5 Basic Practice (part 3 of 3)

Model Performance Assessment

Regression

- Compute $\text{MSE}_{\text{train}} = \frac{1}{N} \sum_{i=1}^{N} (f_{w,b}(x) - y_i)^2$ on training data and ___________.
- ___________ occurred if $\text{MSE}_{\text{train}} \ll \text{MSE}_{\text{test}}$.
- A useful model outperforms the mean model that predicts $\hat{y} = wx + b = 0x + b = _____$.

Classification

A confusion matrix can help diagnose mistakes:

<table>
<thead>
<tr>
<th>actual $y$</th>
<th>predicted $\hat{y}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td># ______ negative (TN) # false positive (FP)</td>
</tr>
<tr>
<td>1</td>
<td># false negative (FN) # _______ positive (TP)</td>
</tr>
</tbody>
</table>

Several assessment metrics are based on the matrix:

- The two most-frequently used metrics are precision and recall:
  - $\text{precision} = \frac{\#\text{correct positive predictions}}{\#\text{positive predictions}} = \frac{TP}{TP + FP}$, the ability _____ to label as positive a sample that is negative
  - $\text{recall} = \frac{\#\text{correct positive predictions}}{\#\text{positive examples}} = \frac{TP}{TP + FN}$, the ability to _____ all positive examples

  e.g. In document retrieval with “relevant” = positive = 1:
  - ___________ is proportion of relevant documents in the returned list.
  - _______ is proportion of relevant documents returned to relevant documents available.

  e.g. In spam (= positive = 1) detection, we want high ___________ to avoid calling a message spam when it is not (FP). We accept lower ___________ putting some spam in our inbox (FN).

  We usually must choose between precision and recall, e.g.:
  - Assign higher weight to examples of a specific class.
  - Tune hyperparameters to maximize one.
  - Vary decision threshold, e.g. make a positive prediction only if model probability is higher than a number larger than 0.5.

- **Accuracy**
  - $\text{accuracy} = \frac{\#\text{predictions}}{\#\text{predictions}} = \frac{TP + TN}{TN + FN + FP + TP}$

  Accuracy is useful when errors in both (or all) classes are equally important.

---

1 Make a similar matrix for multiclass classification. For the metrics, call one class positive and others negative.

2 Cost-sensitive accuracy assigns a positive cost to each mistake, FN and FP, and multiplies in those costs when calculating accuracy.
• **Area Under ROC Curve (AUC)**

An ROC curve assesses a classifier that returns a probability along with its prediction.

- **True positive rate** = TPR = \( \frac{\text{#true positive predictions}}{\text{#positive examples}} \) = proportion of positive examples predicted correctly
- **False positive rate** = FPR = \( \frac{\text{#false positive predictions}}{\text{#negative examples}} \) = proportion of negative examples predicted correctly

To draw ROC curve,

- use each of several values \( t \in [0,1] \) (e.g. from a model) as a prediction probability threshold \( t \), predict labels, and find TPR and FPR. Note:
  * \( t = 0 \Rightarrow \hat{y} = 1 \) for every \( x \), so \((1, 1)\) is on each ROC curve
  * \( t = 1 \Rightarrow \hat{y} = 0 \) for every \( x \), so \((0, 0)\) is on each ROC curve
- plot resulting \( \{(x=\text{FPR}, y=\text{TPR})\} \) pairs

The higher the **area under the ROC curve (AUC)**, the better:

- The diagonal line \( \leftrightarrow \) corresponds to random guessing and has AUC = \( \) because in \( N \) trials with \( P(y = 1) = p \), guessing with \( P(\hat{y} = 1) = c \), we expect this matrix, TPR and FPR:

<table>
<thead>
<tr>
<th>actual ( y )</th>
<th>predicted ( \hat{y} )</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TN = ( )</td>
<td>FPR = ( )</td>
</tr>
<tr>
<td>1</td>
<td>FN = ( )</td>
<td>TPR = ( )</td>
</tr>
</tbody>
</table>

So \( \) is on the ROC curve for each \( c \in [0,1] \).

- AUC < 0.5 (\( \) than guessing) indicates a problem.
- AUC = 1 corresponds to a \( \) classifier, as it allows TPR=1 with FPR=0.

Select a threshold that gives TPR near 1 with FPR near 0.

**Python**

```python
from sklearn.metrics import (confusion_matrix, precision_score, recall_score, accuracy_score, roc_auc_score, roc_curve, RocCurveDisplay)

confusion_matrix(y_true, y_pred)
```

---

3 "ROC" refers to “receiver operating characteristic” from radar engineers detecting enemy objects in battlefields. Two radar receiver operating characteristics are TPR and FPR.

4 e.g. logistic regression, decision tree, kNN

5 Plotting \( (x=\text{FPR}, y=\text{TPR}) \) is like plotting a hypothesis test’s \( (x = \text{type I error rate} \alpha, y = \text{power} \ 1 - \beta) \).
• `precision_score(y_true, y_pred)`

• `recall_score(y_true, y_pred)`

• `accuracy_score(y_true, y_pred)` (or use `clf.score(X, y)` as before)

• `roc_auc_score(y_true, y_score)` gives AUC if `y_true = \{y_i\}` and `y_score = \{P(y_i = 1)\}

• `roc_curve(y_true, y_score)` gives arrays `(FPR, TPR, thresholds)` with (ignoring `i = 0`):
  
  – `FPR[i]` and `TPR[i]` the false and true positive rates, respectively, of predictions from “`score \geq thresholds[i]`”
  – `thresholds[i]` decreasing thresholds on decision function

• `RocCurveDisplay.from_estimator(estimator, X, y)` plots ROC curve for `estimator` or `RocCurveDisplay.from_predictions(y_true, y_pred)` needs `y_pred = \{P(y_i = 1)\}`

To learn more:

• Reference manual:
  
  > .precision_score.html
  > .recall_score.html
  > .accuracy_score.html
  > .roc_auc_score.html
  > .roc_curve.html
  > .RocCurveDisplay.html

• User guide:
  
  > #precision-recall-f-measure-metrics
  > #accuracy-score
  > #roc-metrics

---

6e.g. `svm.SVC()`, `linear_model.LogisticRegression()`, `DecisionTreeClassifier()`, `kNeighborsClassifier()`
Hyperparameter Tuning

A hyperparameter is a parameter that controls learning but is not set by training. Hyperparameter tuning is experimentally finding good values for hyperparameters:

- **Grid search** requires one or several values for each hyperparameter and tries , retaining the best.
- **Random search** requires each hyperparameter to have a statistical distribution. It tries as many randomly-sampled combinations as allows, retaining the best.

After any of these tuning processes, we might try to tune further with.

Cross-validation

Cross-validation steps are:

- Randomly split data into five folds (subsets) \( \{F_1, \ldots, F_5\} \), each containing examples.
- Train model \( i \) on the four folds excluding \( F_i \), for \( i = 1, \ldots, 5 \).
- Evaluate model \( i \) using \( F_i \) as validation data.
- the five values of your performance metric. This reduces the variability of the metric relative to doing a single train-validate split.

Python

- from sklearn.model_selection import cross_val_score
  - `cross_val_score(estimator, X, y)` uses cross validation to evaluate estimator’s score

- Hyperparameter tuning:
  - from sklearn.model_selection import GridSearchCV
    - `clf = GridSearchCV(estimator, param_grid)` creates a cross-validated grid search classifier using an estimator and dictionary `param_grid` of name: value(s) pairs.
  - from sklearn.model_selection import RandomizedSearchCV
    - `clf = RandomizedSearchCV(estimator, param_distributions, n_iter=10)` creates a cross-validated random search classifier using an estimator and a dictionary `param_distributions` of name: [distribution or value(s)] pairs. `n_iter` is the number of parameter settings sampled.

---

7e.g. SVM: C for regularization, \( \gamma \) for `kernel='rbf'`; logistic regression: C for regularization; ID3 decision tree: \( d = \text{max depth}, \epsilon = \text{min impurity decrease} \); kNN: \( k \), choice of metric
- For both:
  * `clf.fit(X, y)` runs `estimator.fit(X, y)` with all combinations in `param_grid` (for GridSearchCV()) or with `n_iter` combinations from `param_distributions` (for RandomizedSearchCV())
  * `clf.best_score_` gives the mean cross-validated score of the best `estimator`
  * `clf.best_params_` gives the best hyperparameter combination on validation data
  * `clf.cv_results_` gives a dictionary of cross validation results that we can display via `print(pd.DataFrame(clf.cv_results_))`
  * `clf.predict(X)`, `clf.predict_proba(X)`, `clf.score(X, y)` use best combination
  * There is also a `scoring=None` parameter which uses `estimator`'s `.score()` method (accuracy for classification, $R^2$ for regression) by default. We can also set it to:
    - for classification: 'accuracy', 'precision', 'recall', 'roc_auc', others
    - for regression: 'r2' ($R^2$), others

To learn more:

- Reference manual:

- User guide: