MS 506 Probability and Statistical Inference

Lecture 30: Classifier Evaluation

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

Evaluating a classification model

1. We require a model evaluation procedure that can quantify the quality of performance of a classification model
2. This procedure needs to produce a numerical quantity that can be used to compare models

In [18]:
```
## WIll use the breast cancer dataset
bc =load_breast_cancer()
X,y = bc.data,bc.target ## Getting the feature matrix and target
```

In [19]:
```
print(bc.DESCR)
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the
class

    :Attribute Information:
        - radius (mean of distances from center to points on the peri
meter)
```

```
        - texture (standard deviation of gray-scale values)

        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the
three
        worst/largest values) of these features were computed for eac
h image,
        resulting in 30 features.  For instance, field 0 is Mean Radi
us, field
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
    concave points (mean):                0.0    0.201
    symmetry (mean):                      0.106  0.304
    fractal dimension (mean):             0.05   0.097
    radius (standard error):              0.112  2.873
    texture (standard error):             0.36   4.885
    perimeter (standard error):           0.757  21.98
    area (standard error):                6.802  542.2
    smoothness (standard error):          0.002  0.031
    compactness (standard error):         0.002  0.135
    concavity (standard error):           0.0    0.396
    concave points (standard error):      0.0    0.053
    symmetry (standard error):            0.008  0.079
    fractal dimension (standard error):   0.001  0.03
    radius (worst):                       7.93   36.04
    texture (worst):                      12.02  49.54
```

```
    perimeter (worst):                    50.41  251.2
    area (worst):                         185.2  4254.0
    smoothness (worst):                   0.071  0.223
    compactness (worst):                  0.027  1.058
    concavity (worst):                    0.0    1.252
    concave points (worst):               0.0    0.291
    symmetry (worst):                     0.156  0.664
    fractal dimension (worst):            0.055  0.208
    ===================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangas
arian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) dataset
s.
https://goo.gl/U2Uwz2 (https://goo.gl/U2Uwz2)

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References
```

```
   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature
extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposi
um on
     Electronic Imaging: Science and Technology, volume 1905, pages 8
61-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer di
agnosis and
     prognosis via linear programming. Operations Research, 43(4), pa
ges 570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learnin
g techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Let
ters 77 (1994)
     163-171.
```

In [20]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.5,r
```

Fitting some classification model

In [21]:
```
clf = DecisionTreeClassifier(random_state=0).fit(X_train,y_train)
pred_test_y = clf.predict(X_test)
metrics.accuracy_score(y_test,pred_test_y)
```

Out[21]:
```
0.9298245614035088
```

Problem with accuracy as a measure

In [22]:
```
set(y_test)
```

Out[22]:
```
{0, 1}
```

In [24]:
```
len(y_test)
```

Out[24]:
```
285
```

In [25]:
```
print(f'Number of y = 0 samples: {len(y_test[y_test==0])}')
print(f'Number of y = 1 samples: {len(y_test[y_test==1])}')
```

```
Number of y = 0 samples: 101
Number of y = 1 samples: 184
```

If the model doesnt learn anything and predicts class 1 everytime:

In [26]:
```
print(f'Accuracy: {184/(184+101)}')
```

```
Accuracy: 0.6456140350877193
```

Hence our model is learning something useful by bringing the accuracy to 92.9% from 64.5%

Now suppose the classes were even more imbalanced:

```
1. 20
0. 2
```

Now if a model predicts class 1 everytime

In [27]:
```
print(f'Accuracy: {20/22}')
```

```
Accuracy: 0.9090909090909091
```

Hence for highly class imbalanced datasets, without learning anything your model can have a high accuracy. Hence Accuracy is not always a good measure of performance.

Exercise

Suppose you have a test dataset with 4 classes with the following distribution

1. Class 0: 3 samples
2. Class 1: 4 samples
3. Class 2: 2 samples
4. Class 4: 25 samples

What is the worst possible accuracy your model should atleast achieve for it to be a valid classification learner

In [28]:
```
25/(25+3+4+2)
```

Out[28]:
```
0.7352941176470589
```

Hence, classification accuracy alone cant show the full picture

As in it misses the data distribution

Confusion matrix

In [10]:
```
confusion = metrics.confusion_matrix(y_test,pred_test_y)
confusion
```

Out[10]:
```
array([[ 90,  11],
       [  9, 175]])
```

	Predicted Class 0	Predicted Class 1
Actual Class 0	90	11
Actual Class 1	9	175

Size is 2 x 2 as we just have 2 possible labels

1. True Positive (TP): Predicted Class 1 when the true class was also 1
2. True Negative (TN): Predicted Class 0 when the true class was also 0
3. False Positive (FP): Predicted Class 1 when true class was 0 (Predicting cancer when the person is healthy: Type I error)
4. False Negative (FN): Predicted Class 0 when true class was 1 (Predicting healthy when person has cancer: Type II error)

Threshold based metrics

They summarize the fraction, ratio, or rate of when a predicted class does not match the expected class in a holdout dataset

1. Classification accuracy

In [30]:
```
TP = confusion[1,1]
TN = confusion[0,0]
FP = confusion[0,1]
FN = confusion[1,0]
print('Model accuracy from confusion matrix: ',(TP+TN)/(TP+TN+FP+FN))
print('Model accuracy from sklearn: ',metrics.accuracy_score(y_test,pr
```

```
Model accuracy from confusion matrix:  0.9298245614035088
Model accuracy from sklearn:  0.9298245614035088
```

2. Classification error

In [32]:
```
print('Model error from confusion matrix: ',(FP+FN)/(TP+TN+FP+FN))
print('Model error from sklearn: ',1 - metrics.accuracy_score(y_test,p
```

```
Model error from confusion matrix:  0.07017543859649122
Model error from sklearn:  0.07017543859649122
```

3. Sensitivity or Recall

When the actual label is 1 (True) how often is the prediction correct

In [35]:
```
print('Sensitivity from confusion matrix: ',(TP)/(TP+FN))
print('Sensitivity from sklearn',metrics.recall_score(y_test,pred_test
```

```
Sensitivity from confusion matrix:  0.9510869565217391
Sensitivity from sklearn 0.9510869565217391
```

In [36]:
```
175/(175+9)
```

Out[36]:
```
0.9510869565217391
```

4. Specificity

When the actual label is 0(False), how often is the prediction correct

In [37]:
```
print('Specificity from confusion matrix: ',(TN)/(TN + FP))
```

```
Specificity from confusion matrix:  0.8910891089108911
```

Exercise:

1. For our case, sensitivity is coming out to be greater than specificity. Is it good or bad ? Explain

5. False Positive rate

When the actual value is 0(False) how often is the prediction incorrect

In [38]:
```
print('False positive rate confusion matrix: ',(FP)/(TN +FP))
```

```
False positive rate confusion matrix:  0.10891089108910891
```

6. Precision

When we predict 1(True), how often is the prediction correct

In [17]:
```
print('Precision from confusion matrix: ',(TP)/(TP +FP))
print('Precision from sklearn: ',metrics.precision_score(y_test,pred_t
```

```
Precision from confusion matrix:  0.9408602150537635
Precision from sklearn:  0.9408602150537635
```

While selecting a classification model, you can focus on either of these metrics

Exercise:

1. For each of the metric, comment on whether you would prefer a higher value or lower value
   A. Classification Accuracy
   B. Classification Error
   C. Sensitivity
   D. Specificity
   E. False Positive rate
   F. Precision
2. Think of an example where computing and analyzing Precision is very important

In [ ]:
```
```