METRICS

Dr. Brian Mc Ginley

EVALUATING MACHINE LEARNING MODELS

- How predictive is our trained model?
  - For regression, usually
    - Mean Squared Error (MSE) : What we used for linear regression
    - R^2
    - p-values (Hypothesis testing)
    - F-stats (ANOVA)
    - AIC/BIC
  - For classification, many options
    - ROC curve
    - Lift Chart
    - Accuracy score
    - F1 Score
    - Recall, Precision

CHOOSING A METRIC

- The procedure of systematically choosing a set of predictors (parameters) that have a significant relationship with the response variable is called variable selection (training).
- But which metric (F-stats, p-values [hypothesis testing], R^2, AIC/BIC) should we use to determine the significance of a set of predictors?
- Rather than relying on a single metric, we should use multiple metrics and double check with common sense.

CLASSIFICATION EVALUATION

- For evaluating Classification algorithms, often people just see how accurate the predictions are on the test set

```
y_pred = model.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

- Will give a score (≤ 1) which is the fraction of how many predictions were correct.
- This stores the y predictions separately (which may be useful later) but the model has a method itself that can be used if you just want the accuracy score

```
model.score(X_test, y_test)
```

- This one line does the predicting and comparing of predictions in the same step.

ACCURACY SCORE

- Accuracy score is a sometimes called a very *harsh* metric and it does not tell us how a classifier performs for particular classes.
- The number digit classifier has 10 possible classes, i.e. 0,1,2,3,4,5,6,7,8,9. Maybe we just care about the performance of the number 7.
- Also, is it over-predicting some of the classes? (Is it falsely identifying 1's as 7, i.e. is it a false-positive for 7 and a false-negative for 1)

PRECISION/RECALL

- For each category we can measure two important metrics, *Precision* and *Recall*
- **Precision**: how many selected were correct.
- **Recall**: how many correct were found? A Classification Report can be outputted from our test data

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predicted))

             precision    recall  f1-score   support

          0       1.00      1.00      1.00        54
          1       0.89      0.94      0.92        35
          2       0.94      0.89      0.91        36

avg / total       0.95      0.95      0.95       125
```

PRECISION/RECALL

	precision	recall	f1-score	support
0	1.00	1.00	1.00	54
1	0.89	0.94	0.92	35
2	0.94	0.89	0.91	36

- The above is showing the performance for 3 categories (0, 1, 2). The support is the number of occurrences of each class in y true.
- Category 1 has precision 0.89. This means that 89% of items predicted to be in Category 1 were correct. The other 11% were incorrectly identified as Category 1 (false-positive for category 1)
- Category 1 has recall 0.94. This means that 94% of the items that should've been predicted in Category 1 were found. The other 6% were incorrectly identified in some other category (false-negative for category 1)

- **Precision** is defined as the number of relevant retrieved instances, divided by the total number of retrieved instances (in that category).

$$Precision = \frac{\#relevant - retrieved - instances}{\#total - retrieved - instances}$$

- **Recall** is defined as the number of relevant retrieved instances, divided by the total number of relevant instances (in that category).

$$Recall = \frac{\#relevant - retrieved - instances}{\#total - relevant - instances}$$

- *sklearn* says:

$$Precision = \frac{tp}{tp + fp} \qquad Recall = \frac{tp}{tp + fn}$$

- where *tp* is number of true positives, *fp* is number of false positives and *fn* is number of false negatives

CANCER EXAMPLE

- Suppose we build a model for cancer diagnosis and we have a test sample of 100 patients. We have the following table describing the performance of our model (this is called a confusion matrix (I will come back to this)):

		Predicted	
		Cancer = Yes	Cancer = No
Actual	Cancer = Yes	True Positive (TP) = 25	False Negative (FN) = 5
	Cancer = No	False Positive(FP) = 5	True Negative(TN) = 65

- We can read precision from the columns
  - Precision for Cancer Yes = 25/(25+5) = 0.83
  - Precision for Cancer No = 65/(65+5) = 0.928
- We can read recall from the rows
  - Recall for Cancer Yes = 25/(25+5) = 0.83
  - Recall for Cancer No = 65/(65+5) = 0.928

ACCURACY REVISITED

- **Note**: Accuracy works quite well when the class distribution is similar but very often our data is imbalanced (cancer analysis, we hopefully have many more patients without cancer than with - imbalanced set).

$$Accuracy = \frac{tp + tn}{tp + fp + tn + fn}$$

- Accuracy is used when the True Positives and True negatives are more important
- The accuracy in this case is = 90% which is a high enough number for the model to be considered as 'accurate'.
  - However, there are 5 patients who actually have cancer, and the model predicted that they don't have it. Obviously, this is too high a cost. Our model should try to minimize these False Negatives.

F1-SCORE

- F1-score is a better metric when there are imbalanced classes.
- F1-score is calculated as the harmonic mean of Precision and Recall.
- F1-Score gives a better measure of the incorrectly classified cases than Accuracy.
- IF we let P and R stand for precision and recall respectively then

$$\mathbf{F1} = 2\frac{P.R}{P+R}$$

	precision	recall	f1-score	support
0	1.00	1.00	1.00	54
1	0.89	0.94	0.92	35
2	0.94	0.89	0.91	36

EXTREME EXAMPLE

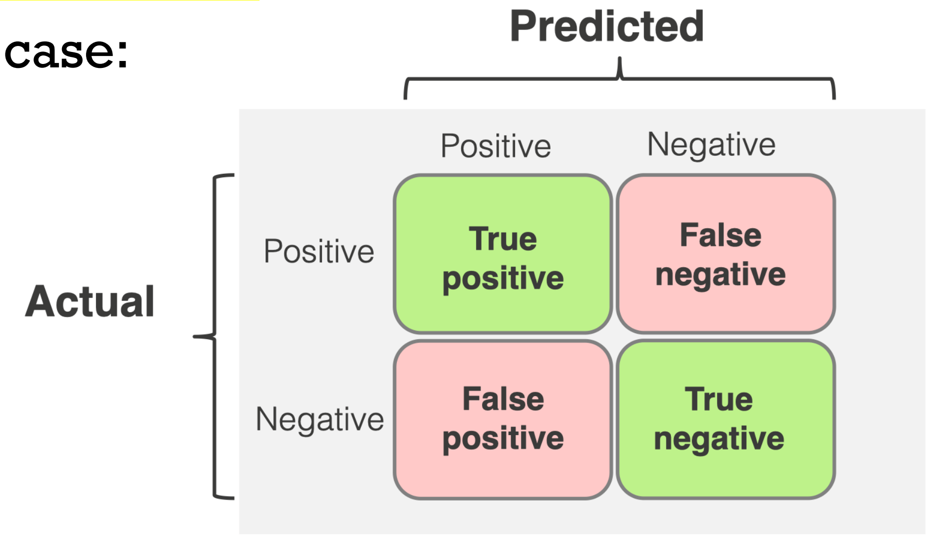
- Of all women who receive regular mammograms, about 10 percent will get called back for further testing and of those, only about 0.5 percent will be found to have cancer.
  - For the women that are called back, 99.5% of them will not have cancer.
  - So, if my model is:

```
def has_cancer(X):
  return false
```

- Then my model will be 99.5% accurate for these women! But it misses all the ones that have cancer.

CONFUSION MATRIX

- Another way of evaluating the accuracy of a classification is to compute a confusion matrix.
- By definition, a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i but predicted to be in group j.
- The diagonals are therefore the "correct" predictions, while the off diagonals are where something has been mis-categorised
- 2-class problems are a particular case:



CONFUSION MATRIX

- As an example in sklearn:

```
from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

- We can use confusion matrices to see where exactly our misclassifications are happening and if necessary, tune our ML algorithm. Precision and Recall are read from a confusion matrix.

OPTIMISING TO THESE SCORES?

- We've seen how we want to get the best metrics: f1-score, accuracy etc.
  - Metrics on a dataset is what we care about (performance)
  - However, we often cannot directly optimize for the metrics
- Our loss/cost function should reflect the problem we are solving and the training technique.
  - We then hope it will yield models that will do well on our dataset

TRADE-OFFS

- Precision and recall usually trade off against each other, meaning that improving one metric reduces the other.
- E.g. Varying the threshold in Logistic Regression will impact the balance between precision and recall