GENERAL ASSEMBLY

INTRO TO DATA SCIENCE
LECTURE :KNN, SVM, ROC

K-NEAREST NEIGHBORS

KEY CONCEPTS - K-NEAREST NEIGHBORS (K-NN)

- One of the simplest classification methods because the classification is done just by looking at the K closest examples in the training set
- ‘Close’ in the sense of some distance or similarity measure
- Usually use Euclidean distance
- The class prediction for the new example is the result of the vote of the K-nearest neighbors - taking the majority result

KEY CONCEPTS - K-NEAREST NEIGHBORS (K-NN)

- K-NN - prediction speed will be proportional to the number of training examples and the dimensionality of your feature set

- Can be a good algorithm for small datasets, but does not scale well

- Two parameters required for this algorithm:

    - The neighborhood cardinality - K
    - The similarity measure, default is euclidean distance or L2

KEY CONCEPTS - K-NEAREST NEIGHBORS (KNN)

- sklearn: kNeighborClassifier() and kNeighborRegressor()

- The regressor simply takes the average of the values of the k nearest neighbors, i.e. it's the average of the neighborhood

SUPPORT VECTOR MACHINES

KEY CONCEPTS - SUPPORT VECTOR MACHINE

- Powerful advanced supervised learning algorithm
- Classification and Regression
- Will fit both linear and non-linear functions
- Very popular
- ***Does not output a probability***, but rather makes a prediction directly
  - there are ‘work arounds’ in sklearn, where estimation of probabilities can be made
- Referred to as a Large Margin Classifier

Key Concepts - Kernels & Non-linear Decision Boundaries

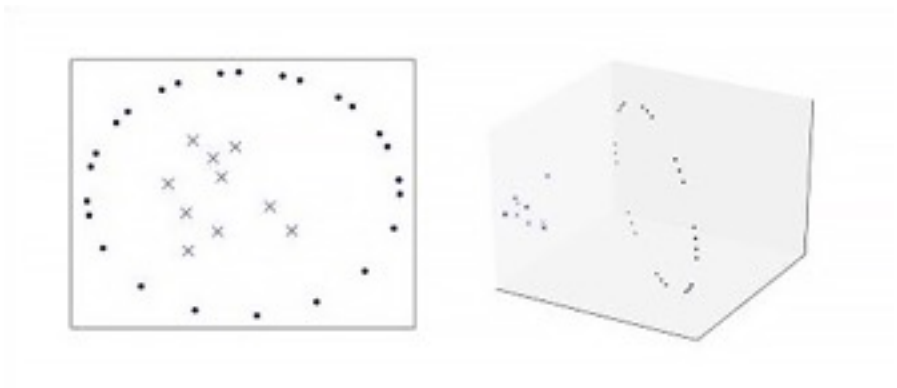
- One method, as we have explored already, of producing a non-linear decision boundary has involved the production of polynomial features

- SVM's can be adapted to produce non-linear decision boundaries by using kernels

KEY CONCEPTS - KERNELS & THE KERNEL TRICK

- Previously we added polynomial features to a ***linear*** model to create a non-linear decision boundary

- This works by creating new features, and thus increasing the dimensionality of feature space, such that the dataset becomes linearly separable in the high dimensional space!



from Mastering Machine Learning with scikit-learn, Gavin Hackeling

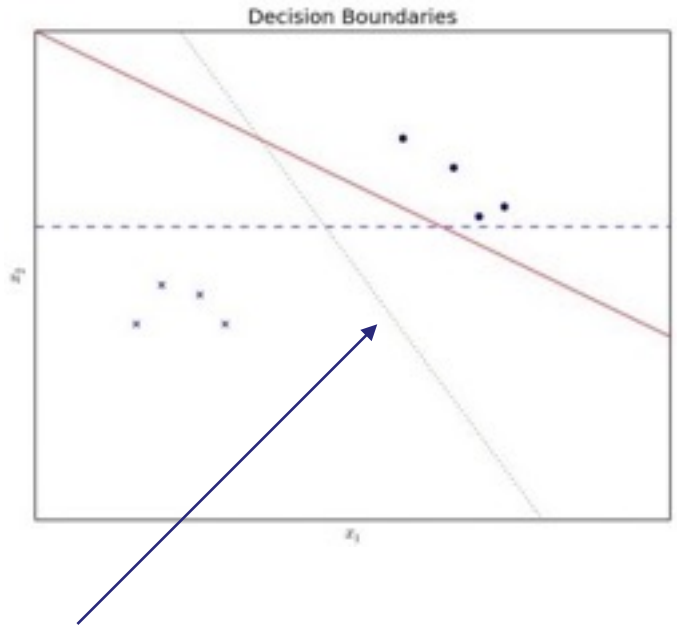
KEY CONCEPTS - KERNELS & THE KERNEL TRICK

- This is how you express a non-linear function using a linear model
- The so-called kernel trick is a different method to achieve the same thing as mapping features to a high dimensional space (without actually having to do it!)
- Kernels produce the same values through a different series of operations that can usually be computed more efficiently

KEY CONCEPTS - MAXIMUM MARGIN CLASSIFICATION & SUPPORT VECTORS

Consider 2 linearly separable classes, and possible decision boundaries

Which decision boundary is likely to generalize the best?, i.e.work on the test set and real world data the best?

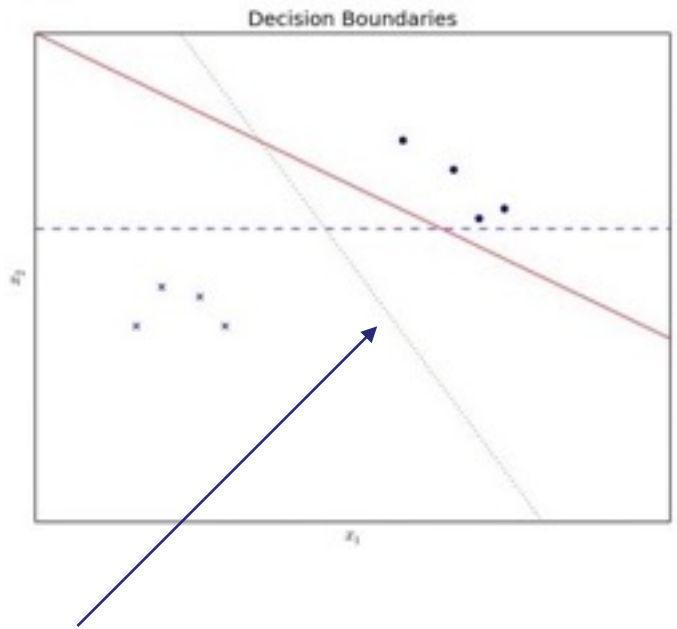


Intuitively this is the best

from Mastering Machine Learning with scikit-learn, Gavin Hackeling

KEY CONCEPTS - MAXIMUM MARGIN CLASSIFICATION & SUPPORT VECTORS

The most confident predictions are those made where the points are most distant from the decision boundary



line represents p = 0.5

from Mastering Machine Learning with scikit-learn, Gavin Hackeling

KEY CONCEPTS - SUPPORT VECTOR MACHINE

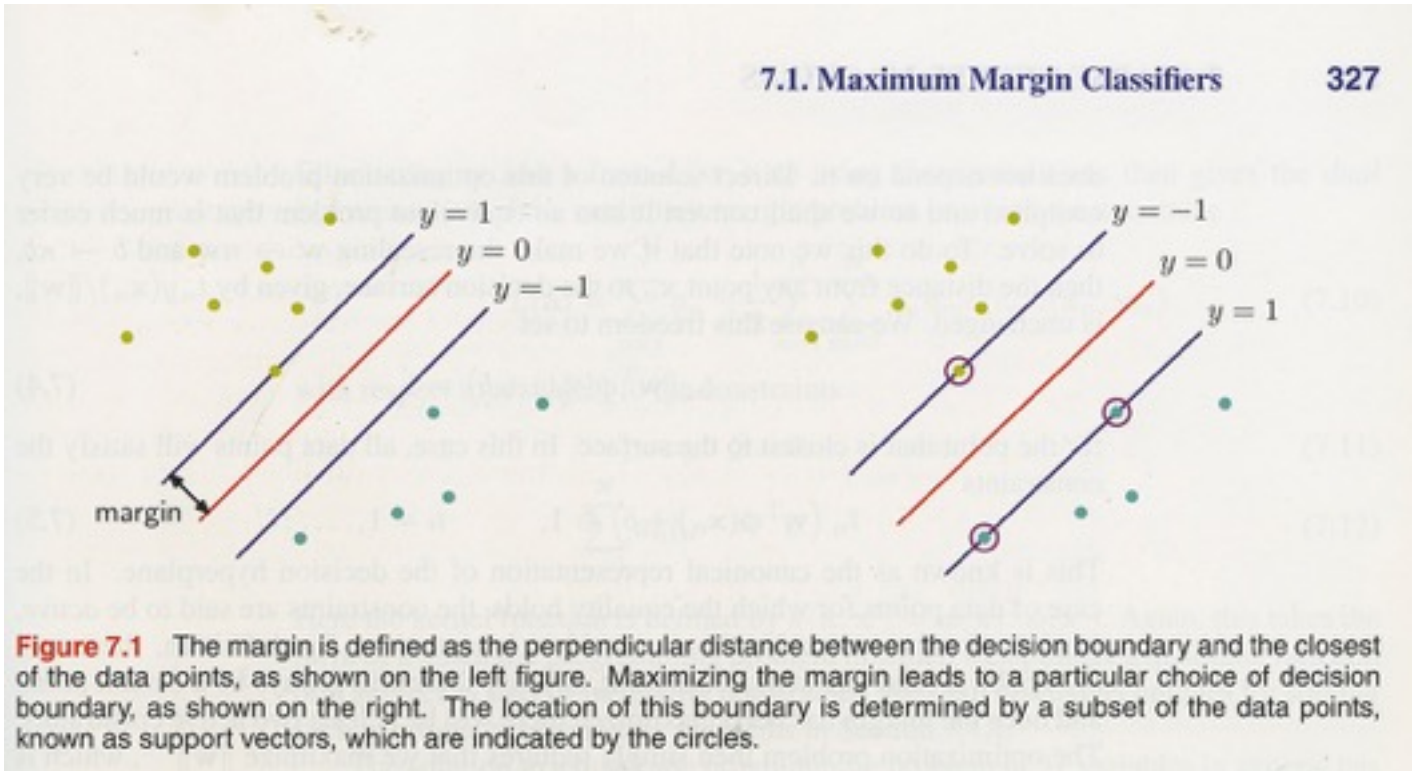


Figure 7.1 The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left figure. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles.

from Pattern Recognition and Machine Learning, Christopher M. Bishop

KEY CONCEPTS - SUPPORT VECTOR MACHINE

- The best decision boundary is that with the maximum margin
- This results in a modified optimization problem
- One benefit of this particular optimization problem is that is has a guaranteed global minimum
- Effectively you can find parameters that maximize the margin
- The problem is sometimes solved using Sequential Minimal Optimization (SMO)

KEY CONCEPTS - SUPPORT VECTOR MACHINE - SKLEARN

- Sklearn offers 2 implementations based upon different libraries

1. LIBSVM

    - this is a complete library of SVM classification (SVC) and regression (SVR) implementations

2. LIBLINEAR

    - this is a linear classification implementation for large datasets, especially sparse text

KEY CONCEPTS - SUPPORT VECTOR MACHINE - SKLEARN

- These are both well respected, well tested open source libraries, originally developed at the National Taiwan University
- Both written in C++
- They are both fast and reliable

KEY CONCEPTS - SUPPORT VECTOR MACHINE - SKLEARN SUPPORT

- LIBSVM
    - must set the cache_size parameter which allocates memory for the kernel cache (at the heart of the algorithm)
    - specified in Mb.
    - 200 is the default, 500 - 1000 is a better setting
- Both algorithms require ndarray or sparse matrices using a dtype of float64
    - they can convert internally, but this may slow performance

KEY CONCEPTS - SCALE-UP - LIBSVM VS LIBLINEAR

- SVM for classification (SVC) has a relationship to the number of training examples.
- Computational complexity grows with the cube of the size of the dataset!
- Linear SVC scales linearly
- Liblinear is particularly suited for linear SVC on large datasets

KEY CONCEPTS - SUPPORT VECTOR MACHINE - TUNING

- Tuning SVMs

- SVM are sensitive to variables of different scale and large numbers
  - the algorithm can be dominated by features of a large range or variance
  - extremely low or high numbers may cause problems to the optimization process
  - range scaling to [-1, 1] is advisable, or unit mean, zero variance

- Unbalanced classes can also be a problem
  - the algorithm will tend to learn the more frequent pattern
  - the are function parameters that can be used to compensate for this
  - this can be complicated and is beyond our scope - but be aware that very skewed datasets may produce unexpected results

KEY CONCEPTS - SUPPORT VECTOR MACHINE - SKEWED DATA

- Can you re-sample your data to balance the classes?
- If not you can weight the C parameter according to the frequency of the class
- The are other function arguments available to help with this problem depending upon the implementation that you use
- When using a SVM consult the library documentation for the implementation you are using!!

KEY CONCEPTS - SUPPORT VECTOR MACHINE - HYPER-PARAMETERS

- Sklearn actually provides a number of SVM implementations
- I have focused on SVM - SVC and SCR
- The hyper-parameters are implementation dependent, so read the documentation carefully

KEY CONCEPTS - SUPPORT VECTOR MACHINE - HYPER-PARAMETERS FOR SVM

- C - regularization parameter.
  - ***Large C = low regularization***, meaning lower bias, higher variance
  - ***Small C = high regularization***, meaning higher bias, lower variance

- Choice of Kernel - similarity function
  - None! - “linear” kernel (don’t produce features, f)
    - Linear decision boundary
  - Gaussian kernel (called an ‘rbf’ or Radial Basis Function)
    - Complex, non-linear decision boundary
    - Must perform feature scaling!!

Key Concepts - Support Vector Machine - A Note on Similarity Functions

- Sometimes you have to provide the similarity function - depends on the package
- Not all similarity functions are “valid” kernels
- Kernels need to satisfy a technical condition called Mercer’s Theorem
- The kernel must ensure convergence
- Linear, Gaussian kernels obviously satisfy this, and are the most popular
- But also:
  - Polynomial kernel
  - Other more esoteric kernels - string kernel, chi-square kernel, etc etc

KEY CONCEPTS - SUPPORT VECTOR MACHINE - HYPER-PARAMETERS

- Gamma - Kernel scale parameter
  - Large Gamma = rapid fall off

- Gamma is related to how spread out your data points are. If they are very far from each other (which would happen in a very high dimensional space for example), then you don't want the kernel to drop off quickly, so you would use a small gamma
  - Thus libSVM uses a default of 1/num_features (NB:features, NOT samples)
  - Smooth Kernel (small gamma) with slow drop off, gives higher bias, and lower variance
  - Abrupt Kernel (large gamma) with quick drop off, gives lower bias, and higher variance

KEY CONCEPTS - SUPPORT VECTOR MACHINE - MULTI-CLASS CLASSIFICATION

- Multi-class classification is built into most packages
- Can use one-vs-all method if necessary
  - For K classes train K SVMs, one vs the-rest

KEY CONCEPTS - SVM OR LOGISTIC REGRESSION? BROAD USAGE GUIDELINES

- N is features, m is training examples
- LR - Logistic Regression, SVM - Support Vector Machine

- N is large relative to m [n>m, n=10000, m=10..1000]
  - use LR, or SVM with a linear kernel
  - not enough data to fit a complicated function

- N is small, m is intermediate [n=1-1000, m=10..10000]
  - use SVM with a Gaussian kernel

- N is small, m is large [n=1-1000, m=50000+]
  - Try to add/create more features
  - Use SVM with a linear kernel, or LR

EVALUATING CLASSIFIERS - PRECISION, RECALL AND F1-SCORE

Key Concepts - Evaluating Classifiers

- True positive: correctly classifying a positive case
- True negative: correctly classifying a negative case
- False positive: a negative case incorrectly classified as positive
- False negative: a positive case incorrectly classified as negative

KEY CONCEPTS - EVALUATING CLASSIFIERS

- We have focussed on classification accuracy as measure of classification performance
- While this is perfectly fine it omits other considerations, such as not examining the FP/FN
- If, for example, you had a classifier to detect malignant tumors then you would be interested in more than just the overall accuracy
- Failing to detect a malignant tumor (a falsely negative outcome) is a more severe error than identifying a malignant tumor, when in fact it is not (malignant)

- Skewed classes, example: consider cancer diagnosis
- Classification model giving 1 if the person has cancer and 0 if not
- Suppose you have a 1% error on your test set (99% correct)
- Suppose the incidence of cancer in our data is 0.5%
- If your algorithm just predicted 0 all the time, then your error will be 0.5% reflecting the cancer incidence in the training, validation and test sets

```
def predict(x):
    return 0
```

KEY CONCEPTS - ERROR METRICS

- Simply classification accuracy does not tell you the whole story
- In the presence of rare cases, such as a person having cancer, we want to predict y=1
- We need error metrics to reflect how well we are doing

KEY CONCEPTS - ERROR METRICS - PRECISION

Precision:

- Of all the patients we predicted y = 1, what fraction actually has cancer
- TP/(all predicted postive)
- Precision = TP/(TP+FP)

KEY CONCEPTS - ERROR METRICS - RECALL

Recall:

- Of all the patients that actually have cancer, what fraction did we correctly detect as having cancer
- TP/(number of actual positives)
- Recall = TP/(TP + FN)
- In the medical domain often referred to as sensitivity

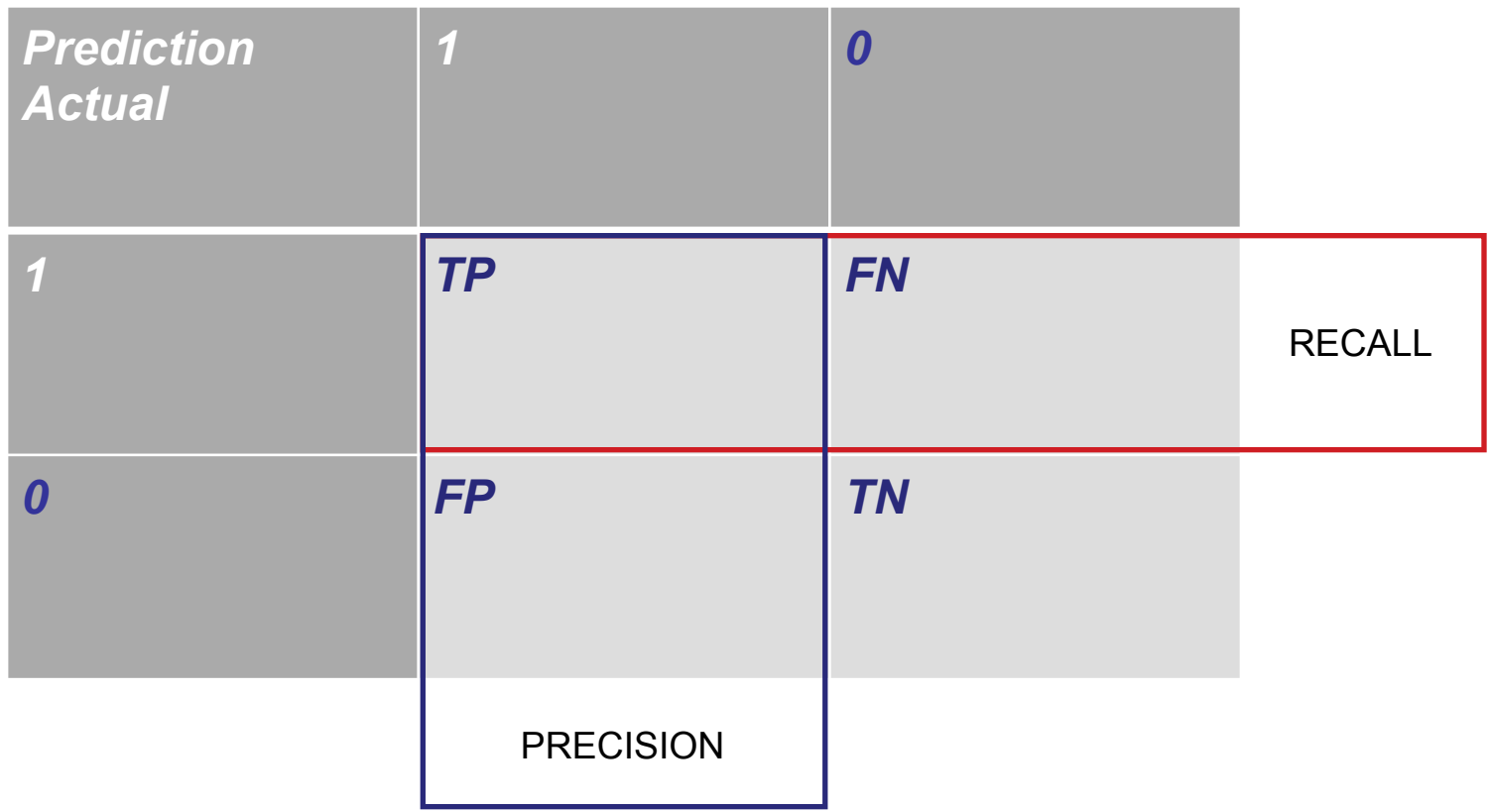
KEY CONCEPTS - ERROR METRICS - CONVENTION

Precision and Recall convention:

- set y = 1 in the presence of the rare class

KEY CONCEPTS - ERROR METRICS - PRECISION AND RECALL

In the case of 2 classes



Key Concepts - Error Metrics - Precision and Recall

- May need to control the trade-off between precision and recall
- Suppose we want to predict cancer (y = 1) if we are very confident
- Modify the probability threshold for predicting y = 1
- Usually if p > 0.5 then y = 1.
- Change this to if p > 0.9 then y = 1.
- This results in high precision and lower recall, improved TP/(all predicted positive)

On the other hand:

- Suppose we want to avoid missing too many cases of cancer (i.e. avoid the falsely negative).
- A false negative has high implication in our cancer example. Telling someone they don't have cancer when in fact they do
- So when in doubt, predict y = 1
- In this change the cutoff to $p > 0.3$ then $y = 1$.
- This leads to a high recall classifier, with lower precision.

Key Concepts - Error Metrics - Precision and Recall

- For any given classifier you need to decide the kind of classifier you want - the trade off between precision and recall.
- You do this by setting the critical value at which your classifier assigns input to target

KEY CONCEPTS - ERROR METRICS - F1-SCORE

- Comparing different precision/recall numbers
- How do you tell which classifier is best when you have different classifiers producing different sets of precision/recall numbers
- Ideally we still want a single number evaluation metric
- F1 Score or F Score

KEY CONCEPTS - ERROR METRICS - F1-SCORE

- F1-score penalizes classifiers with imbalance between their precision and recall scores

KEY CONCEPTS - ERROR METRICS - F1-SCORE

- Average = (P + R)/2??? Not a good solution, because the average will be influenced by having P or R at the extremes of the values that they can take (1!)
- F1 Score = 2 (PR)/(P+R) - the product term in the numerator ensures models at the extreme (i.e. very high precision or very high recall) are weighted less
- F1 Score is a weighted average of Precision and Recall
- F1 Score ranges between 0 and 1
- Can use cross validation to tune the threshold to maximize the F1 score

KEY CONCEPTS - ERROR METRICS - F1-SCORE

	Precision	Recall	Average	F1-Score
Algorithm A	0.5	0.4	0.45	0.444
Algorithm B	0.7	0.1	0.4	0.175
Algorithm C	0.02	1.0	0.51	0.0392

KEY CONCEPTS - SKLEARN METRICS

sklearn metrics:

- accuracy_score() Also note that models provide a .score() function
- precision_score()
- recall_score()
- f1_score()
- classification_report() - provides precision, recall and f1
- The reason for having them available separately is so you can cross-validate against them

KEY CONCEPTS - THE MULTI-CLASS CASE

- Consider each class separately

Remember:

- Precision is the fraction of events where we correctly declared y out of all instances where the algorithm declared y.
- TP/all predicted

- Recall is the fraction of events where we correctly declared y out of all of the cases where the true of state of the world is y
- TP/all actual

KEY CONCEPTS - THE MULTI-CLASS CASE

- Iris data confusion matrix
- Each class must be considered in isolation

Prediction Actual	*Setosa*	*Versicolor*	*Virginica*
Setosa	*TN = 23*	*FP = 0*	
Versicolor	*FN = 0*	*TP = 28*	*FN = 1*
Virginica		*FP = 3*	*TN = 20*

KEY CONCEPTS - SKLEARN CLASSIFICATION REPORT

- The sklearn classification report
  - *from sklearn.metrics import classification report*

```
In [46]: print "Accuracy {:5.2f}".format(clf.score(X_test, y_test))
         Accuracy  0.95

In [27]: print classification_report(y_test, y_hat)
                      precision    recall  f1-score   support

                   0       1.00      1.00      1.00        23
                   1       0.90      0.97      0.93        29
                   2       0.95      0.87      0.91        23

         avg / total       0.95      0.95      0.95        75
```

KEY CONCEPTS - ERROR METRICS - FALL-OUT

- Sensitivity - If a disease is present then the test will pick it up
- Specificity - What other diseases are also picked up when the disease is not present

Wikipedia:

- If 100 patients known to have a disease were tested, and 43 test positive, then the test has 43% sensitivity.

- If 100 with no disease are tested and 96 return a negative result, then the test has 96% specificity.

- Want tests of high sensitivity and specificity

KEY CONCEPTS - ERROR METRICS - FALL-OUT (1 - SPECIFICITY)

Fall-out:

- Of all the patients we incorrectly predicted y = 1, what fraction actually has not got cancer
- FP/(number of actual negatives)
- Fall-out = FP/(TN+FP)
- Also referred to as 1-specificity

EVALUATING CLASSIFIERS - ROC & AUC

KEY CONCEPTS - RECEIVER OPERATING CHARACTERISTIC (ROC)

- A graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied
- ROC is ***insensitive*** to skewed data
- The curve is created by plotting the TP (or hit) rate against the FP (or miss) rate at various threshold settings

KEY CONCEPTS - RECEIVER OPERATING CHARACTERISTIC (ROC)

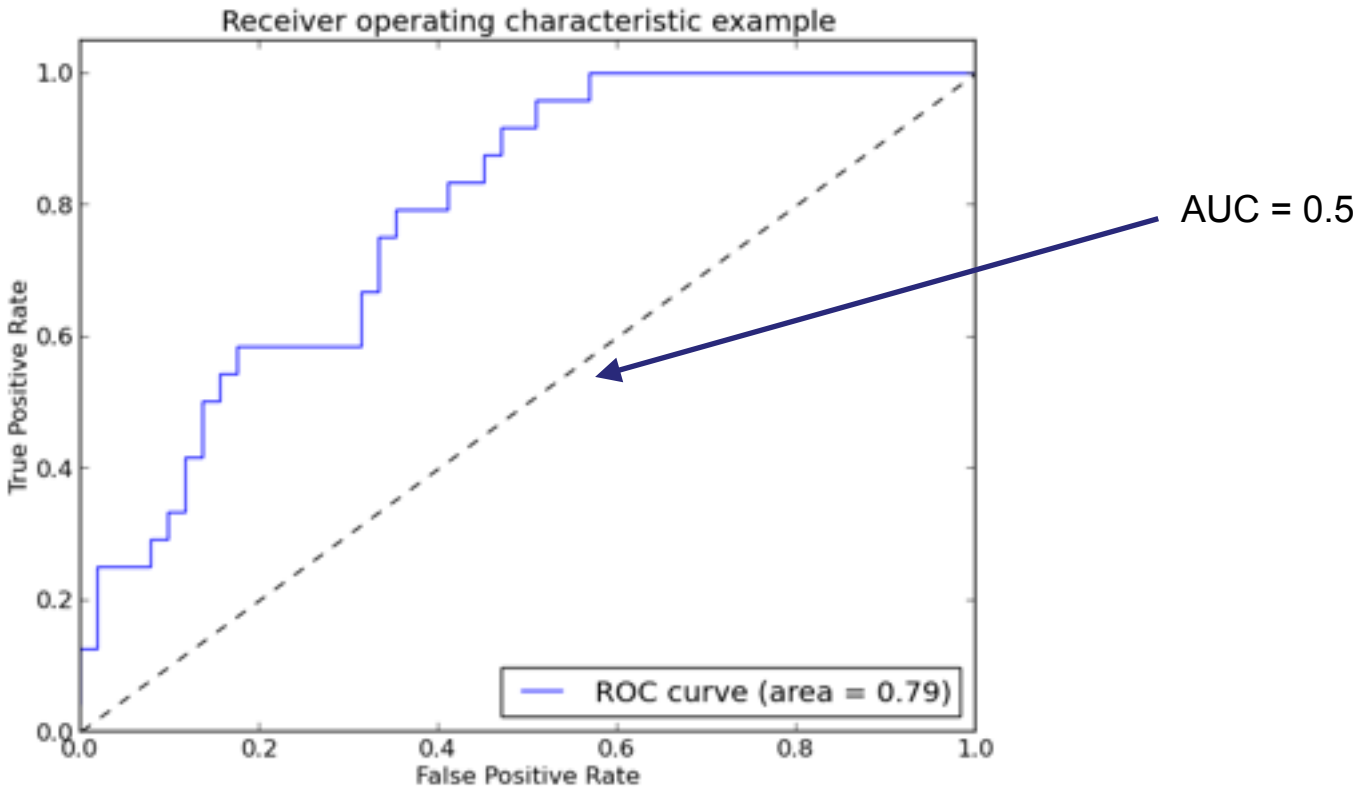
- The rate of correct positive results vs the rate of incorrect results

- The area under the curve represents how well the classifier performs with respect to a random classifier, whose AUC = 0.5

KEY CONCEPTS - RECEIVER OPERATING CHARACTERISTIC (ROC)

- Developed by “receiver operators” during WWII for radar-signal detection methodology (signal-to-noise), hence “Radar Receiver Operator Characteristic”)
- Used extensively in medical and psychological test evaluation
- More recently used in machine learning

KEY CONCEPTS - RECEIVER OPERATING CHARACTERISTIC (ROC)



KEY CONCEPTS - RECEIVER OPERATING CHARACTERISTIC (ROC)