How to Choose and Improve Machine Learning Models

- [ ] What models should I try?
- [ ] How do I choose between models?
- [ ] How do I know I have the best model?

WARNING: This session will feel like you're driving in the mountains - exciting but a little scary with lots of switchbacks!

STEP 1 Take the data set and split it into a training and test data set.

Depending on how large your dataset is, you'll take

70% / 30%
Train Test

Common ways to split

OR

80% / 20%
Train Test

Smaller

Larger

STEP 2: Choose a bunch of models

What models should I use?
What kind of problem is it?

Predict a numerical value

- regression, linear/non-linear
- SVM, linear/non-linear
- Tree, Random Forest
- Neural Network

Predict a class/category

- logistic regression, linear/non-linear
- SVM, linear/non-linear
- Tree, Random Forest
- Neural Network
- Naive Bayes

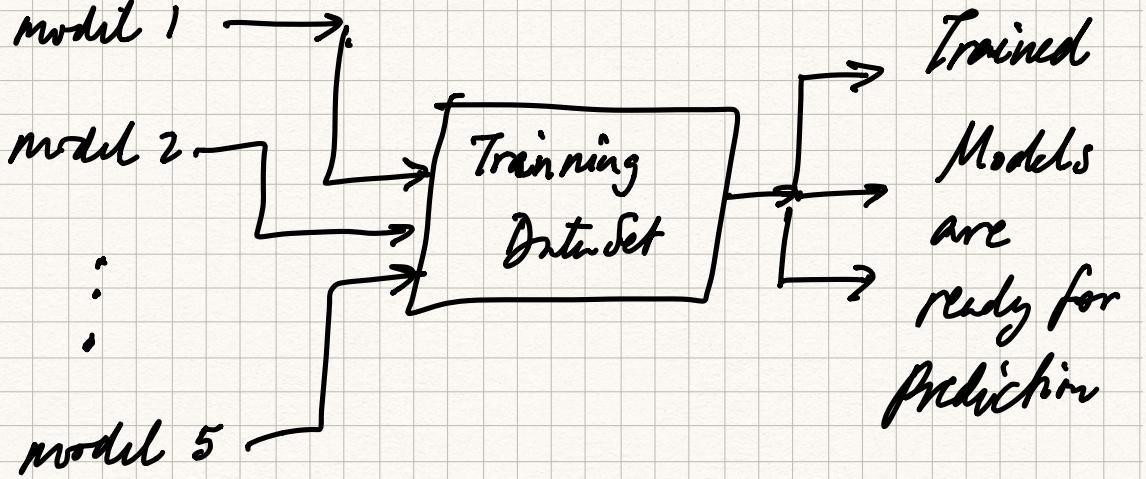
pretty much the same models for both kinds of predictions

□ exceptions to the rule

STEP 3 Train each model using the training data set.

Note: For each model, use the model's default hyperparameters.

↑ Know where to find and set these in Orange



STEP 4: Measure the performance of each trained model on the test data set.

How to measure the performance of a model?

For numerical prediction:
- MSE - Mean Square Error
- RMSE - Root Mean Square Error

For class/category prediction:
- F1
- Accuracy (if you know the target values are balanced)

Make yourself a table

	F1
model 1	78%
⋮	
model 5	92%

← This model looks great

But... is it?

OK, I know which model performed the best. But I just used default values for each model.

- What if these defaults change?
- Will the relative performance of the models change?

Probably!

So what do I do now?

Let's take a pause and look at how models work.

Complexity of a Model

The Complexity of a model depends on:

- The features
- The model's hyperparameters
- the amount of training data

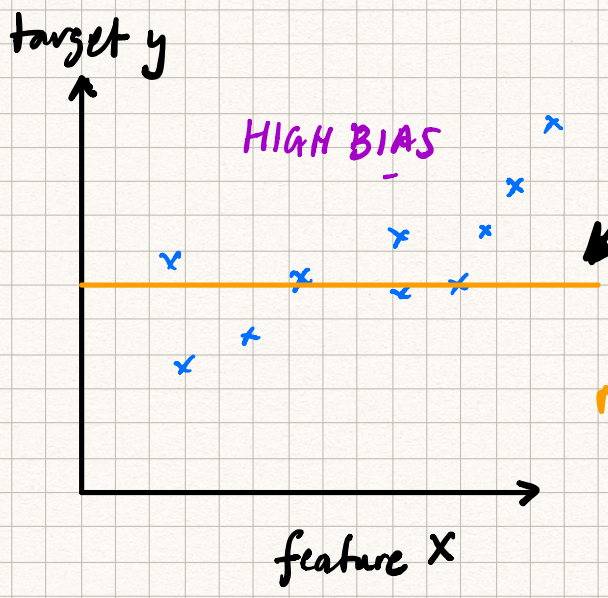
Let's look at a model's hyperparameters in Orange

- Regularization (regressors & classifiers)
- Linear / Non-Linear (SVM)
- Layers, # of units (Neural Network)

Let's think about model complexity by asking what it means for a model to learn from data.

There are 2 extreme ways to learn from data.

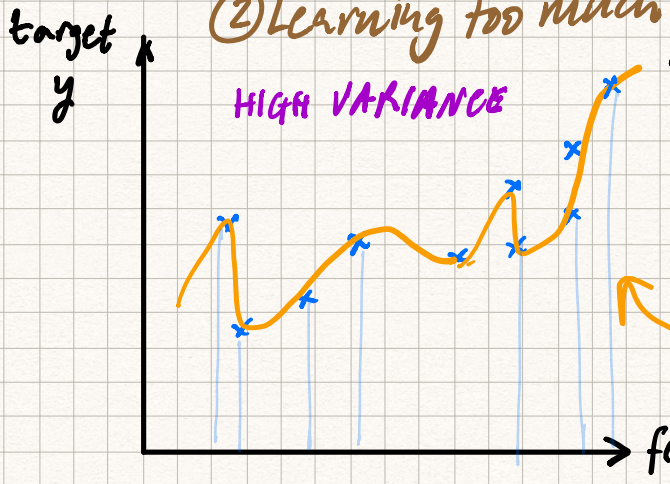
① Learning too little



always predict the same value no matter the feature value

Underfitting

② Learning too much

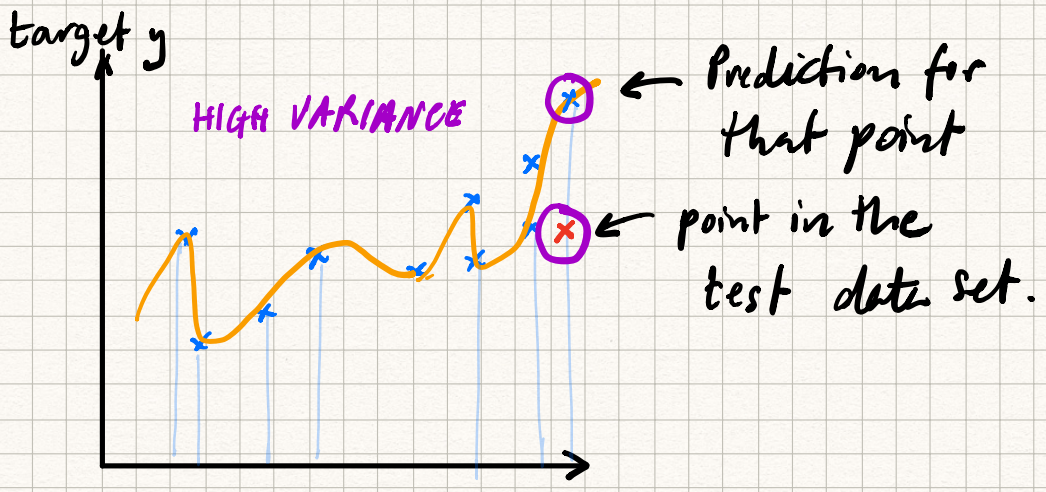


Squeezing every last drop out of the data!

Overfitting

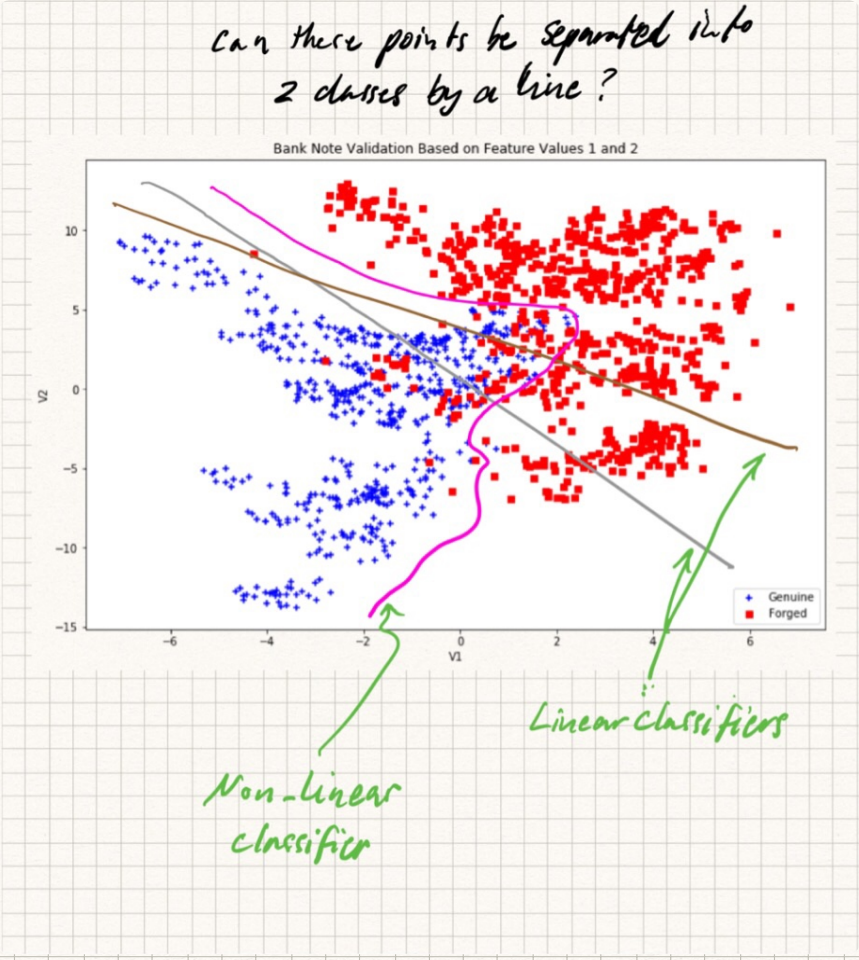
Note: these are not scatterplots.

Why is learning too much a bad thing?



When models learn too much they make (wild) mistakes. This makes them error prone.

Learning too little or too much in classification problems



Models that learn too little

Models that learn too much

High Bias = Learning too little

= Low Complexity

Risk of underfitting

High Variance = Learning too much

= High Complexity

Risk of overfitting

OK, back to where we were

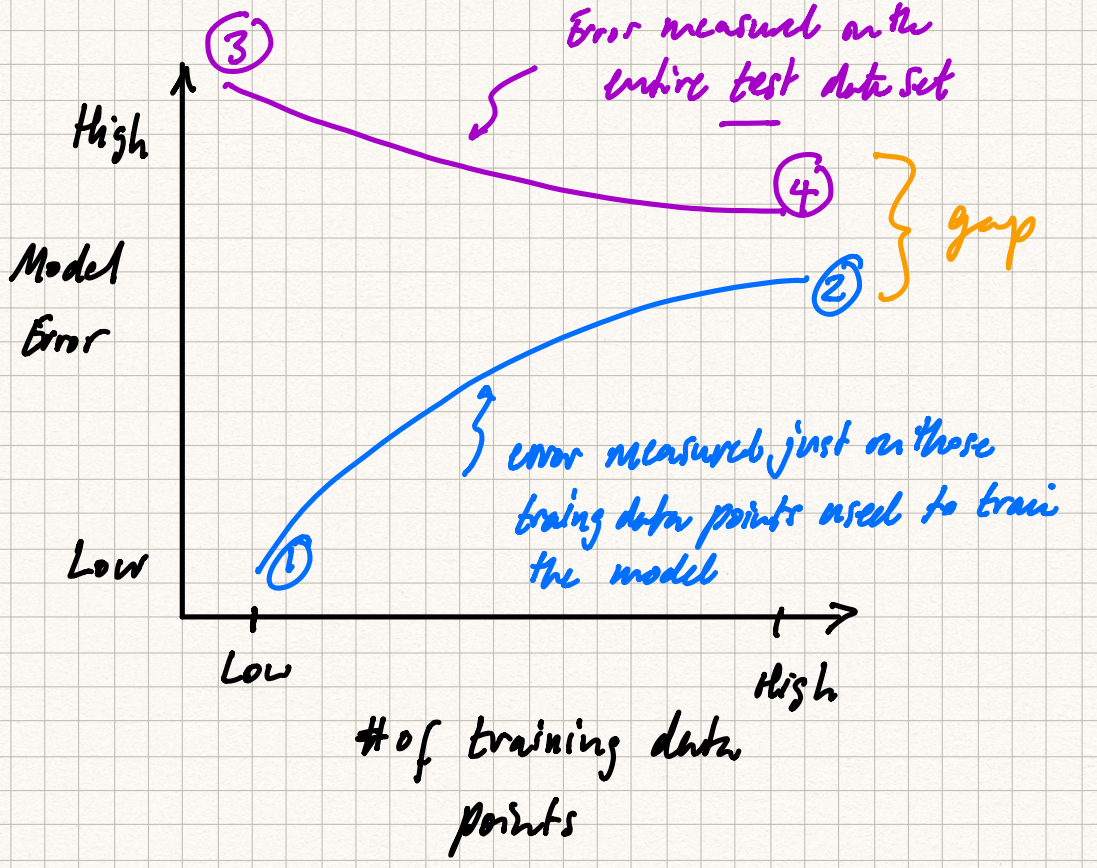
before we got side tracked ...

Learning Curves

For a given model and dataset, learning curves allow us to determine whether and in which direction the model's hyper-parameters need to be tuned.

In other words, a way to find out if a model has high bias or high variance (or neither)

How to Construct a learning Curve



(1) when the # of train data points is low model error on just these closer data points will be low.

(2) when the # of train data points is high, model error on these data points will rise up to a point and stabilize.

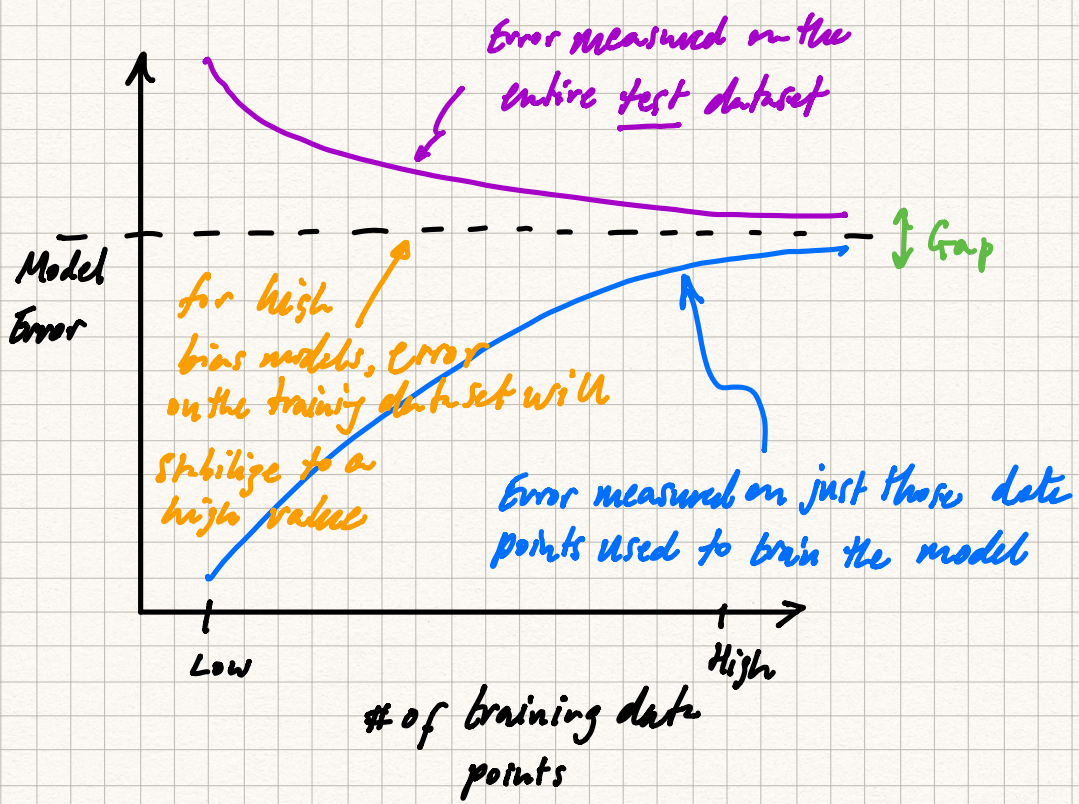
③ when the model is trained on just a few data points, the error on the entire test dataset will be large.

④ when the model is trained on pretty much the entire training data set, the error on the entire test set will drop to its lowest value and stabilize there.

The gap between ④ and ② gives an indication of whether a model has high bias or high variance.

Let's see how this works...

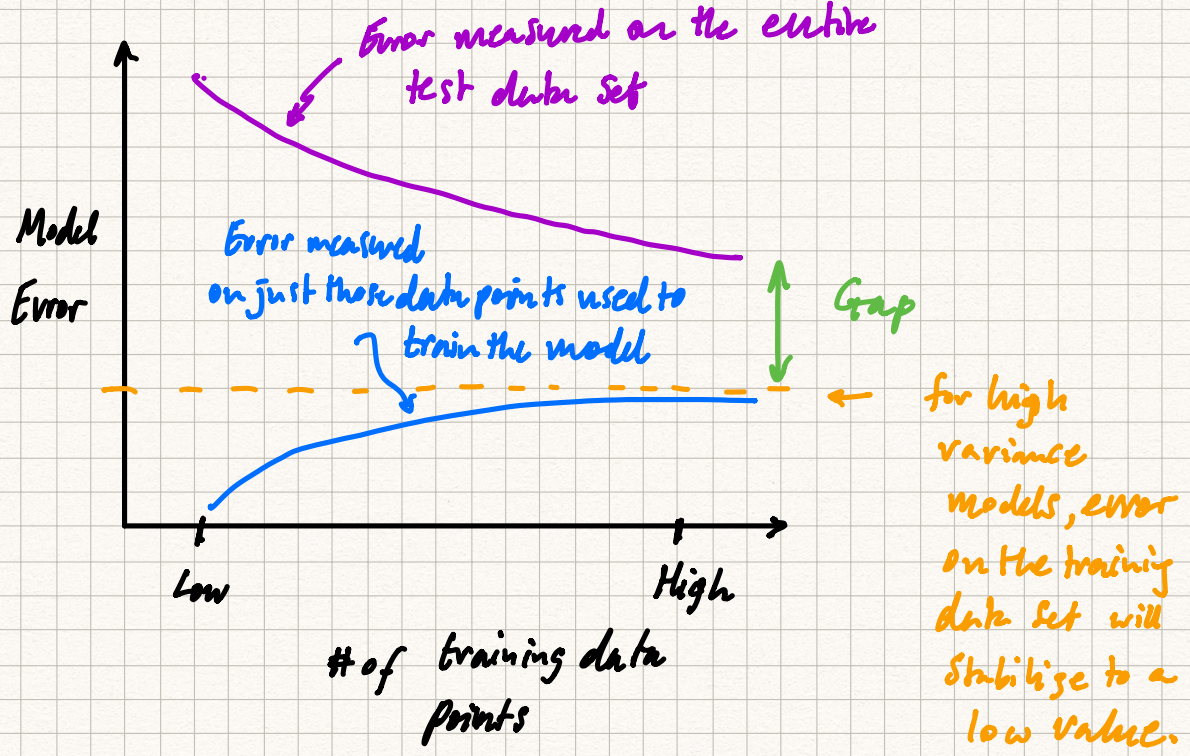
Learning Curves for High Bias Models



The error of a high bias model, as measured on the test dataset is going to be high, no matter how much training data is used to train the model.

The gap between the — and — lines will be SMALL.

Learning Curves for High Variance Models



The error of a high variance model, as measured on the test data set is going to be <u>much higher</u> than its error as measured on the training data set.

The gap between the — and — lines will be <u>LARGE</u>.

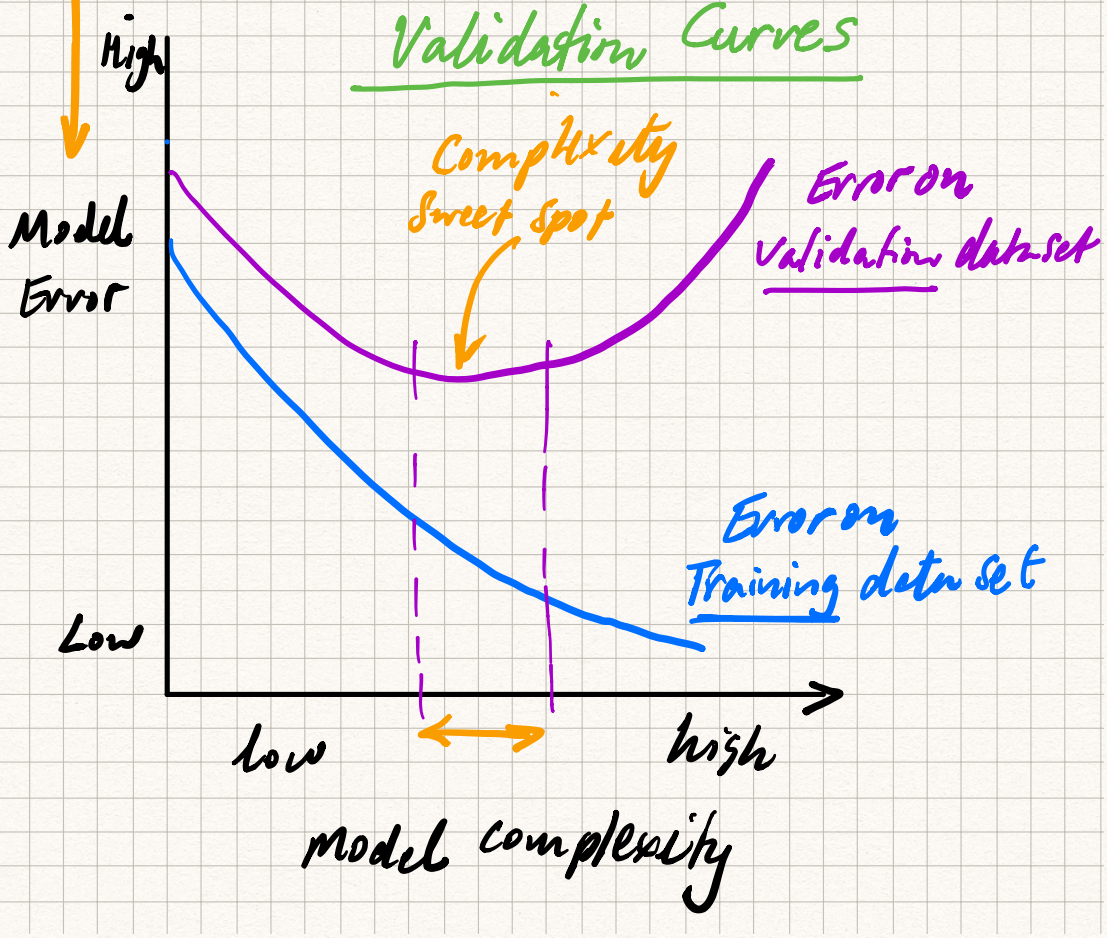
When a model has high bias, adding more data will not help. We need to make the model more Complex to improve its performance.

When a model has high variance,

- [x] making it less Complex will improve its performance.
- [x] Adding more data to the training dataset will also improve performance.

STEP 5 (finally!): Once you know if model complexity needs to be increased or decreased, proceed to systematically adjust the model's hyperparameters in that direction.

CAUTION: when F1 is high error is low
when F1 is low error is high



But now we see this thing called the Validation set. What is it and why do we need it?

Read Training, Validation, and Test Data Sets.

By using k-fold cross validation, you reduce the chance that your model overfits as you adjust its complexity to find the sweet spot of model complexity that results in the best performance for that model and the given data set.

Wow! That's a lot of stuff to think through!

Let's Summarize

How do you pick the best model for the problem at hand?

1) Pick a bunch of appropriate models.

2) For each one, use learning curves to figure out which way to tweak the complexity of the model.

3) Use k-fold cross validation to measure the performance of the model as you tweak it.

→ (Contd.)

Summary (Contd.)

4) Find the combination of hyper-parameter values that produces the best performance for that model.

5) For this setting of hyper-parameters, measure the performance of the model on the TEST data set. This is the true performance of the model.

6) Compare the true performance of the models — choose the one with the highest performance.