POLYNOMIAL REGRESSION

Overfitting/Tuning Explained

OVERVIEW

You will learn about:

- Overfitting in detail.
- Circumstances that make *overfitting* more likely to occur.
- Consequences of overfitting when predicting new data.
- Hyper-parameter tuning to avoid *overfitting*.
  - Validation
  - Cross Validation»

OVERFITTING

If a model performs well when approximating the training data but does not perform well when it faces new data to predict outcomes.

Overfitting is one of the most pressing and still not fully solved problems in machine learning.»

CIRCUMSTANCES THAT CAN LEAD TO OVERFITTING

- If the training dataset does not have a sufficient number of observations.
- If the model considers many variables and thus contains many parameters to calibrate.
- If the underlying machine learning model is highly non-linear.»

THE DATA

In what follows we use the Kings County Real Estate dataset.

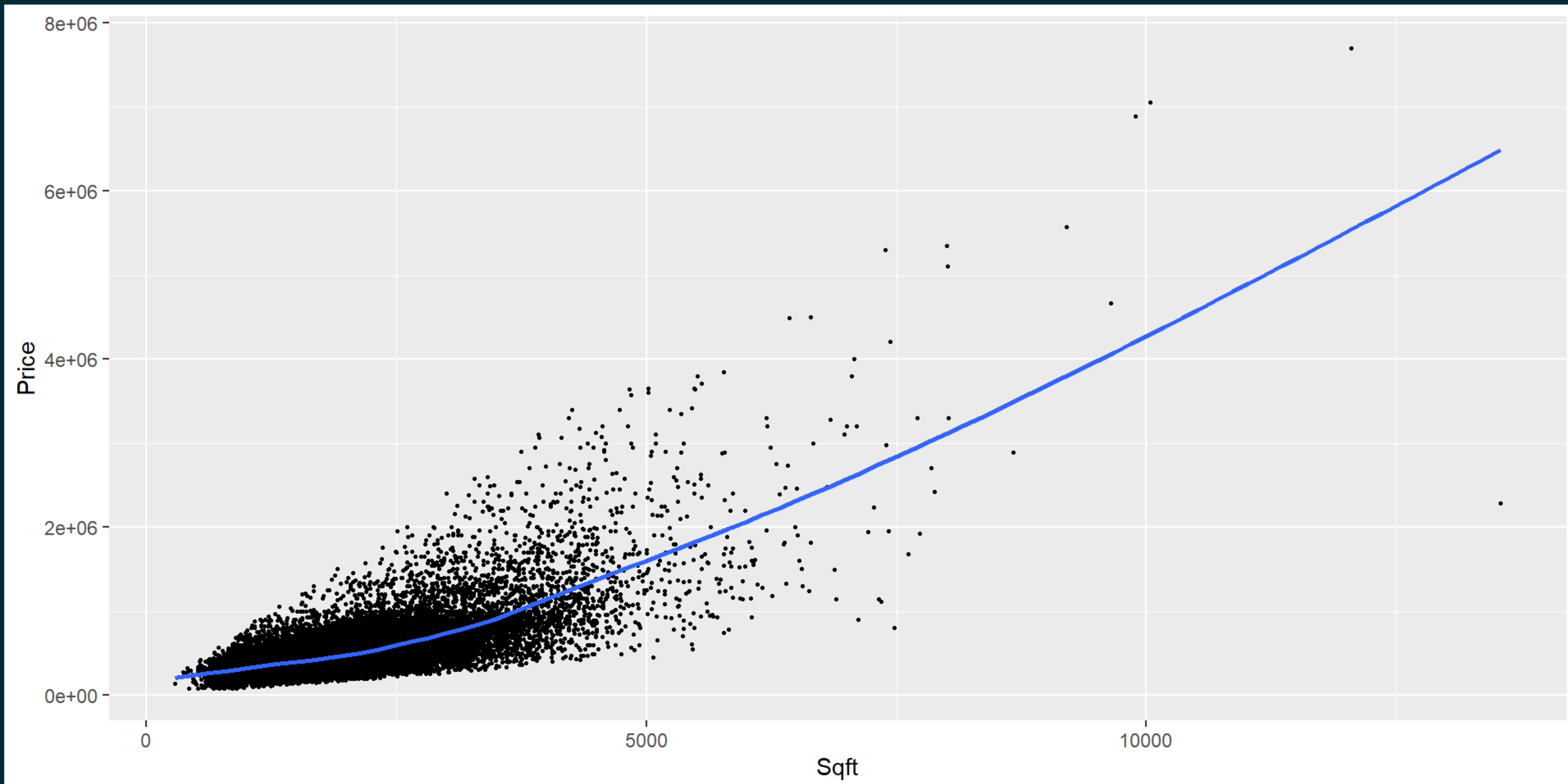
▶ Code

We want to demonstrate *overfitting*. Therefore, we ceate conditions that **likely trigger overfitting**. Consequently, we work only with a very small training dataset (20 observations=0.1% of total observations. All other observations become testing data:

▶ Code

DATA VISUALIZATION

There seems to be a non-liner trend:



TRAINING DATA STRUCTURE

▶ Code

	Price	Sqft
1	153503	1240
2	199500	1750
3	234950	1720
4	246000	2120
5	355000	1240
6	385000	2090
7	365000	910
8	349000	1690
9	474950	2030
10	450000	1540
11	465000	2020
12	445000	1630
13	568000	2110
14	660000	2470
15	530000	1260
16	600000	2090
17	1150000	3830

POLYNOMIAL REGRESSION

Regular univariate prediction equation:

$$\widehat{Price} = \beta_1 Sqft + \beta_2$$

Polynomial univariate prediction equation (degree 5):

$$\begin{aligned}\widehat{Price} = \beta_1 Sqft + \beta_2 Sqft^2 + \beta_3 Sqft^3 \\ + \beta_4 Sqft^4 + \beta_5 Sqft^5 + \beta_6\end{aligned}$$

POLYNOMIAL REGRESSION

Polynomial univariate prediction equation (degree 5):

$$\begin{aligned}\widehat{Price} &= \beta_1 Sqft + \beta_2 Sqft^2 + \beta_3 Sqft^3 \\ &+ \beta_4 Sqft^4 + \beta_5 Sqft^5 + \beta_6\end{aligned}$$

We create $Sqft^2$, $Sqft^3$, $Sqft^4$, and $Sqft^5$ as new variables in the data and treat them as they were separate variables in a multivariate regression.

This makes the regression **linear in variables** but **non-linear in data**.

Consequently, we can use OLS to find the optimal βs.»

HOW THE DATA WOULD LOOK LIKE

▶ Code

	Price	Sqft	Sqft2	Sqft3	Sqft4	Sqft5
1	221900	1180	1392400	1643032000	1.938778e+12	2.287758e+15
2	538000	2570	6604900	16974593000	4.362470e+13	1.121155e+17
3	180000	770	592900	456533000	3.515304e+11	2.706784e+14
4	604000	1960	3841600	7529536000	1.475789e+13	2.892547e+16
5	510000	1680	2822400	4741632000	7.965942e+12	1.338278e+16
6	1230000	5420	29376400	159220088000	8.629729e+14	4.677313e+18

COMPARING REGULAR OLS AND POLYNOMINAL REGRESSION (DEGREE=5) 🤓

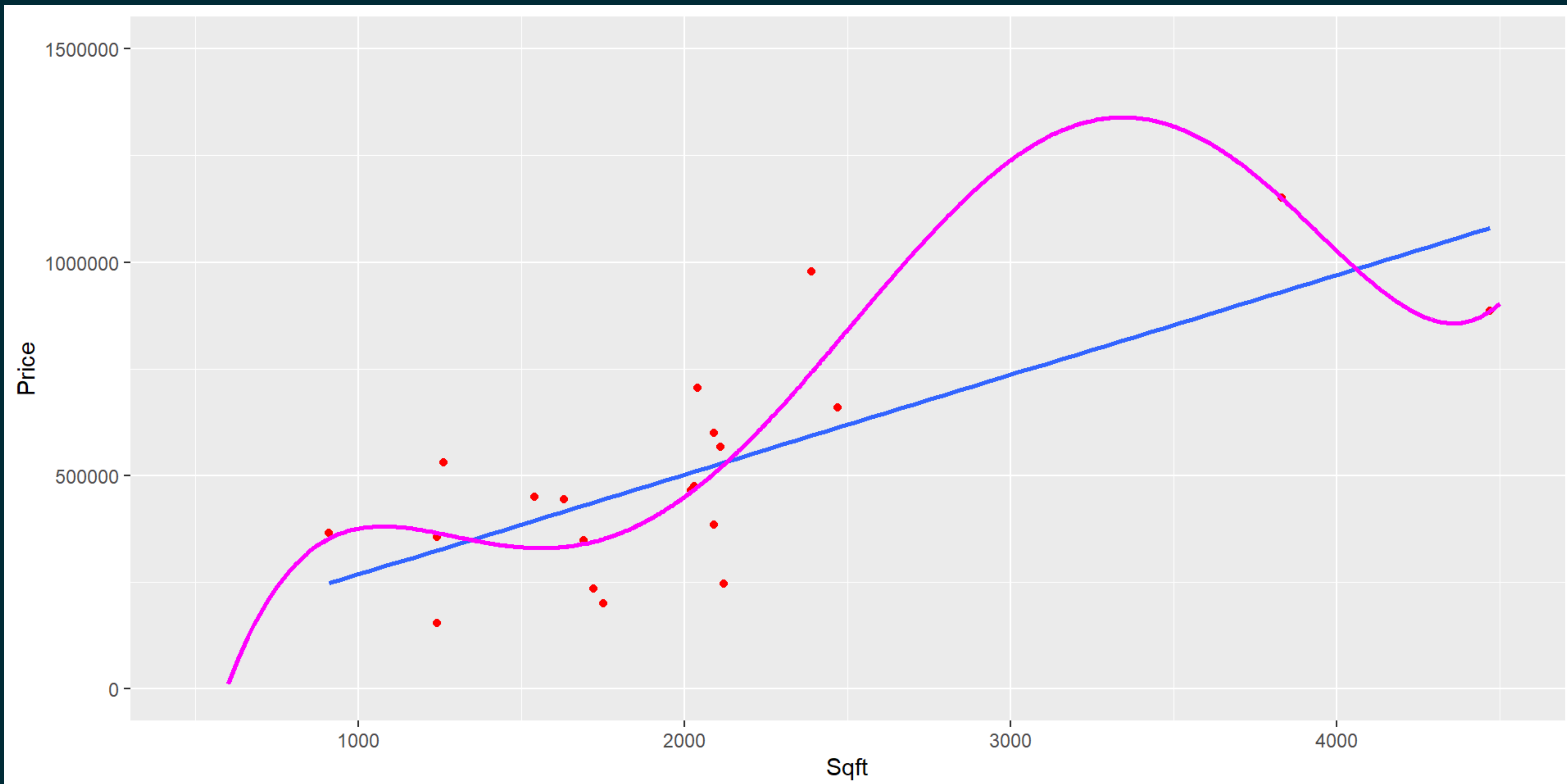
Code to compare is linked in the footer of this slide.

POLYNOMIAL REGRESSION (DEGREE=5) VS. REGULAR OLS

APROXIMATION OF THE TRAINING DATA

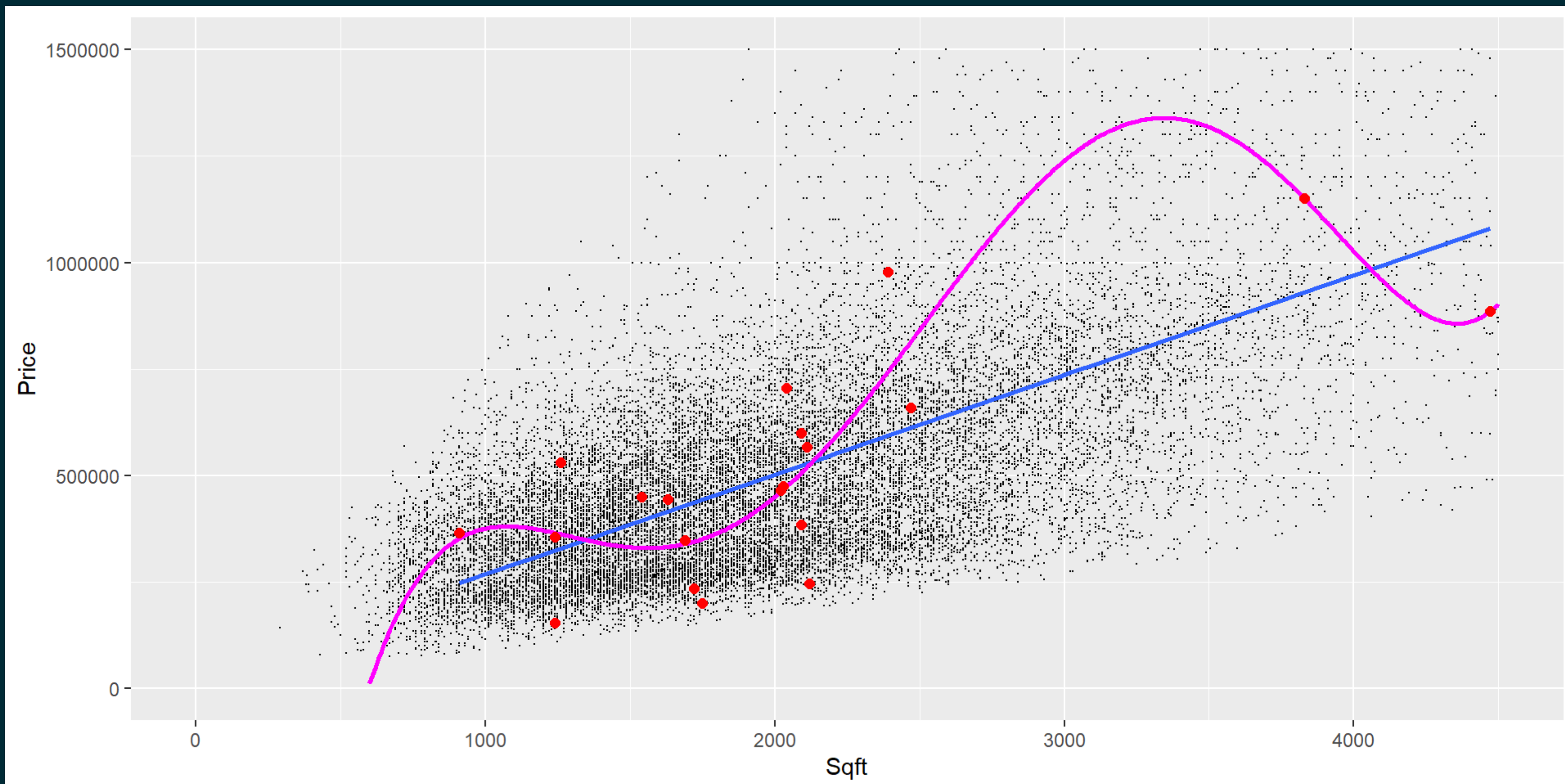
POLYNOMIAL REGRESSION (DEGREE=5) VS. REGULAR OLS

APROXIMATION OF THE TRAINING DATA



POLYNOMIAL REGRESSION (DEGREE=5) VS. REGULAR OLS

TRAINING AND TESTING DATA PERFORMANCE

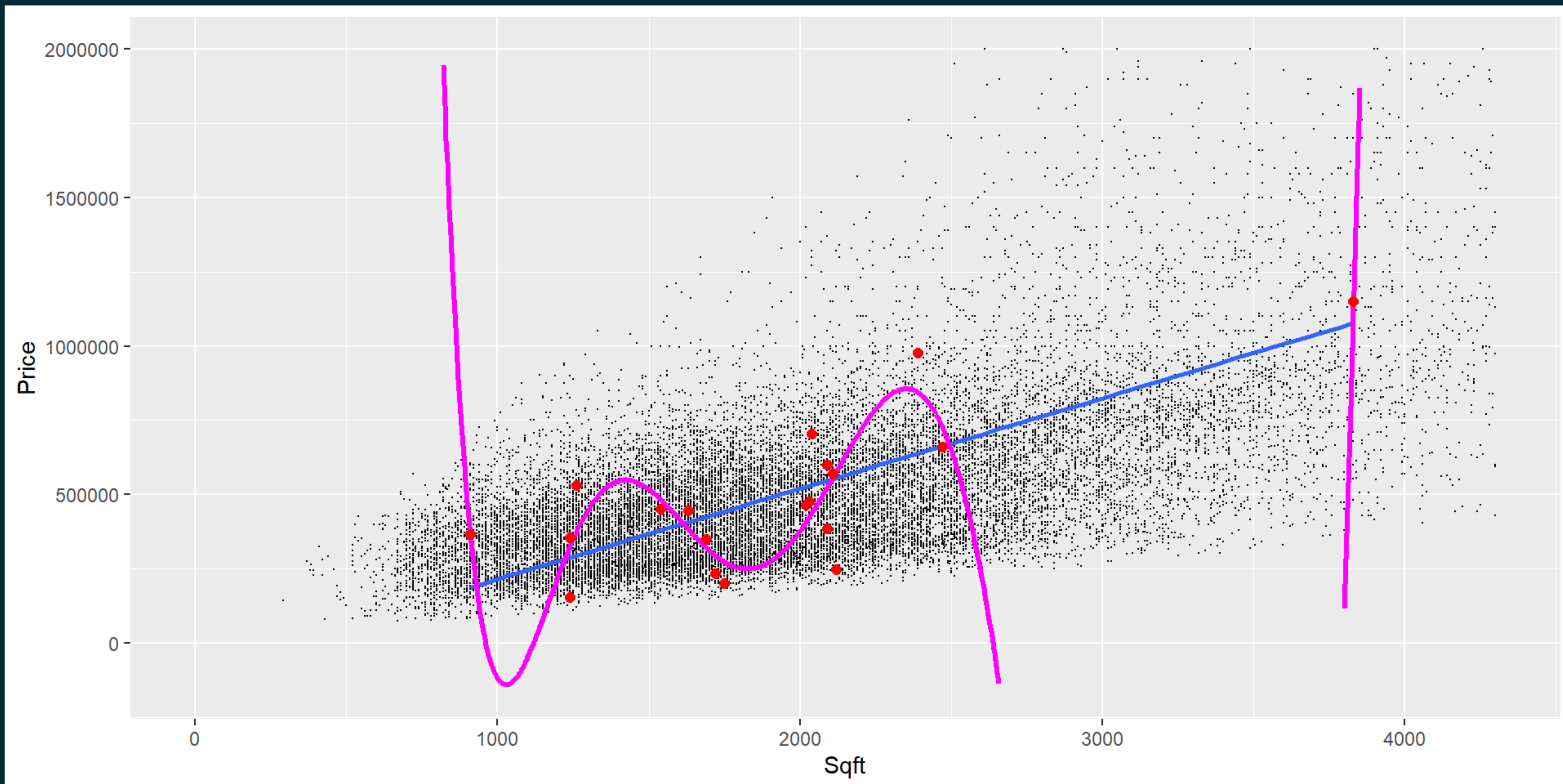


$$\widehat{Price} = \beta_1 Sqft + \beta_2 Sqft^2 + \beta_3 Sqft^3 + +\beta_4 Sqft^4 + \beta_5 Sqft^5 + \beta_6$$

https://ai.lange-analytics.com/

POLYNOMIAL REGRESSION (DEGREE=10) VS. REGULAR OLS

TRAINING AND TESTING DATA PERFORMANCE



$$\widehat{Price} = \beta_1 Sqft + \beta_2 Sqft^2 + \beta_3 Sqft^3 + \cdots + \beta_{10} Sqft^{10} + \beta_{11}$$

https://ai.lange-analytics.com/

SUMMARY: POLYNOMIAL REGRESSION

- If we do not have enough data polynomial regression with a high degree might lead to overfitting
- What is the right degree?
- We could try different degrees (e.g., 2, 3, 4, … 10) and see which model performs best.
- Which data are we using to measure performance? Training data (overfitting) and testing data (cannot be used for model optimization) are out.
- We could split off data from the training dataset (**validation data**). These *validation data* are not used to calculate the βs. Instead, they are used to find the best setting for the degree of polynomial regression (aka hyper-parameter of polynomial regression).

HYPER-PARAMETERS

- Hyper-Parameters are parameters other than the β parameters, because they can not be optimized by the optimizer.
- Hyper-Parameters are like settings for a machine learning model such as the number of polynomials (e.g., $Sqft^N$) to be considered for polynomial regression. Another example are the number of k Nearest Neighbors.
- Hyper parameters often make a model more or less complex and thus influence the quality of predicting but also the chance of overfitting.»

PROBLEMS OF SPLITTING VALIDATION DATA OFF THE TRAINING DATA

- Reduces data left over to train (finding optimal βs).
- If the training dataset is big enough this is no problem. Otherwise, it is a problem!

CROSS VALIDATION (4-FOLD)

For each hyper-parameter setting:

1. Splits off validation data from training data (e.g. last quarter)
2. Runs the model and calculates metrics based on validation data.
3. Splits off validation data from training data (next quarter)
4. Repeats steps 2 – 3 four times.

We end up with four results for each hyper-parameter setting. We calculate the average of the four results as an result for that specific hyper parameter.

CROSS VALIDATION FOR POLYNOMIAL REGRESSION AND THE KING COUNTY REALESTATE DATASET

MORE REALISTIC DATASPLIT: 80% TRAINING, 20% TESTING

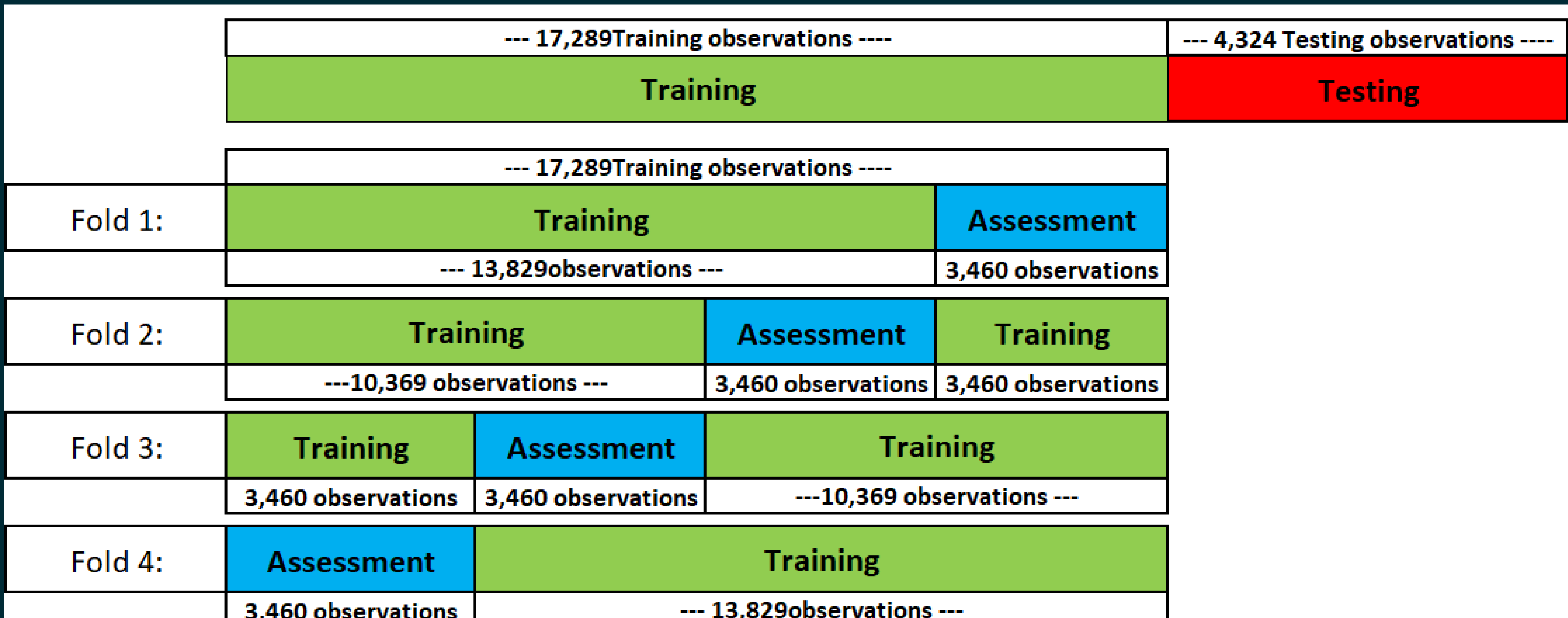
```
set.seed(987)

Split80=DataHousing %>%
  initial_split(prop = 0.8, strata = Price, breaks = 5)
DataTrain=training(Split80)
DataTest=testing(Split80)

print(Split80)
```

```
<Training/Testing/Total>
<17289/4324/21613>
```

CROSSVALIDATION – THE IDEA BEHIND IT



10 STEPS TO CREATE A MODEL, TUNE IT, AND PREDICT

The **10 general steps** are:

1. Generating **training and testing data** with `initial_split()`, `training()`, `testing()`.
2. Create **recipe** to determine predictor and outcome variables. Optionally add one or more `step_X()` commands.
3. Create **model design** and mark parameters to be tuned (use `tune()`) without `fit()`
4. Create **workflow** by `add_recipe()` and `add_model()`
5. Create a **hyper-parameter grid** containing the hyper-parameter combinations to be validated.
6. Create **cross validation datasets** (aka *resamples*) containing the folds (use command `vfold()`).
7. **Tune** the machine learning model with `tune_grid()` and track specific metrics defined by `metric_set()`. **Runs all hyper-parameter combinations for all folds.**
8. **Extract the best hyper-parameter combination** from the tuning results based on selected metrics (use `select_best()`)
9. **Finalize the model** by training it with the full set of training data with the best hyper-parameter combination (see `finalize_workflow() %>% fit()`).
10. **Assessing predictive quality** of the final model by using the testing dataset to predict (see `augment() %>% metrics()`).

RUN ALL 10 STEPS TO TUNE THE REAL ESTATE MODEL 🤓

Code to run all 10 steps is linked in the footer of this slide.

EXERCISE FROM AIBOOK 🤓

Use k-Nearest Neighbors to estimate the color of a wine

Click the link in the footer of this slide to start the exercise.

RESEARCH PROJECT 🤓

Click the link in the footer of this slide to download a skeleton of the R script for the research project.