Exploring the Breast-Cancer Dataset

We will use Recursive Feature Elimination (RFE): Tree Based and Gradient Boosting Estimators

We will employ RFE using:

(1) the Tree based and

(2) the Gradient Boosting

estimators.

**** Note: we will like to compare these results to 'SelectFromModel' method. So we will start with the later.***

In [1]:

```
%matplotlib inline

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.feature_selection import SelectFromModel, RFE
from sklearn.metrics import accuracy_score
```

In [3]:

```
from sklearn.datasets import load_breast_cancer
```

We Import our Data

In [4]:

```
cancer = load_breast_cancer()
cancer.keys()
```

Out[4]:

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

In [5]:

```
cancer.target_names
```

Out[5]:

```
array(['malignant', 'benign'], dtype='<U9')
```

In [6]:

```
X = pd.DataFrame(cancer.data, columns=cancer.feature_names)
```

In [7]:

```
y = cancer.target
```

In [8]:

```
X.head()
```

Out[8]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	wors perimete
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.6
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.8
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.5
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.8
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.2

5 rows × 30 columns

In [9]:

```
X.shape, y.shape
```

Out[9]:

```
((569, 30), (569,))
```

In [10]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.2)
```

In [11]:

```
X_train.shape, X_test.shape
```

Out[11]:

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

Feature Selection by Feature Importance: Using SelectFromModel

We will use SelectFromModel with RandomForestClassifier

In [12]:

```
select = SelectFromModel(RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1))
select.fit(X_train, y_train)
```

Out[12]:

```
SelectFromModel(estimator=RandomForestClassifier(n_jobs=-1, random_state=0))
```

In [13]:

```
select.get_support()
```

Out[13]:

```
array([ True, False,  True,  True, False, False,  True,  True, False,
       False, False, False, False,  True, False, False, False, False,
       False, False,  True, False,  True,  True, False, False, False,
        True, False, False])
```

In [14]:

```
selected_features = X_train.columns[select.get_support()]
selected_features
```

Out[14]:

```
Index(['mean radius', 'mean perimeter', 'mean area', 'mean concavity',
       'mean concave points', 'area error', 'worst radius', 'worst perimeter',
       'worst area', 'worst concave points'],
      dtype='object')
```

In [15]:

```
len(selected_features)
```

Out[15]:

```
10
```

In [16]:

```
X_train_select = select.transform(X_train)
X_test_select = select.transform(X_test)
```

In [17]:

```
X_train_select.shape, X_test_select.shape
```

Out[17]:

```
((455, 10), (114, 10))
```

In [18]:

```
def run_model(X_train, X_test, y_train, y_test):
    rf = RandomForestClassifier(n_estimators=100, random_state=0)
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    print('Accuracy score:', accuracy_score(y_test, y_pred))
```

In [19]:

```
print('---Raw Data (30 features)---')
run_model(X_train, X_test, y_train, y_test)
```

```
---Raw Data (30 features)---
Accuracy score: 0.9649122807017544
```

In [20]:

```
print('---Using Feature Importance (9 features)---')
run_model(X_train_select, X_test_select, y_train, y_test)
```

```
---Using Feature Importance (9 features)---
Accuracy score: 0.9473684210526315
```

We got a score of 94.74% using SelectFromModel with Random Forest Classifier.

Recursive Feature Elimination (RFE): Tree Based Estimator

In [21]:

```
for k in range(1, 31):
    sel = RFE(RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1), n_features_to_select=k)
    sel.fit(X_train, y_train)
    X_train_sel = sel.transform(X_train)
    X_test_sel = sel.transform(X_test)
    print('Selected feature:', k)
    run_model(X_train_sel, X_test_sel, y_train, y_test)
    print()
```

```
Selected feature: 1
Accuracy score: 0.8947368421052632

Selected feature: 2
Accuracy score: 0.9298245614035088

Selected feature: 3
Accuracy score: 0.9473684210526315

Selected feature: 4
Accuracy score: 0.9649122807017544

Selected feature: 5
Accuracy score: 0.9649122807017544

Selected feature: 6
Accuracy score: 0.956140350877193

Selected feature: 7
Accuracy score: 0.956140350877193
```

```
Selected feature: 8
Accuracy score: 0.9649122807017544

Selected feature: 9
Accuracy score: 0.9736842105263158

Selected feature: 10
Accuracy score: 0.9736842105263158

Selected feature: 11
Accuracy score: 0.9649122807017544

Selected feature: 12
Accuracy score: 0.9736842105263158

Selected feature: 13
Accuracy score: 0.9649122807017544

Selected feature: 14
Accuracy score: 0.9736842105263158

Selected feature: 15
Accuracy score: 0.9736842105263158

Selected feature: 16
Accuracy score: 0.9736842105263158

Selected feature: 17
Accuracy score: 0.9824561403508771

Selected feature: 18
Accuracy score: 0.9649122807017544

Selected feature: 19
Accuracy score: 0.9649122807017544

Selected feature: 20
Accuracy score: 0.9736842105263158

Selected feature: 21
Accuracy score: 0.9736842105263158

Selected feature: 22
Accuracy score: 0.9736842105263158

Selected feature: 23
Accuracy score: 0.9649122807017544

Selected feature: 24
Accuracy score: 0.9824561403508771

Selected feature: 25
Accuracy score: 0.956140350877193

Selected feature: 26
Accuracy score: 0.956140350877193

Selected feature: 27
Accuracy score: 0.9649122807017544

Selected feature: 28
Accuracy score: 0.9649122807017544

Selected feature: 29
Accuracy score: 0.9649122807017544

Selected feature: 30
Accuracy score: 0.9649122807017544
```

In [25]:

```python
sel = RFE(RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1), n_features_to_select=17)
sel.fit(X_train, y_train)
X_train_sel = sel.transform(X_train)
X_test_sel = sel.transform(X_test)
print('Selected feature:', 17)
run_model(X_train_sel, X_test_sel, y_train, y_test)
```

```
Selected feature: 17
Accuracy score: 0.9824561403508771
```

In [26]:

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

Out[26]:

```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean concavity', 'mean concave points', 'radius error', 'area error',
       'worst radius', 'worst texture', 'worst perimeter', 'worst area',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry'],
      dtype='object')
```

In [27]:

```
len(features)
```

Out[27]:

```
17
```

Recursive Feature Elimination: Gradient Boost Estimator

In [24]:

```
for k in range(1, 31):
    rfe = RFE(GradientBoostingClassifier(n_estimators=100, random_state=0), n_features_to_select=k)
    rfe.fit(X_train, y_train)
    X_train_rfe = rfe.transform(X_train)
    X_test_rfe = rfe.transform(X_test)
    print('Selected feature:', k)
    run_model(X_train_rfe, X_test_rfe, y_train, y_test)
    print()
```

```
Selected feature: 1
Accuracy score: 0.8771929824561403

Selected feature: 2
Accuracy score: 0.9035087719298246

Selected feature: 3
Accuracy score: 0.9649122807017544

Selected feature: 4
Accuracy score: 0.9736842105263158

Selected feature: 5
Accuracy score: 0.9649122807017544

Selected feature: 6
Accuracy score: 0.9912280701754386

Selected feature: 7
Accuracy score: 0.9736842105263158

Selected feature: 8
Accuracy score: 0.9649122807017544

Selected feature: 9
Accuracy score: 0.9736842105263158

Selected feature: 10
Accuracy score: 0.956140350877193

Selected feature: 11
Accuracy score: 0.956140350877193

Selected feature: 12
Accuracy score: 0.9736842105263158

Selected feature: 13
Accuracy score: 0.956140350877193

Selected feature: 14
Accuracy score: 0.956140350877193

Selected feature: 15
Accuracy score: 0.9649122807017544

Selected feature: 16
Accuracy score: 0.956140350877193
```

```
Selected feature: 17
Accuracy score: 0.9649122807017544

Selected feature: 18
Accuracy score: 0.9473684210526315

Selected feature: 19
Accuracy score: 0.9649122807017544

Selected feature: 20
Accuracy score: 0.9473684210526315

Selected feature: 21
Accuracy score: 0.9649122807017544

Selected feature: 22
Accuracy score: 0.9649122807017544

Selected feature: 23
Accuracy score: 0.9649122807017544

Selected feature: 24
Accuracy score: 0.9649122807017544

Selected feature: 25
Accuracy score: 0.9736842105263158

Selected feature: 26
Accuracy score: 0.9736842105263158

Selected feature: 27
Accuracy score: 0.9649122807017544

Selected feature: 28
Accuracy score: 0.9649122807017544

Selected feature: 29
Accuracy score: 0.9649122807017544

Selected feature: 30
Accuracy score: 0.9649122807017544
```

In [28]:

```
rfe = RFE(GradientBoostingClassifier(n_estimators=100, random_state=0), n_features_to_select=6)
rfe.fit(X_train, y_train)
X_train_rfe = rfe.transform(X_train)
X_test_rfe = rfe.transform(X_test)
print('Selected feature:', 6)
run_model(X_train_rfe, X_test_rfe, y_train, y_test)
```

```
Selected feature: 6
Accuracy score: 0.9912280701754386
```

In [30]:

```
features = X_train.columns[rfe.get_support()]
features
```

Out[30]:

```
Index(['mean concave points', 'area error', 'worst texture', 'worst perimeter',
       'worst area', 'worst concave points'],
      dtype='object')
```

We got a score of 99.13% using RFE with Gradient Boosting Classifier.