MMA/MMAI 869
Machine Learning and AI

Cross-Validation

Stephen Thomas

Updated: October 29, 2022

Smith SCHOOL OF BUSINESS | Queen's University

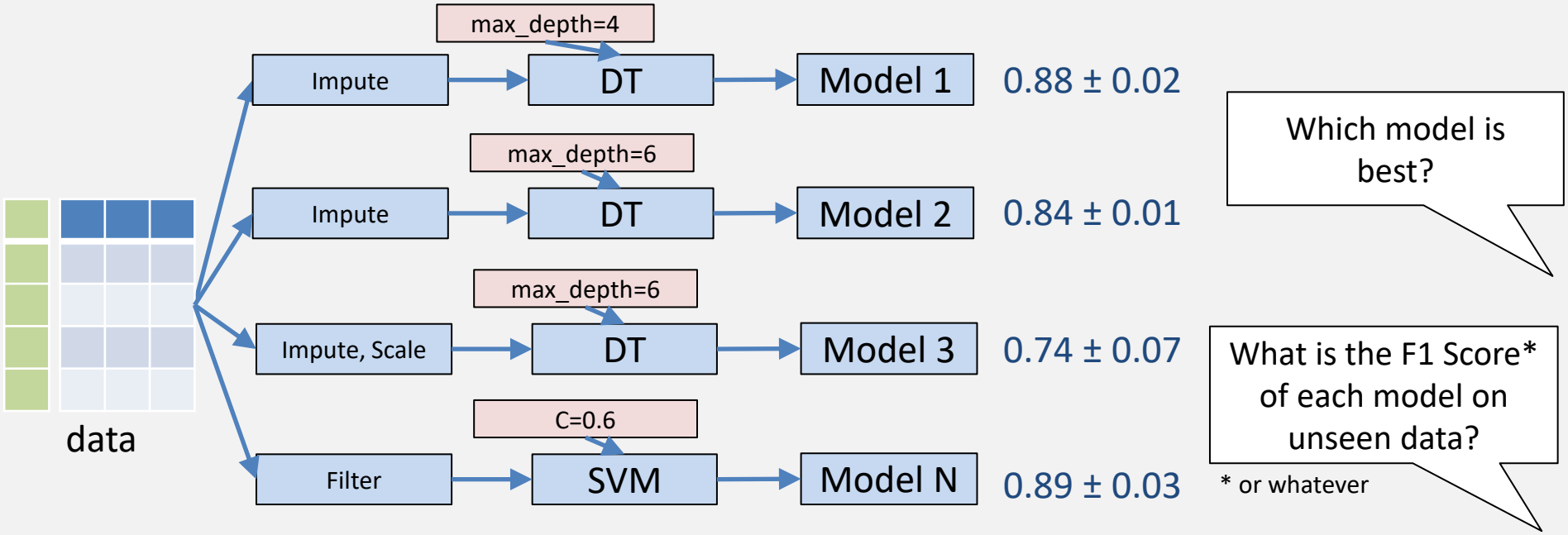
Outline

- How to estimate a model's future performance?
  - How well does it generalize to new, unseen data?
- How to choose the best model (out of many candidates)?

Answer: Cross-Validation

CROSS-VALIDATION

Why Cross-Validation?



- Will have dozens of candidate models
  - Some will overfit
  - Some will underfit
  - Some will be terrible
  - Some will be great
- CV gives an accurate/robust estimate of F1 score on unseen data
  - *"A resampling technique for estimating model performance"*

Other Model Validation Methods

- K-fold Cross-Validation ← Recommended
- Holdout Method
- Repeated Holdout / Shuffle Split
- Repeated K-fold Cross-Validation
- Leave One Out
- Generalized Cross-Validation
- Bootstrapping

Also good

- … and variants for Time Series, Group, and other non-i.i.d. data

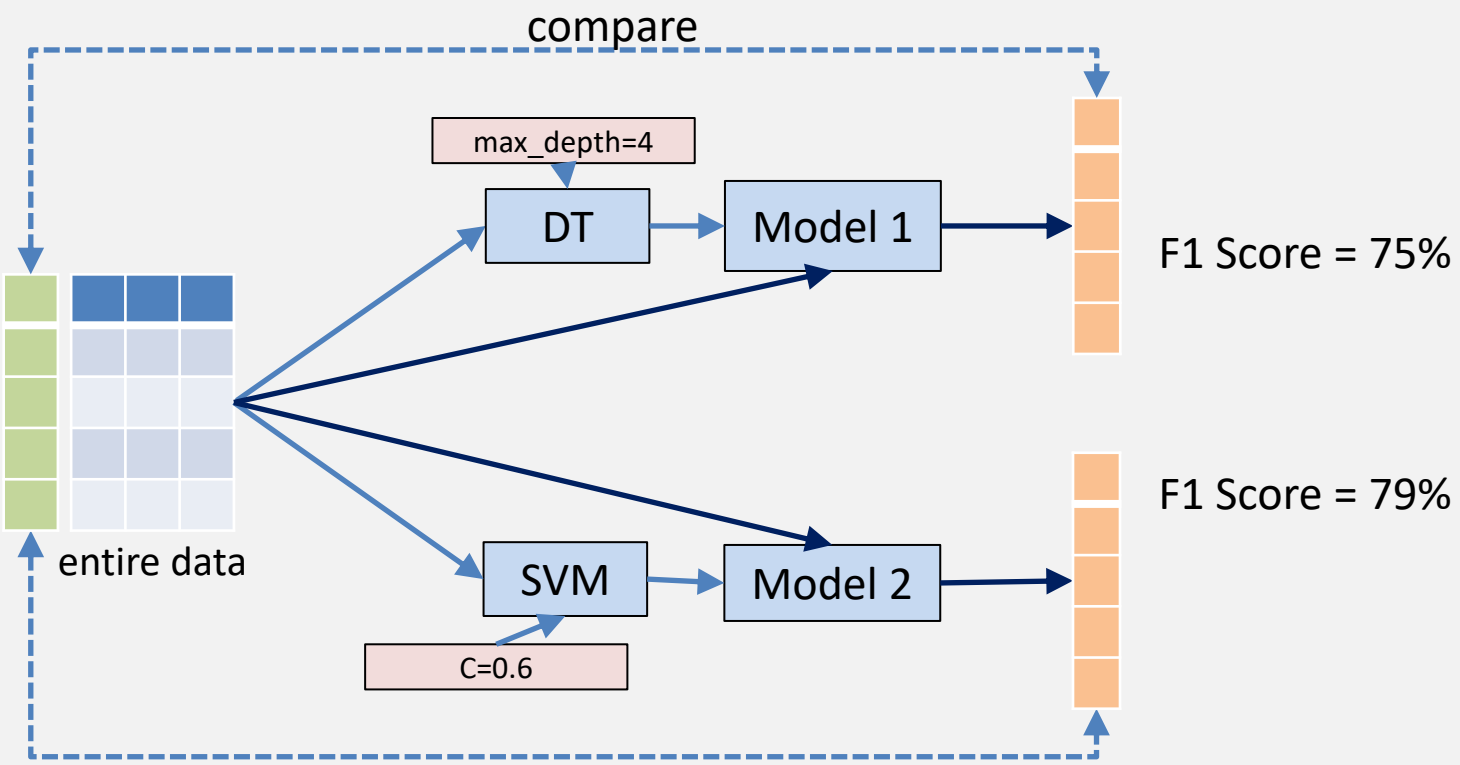
Cross-Validation is a Test

- Military wants to find the strongest individuals

- Cross-validation is like a test of strength
  - CV won't make the individuals stronger
  - CV will tell the military which individuals are stronger

Why Resample?

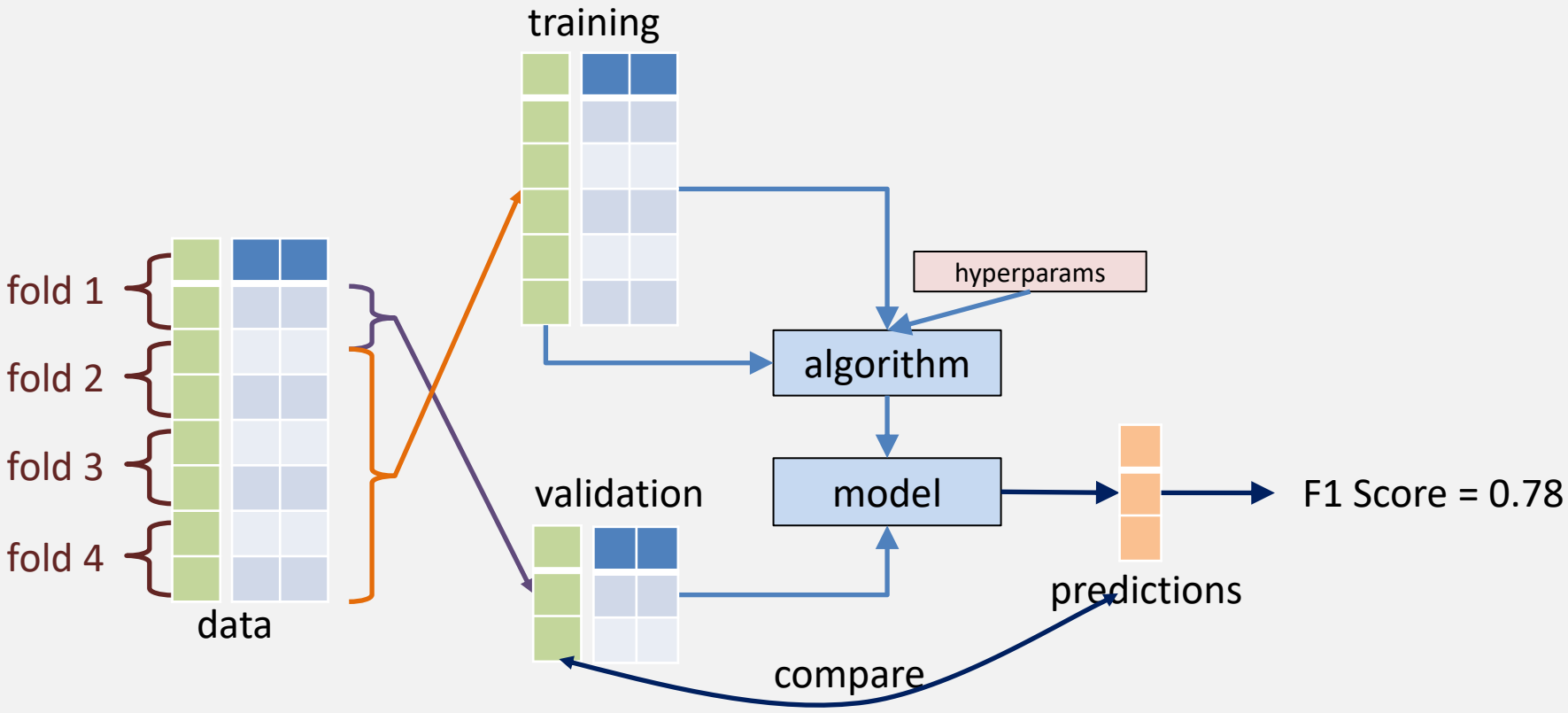
Just Use All the Data to Find F1 Score!



- Doesn't work! Why?

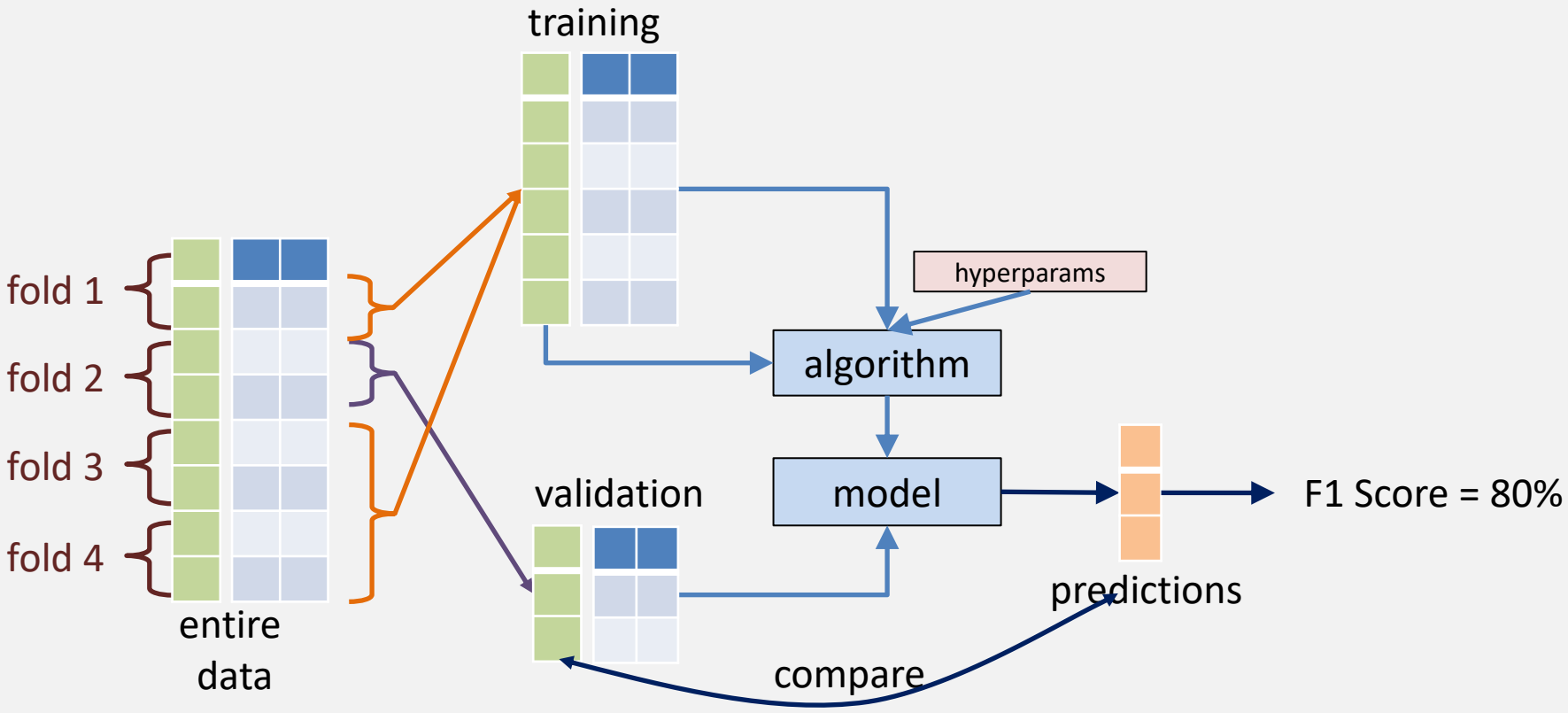
K-Fold Cross Validation

- Randomly split the data into K "folds" (i.e., subsets)
- K times:
  - Set one fold for *validation*
  - Set all other folds for *training*
  - Train the model like normal, evaluate predictions like normal



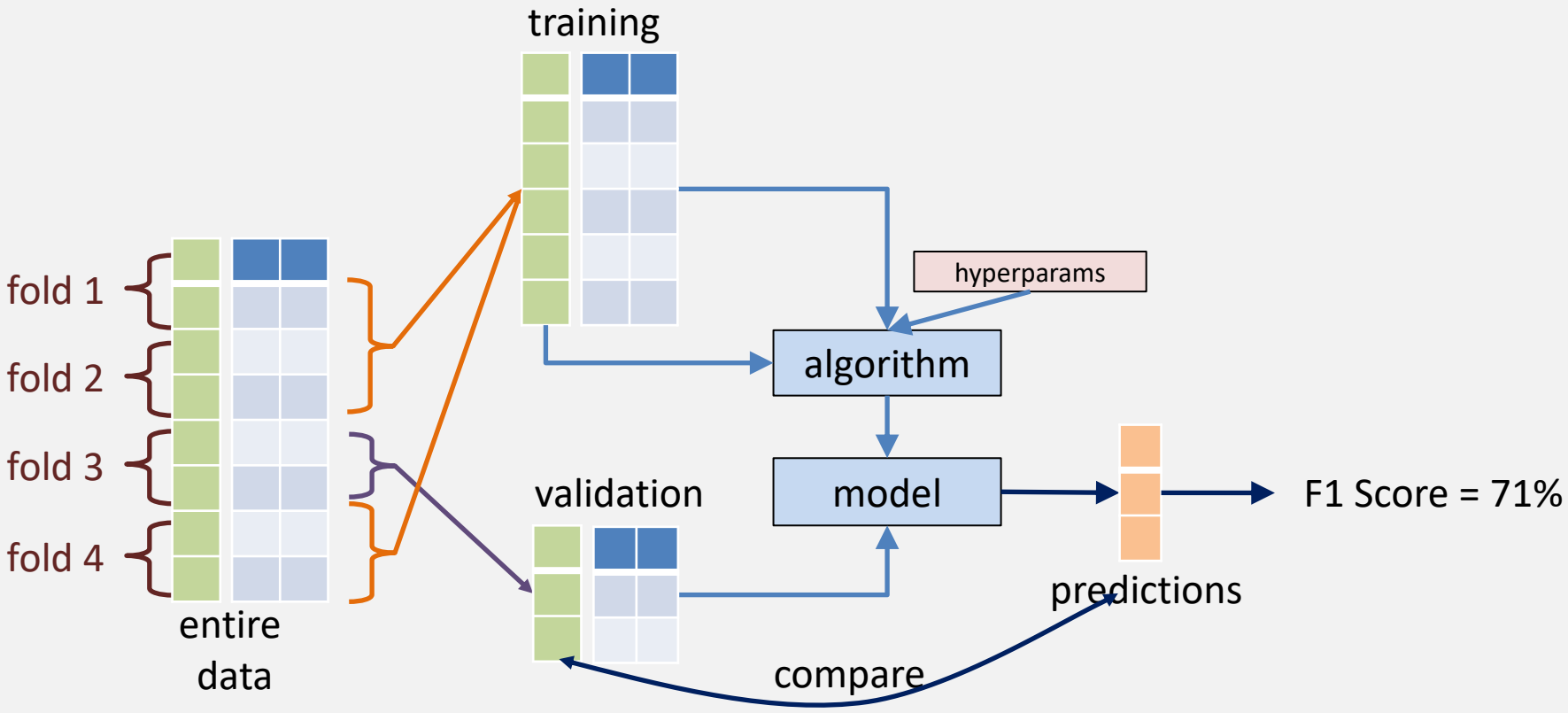
K-Fold Cross Validation

- Do the same thing, with the next fold as validation.
- F1 Scores = [.78, .80]



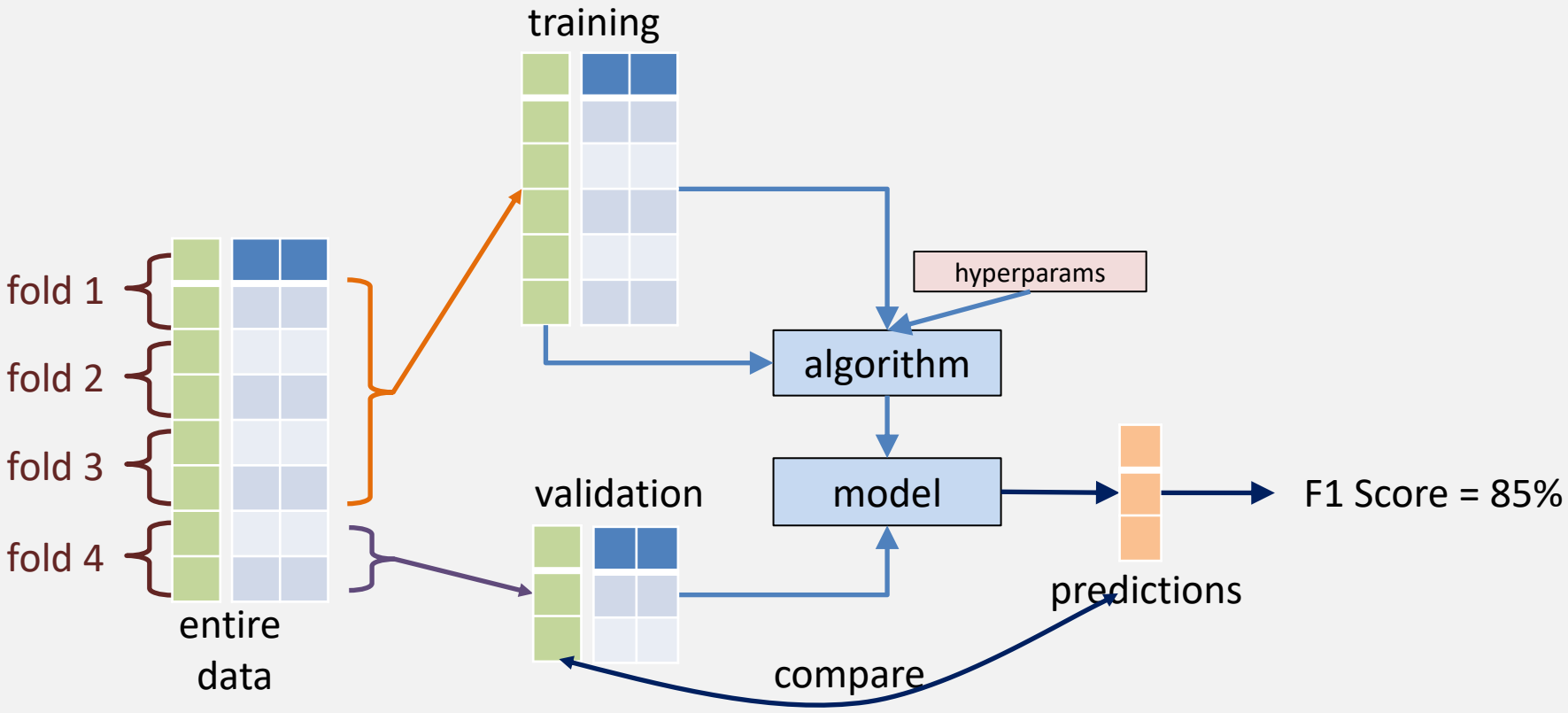
K-Fold Cross Validation

- Do the same thing, with the next fold as validation.
- F1 Scores = [.78, .80, .71]



K-Fold Cross Validation

- Do the same thing, with the last fold as validation.
- F1 Scores = [.78, .80, .71, .85]
- F1 score on future data will be 0.78 ± 0.05



Example

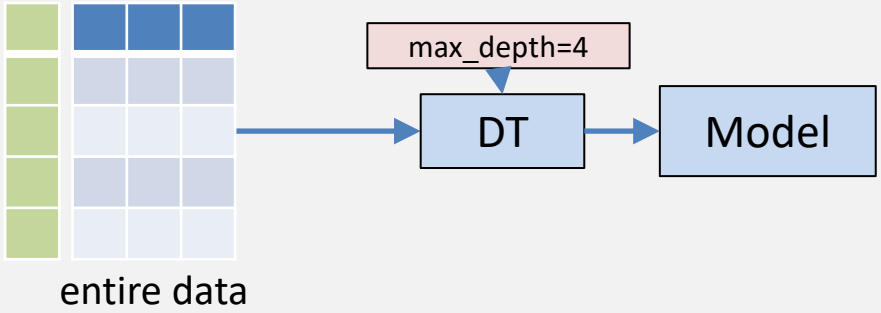
```
from sklearn.model_selection import cross_val_score
clf_cv = DecisionTreeClassifier(random_state=42, criterion="entropy",
                                min_samples_split=2, min_samples_leaf=1, max_depth=100, max_leaf_nodes=500)

scores = cross_val_score(clf_cv, X, y, cv=10, scoring="accuracy")
```

```
scores
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

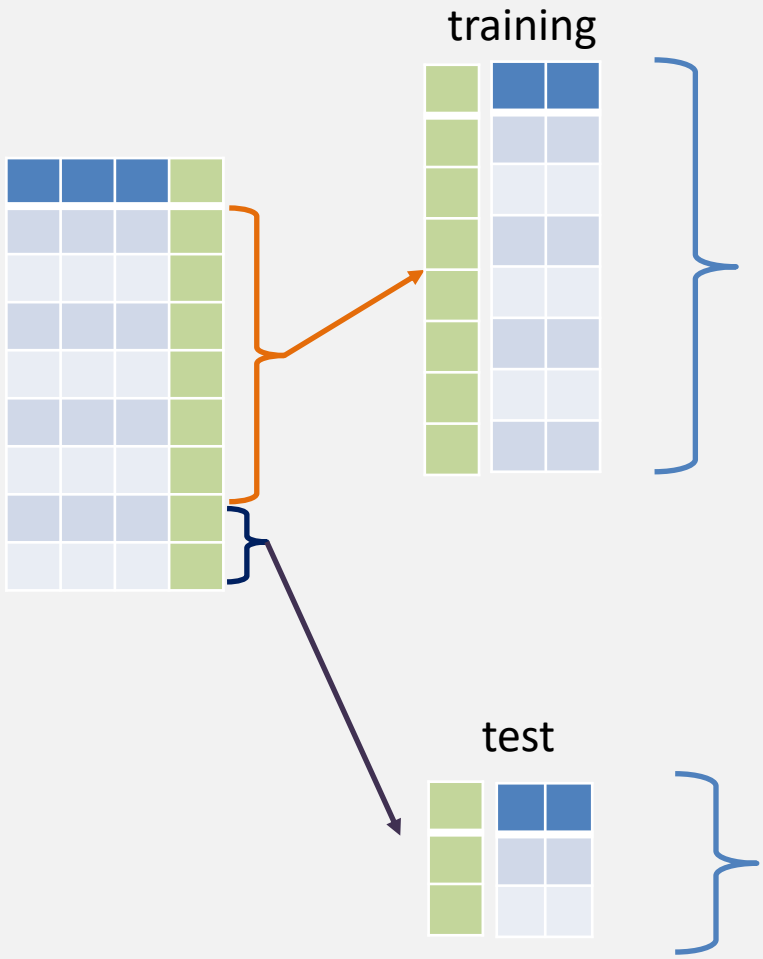
```
array([0.88235294, 0.82      , 0.86      , 0.8       , 0.88      ,
       0.92      , 0.82      , 0.88      , 0.82      , 0.89795918])
Accuracy: 0.86 (+/- 0.08)
```

Note 1: Final Production Model

- During CV, K models will be created
  - Which do you use for production?
  - None!
- After you've done CV and selected best model, retrain model "one last time" with full data



Note 2: Holdout Testing Set for HT



Use CV to find best hyperparameter values

Used to evaluate the model with best hyperparameter values

Note 3: FE, CV, and Leakage

- Next session, we will discuss best practices to avoid data leakage while engineering features

QUICK CHECK

Quick Check

- T or F? If you don't do CV, your model will overfit.

- T or F? CV will ensure you select the best model.

RESOURCES

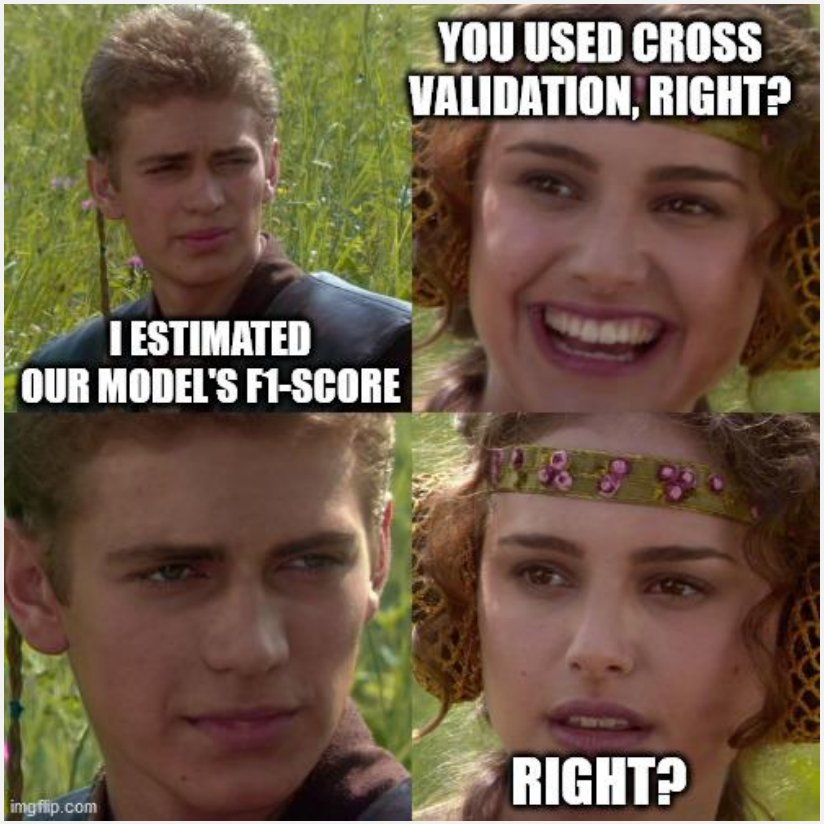
Resources

- Coding Tutorial Files by Uncle Steve
  - https://github.com/stepthom/869_course/tree/main/classification

SUMMARY

Summary

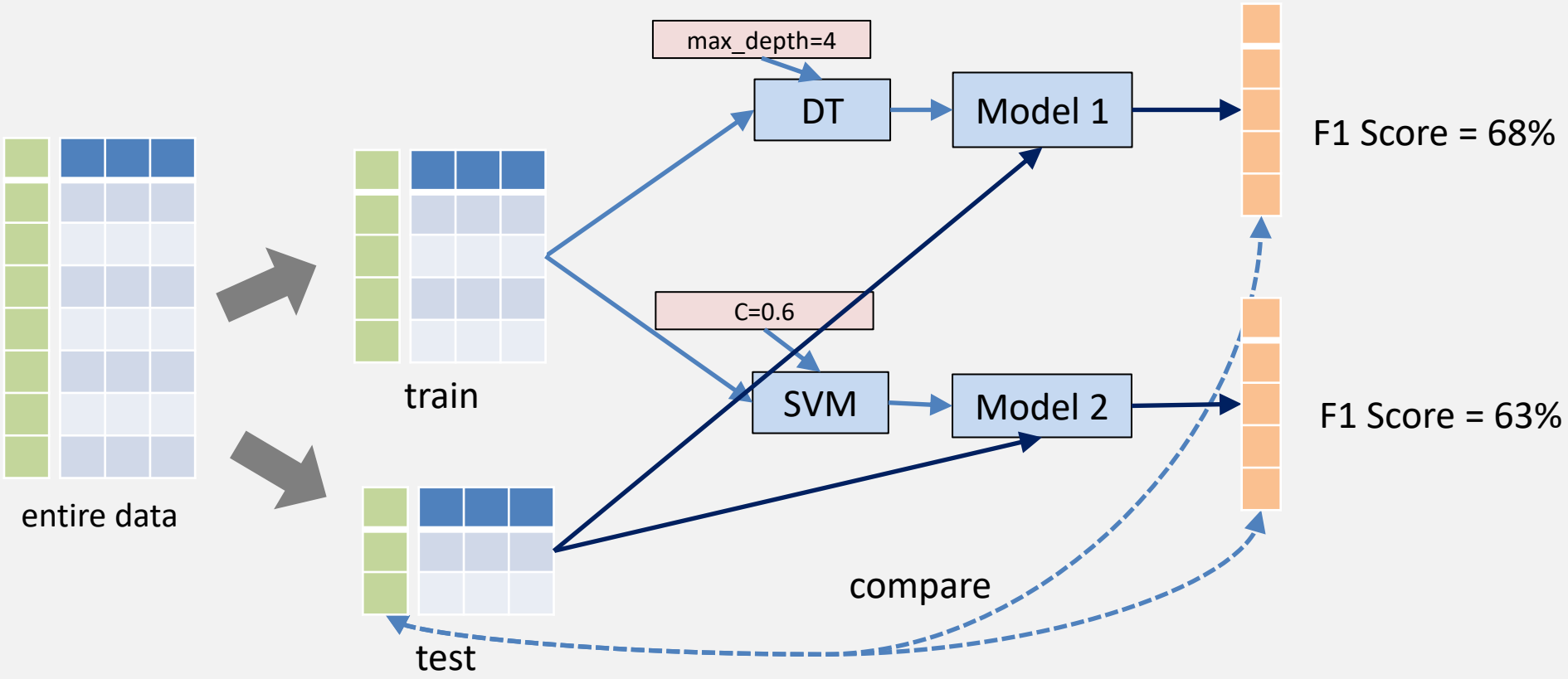
- **Cross-validation:** Robust way to assess/estimate model's performance on [future, unseen, unlabeled] data
  - So you can select which model is best
  - … with confidence



APPENDIX

Holdout Method

- ***Holdout Method***: Randomly divide the data into two subsets
  - Rule of thumb: 80%/20% split
- *Train set:* Used to train the models
- *Test set* (also called *holdout set*): used to evaluate model's predictions
  - We pretend the test data is "future data"

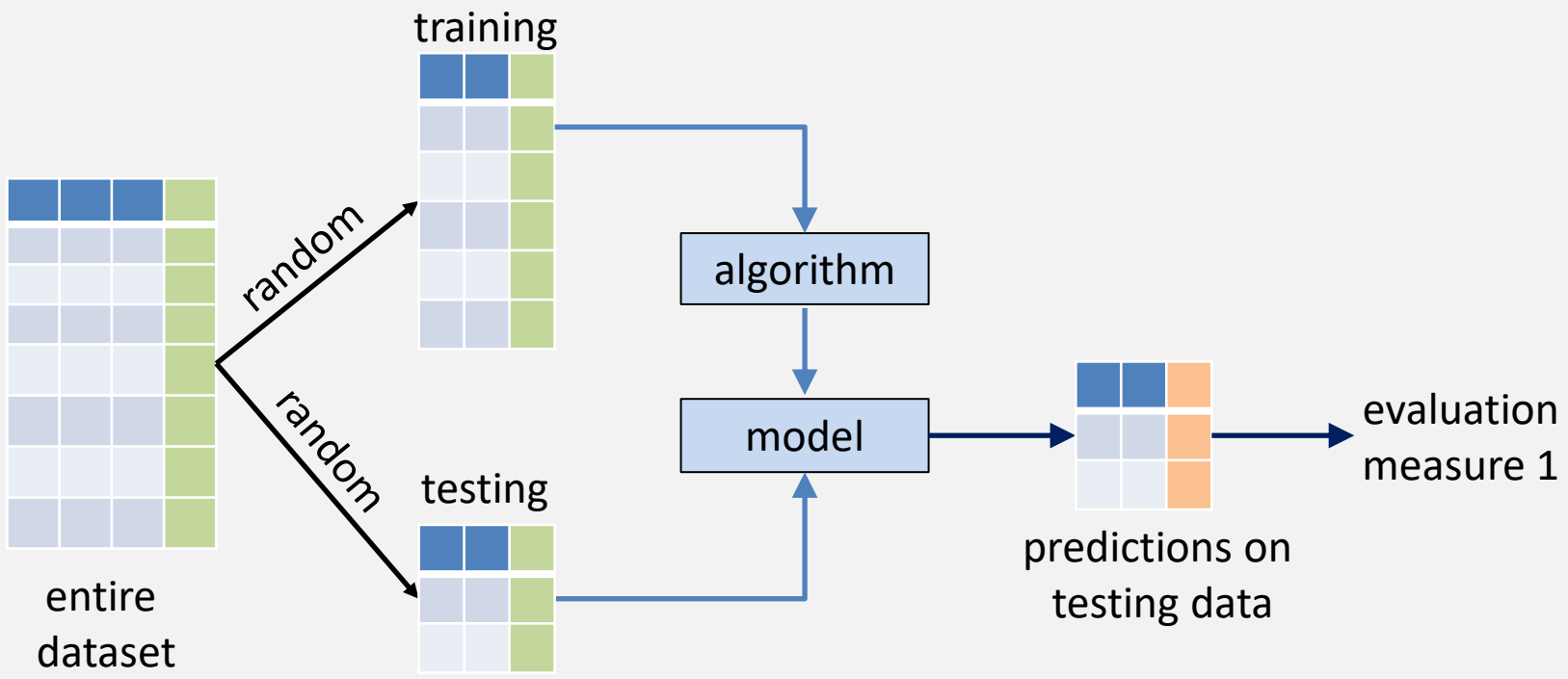


Example

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=48)
clf_holdout = DecisionTreeClassifier(random_state=42, criterion="entropy",
                            min_samples_split=2, min_samples_leaf=1, max_depth=100, max_leaf_nodes=500)
clf_holdout.fit(X_train, y_train)
y_pred_holdout = clf_holdout.predict(X_test)
```

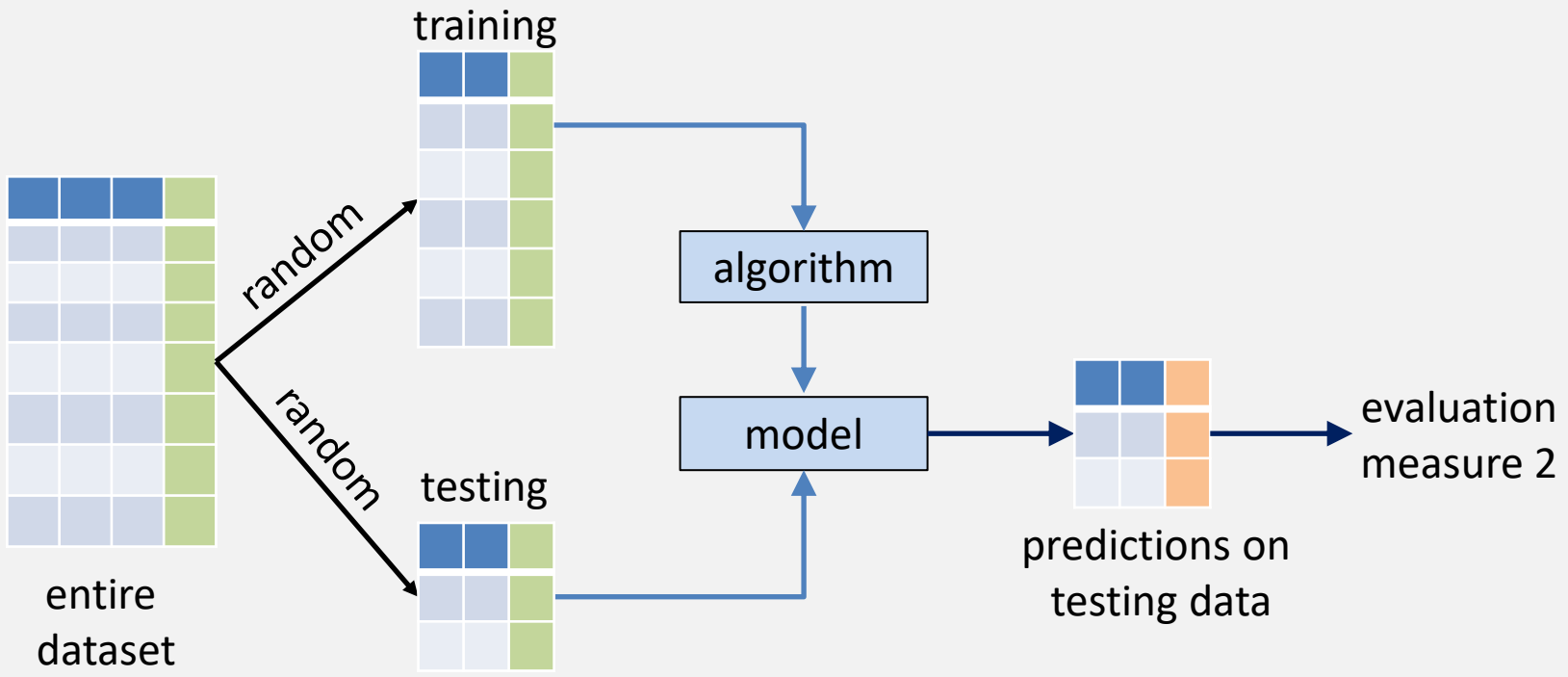
Bootstrapping (1/4)

- Also called "repeated random subsampling"
- Same as the holdout method, except you do it many times:
  - Divide data into testing and training, get evaluation measure



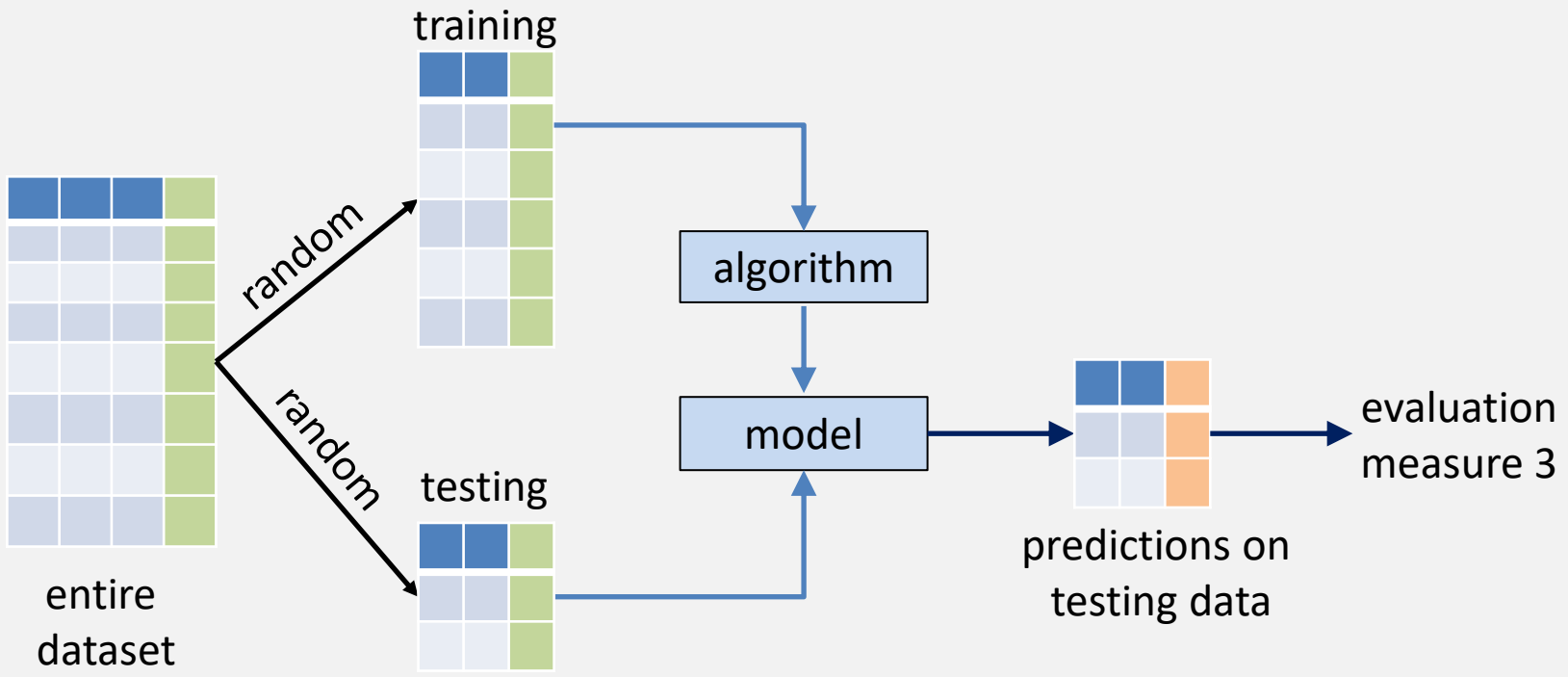
Bootstrapping (2/4)

- Also called "repeated random subsampling"
- Same as the holdout method, except you do it many times:
  - Divide data into testing and training, get evaluation measure
  - Repeat as many times as you want



Bootstrapping (3/4)

- Also called "repeated random subsampling"
- Same as the holdout method, except you do it many times:
  - Divide data into testing and training, get evaluation measure
  - Repeat as many times as you want



Bootstrapping (4/4)

- Also called "repeated random subsampling"
- Same as the holdout method, except you do it many times:
  - Divide data into testing and training, get evaluation measure
  - Repeat as many times as you want
  - Take average of evaluation measures

$$\text{Final Evaluation measure} = \frac{EM1 + EM2 + EM3}{3}$$

K-Fold Cross Validation (1/6)

Similar to bootstrapping, except each round, make sure you use a different subset for training.

Example

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(random_state=42, criterion="entropy",
                             min_samples_split=2, min_samples_leaf=1, max_depth=100, max_leaf_nodes=500)
clf.fit(X, y)

y_pred_dt = clf.predict(X)
```

Example

```
from pandas_ml import ConfusionMatrix

print(ConfusionMatrix(y, y_pred_dt))
```

```
Predicted  False  True  __all__
Actual
False        251     0      251
True           0   249      249
__all__      251   249      500
```

```
from sklearn.metrics import classification_report

print(classification_report(y, y_pred_dt, target_names=class_names))
```

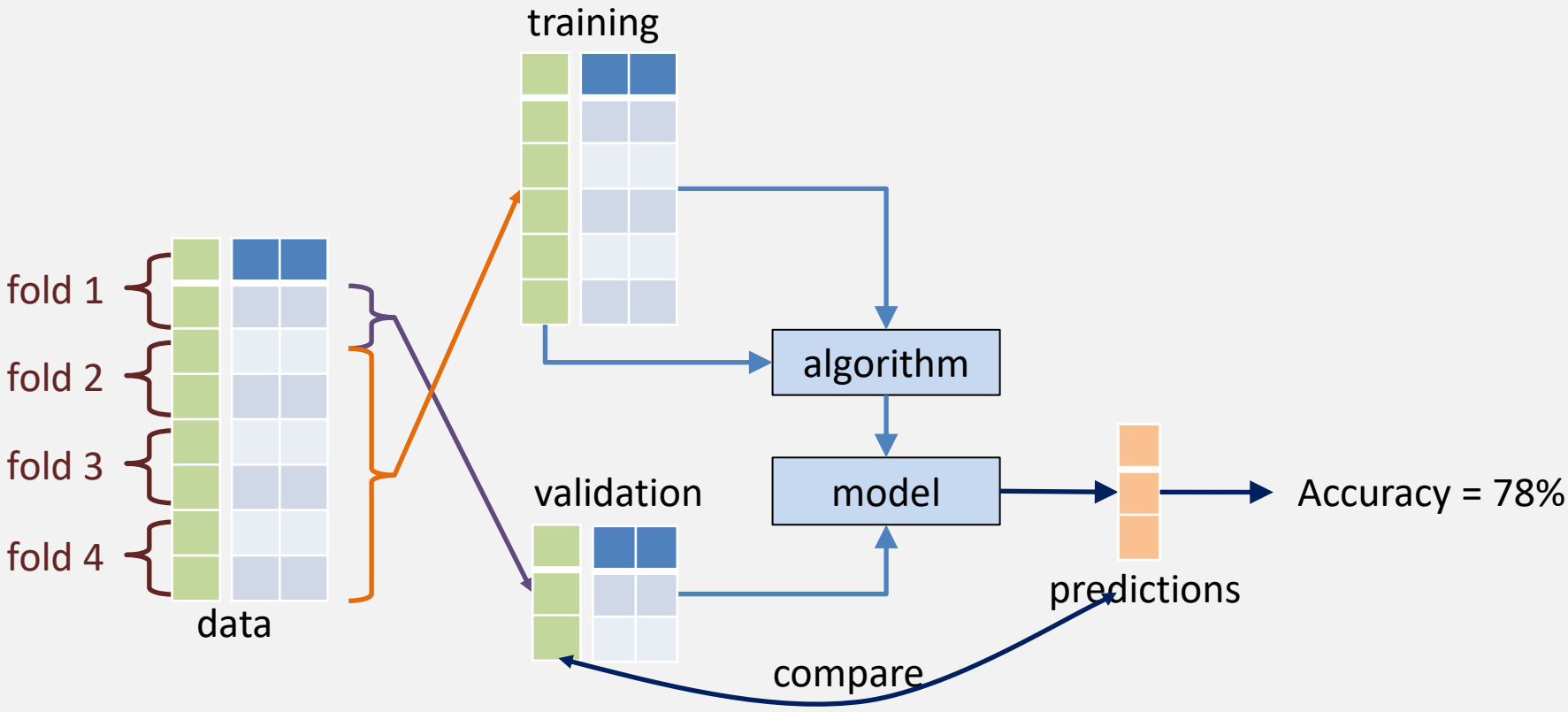
```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00       251
          1       1.00      1.00      1.00       249

avg / total       1.00      1.00      1.00       500
```

```
from sklearn.metrics import accuracy_score, cohen_kappa_score, f1_score, log_loss

print("Accuracy = {:.2f}".format(accuracy_score(y, y_pred_dt)))
print("Kappa = {:.2f}".format(cohen_kappa_score(y, y_pred_dt)))
print("F1 Score = {:.2f}".format(f1_score(y, y_pred_dt)))
print("Log Loss = {:.2f}".format(log_loss(y, y_pred_dt)))
```

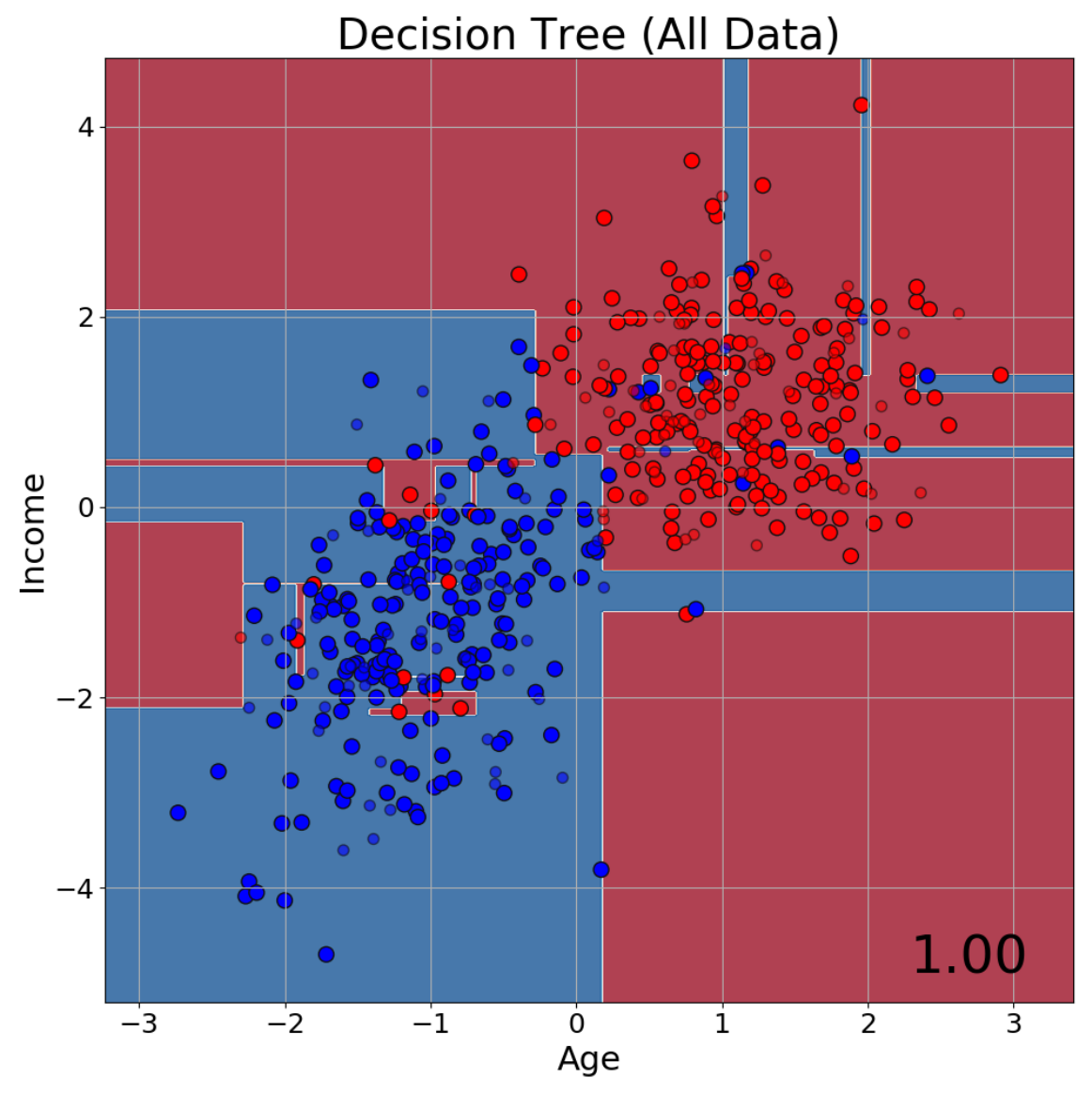
```
Accuracy = 1.00
Kappa = 1.00
F1 Score = 1.00
Log Loss = 0.00
```

K-Fold Cross Validation

- Best method to test future performance
  - Randomly split the data into K "folds" (i.e., subsets)
  - K times:
    - Set one fold for *validation*
    - Set all other folds for training
    - Train the model like normal, evaluate predictions like normal

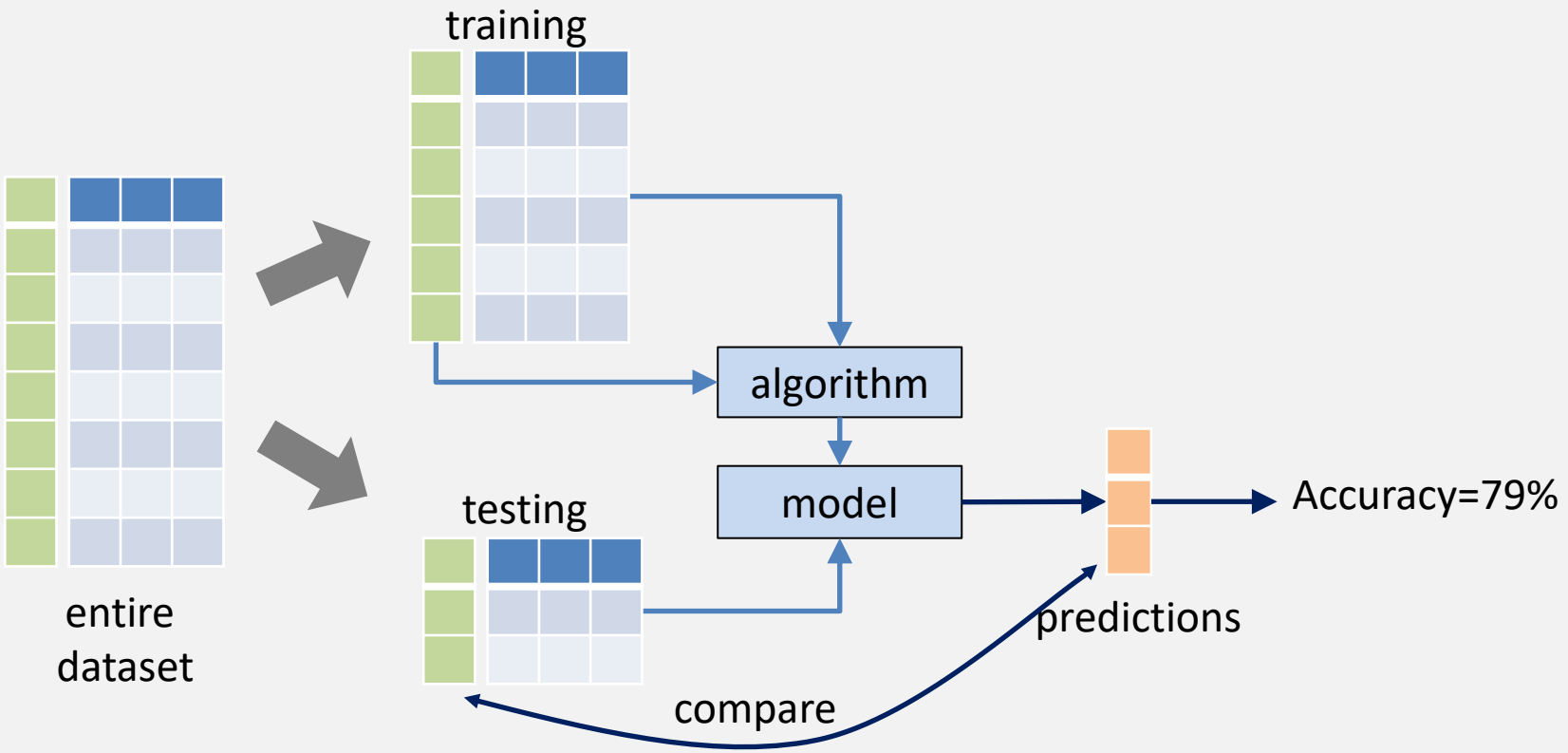


Example



Holdout Method

- ***Holdout Method***: Randomly divide the data into two subsets
  - Rule of thumb: 80%/20% split
- *Training set:* Used to train the model
- *Test set* (also called *holdout set*): used to evaluate model's predictions
  - We pretend that the test data is "future data"



Why Model Validation?

Smith SCHOOL OF BUSINESS Queen's University

During a typical ML project, you will ask yourself:

How well will my model perform in the future?	• Called "model evaluation"
Which algorithm should I use?	• "Algorithm selection"
Which values for hyperparameters are best?	• "Hyperparameter tuning" • "Model selection"
Which variables should I use?	• "Feature/variable selection"

- ***Model Validation*** are ways to robustly quantify the performance of *one* model
- Way to estimate how well the model will work in the future, when the data is not labeled!
- Can use to compare two models/algorithms/hyperparameter values/etc.
- Different model validation techniques:
  - Hold out
  - K-fold cross validation

Leave One Out (LOO)

- Special case of K-Fold CV, where K=N
  - i.e., each fold only has one instance
- Not practical; rarely used