ID2214 Programming for Data Science – Evaluating Predictive Models

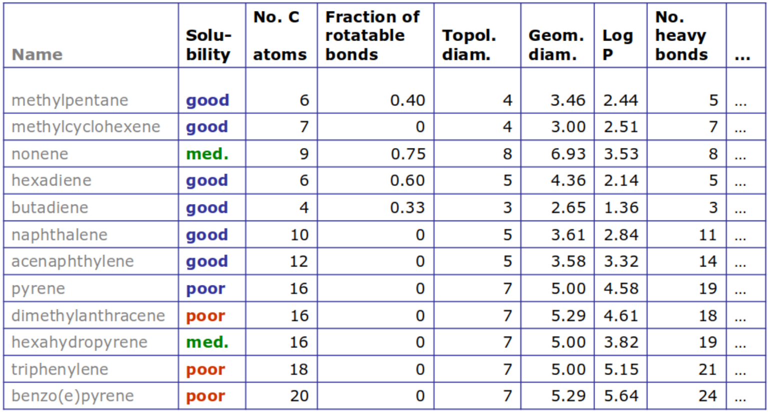
Henrik Boström

Prof. of Computer Science - Data Science Systems
Dept. of Software and Computer Systems
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
bostromh@kth.se

November 7, 2018

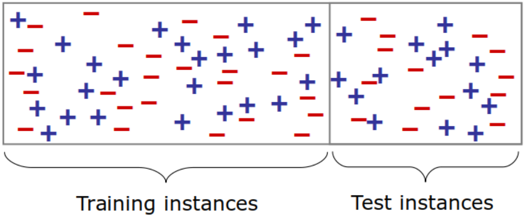
Outline

Predictive Modeling



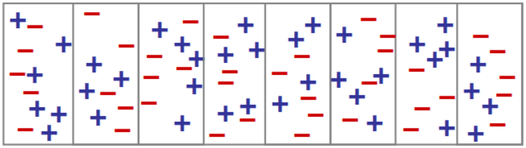
Name	Solu-bility	No. C atoms	Fraction of rotatable bonds	Topol. diam.	Geom. diam.	Log P	No. heavy bonds	...
methylpentane	good	6	0.40	4	3.46	2.44	5	...
methylcyclohexene	good	7	0	4	3.00	2.51	7	...
nonene	med.	9	0.75	8	6.93	3.53	8	...
hexadiene	good	6	0.60	5	4.36	2.14	5	...
butadiene	good	4	0.33	3	2.65	1.36	3	...
naphthalene	good	10	0	5	3.61	2.84	11	...
acenaphthylene	good	12	0	5	3.58	3.32	14	...
pyrene	poor	16	0	7	5.00	4.58	19	...
dimethylanthracene	poor	16	0	7	5.29	4.61	18	...
hexahydropyrene	med.	16	0	7	5.00	3.82	19	...
triphenylene	poor	18	0	7	5.00	5.15	21	...
benzo(e)pyrene	poor	20	0	7	5.29	5.64	24	...

Split (Hold-out)



- *Stratified split* = class proportions are approximately the same

N-fold Cross-Validation



- The procedure is called *leave-one-out cross-validation*, if N = no. of instances
- A common choice is N = 10
- Stratification may also be employed here; *stratified cross-validation*

Classification

		Predicted class	
		+	-
Actual class	+	true positive (tp)	false negative (fn)
	-	false positive (fp)	true negative (tn)

Performance Metrics for Classification

- Accuracy; fraction of correct predictions

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

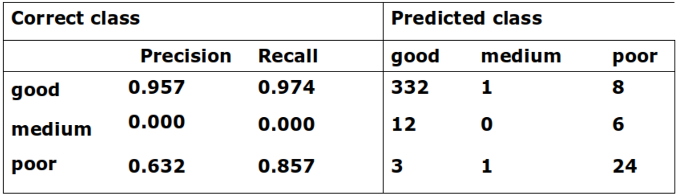
- Precision; fraction of correct predictions for a class

$$Precision = \frac{tp}{tp + fp}$$

- Recall; fraction of certain class correctly predicted

$$Recall = \frac{tp}{tp + fn}$$

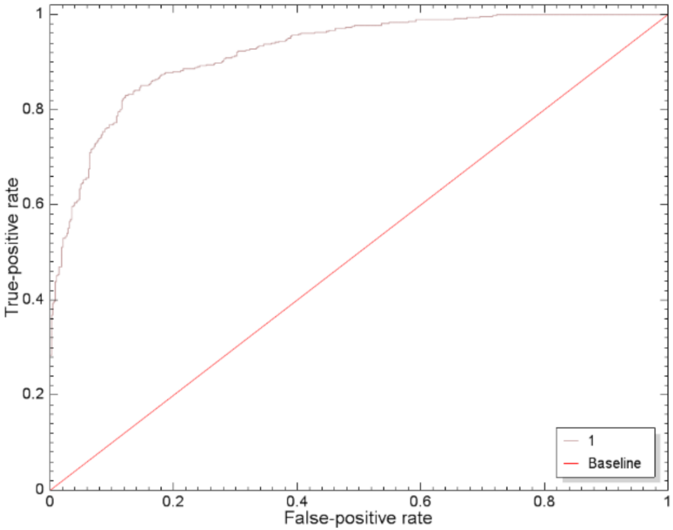
Performance Metrics for Classification (cont.)



Correct class			Predicted class		
	Precision	Recall	good	medium	poor
good	0.957	0.974	332	1	8
medium	0.000	0.000	12	0	6
poor	0.632	0.857	3	1	24

$$Accuracy = \frac{332 + 0 + 24}{341 + 18 + 28} \approx 91.99\%$$

Receiver Operating Characteristic (ROC) Curve

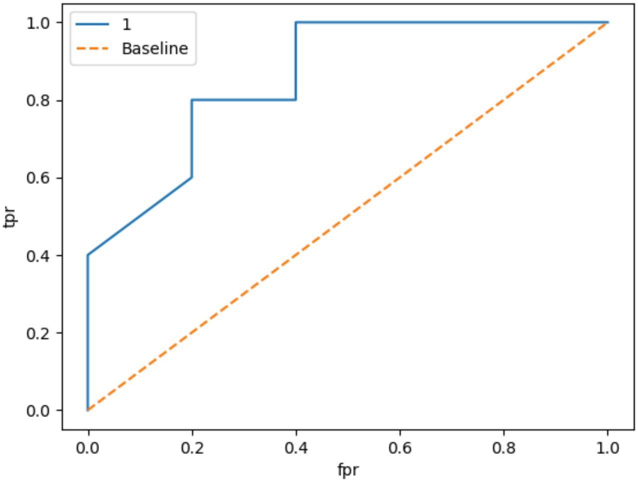


Plotting

- Plotting a ROC curve

```
import matplotlib.pyplot as plt

pos = [1,1,1,1,0,1,0,0]
neg = [0,0,1,0,1,0,2,1]

tpr = [cs/sum(pos) for cs in np.cumsum(pos)]
fpr = [cs/sum(neg) for cs in np.cumsum(neg)]

plt.plot([0.0]+fpr+[1.0],[0.0]+tpr+[1.0],"-",label="1")
plt.plot([0.0,1.0],[0.0,1.0],"--",label="Baseline")
plt.xlabel("fpr")
plt.ylabel("tpr")
plt.legend()
plt.show()                    # To save: plt.savefig("ROC")
```

Plotting (cont.)



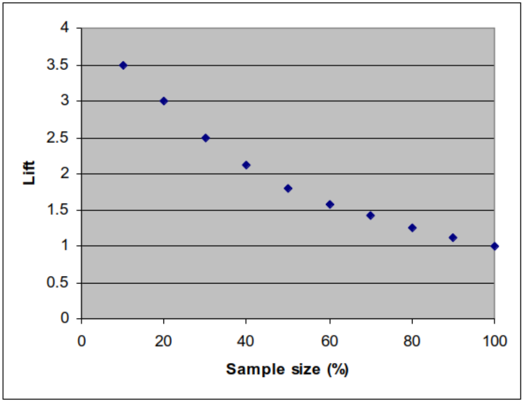
Receiver Operating Characteristic (ROC) Curve (cont.)

- The area under the ROC curve (AUC) $=$ the probability of an example belonging to the class being ranked ahead of an example not belonging to the class
- For binary classification tasks, the AUC will be the same for both classes.
- In case there are more than two classes, the resulting AUC may be calculated as the weighted average of the individual AUCs, using relative class frequencies as the weights.

Calculating the Area under ROC Curve (AUC)

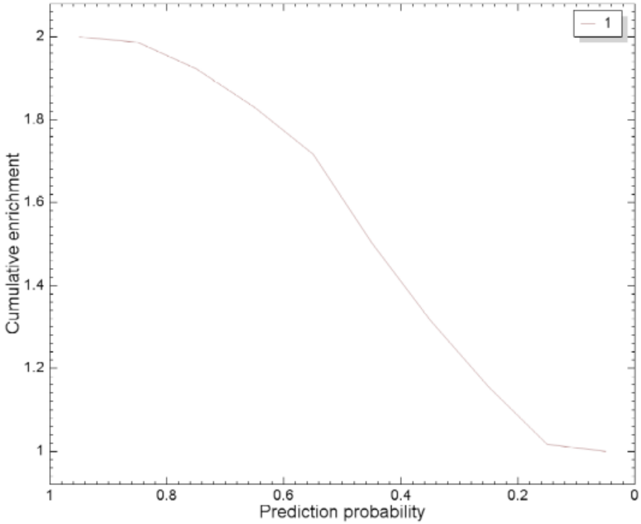
```
Input: (s_1,tp_1,fp_1), ..., (s_n,tp_n,fp_n) (sorted
triples of scores, no. true and false pos. with the
scores wrt some class c), Tot_tp, and Tot_fp
Output: AUC

AUC = 0
Cov_tp = 0
for i = 1 to n
   if fp_i = 0 then Cov_tp += tp_i
   else if tp_i = 0 then
        AUC += (Cov_tp/Tot_tp)*(fp_i/Tot_fp)
   else
        AUC += (Cov_tp/Tot_tp)*(fp_i/Tot_fp)+
               (tp_i/Tot_tp)*(fp_i/Tot_fp)/2
        Cov_tp += tp_i
```

Cumulative Lift Chart



$$Lift = \frac{\frac{tp}{tp+fp}}{\frac{TP}{TP+FP}}$$

Cumulative Lift Chart (alt.)



Evaluating Predicted Probabilities

- Brier score (*quadratic loss*); mean squared error of the predicted probabilities

$$\textit{Brier score} = \frac{1}{n}\sum_{i=1}^{n}(p_i - o_i)^2$$

where p_i are the predicted and o_i the actual (observed) probabilities for test instance i, where typically all values are zero in o_i, except one (corresponding to the true class label)

Evaluating Predicted Probabilities (cont.)

- Log loss (*informational loss*); mean logarithm of the predicted probabilities for the true class labels

  $$Log\ loss = -\frac{1}{n}\sum_{i=1}^{n} o_i\ log\ p_i$$

  where p_i are the predicted and o_i the actual (observed) probabilities for test instance i, where typically all values are zero in o_i, except one (corresponding to the true class label)

Performance Metrics for Regression

- Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(p_i - o_i)^2$$

- Root Mean Squared Error (RMSE)

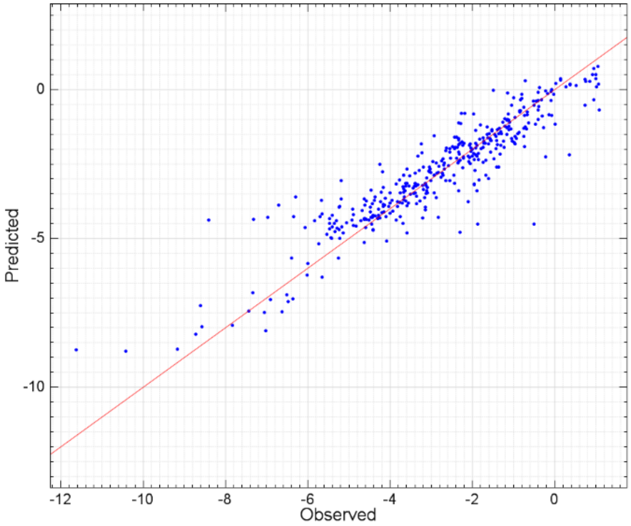
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - o_i)^2}$$

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - o_i|$$

where p_i is the predicted and o_i the actual (observed) target (regression) value for test instance i

Predicted vs. Observed Plot



Performance Metrics for Regression (cont.)

$$\textit{Pearson (product moment) correlation coefficient} =$$

$$\frac{\sum p_i o_i - n\bar{p}\bar{o}}{\sqrt{(\sum p_i^2 - n\bar{p}^2)}\sqrt{(\sum o_i^2 - n\bar{o}^2)}}$$

where p_i is the predicted and o_i the actual (observed) target (regression) value for test instance i, and $\bar{p}$ and $\bar{o}$ are the corresponding averages

Empirical Investigations

What is to be investigated?

- What is the predictive performance of *model* M on new (unseen) data?
- Is there a (significant) difference between *model* M_1 and M_2 (or between M_1, M_2, M_3, $\ldots$)?
- Is there a significant difference between *algorithm* A_1 and A_2 (or between A_1, A_2, A_3, $\ldots$)?

Predictive Performance of a Model

- We assume that we have chosen some performance metric, e.g., accuracy or MSE, for which we can obtain a sample of measurements for the model
- Given the sample, we may infer a *confidence interval*, i.e., by following the procedure we will with high probability obtain an interval that contains the true performance of the model
- Based on the confidence interval, we may perform statistical hypothesis testing, e.g., conclude that the model performance is significantly different from some baseline level, i.e., the baseline falls outside the confidence interval

Predictive Performance of a Model (cont.)

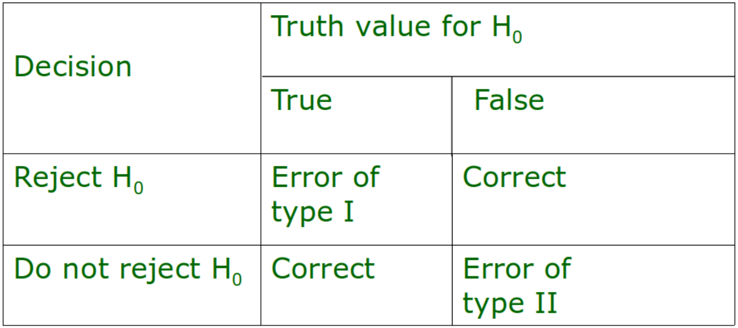
Questions to ask when evaluating the performance:

- Is the sample representative, i.e., randomly sampled from the target distribution?
- Is it independent, e.g., parameter settings, model choice, etc. were not based on the data?
- Is the sample size (N) large enough?
  - For highly skewed distributions, N needs to be higher than 30, to allow for using the normal distribution when inferring confidence intervals
  - For a proportion P, e.g., accuracy, $PN > 10$ and $(P-1)N > 10$
- Is there actually a (single) underlying population from which the sample is drawn?
  - E.g., can the scores obtained from 10-fold cross-validation be considered to be a sample drawn from some population?

Comparing Two Models

- The null hypothesis assumes there is no difference between the models
- A (sufficiently large) random sample is collected, e.g., measured differences in predictive performance
- A significance level is chosen and if measurements of the sample deviate significantly from what can be expected if the null hypothesis is true, then the null hypothesis is rejected, and the alternative hypothesis is accepted

Statistical Errors



Decision	Truth value for H_0	
	True	False
Reject H_0	Error of type I	Correct
Do not reject H_0	Correct	Error of type II

Comparing Multiple Models

- When performing multiple pairwise comparisons, the elevated risk of type I error may be controlled, by *Bonferroni correction*, i.e., divide the significance level by the number of tested hypotheses

Comparing Multiple Algorithms

- When comparing multiple algorithms over multiple datasets, the standard procedure is to employ a *Friedman test*, followed by some suitable *post hoc* test, e.g., Nemenyi, to reject (some of) the pairwise hypotheses, see e.g.,

  Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research, 7, pp.1-30

  Python implementation: https://docs.orange.biolab.si/3/data-mining-library/reference/evaluation.cd.html

  Garcia, S. and Herrera, F., 2008. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research, 9, pp.2677-2694.

  Java implementation: http://sci2s.ugr.es/keel/multipleTest.zip

Summary

- The choice of performance metric is (at least) as important as the choice of learning algorithm; careful consideration of what needs to be optimized is required
- Common traps should be avoided, e.g., dependencies between training and test data, over-fitting the test set by repeated experimentation.
- A careful formulation of one or more null hypotheses are needed for empirical investigations; what exactly are to be compared, etc.