Model Evaluation

Justin Silverman

Penn State University

02/04/2021

Table of Contents

Motivation

We have discussed algorithmic approaches to choosing the best model (e.g., held-out sets and cross validation), even ways of tuning hyper-parameters based on which gives a "best" model during model training.

But we have not actually discussed what "best" model means. There are many definitions of "best".

Typically we define "best" as something that maximizes a given model evaluation metric. We are going to introduce a number of evluation metrics in this lecture.

A note on Scikit-Learn

Each estimator has a `score` method which provides a default evaulation metric. In the previous lecture we used the default frequently. But the default is just that, a default choice, not the best choice. This lecture will help you understand the subtleties of different evaluation metrics.

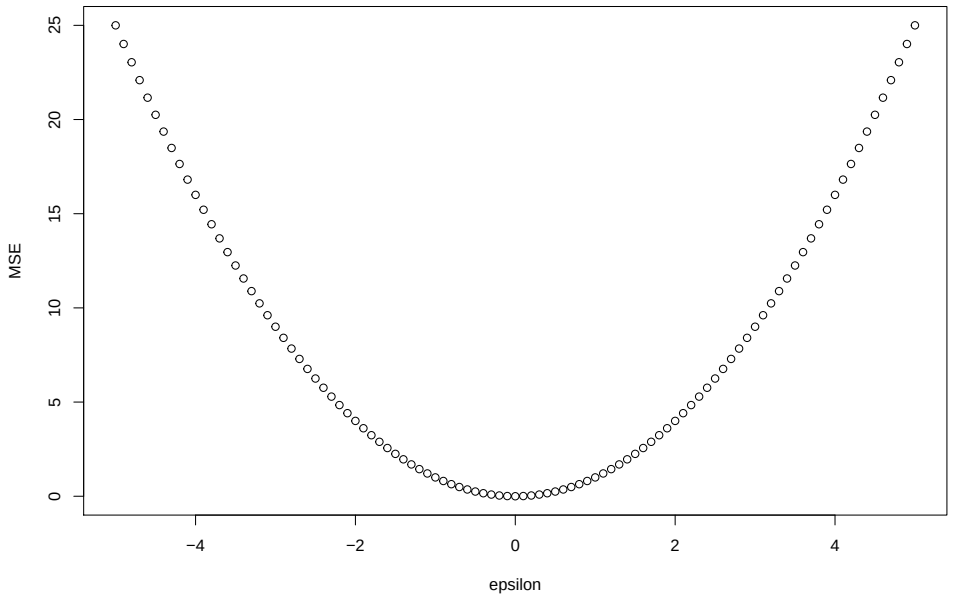
Section 1

Regresion Metrics

Mean Squared Error

$$\text{MSE}(y, \hat{y}) = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2$$

Let $\epsilon_i = y_i - \hat{y}_i$

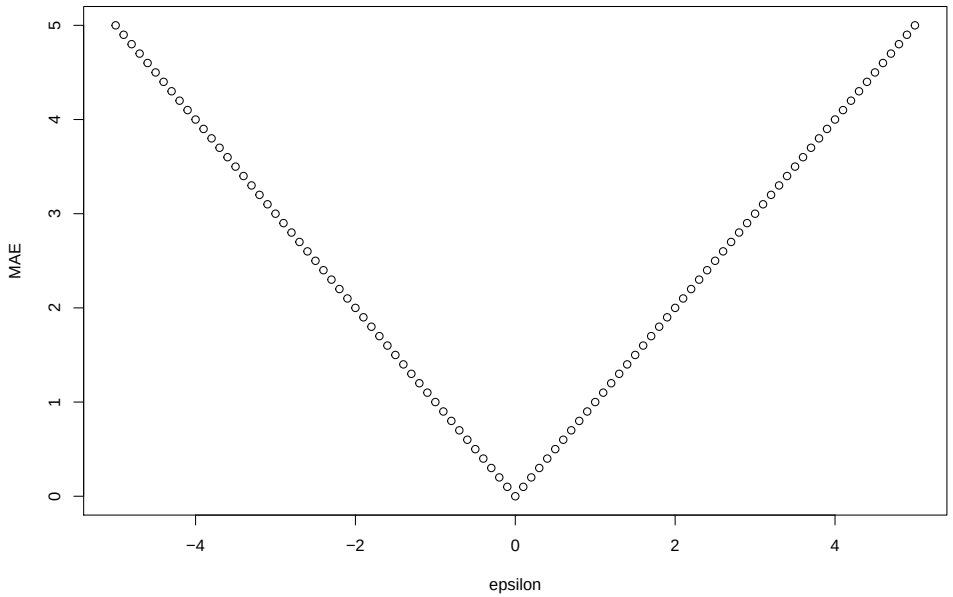



```
from sklearn.metrics import mean_squared_error
```

Mean Absolute Error

$$\text{MAE}(y, \hat{y}) = \frac{1}{n} \sum_i |y_i - \hat{y}_i|$$

Let $\epsilon_i = y_i - \hat{y}_i$




```
from sklearn.metrics import mean_absolute_error
```

Max Error

$$\text{Max Error}(y, \hat{y}) = \max(\{|y_1 - \hat{y}_1|, \ldots, |y_n - \hat{y}_n|\})$$

```
from sklearn.metrics import max_error
y_true = [3, 2, 7, 1]
y_pred = [9, 2, 7, 1]
max_error(y_true, y_pred)
```

```
## 6
```

Median Absolute Error

$$\text{MedAE}(y, \hat{y}) = \text{median}(\{|y_1 - \hat{y}_1|, \ldots, |y_n - \hat{y}_n|\})$$

Let $\epsilon_i = y_i - \hat{y}_i$

The median makes it fairly robust to outliers.

```
from sklearn.metrics import median_absolute_error
```

Goodness-of-Fit Statistics

Various measure of how well a model fits the data. These are more based in statistical inference and are less frequently used in machine learning.

- R^2
- Likelihood Ratio
- Akaike Information Criteria (AIC)
- Bayesian Information Criteria (BIC)
- Marginal Likelihood (aka Model Evidence)

Note: AIC, BIC, and Marginal Likelihood all involve a penalization for increasing model parameters that helps avoid over-fitting.

Explained Variance

$$\text{explained variance}(y, \hat{y}) = 1 - \frac{\text{Var}(y - \hat{y})}{\text{Var}(y)}$$

Best possible score is 1, lower values are worse.

```
from sklearn.metrics import explained_variance_score
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
explained_variance_score(y_true, y_pred)
```

R^2

Explained variation is closely related to R^2.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$$

where $\bar{y}$ is the mean of y.

If the mean error is 0 then R^2 and explained variation are identical. Otherwise explained variation adjusts for potential biased error.[1]

```
from sklearn.metrics import r2_score
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
r2_score(y_true, y_pred)
```

[1] Note that $\text{Var}(y - \hat{y}) = \sum_i (\epsilon_i - \bar{\epsilon})^2 / n$

Section 2

Classification Metrics

Classification in Brief

Like regression, Classification typically some involves predicting some outcome y based on some covariates x. However, unlike regression where y is typically continuous; in classification y typically takes on one of a set of k unordered labels.

Binary Classification $k = 2$ – *e.g.*, $y \in \{sick, healthy\}$, $y \in \{on, off\}$, $y \in \{yes, no\}$

Multiclass Classification $k > 2$. – *e.g.*, $y \in \{dog, cat, horse, car\}$

Multilabel Classification $k \geq 2$. – could be binary or multiclass, each observation i has more than one label. *e.g.*, "that is a picture of a horse, a dog, and a car".

Classification Compared to Regression

In regression we are often concerned with how close a model got to the true answer– not whether it got the exact right answer which is typically impossible. (e.g., often predicting 2.00001 when the true value is 2 is pretty good)

In classification, since we are dealing with discrete classes, it makes sense to ask whether a model recovered the exact right labels. (e.g., if the picture was of a horse but the model thought it was a car, that is not so good)

In Short, in Classification we often deal with how often the model was exactly right, not just if it got close.

Confusion Matrix for Binay Classification and Key Evaluation Metrics Defined

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$		

Figure 1: Image from Wikipedia

Confusion Matrix for Multiclass Classification

```
from sklearn.metrics import confusion_matrix
y_true = [2, 0, 2, 2, 0, 1]
y_pred = [0, 0, 2, 2, 0, 2]
confusion_matrix(y_true, y_pred)
```

```
## array([[2, 0, 0],
##        [0, 0, 1],
##        [1, 0, 2]])
```

Accuracy

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} I(\hat{y}_i = y_i)$$

```
import numpy as np
from sklearn.metrics import accuracy_score
y_pred = [0, 2, 1, 3]
y_true = [0, 1, 2, 3]
accuracy_score(y_true, y_pred)
accuracy_score(y_true, y_pred, normalize=False)
```

Problems with Accuracy

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} I(\hat{y}_i = y_i)$$

What if you have two labels (e.g., disease and healthy) but one label is far more prevalent than the other? For example, classifying whether an individual has some rare disease.

You could have very good accuracy by just always predicting that a person is healthy (yet this model would likely be useless).

Simple Key Metrics

Sensitivity (aka Recall) TP/(TP + FN), How many of the positives did we correctly classify?

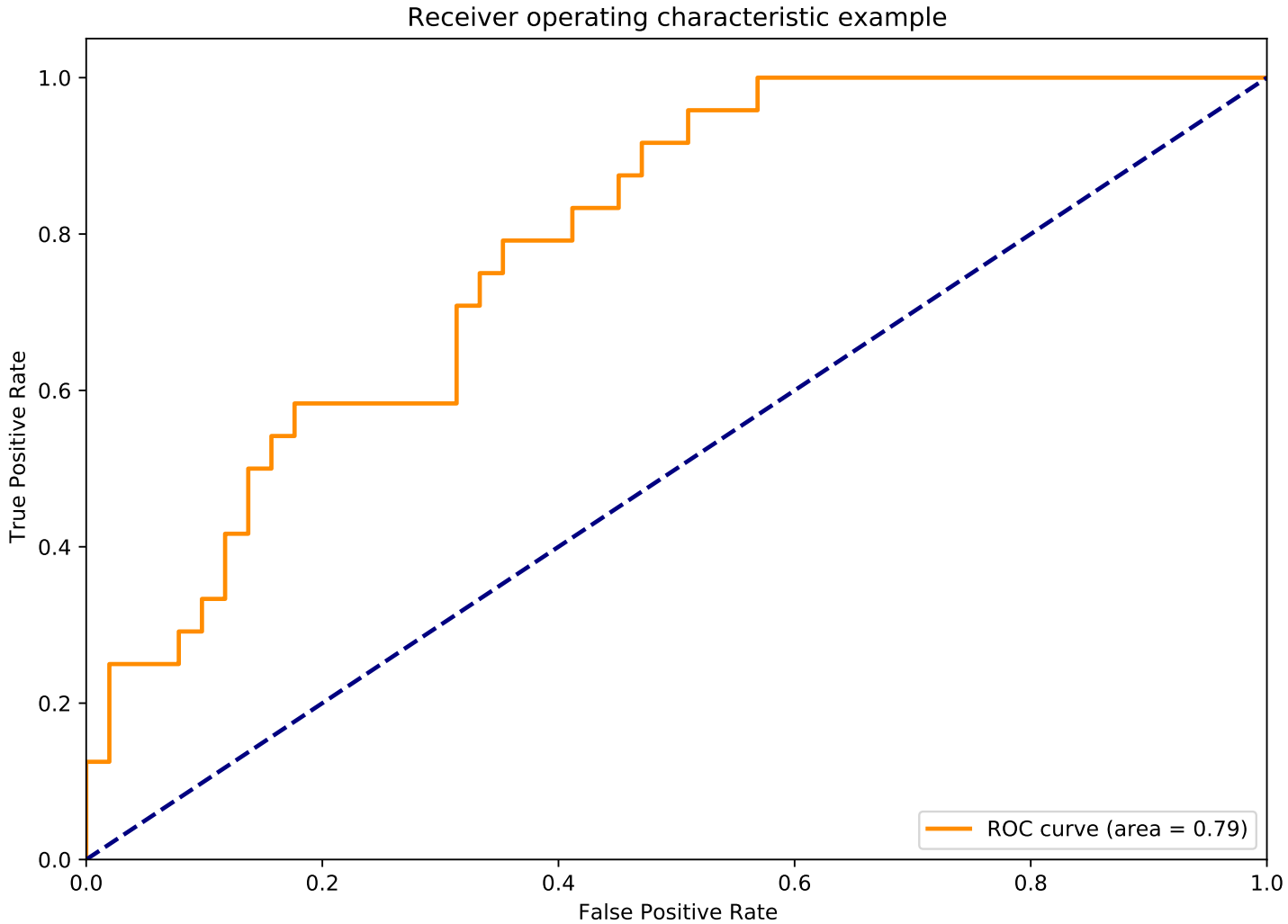
Specificity TN/(TN+FP), How many of the negatives did we correctly classify?

Positive Predictive Value (aka Precision) TP/(TP+FP), How many of the positives that we classified were correct?

Negative Predictive Value TN/(TN+FN), How many of the negatives that we classified were correct?

For some reason machine learning researchers prefer the precision/recall terminology.

Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC)



F_1

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

More can be found at: https://en.wikipedia.org/wiki/F-score

Section 3

Clustering Metrics

Clustering Metrics

This can be difficult. We focus on clustering metrics when we learn clustering later in the semester.

Clustering in Brief

Usually an unsupervised learning problem. – This makes evaluating results difficult.

Like classification, clustering typically involves predicting some label y based on some covariate x. But here we rarely have y truly labeled and instead we are interested in inferring which data points / samples share the same labels.

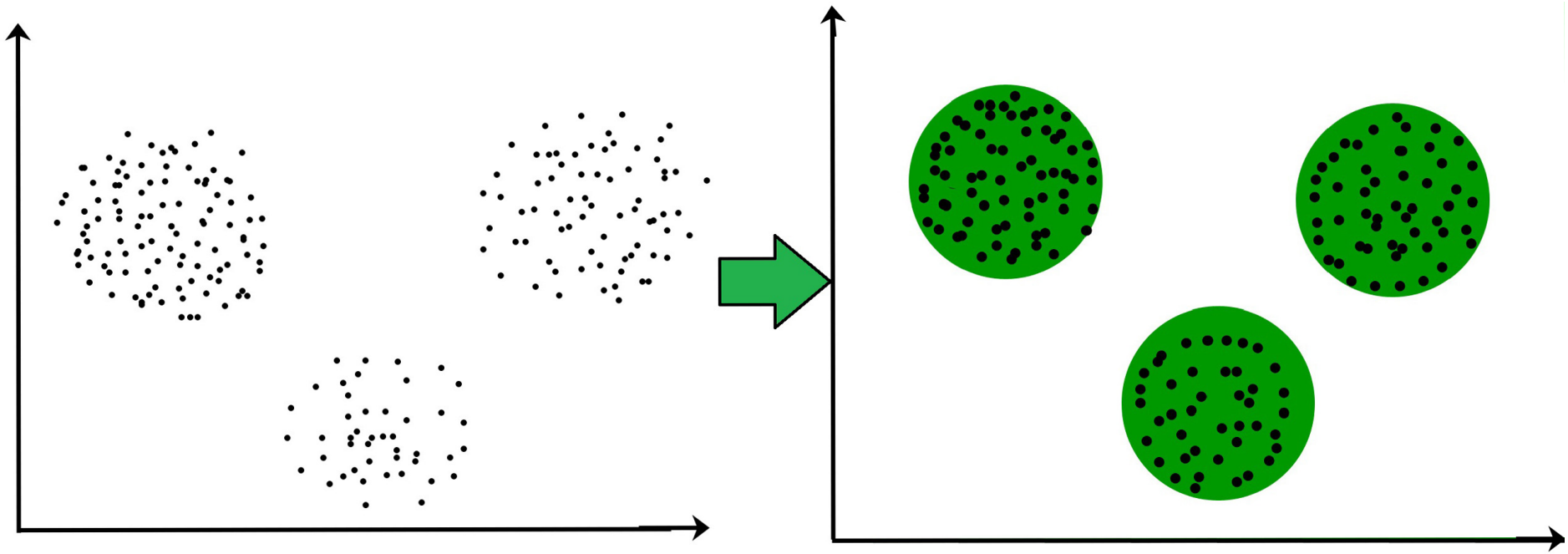


Figure 2: Image from geeksforgeeks.com

List of Clustering Metrics

- Adjusted Rand Index (*)
- Variation of Information (*)
- Mutual Information (*)
- Silhouette Coefficient
- Contingency Matrix (*)

(*) Requires true cluster labels be known

Section 4

Dummy Estimators

Dummy Estimators

These are less evaluation metrics and more sanity checks.

Dummy Estimators for Classification

Does your model beat[2] a simple classification model.

Does your model beat a simple classifier that just predicts whatever label was most frequent in the training data?

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.dummy import DummyClassifier
from sklearn.svm import SVC # support vector classfier

X, y = load_iris(return_X_y=True)
y[y != 1] = -1
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# Fit model we care about
clf = SVC(kernel='linear', C=1).fit(X_train, y_train)
clf.score(X_test, y_test)

# Fit dummy classifier as reference
```

```
## 0.631578947368421
```

```
clf = DummyClassifier(strategy='most_frequent', random_state=0)
clf.fit(X_train, y_train)
```

```
## DummyClassifier(random_state=0, strategy='most_frequent')
```

```
clf.score(X_test, y_test)
```

```
## 0.5789473684210527
```

[2] using one of the previously described evaluation metrics

Dummy Estimators for Regression

- Does your model beat a regression model that always predicts the mean of the training set?

Dummy Estimators for Regression

- Does your model beat a regression model that always predicts the mean of the training set?
- Does your complex regression model beat a simple linear regression model?

Dummy Estimators for Regression

- Does your model beat a regression model that always predicts the mean of the training set?
- Does your complex regression model beat a simple linear regression model?
- Does your model beat an expert who makes manual predictions?