code to calculate the accuracy of the AdaBoost models as we increase the number of trees from 1 to 50 with a step of 3 in the lines:

'estimators = list(range(1, 50, 3))'

'for n_est in estimators:'

We finally end up with the accuracy of all the models in a single list abc_scores.

In [10]:
```
# adaboost with the tree as base estimator

estimators = list(range(1, 50, 3))

abc_scores = []
for n_est in estimators:
    ABC = AdaBoostClassifier(
    base_estimator=shallow_tree,
    n_estimators = n_est)

    ABC.fit(X_train, y_train)
    y_pred = ABC.predict(X_test)
    score = metrics.accuracy_score(y_test, y_pred)
    abc_scores.append(score)
```

In [11]:
```
abc_scores
```

Out[11]:
```
[0.9473684210526315,
 0.956140350877193,
 0.9473684210526315,
 0.9385964912280702,
 0.9385964912280702,
 0.9473684210526315,
 0.9473684210526315,
 0.9649122807017544,
 0.956140350877193,
 0.956140350877193,
 0.9736842105263158,
 0.9736842105263158,
 0.9736842105263158,
 0.9736842105263158,
 0.9824561403508771,
 0.9824561403508771,
 0.9824561403508771]
```

In [12]:
```
# plot test scores and n_estimators
# plot
plt.plot(estimators, abc_scores)
plt.xlabel('n_estimators')
plt.ylabel('accuracy')
plt.ylim([0.85, 1])
plt.show()
```