5 Basic Practice (part 3 of 3)

Model Performance Assessment

Regression

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

Classification

A *confusion matrix* can help diagnose mistakes:

	predicted $\hat{y}$	
actual y	0	1
0	# _________ negative (TN)	# false positive (FP)
1	# false negative (FN)	# _____________ positive (TP)

Several assessment metrics are based on the matrix:[1]

- The two most-frequently used metrics are precision and recall:

  $precision = \dfrac{\#\text{correct positive predictions}}{\#\text{positive predictions}} = \dfrac{\text{TP}}{\text{TP}+\text{FP}}$, the ability _____ to label as positive a sample that is negative

  $recall = \dfrac{\#\text{correct positive predictions}}{\#\text{positive examples}} = \dfrac{\text{TP}}{\text{TP}+\text{FN}}$, 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

  $accuracy = \dfrac{\#\text{\_\_\_\_\_\_\_\_\_\_\_\_\_ predictions}}{\#\text{predictions}} = \dfrac{\text{TP}+\text{TN}}{\text{TN}+\text{FN}+\text{FP}+\text{TP}}$

  Accuracy is useful when errors in both (or all) classes are equally important.[2]

---

[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[3] curve assesses a classifier that returns a probability along with its prediction.[4]

  - *True positive rate* $= \text{TPR} = \dfrac{\#\text{true positive predictions}}{\#\text{positive examples}} = \dfrac{\text{TP}}{\text{TP}+\text{FN}}$ (= __________) = proportion of positive examples predicted correctly
  - *False positive rate* $= \text{FPR} = \dfrac{\#\text{false positive predictions}}{\#\text{negative examples}} = \dfrac{\text{FP}}{\text{FP}+\text{TN}}$ = proportion of negative examples predicted ____________

  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:
    - $t = 0 \implies \hat{y} = 1$ for every $\mathbf{x}$, so (1, 1) is on each ROC curve
    - $t = 1 \implies \hat{y} = 0$ for every $\mathbf{x}$, so (0, 0) is on each ROC curve
  - plot resulting $\{(x = \text{FPR}, y = \text{TPR})\}$ pairs[5]

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

  - The diagonal line ____________ 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:

	predicted $\hat{y}$		
actual y	0	1	rate
0	TN = ____________	FP = ____________	FPR = ____________
1	FN = ____________	TP = ____________	TPR = ____________

    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

- `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 =FPR, y =TPR) is like plotting a hypothesis test's (x = type I error rate $\hat{\alpha}$, y =power $1 - \hat{\beta}$).

- `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`;[6] or `RocCurveDisplay.from_predictions(y_true, y_pred)` needs `y_pred` $= \{P(y_i = 1)\}$

To learn more:

- Reference manual:

  `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html`

  `.precision_score.html`

  `.recall_score.html`

  `.accuracy_score.html`

  `.roc_auc_score.html`

  `.roc_curve.html`

  `.RocCurveDisplay.html`

- User guide:

  `https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix`

  `#precision-recall-f-measure-metrics`

  `#accuracy-score`

  `#roc-metrics`

  `https://scikit-learn.org/stable/visualizations.html` (plot ROC curve)

[6] e.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.[7]

- *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.

[7] e.g. SVM: C for regularization, γ for `kernel='rbf'`; logistic regression: C for regularization; ID3 decision tree: d = `max_depth`, ϵ = `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:

  https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

  https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

  https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

- User guide:

  https://scikit-learn.org/stable/modules/cross_validation.html

  https://scikit-learn.org/stable/modules/grid_search.html

  https://scikit-learn.org/stable/modules/grid_search.html#randomized-parameter-search

  https://scikit-learn.org/stable/modules/model_evaluation.html (for `scoring`; search for "custom")