Chair of Decision Sciences and Systems
TUM School of Computation, Information and Technology
Technical University of Munich

Business Analytics & Machine Learning
Tutorial sheet 7: Model Evaluation and Selection – Solution

Prof. Dr. Martin Bichler, Prof. Dr. Jalal Etesami
Julius Durmann, Markus Ewert, Johannes Knörr, Yutong Chao

Exercise T7.1 *Metrics*

Assume your Business Analytics team has trained a binary classifier with the following results on unseen test data:

True Class	0	0	1	1	0	1	0	1	0	1
Predicted Class	0	1	1	0	0	0	0	1	1	0

a) State the confusion matrix for the ground truth and predicted labels given in the following table and interpret the results.

b) With these results, calculate recall, false alarm rate, precision, specificity and accuracy.

Solution

a) The confusion matrix, where columns represent what the model predicts and rows what the actual classes are, is given as:

	0	1
0	TN = 3	FP = 2
1	FN = 3	TP = 2

b) You may consult [1] for an overview of different metrics.

Let us first check which predictions are correct:

True Class	0	0	1	1	0	1	0	1	0	1
Predicted Class	0	1	1	0	0	0	0	1	1	0
	TN	FP	TP	FN	TN	FN	TN	TP	FP	FN

1) Recall (aka. true positive rate, sensitivity, hit rate): “How many positive instances have been predicted to be positive?”

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{2}{2+3} = 0.4.$$

2) False alarm rate (aka. false positive rate): “How many negative instances have been predicted to be positive?”

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = \frac{2}{2+3} = 0.4.$$

3) Precision (aka. positive predictive values): “How many positively predicted instances have actually been positive?”

$$\text{PRE} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{2}{2+2} = 0.5.$$

4) Specificity (aka. true negative rate): “How many negative instances have been predicted to be negative?”

$$\text{TNR} = \frac{\text{TN}}{\text{FP} + \text{TN}} = \frac{3}{2+3} = 0.6 \qquad (= 1 - \text{FPR}).$$

5) Accuracy: “How many instances have been predicted correctly?”

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} = \frac{2+3}{2+2+3+3} = 0.5.$$

Exercise T7.2 *Gain Curve, Lift Curve, ROC Curve*

Working for a car rental company, you have trained a classification model that predicts whether a car requires maintenance based on several input features. You have reserved some validation data, on which to you now apply the model and receive the following results:

	True Class	**Probability of Classifier**
Car 1001	-	0.24
Car 1002	-	0.02
Car 1003	+	0.82
Car 1004	-	0.67
Car 1005	+	0.94
Car 1006	-	0.78
Car 1007	-	0.37
Car 1008	+	0.63
Car 1009	+	0.17
Car 1010	+	0.73

Table 1 Your classifier results

Each row represents a car, which either requires maintenance (positive label) or not (negative label). The probabilities indicate the likelihood of being classified as requiring maintenance. The default cutoff value is 0.65.
Draw the

a) Gain Curve

b) Lift Curve

c) ROC Curve

of the classifier. Name the axes and write down the (x,y)-coordinates for every point. Mark the point corresponding to the default cutoff value. Would you argue that this cutoff value is appropriate? Explain your reasoning.

	True Class	Probability of Classifier
Car 501	+	0.78
Car 502	+	0.81
Car 503	-	0.38
Car 504	-	0.27
Car 505	-	0.55
Car 506	-	0.01
Car 507	-	0.15
Car 508	+	0.62
Car 509	+	0.93
Car 510	-	0.49

Table 2 Your colleague's classifier results

d) Your colleague has trained an alternative model on the entire dataset and evaluates the model on a subset thereof:

Your colleague argues:
"*This is the best possible classifier. Its gain curve is above the gain curve of a random classifier, and therefore it clearly dominates a random classifier. Moreover, the lift curve of my classifier will be a constant function at 2, whereas the lift curve of your classfier is monotonically decreasing. We should employ my classifier instead of yours.*"

Explain three reasons why you disagree with your colleague's statement.

Solution

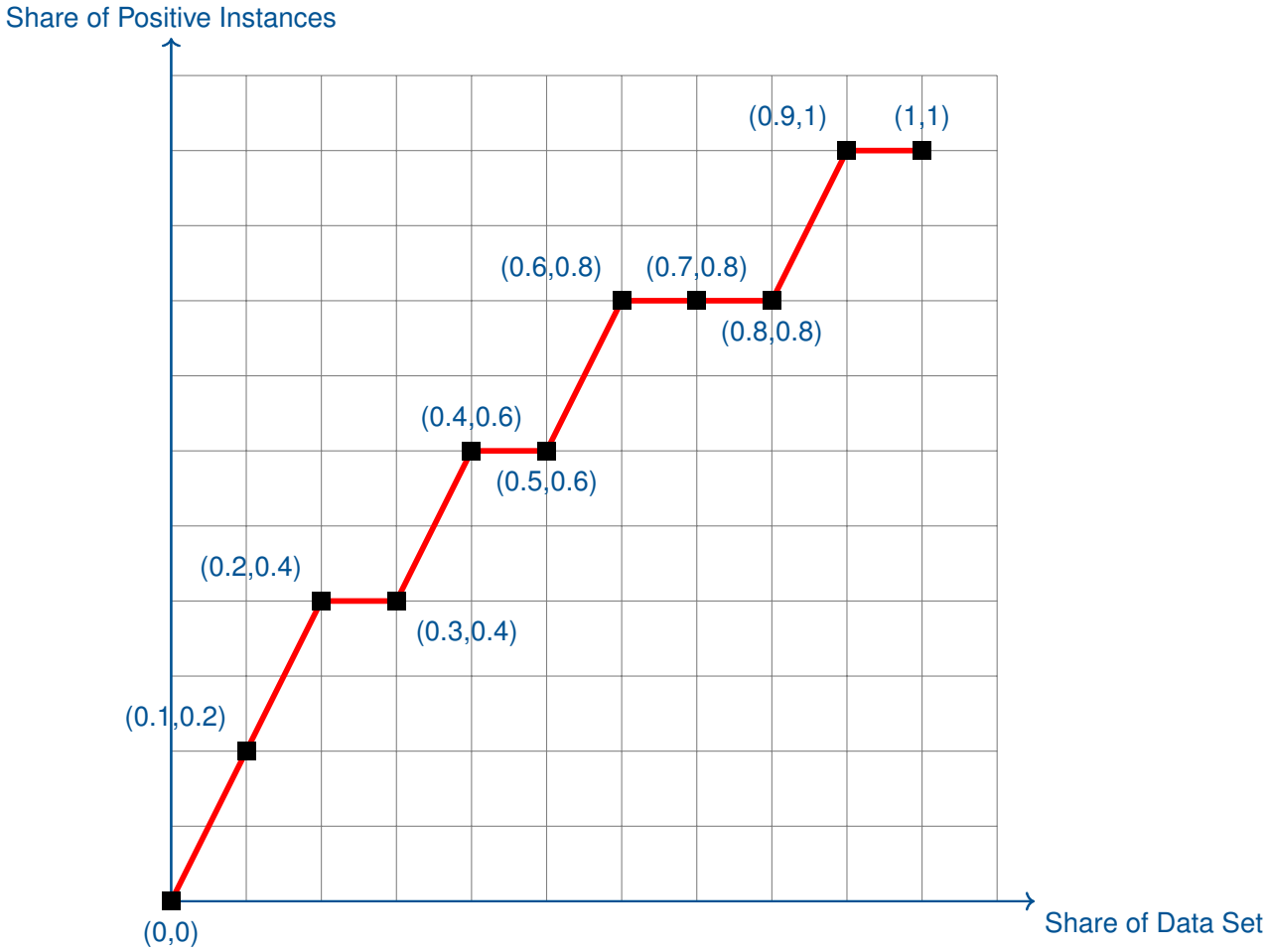

Figure 1 Gain Curve

a) Cutoff corresponds to point (0.5,0.6). It is a poor choice, as a higher cutoff value would capture the same share of positive instances with a smaller share of the data.

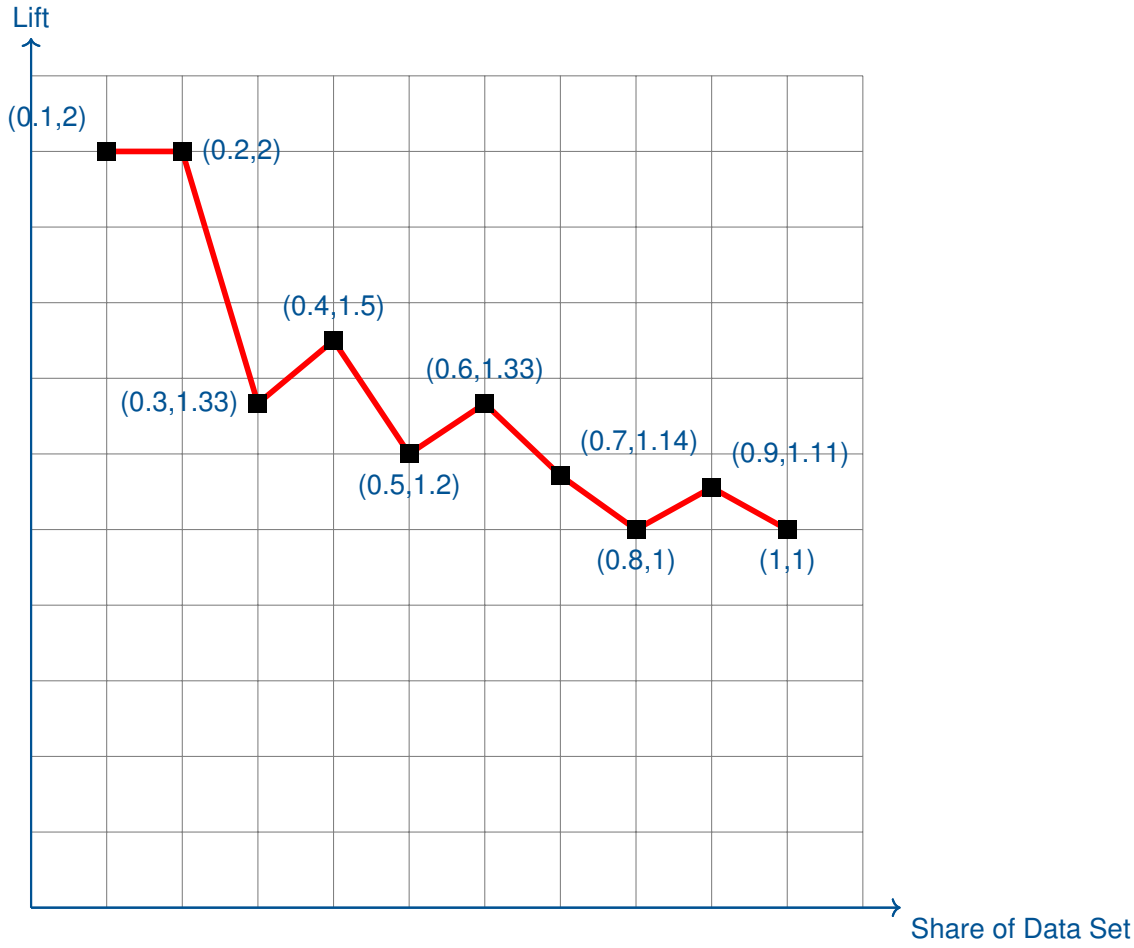

Figure 2 Lift Curve

b) Cutoff corresponds to point (0.5,1.2). It is a poor choice, as it is a local minimum.

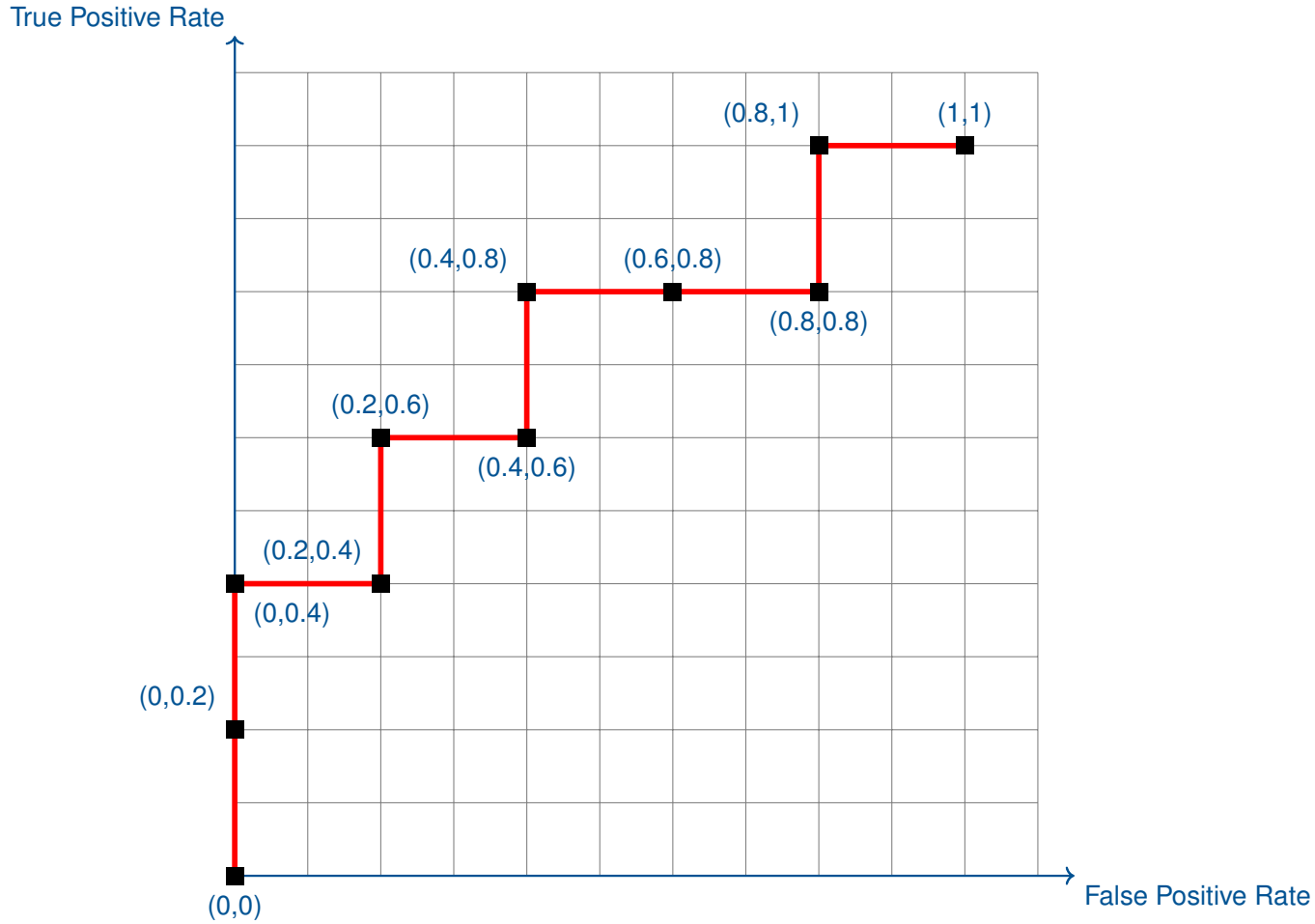

Figure 3 ROC Curve

c) Cutoff corresponds to point (0.4,0.6). It is a poor choice, as a lower cutoff value could increase the True Positive Rate with the same False Positive Rate (or vice versa)

d)
a) The colleague evaluates the model on (a subset of) the training set. This indicates overfitting. The model might perform poorly on unseen data and we should not employ it.

b) The lift curve will not be a constant function at 2. It will start at 2, but then monotonically decrease with the first "-" instance.

c) The lift curve of my classifier is not monotonically decreasing. It is neither monotonically increasing nor decreasing.

Exercise T7.3 *Churn prediction*

Working for a large e-commerce enterprise, you are given a dataset for customer churn prediction. The file `e-commerce-dataset.xlsx` contains a description of all variables in the sheet `Data Dict` as well as the raw data itself in sheet `E_Comm`.

You intend to find a good random forest model to predict future customer churns.

Note: Use the py-script `churn.py`.

You begin with some data preparations. You remove all cases with missing data and factorize all variables where necessary.

a) Your colleague proposes to train the model on the entire dataset and argues to tune the `n_estimators` and `max_features` parameters of `sklearn.ensemble.RandomForestClassifier` until the training accuracy is maximized. Do you agree? If not, which issues can you identify with this approach?

Your colleague has little understanding of the issues that you raised. You decide to illustrate your ideas by means of the first twenty instances in your data $\{1, 2, \ldots, 20\}$.

b) You decide to split your dataset into training and test sets with an 80-20 % split and perform a 4-fold cross-validation. Using the small dataset, design an exemplary split of the data. Show how you would partition the data for the 4-fold cross validation. Explain the purposes of each subset and which operations/actions you perform with each subset.

You now turn back to the original dataset.

c) Perform training, 4-fold cross-validation, and testing with a 60-20-20 % split in Python. Use the precision as metric for model selection. Build a confusion matrix for the test set and report precision, accuracy, and recall.

d) On another dataset, you notice missing values for some numeric attributes in the test set. Your colleague suggests imputing these missing values by the mean of this attribute across all test instances. Do you agree? Explain your reasons.

Solution

a) See `solution.py`. This approach will result in an overfitted model with high variance. This means that the model will probably not be able to generalize to the population. Hence, when our model is given a new dataset which is different from the training data but still from the same population, it will likely perform poorly.

b) An exemplary solution will be:

1) 80-20 train/test-split: Here we simply split the data once into a training set: $\{1, 2, \ldots, 16\}$ and a test set $\{17, 18, 19, 20\}$.
2) 4-fold cross-validation: We partition the remaining training data into $k = 4$ folds, e.g., $\{1, 2, 3, 4\}$, $\{5, 6, 7, 8\}$, $\{9, 10, 11, 12\}$, and $\{13, 14, 15, 16\}$.
3) During each iteration of cross-validation, a learning model (in this case a decision tree) will be fitted on the training data of the corresponding iteration and will be validated on the validation set of the iteration:

   For each iteration, the value of the chosen evaluation metric (accuracy, recall, F1 score, etc.) will be calculated, and the average of those values will be used to select an appropriate model.
4) The test set $\{17, 18, 19, 20\}$ can be thought of as a proxy of unseen data, and it will not be used until the very end of the modeling phase. It will only be used after we are satisfied with our model selection. Applying the model on the test set and calculating the evaluation metric allows for an assessment of the overall goodness of the model.

Iteration	Training set	Validation set
1	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$	$\{13, 14, 15, 16\}$
2	$\{1, 2, 3, 4, 5, 6, 7, 8, 13, 14, 15, 16\}$	$\{9, 10, 11, 12\}$
3	$\{1, 2, 3, 4, 9, 10, 11, 12, 13, 14, 15, 16\}$	$\{5, 6, 7, 8\}$
4	$\{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$	$\{1, 2, 3, 4\}$

c) See `solution.py`.

d) Imputing the missing values with the mean of the dataset is a common approach. However, the imputation should always be based on the mean of the training set, and not on the test set. The test set is a proxy for unseen data and therefore it must not be used for imputation.

This also means that the imputation should be done after the train-test-split, since imputing the missing data from the full dataset will expose information about the test data to the training data, causing the phenomenon referred to as *data leakage*. This will result in overfitting and having an overly optimistic evaluation of the model.

Exercise T7.4 *Leave-one-out cross-validation*

Consider a binary classification task where both classes have an equal probability of 50%. Further, assume you are given a data set reflecting this by consisting of an equal amount of positive and negative instances $\mathcal{D}$ with sample size $|\mathcal{D}| = 64$. You plan to use leave-one-out cross-validation to evaluate a simple majority classifier.

a) What is the true (population) error?

b) What is the the leave-one-out error estimate?

c) Describe the problem that you observe with these results.

Solution

a) The true (population) error is simply given as 50%.

b) Let us now calculate the estimated error. Without loss of generality (w.l.o.g.), assume that we leave out a positive instance in our first fold. Now there is a majority of 32 negative instances (and 31 positive instances) in the remaining training data. Therefore, the majority classifier has an error of 100% on the left out (positive) instance. The same argument can be made for folds with a negative instance left out. Averaging over all 64 folds the error remains at 100%.

c) Consequently, we drastically overestimate the error of the majority classifier. This example showcases a possible failure of leave-one-out cross validation.