

You scaled your data for a random forest.

D8A.ACADEMY/RESOURCES/NORM-VS-STANDARD

THE TWO

standardize	$(x - \text{mean}) / \text{std}$
normalize	$(x - \text{min}) / (\text{max} - \text{min})$
z-score vs 0-1	← names get swapped.

WHO NEEDS IT

distance models	KNN, k-means, SVM
gradient descent	linear, logistic, nets
regularized	L1/L2 punish big scales

WHO DOES NOT

trees	splits ignore scale
random forest	same
boosting	← scaling changes nothing.

PICK ONE

standardize	the default
normalize	nets, images, bounded
robust scaler	outliers, uses IQR

DO IT RIGHT

fit on train	transform train + test
inside a Pipeline	CV stays leak-free
save the scaler	production needs it

GOTCHAS

fit on all data	test leaks into mean
one-hot columns	do not scale 0/1
target scaled	unscale the predictions

WORTH MEMORIZING

01 The scaler lives inside the Pipeline

Fit on train, transform test, never the other way round. The Pipeline makes that impossible to get wrong.

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

knn = make_pipeline(StandardScaler(),
                    KNeighborsClassifier())
knn.fit(X_train, y_train) # scaler fits on train only
knn.score(X_test, y_test) # test is transformed, not fit

# trees: no scaler, same result either way
rf = RandomForestClassifier().fit(X_train, y_train)
```

→ `cross_val_score(knn, X, y)` now refits the scaler inside every fold. A scaler fitted on all rows first would leak the test mean into training.

02 Trap: three scaling mistakes

Two are wasted work, one is a leak that inflates your score.

YOU DID	THE EFFECT	INSTEAD
scaled for XGBoost	none, splits ignore scale	skip it, save the step
fit scaler on all rows	test mean leaked in	fit on train, in a Pipeline
KNN, no scaling	salary drowns age	StandardScaler first

→ Interview phrasing: I scale for distance-based and gradient-based models, skip it for trees, and always fit the scaler inside the pipeline so CV stays honest.