recursive_elimination_of_features

February 17, 2019

In [1]:
```
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.datasets import make_classification
X, y = make_classification(n_samples = 100, n_features=100,
                           n_informative=5, n_redundant=2, random_state=101)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=
```

```
D:\Python\Lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufunc size changed, may indica
  return f(*args, **kwds)
```

In [5]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=101, solver="lbfgs")
classifier.fit(X_train, y_train)
print("Accuracy of trening samples: %3f" % classifier.score(X_train, y_train))
print("Accuracy of test samples: %3f" % classifier.score(X_test, y_test))
```

```
Accuracy of trening samples: 1.000000
Accuracy of test samples: 0.766667
```

In [6]:
```
from sklearn.feature_selection import RFECV #Recursive feature elimination with cross-
selector = RFECV(estimator=classifier, step=1, cv=10, scoring='accuracy')
selector.fit(X_train, y_train)
print("Optimal features number: %d" % selector.n_features_)
```

```
Optimal features number: 31
```

In [9]:
```
X_train_s = selector.transform(X_train)
X_test_s = selector.transform(X_test)
classifier.fit(X_train_s, y_train)
print("Accuracy of trening samples: %3f" % classifier.score(X_train_s, y_train))
print("Accuracy of test samples: %3f" % classifier.score(X_test_s, y_test))
```

```
Accuracy of trening samples: 1.000000
Accuracy of test samples: 0.666667
```

```
In [11]: selector.fit(X_train_s, y_train)
         print("Optimal features number: %d" % selector.n_features_)
```

```
Optimal features number: 29
```

```
In [13]: X_train_s2 = selector.transform(X_train_s)
         X_test_s2 = selector.transform(X_test_s)
         classifier.fit(X_train_s2, y_train)
         print("Accuracy of trening samples: %3f" % classifier.score(X_train_s2, y_train))
         print("Accuracy of test samples: %3f" % classifier.score(X_test_s2, y_test))
```

```
Accuracy of trening samples: 1.000000
Accuracy of test samples: 0.666667
```