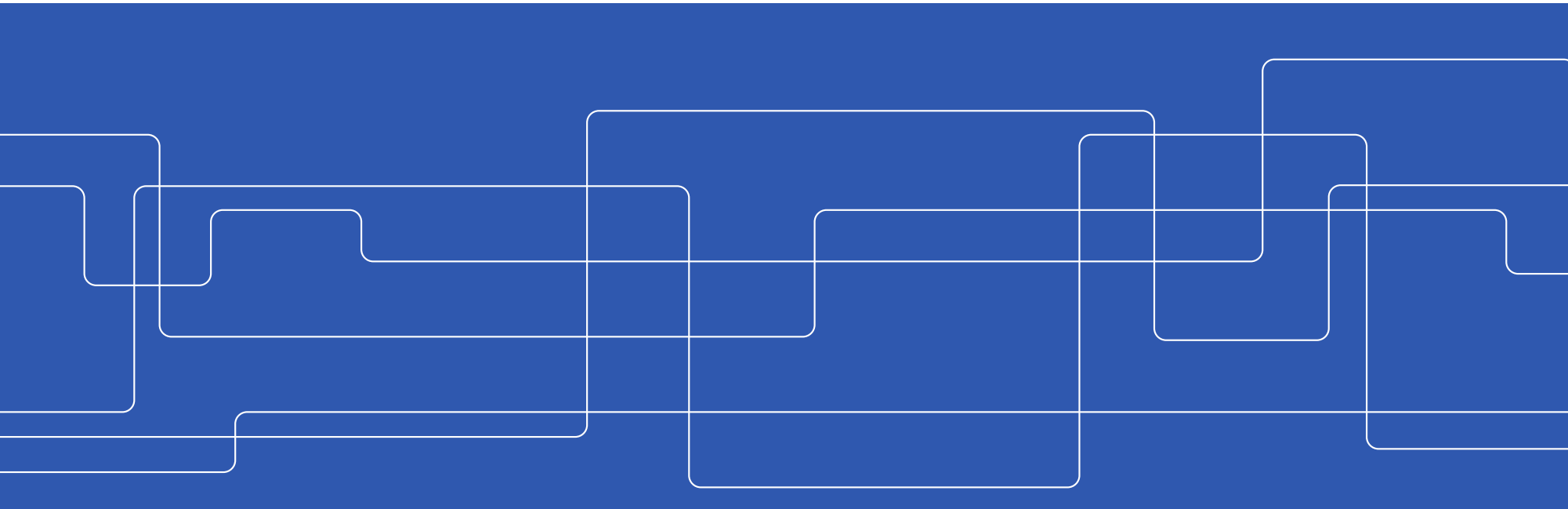
ID2214/FID3214 2020

Programming for Data Science – Evaluating Predictive Models

Henrik Boström

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



Outline

Evaluation protocols

Performance metrics for classification

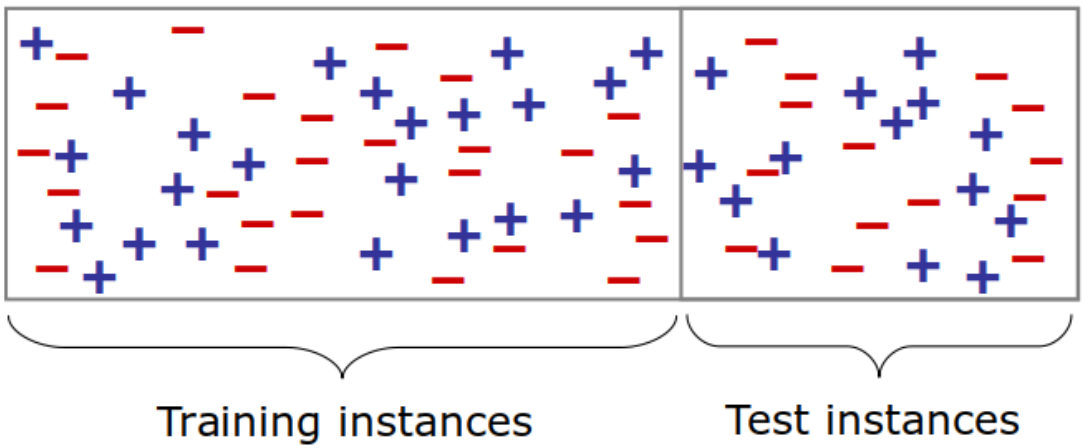
Performance metrics for regression

Example examination questions

Predictive modeling

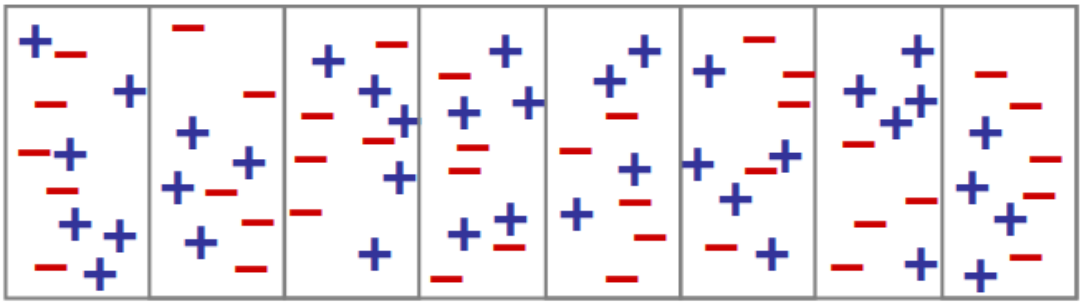
Name	Solu-bility	No. C atoms	Fraction of rotatable bonds	Topol. diam.	Geom. diam.	Log P	No. heavy bonds	...
methylpentane	good	6	0.40	4	3.46	2.44	5	...
methylcyclohexene	good	7	0	4	3.00	2.51	7	...
nonene	med.	9	0.75	8	6.93	3.53	8	...
hexadiene	good	6	0.60	5	4.36	2.14	5	...
butadiene	good	4	0.33	3	2.65	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: +	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

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

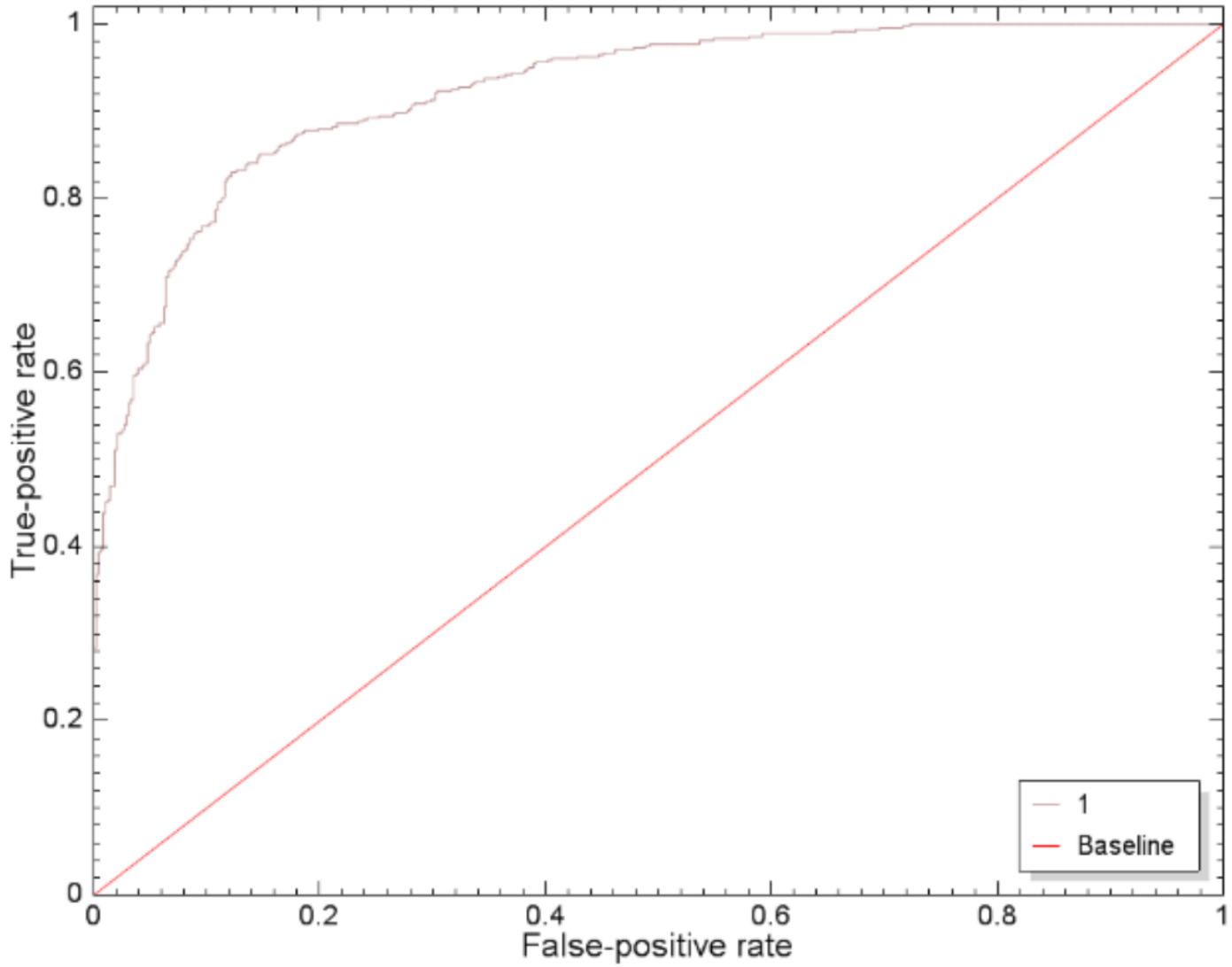
Performance metrics for classification (cont.)

Confusion matrix

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

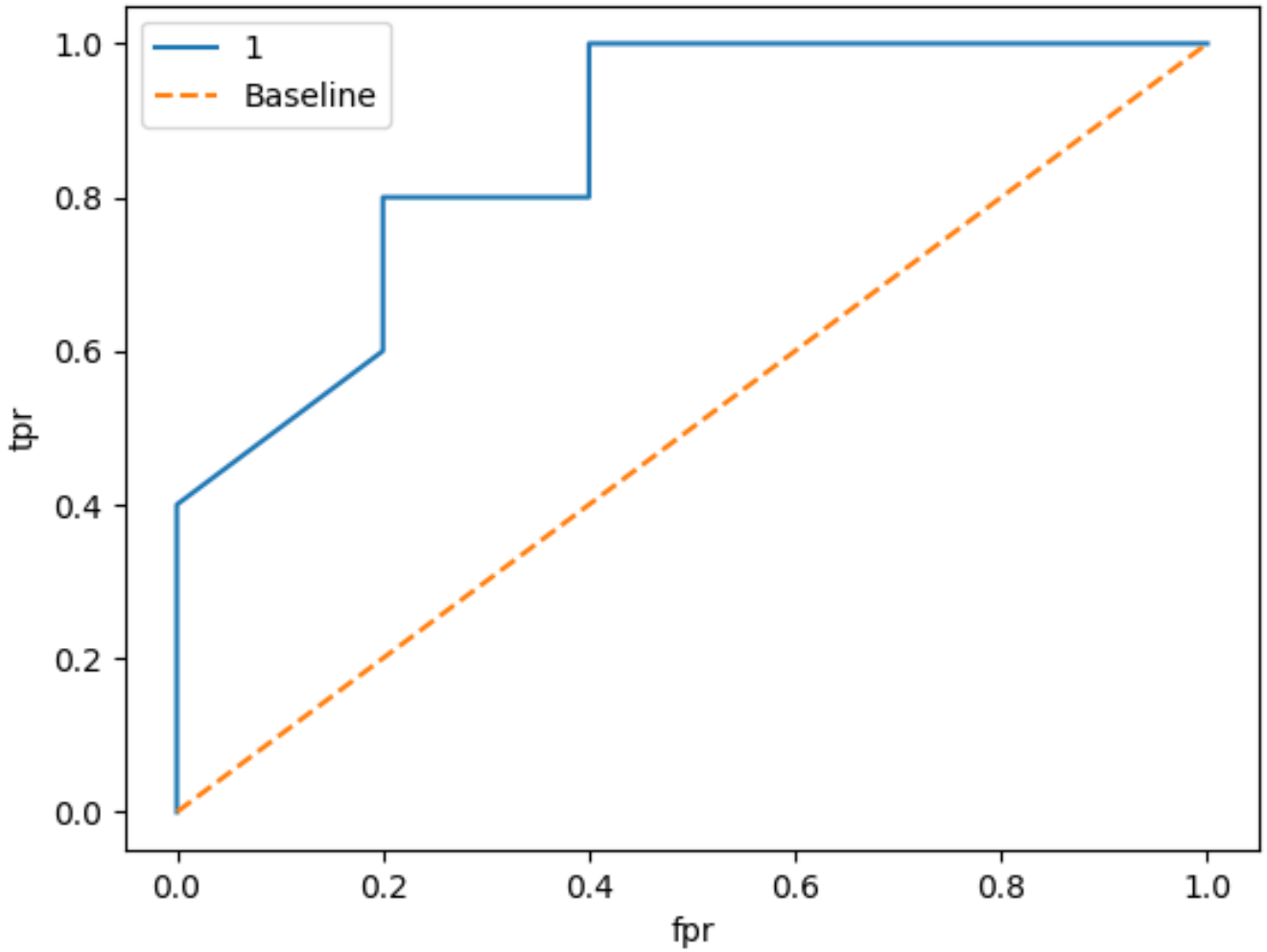
```
import matplotlib.pyplot as plt

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
```

Calculating the AUC (an example)

```
Input:
(0.9,1,0), (0.8,1,0), (0.7,1,1), (0.6,1,0),
(0.5,0,1), (0.4,1,0), (0.3,0,2), (0.2,0,1)

1. Cov_tp = 1
2. Cov_tp = 2
3. AUC = 2/5 * 1/5 + (1/5 * 1/5) / 2 = 0.1
   Cov_tp = 3
4. Cov_tp = 4
5. AUC = 0.1 + 4/5 * 1/5 = 0.26
6. Cov_tp = 5
7. AUC = 0.26 + 5/5 * 2/5 = 0.66
8. AUC = 0.66 + 5/5 * 1/5 = 0.86
```

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)

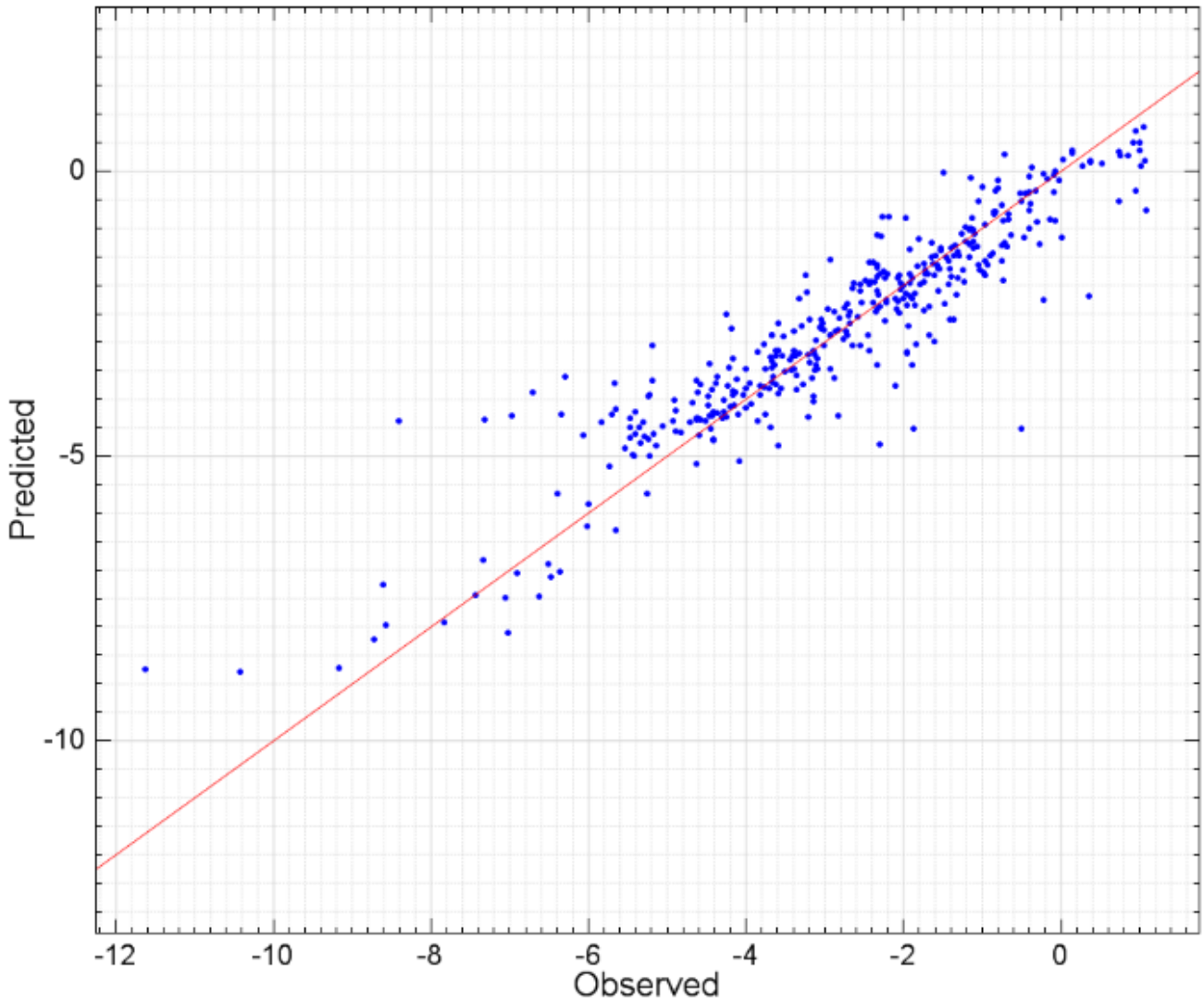
Evaluating predicted probabilities (cont.)

- 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)

Predicted vs. observed plot for regression



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

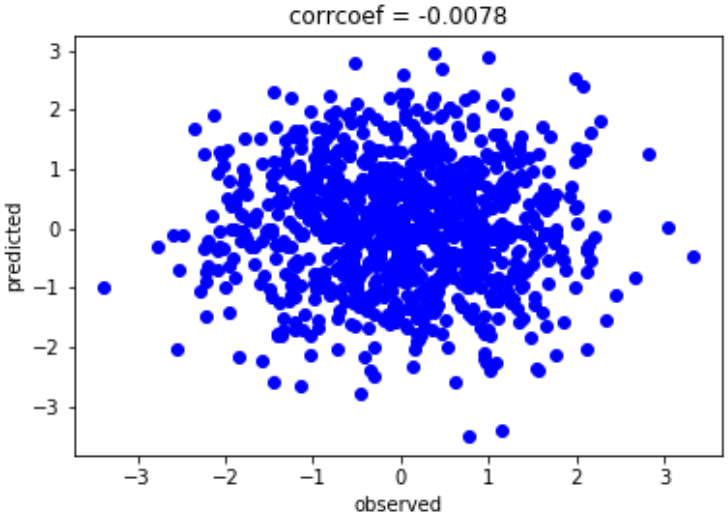
Performance metrics for regression (cont.)

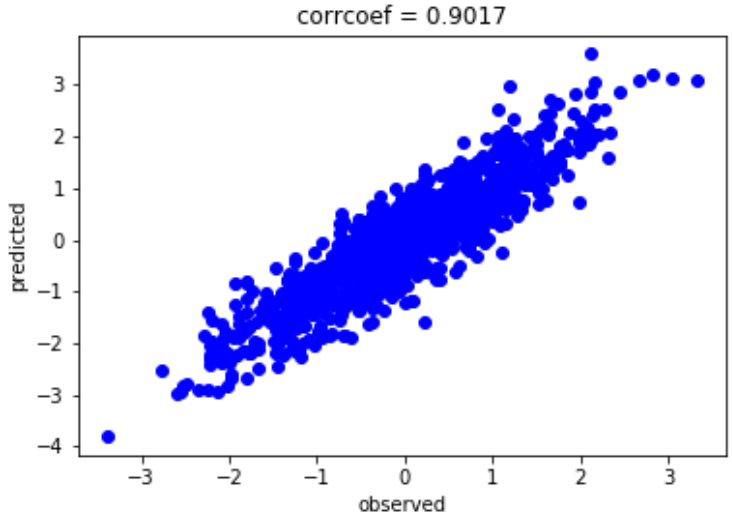
Pearson (product moment) correlation coefficient =

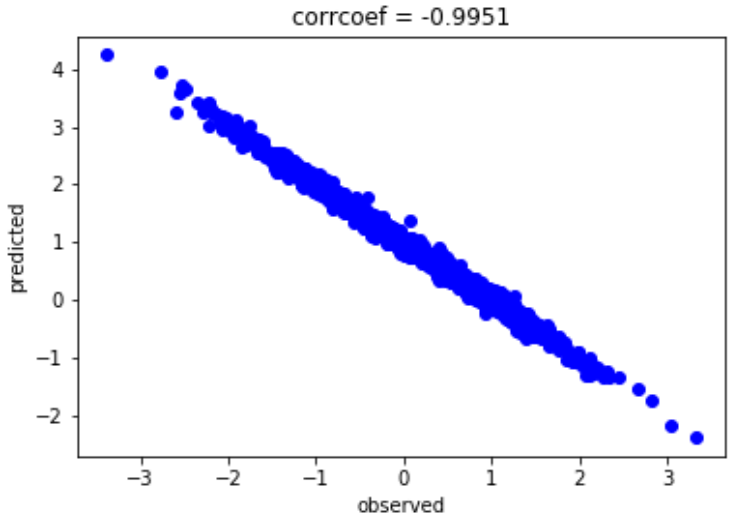
$$\frac{\sum p_i o_i - n\bar{p}\bar{o}}{\sqrt{(\sum p_i^2 - n\bar{p}^2)}\sqrt{(\sum o_i^2 - n\bar{o}^2)}}$$

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

Correlation coefficient







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

Example examination questions

1. Assume that we have two classification models M1 and M2 that are evaluated on five test instances. Show with an example that M1 can have a higher accuracy than M2, while at the same time M2 has a higher area under the ROC curve (AUC) than M1.

2. Assume that we have generated a regression model for predicting the outdoor temperature using a large training set. However, we then find out that the mean-squared error of the model is significantly higher than of a default model, which just predicts the average outdoor temperature in the training set. Could our model still be more useful for prediction tasks than the default model? Explain your reasoning.