Overfitting, Regularization, and Model Selection

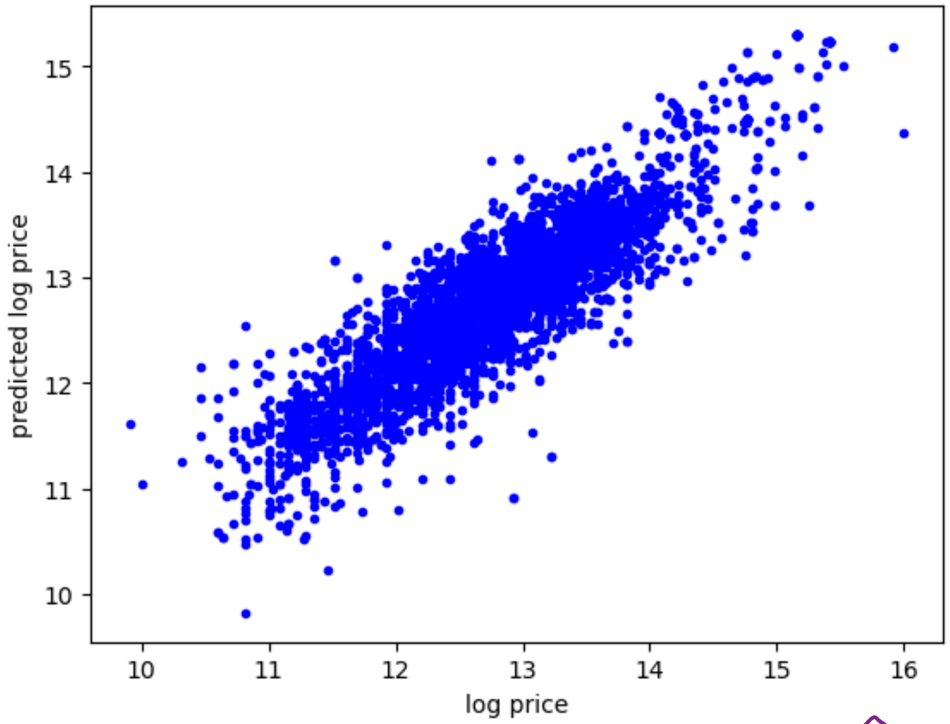
DUKE
FUQUA
SCHOOL OF BUSINESS

Quick recap of homework 1

- CarDekho multiple regression results:

One way to visualize multiple linear regression: plot predictions against real target values:

$\hat{Y}_i$



Y_i

$Y_i - \hat{Y}_i \sim N(0, \sigma^2)$

	coef	std err	t	P>|t|	[0.025	0.975]
const	12.4959	0.059	212.070	0.000	12.380	12.611
year	0.1178	0.002	66.227	0.000	0.114	0.121
km_driven	-0.0008	0.000	-5.354	0.000	-0.001	-0.001
fuel_Diesel	0.3954	0.015	27.002	0.000	0.367	0.424
fuel_Other	-0.1087	0.051	-2.148	0.032	-0.208	-0.009
seller_type_Individual	-0.1047	0.016	-6.728	0.000	-0.135	-0.074
seller_type_Trustmark Dealer	0.3715	0.042	8.849	0.000	0.289	0.454
owner_Other	-0.0820	0.042	-1.963	0.050	-0.164	-0.000
owner_Second Owner	-0.0328	0.015	-2.119	0.034	-0.063	-0.002
owner_Third Owner	-0.0979	0.026	-3.798	0.000	-0.148	-0.047
brand_BMW	0.1236	0.082	1.513	0.130	-0.037	0.284
brand_Chevrolet	-1.7207	0.060	-28.876	0.000	-1.838	-1.604
brand_Datsun	-1.7509	0.084	-20.753	0.000	-1.916	-1.585
brand_Fiat	-1.6262	0.083	-19.481	0.000	-1.790	-1.463
brand_Ford	-1.3177	0.058	-22.858	0.000	-1.431	-1.205
brand_Honda	-1.1191	0.058	-19.433	0.000	-1.232	-1.006
brand_Hyundai	-1.3317	0.054	-24.594	0.000	-1.438	-1.226
brand_Mahindra	-1.1678	0.056	-20.824	0.000	-1.278	-1.058
brand_Maruti	-1.4619	0.054	-27.191	0.000	-1.567	-1.357
brand_Mercedes-Benz	0.2986	0.084	3.540	0.000	0.133	0.464
brand_Nissan	-1.3616	0.072	-18.971	0.000	-1.502	-1.221
brand_Other	-0.2968	0.084	-3.536	0.000	-0.461	-0.132
brand_Renault	-1.5154	0.062	-24.615	0.000	-1.636	-1.395
brand_Skoda	-1.0981	0.070	-15.577	0.000	-1.236	-0.960
brand_Tata	-1.8146	0.056	-32.224	0.000	-1.925	-1.704
brand_Toyota	-0.7292	0.059	-12.350	0.000	-0.845	-0.613
brand_Volkswagen	-1.2389	0.064	-19.276	0.000	-1.365	-1.113

Baseline is Audi

Some fundamental principles

1. Although every problem differs, there is a systematic process for using data to help make better decisions.
2. All data contains structure, but we need to distinguish between the “signal” and the “noise.”
3. The specifics of the problem we are trying to solve should govern the choice of solution techniques, not the other way around.
4. Data and data analytics (or data science) capabilities are strategic assets, and firms need to carefully consider their investments in these assets.

Overfitting and the Super Bowl

- From Nate Silver's *The Signal and the Noise*:

  A once-famous "leading indicator" of economic performance, for instance, was the winner of the Super Bowl. From Super Bowl I in 1967 through Super Bowl XXXI in 1997, the stock market gained an average of 14 percent for the rest of the year when a team from the original National Football League (NFL) won the game. But it fell by almost 10 percent when a team from the original American Football League (AFL) won instead. ***Through 1997, this indicator had correctly "predicted" the direction of the stock market in twenty-eight of thirty-one years. A standard test of statistical significance, if taken literally, would have implied that there was only about a 1-in-4,700,000 possibility that the relationship had emerged from chance alone.***

- Related Washington Post [article]

Quotes on overfitting

First, don't tell your data analysts to figure out what is affecting sales. "The way most analyses go haywire is the manager hasn't narrowed the focus on what he or she is looking for," says Redman. It's your job to identify the factors that you suspect are having an impact and ask your analyst to look at those. "If you tell a data scientist to go on a fishing expedition, or to tell you something you don't know, then you deserve what you get, which is bad analysis," he says. In other words, don't ask your analysts to look at every variable they can possibly get their hands on all at once. If you do, you'll probably find relationships that don't really exist. It's the same principle as flipping a coin: Do it enough times and you'll eventually think you see something interesting, like a bunch of heads all in a row.

- Amy Gallo, HBR, quoting Thomas C. Redman

The act of 'testing on the training set' is anathema in machine learning, the greatest sin you can possibly commit.

- Yann LeCun, Chief AI Scientist at Meta

"The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data."

- John Tukey, famous statistician

If you torture the data long enough, it will confess.

- Ronald Coase

Revisiting the targeted marketing example (Class 1)

Your company offers subscription-based services to customers in a market with many competitors, and you are concerned about retaining customers at the expiration of their contracts. You are considering offering incentives (e.g., discounts on service fees) to some customers near the end of their contracts to retain them. To which customers should you offer incentives?

You ask your data science team to build a model based on historical data. A few days later, they come back later with a model and report it has 100% accuracy on this historical data – the model correctly identifies every customer who renewed their subscription and every one who did not!

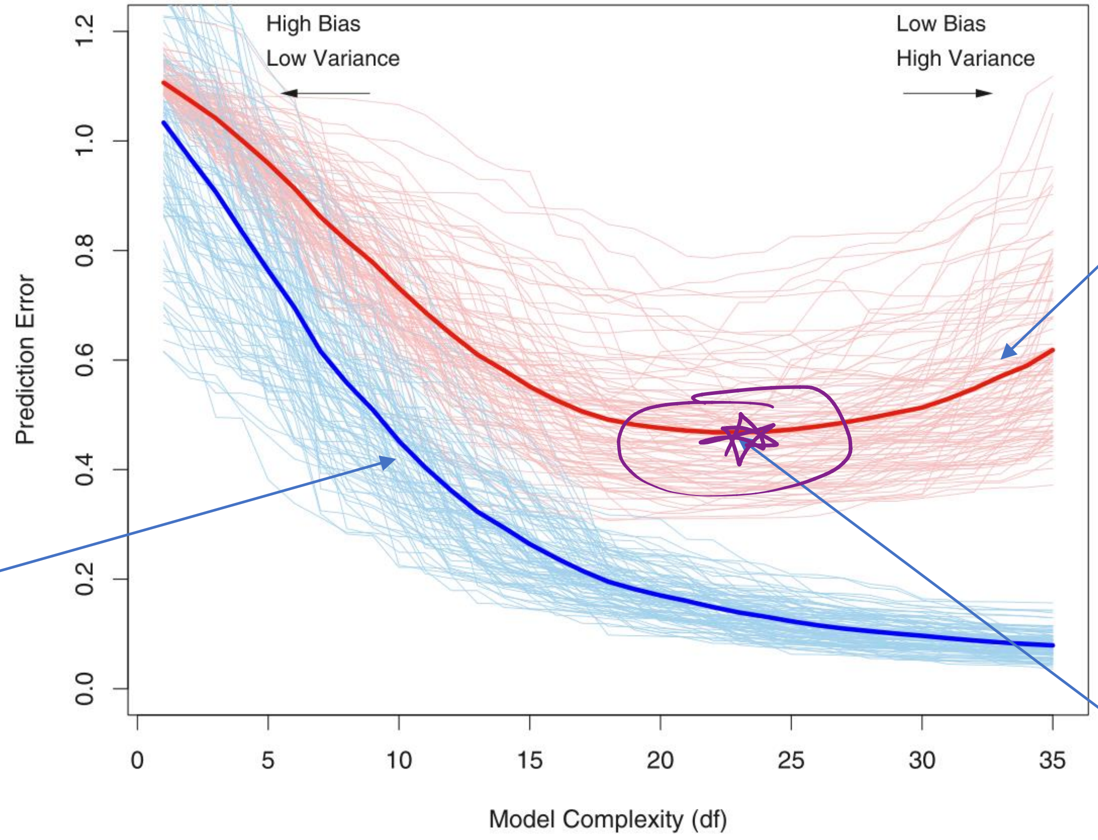
X^+ = set of training data features for customers who renewed

How would you respond?

Predict on new feature x: if x in $X^+ \rightarrow$ renew
if not $\rightarrow$ not renew

“Fitting graphs” illustrate model performance

Performance on test data (or “holdout” or “validation” data)



Performance on training data

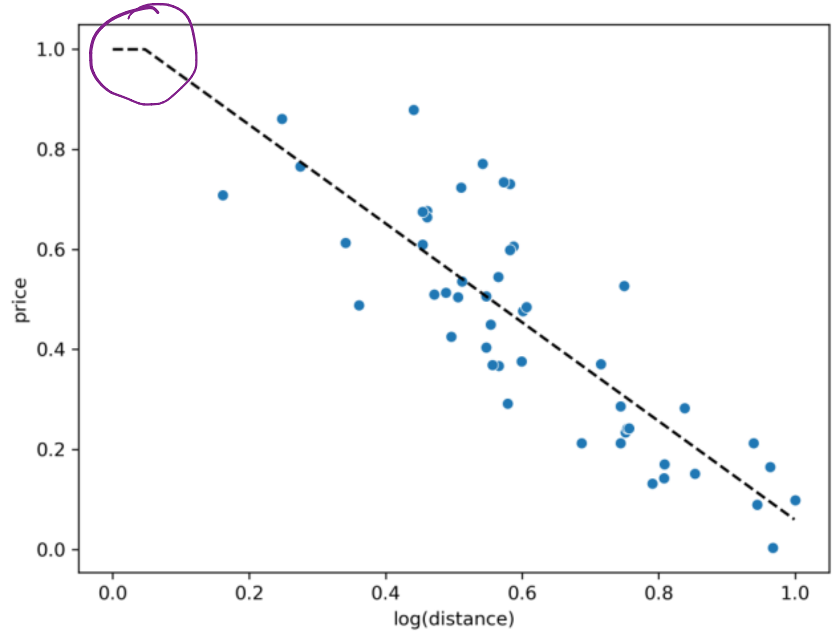
Source: Hastie et al. (2017), *The Elements of Statistical Learning*

Optimal performance (“sweet spot”)

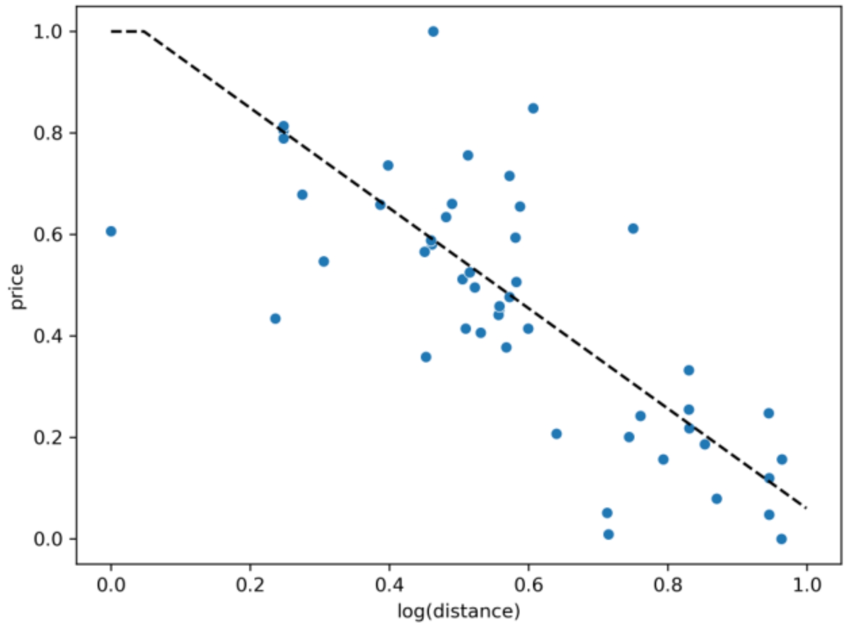
“Complexity” in regression models $\approx$ number of feature variables (or size of their coefficients)

Revisiting the New Taipei City real estate data

- We used polynomial regression on the (log) distance to illustrate the impact of using more feature variables in regression models; the degree of the polynomial controlled the complexity of the model
- Linear regression (degree = 1) results:



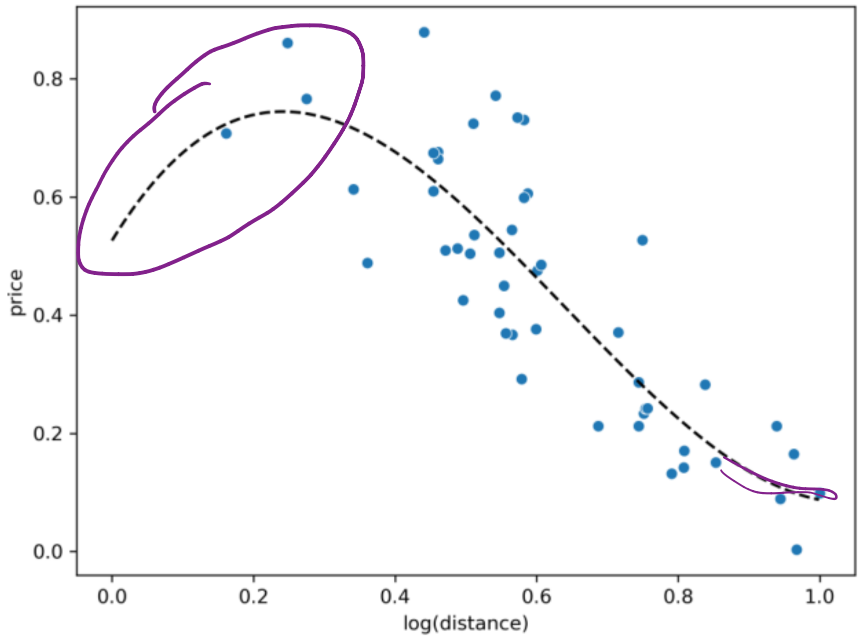
Performance on training data: MSE = 0.014
(1-norm of coefficients = 0.988)

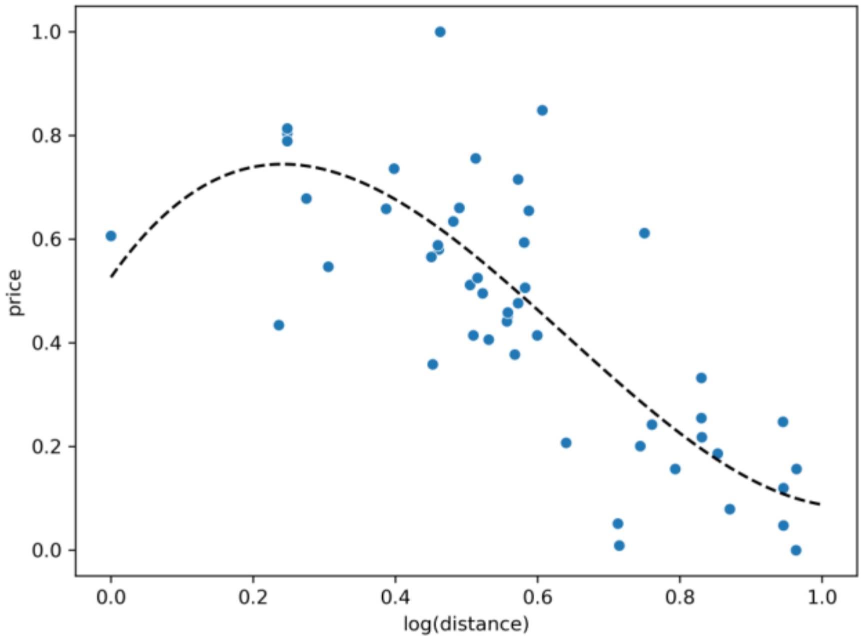


Performance on test data: MSE = 0.028

New Taipei City real estate data (cont.)

- Cubic regression (degree = 3) results:



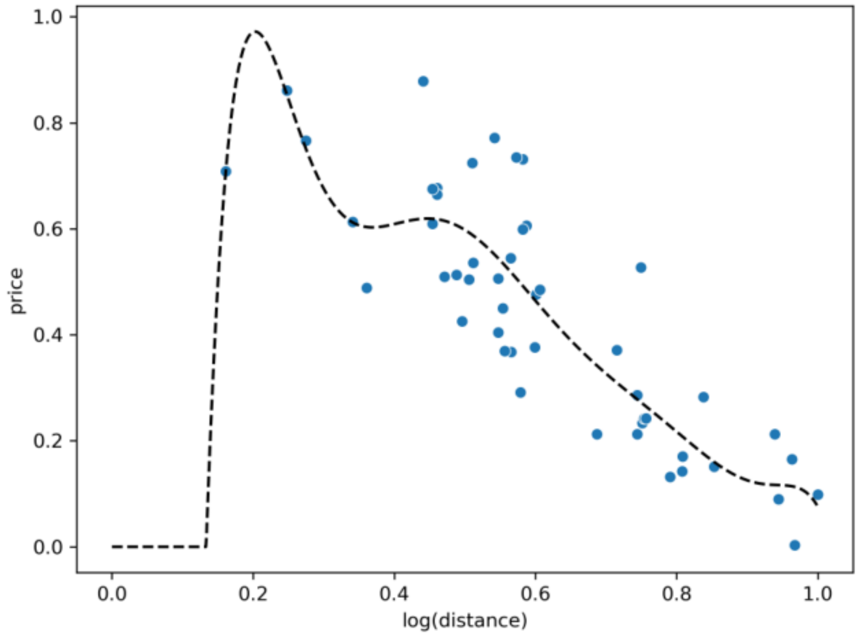


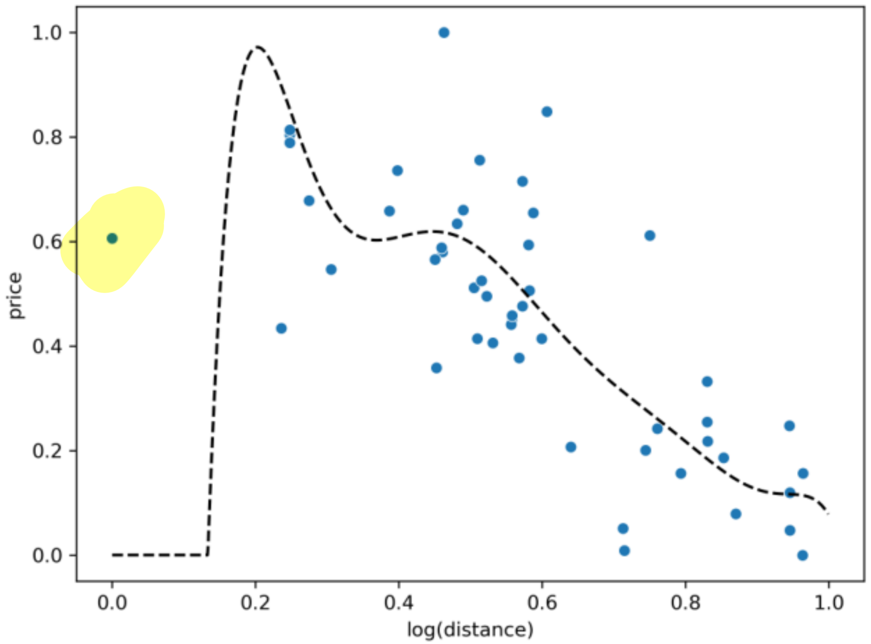
Performance on training data: MSE = 0.013
(1-norm of coefficients = 9.581)

Performance on test data: MSE = 0.023

New Taipei City real estate data (cont.)

- ~~Cubic~~ regression (degree = 8) results:





Performance on training data: MSE = 0.013
(1-norm of coefficients = 160,025.90)

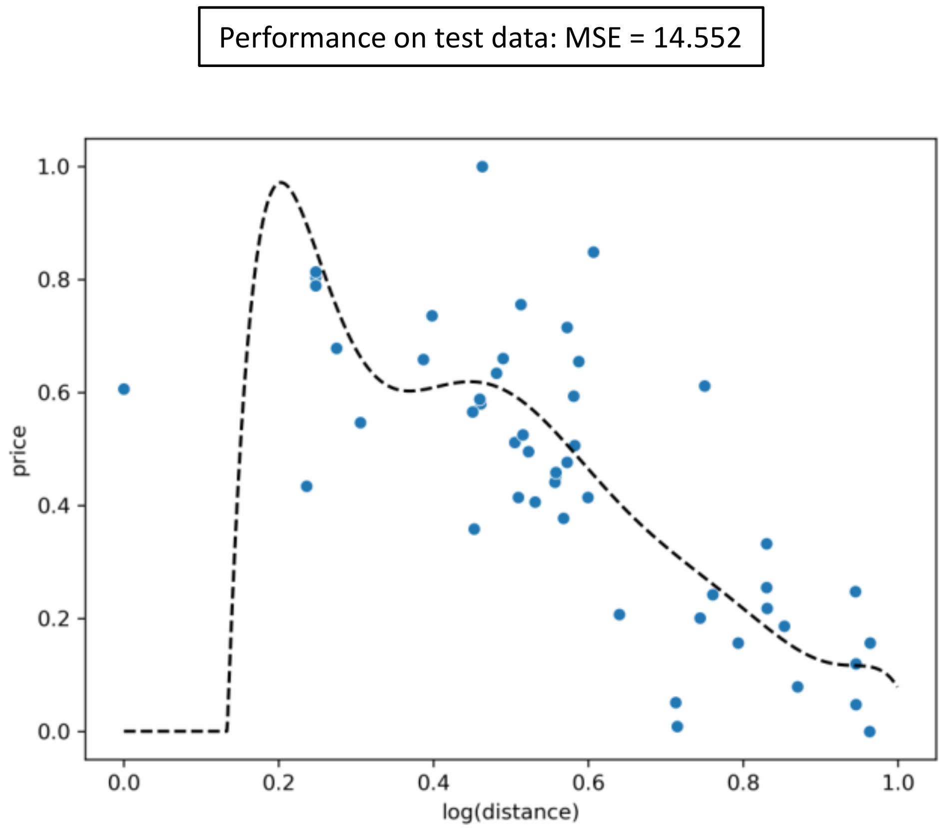
Performance on test data: MSE = 14.552

```
estimator = fit[3]
estimator.coef_
```

```
array([   490.98992062,  -3599.54986736,  14127.67683894, -32829.10374429,
        46630.28698115, -39807.96148164,  18777.20432675,  -3763.12260129])
```

- *Why did the test performance get so much worse moving from degree 3 to 8?*

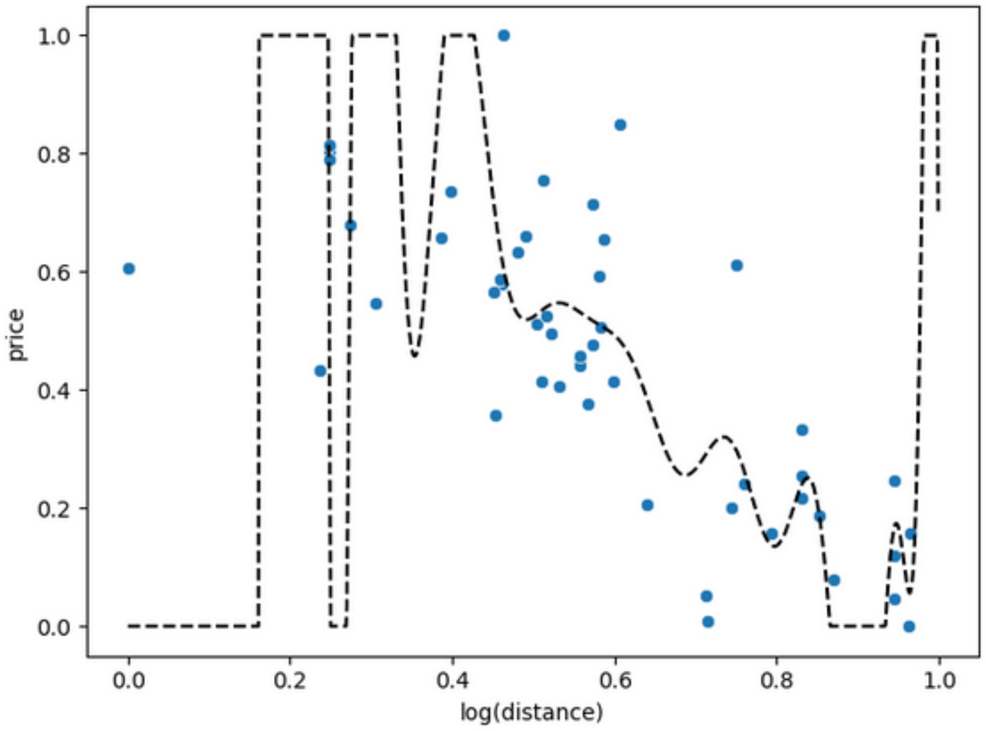
Inspecting the degree 8 predictions




```
In [20]: estimator = fit[3]
         polynomial_features = fit[0]
         X_polynomial = polynomial_features.transform(X_test)
         predictions = estimator.predict(X_polynomial)

In [21]: predictions

Out[21]: array([  0.14512601,    0.52925796,    0.85383026,    0.53111837,
                  0.50711615,    0.61791809,    0.27863581,    0.50711615,
                  0.57615062,    0.89797572,    0.5939551 ,    0.11420244,
                  0.11638016,    0.61063632,    0.61909975,    0.58347773,
                  0.46588266,    0.18340138,    0.52925796,    0.40449572,
                  0.31013263,    0.27205959,    0.58671628,    0.51455135,
                  0.49236387,    0.48372929,    0.11638016,    0.85383026,
                  0.22464081,    0.56606882,    0.85383026,    0.45482545,
                  0.60565864,    0.49465918,    0.31265152,    0.18431851,
                  0.60508555,    0.6640749 ,    0.11638604,    0.61743187,
                  0.11429432,    0.60813895,    0.61811973,    0.75171992,
                  0.61900552,  -26.34450874,    0.58957213,    0.26096887,
                  0.18424133,    0.1602845 ])

In [22]: X_test.values.reshape(1,-1)

Out[22]: array([[0.86985369, 0.55777425, 0.24765499, 0.55650091, 0.57250431,
                 0.46049634, 0.74383532, 0.57250431, 0.52216268, 0.23600512,
                 0.50459612, 0.96366422, 0.94501906, 0.48102254, 0.4505435 ,
                 0.5154241 , 0.59895326, 0.83062182, 0.55777425, 0.63954109,
                 0.71479793, 0.74996192, 0.51225369, 0.567631  , 0.58203439,
                 0.58756059, 0.94501906, 0.24765499, 0.79341008, 0.5306937 ,
                 0.24765499, 0.60604237, 0.48948491, 0.58056053, 0.71252798,
                 0.8297739 , 0.38693624, 0.30501537, 0.94490256, 0.46280568,
                 0.96328589, 0.39775226, 0.45940336, 0.27461717, 0.45216279,
                 0.        , 0.50933739, 0.76026631, 0.82984517, 0.85302838]])
```

Inspecting the degree 21 predictions

Performance on test data: MSE = 370,567,850,890




```
In [28]: predictions

Out[28]: array([-8.14559879e-02,  5.30955511e-01,  8.62278777e-01,  5.32176214e-01,
                 5.17436224e-01,  6.26933294e-01,  3.12464911e-01,  5.17436224e-01,
                 5.44848639e-01,  1.25949181e+01,  5.28300482e-01,  5.72158871e-02,
                 1.66590887e-01,  5.27602392e-01,  7.21751410e-01,  5.39752203e-01,
                 4.93647772e-01,  2.38856512e-01,  5.30955511e-01,  3.87782293e-01,
                 2.95619207e-01,  2.98426825e-01,  5.36624151e-01,  5.21754461e-01,
                 5.09577948e-01,  5.05015570e-01,  1.66590887e-01,  8.62278777e-01,
                 1.37171942e-01,  5.47412115e-01,  8.62278777e-01,  4.83699041e-01,
                 5.19107061e-01,  5.10783392e-01,  2.90675360e-01,  2.36659246e-01,
                 9.51443869e-01,  2.48236604e+00,  1.67567450e-01,  6.09202581e-01,
                 5.81924496e-02,  1.08360787e+00,  6.35928351e-01,  7.67005879e-01,
                 7.04463202e-01, -4.30446133e+06,  5.33496100e-01,  2.61073309e-01,
                 2.36170965e-01,  1.81239325e-01])
```

```
In [29]: X_test.values.reshape(1,-1)

Out[29]: array([[0.86985369, 0.55777425, 0.24765499, 0.55650091, 0.57250431,
                 0.46049634, 0.74383532, 0.57250431, 0.52216268, 0.23600512,
                 0.50459612, 0.96366422, 0.94501906, 0.48102254, 0.4505435 ,
                 0.5154241 , 0.59895326, 0.83062182, 0.55777425, 0.63954109,
                 0.71479793, 0.74996192, 0.51225369, 0.567631  , 0.58203439,
                 0.58756059, 0.94501906, 0.24765499, 0.79341008, 0.5306937 ,
                 0.24765499, 0.60604237, 0.48948491, 0.58056053, 0.71252798,
                 0.8297739 , 0.38693624, 0.30501537, 0.94490256, 0.46280568,
                 0.96328589, 0.39775226, 0.45940336, 0.27461717, 0.45216279,
                 0.        , 0.50933739, 0.76026631, 0.82984517, 0.85302838]])
```

Complete craziness!

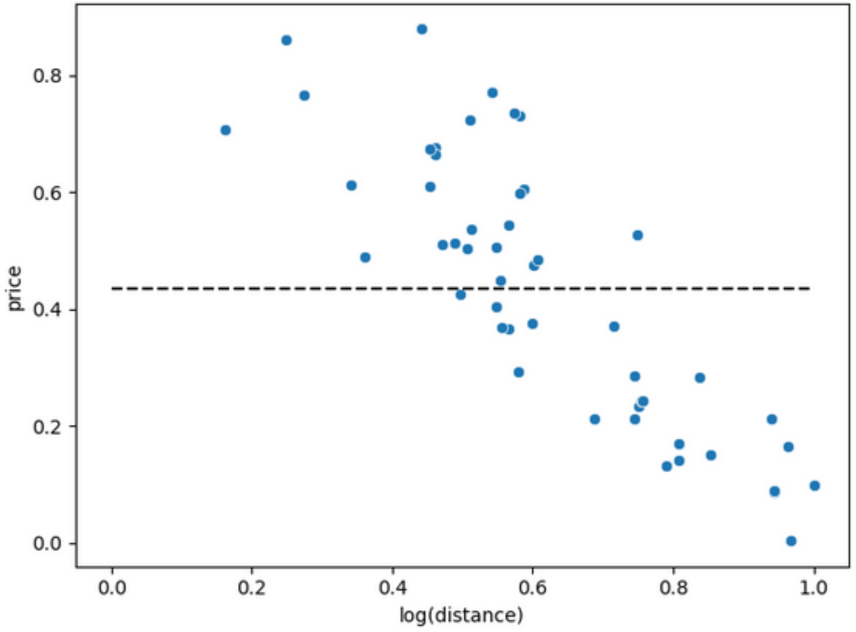
- Fix?

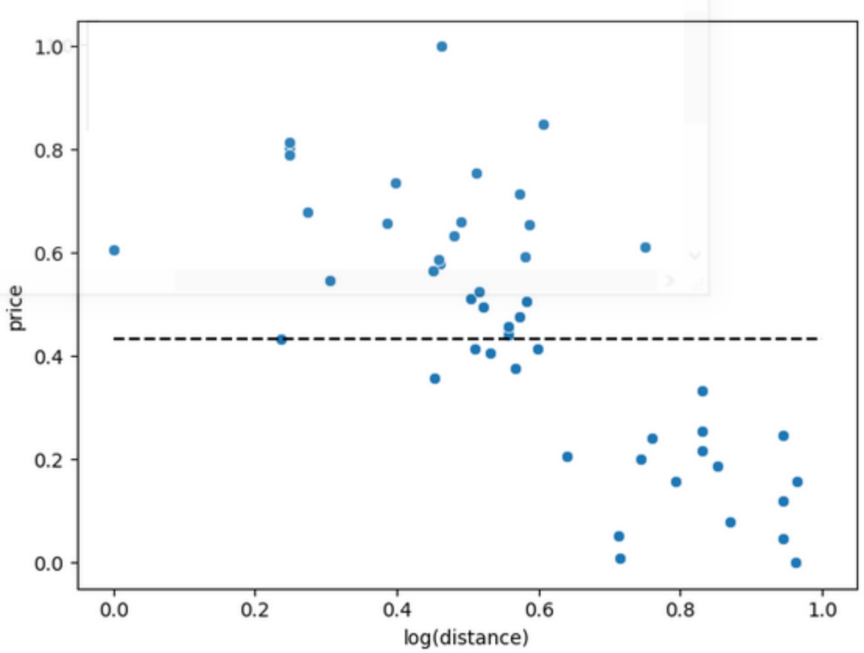
The degree 21 model with constraints

- 21 degree with coefficients >= 0 constraints added:

Degree 21

```
fit = fit_polynomial_regression(X_train, Y_train, degree=21, estimator_kwargs={'positive' : True})
```

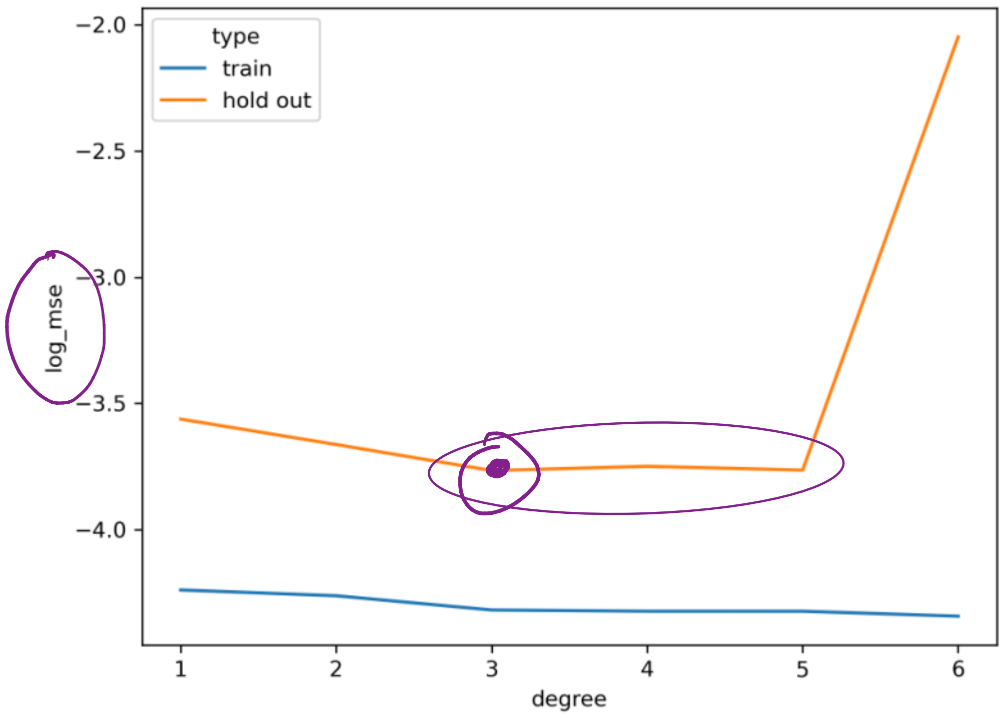




Performance on training data: MSE = 0.051
(1-norm of coefficients = 0.0)

Performance on test data: MSE = 0.059

New Taipei City real estate fitting graph



- Choose the model with the best performance on the test/hold out set
  - Occam's razor principle: if multiple models have the same performance on the test/hold out set, go with the simplest one

Regularization is a principled way to find the "sweet spot" in predictive models

- In ordinary linear regression, we choose the coefficients to minimize the mean squared error (MSE)

- In regularized regression we choose the coefficients to minimize:

$$\text{MSE} + \alpha \cdot \text{Penalty(Coefficients)}$$

Regularization strength (also often called λ)

- Penalty punishes coefficients for being too big; typical choices:
  - Ridge (or "L2" or "Tikhonov"): sum of squares of coefficients
  - Lasso (or "L1"): sum of absolute values of coefficients
- Ultimately, we take the model corresponding to the α that gives the best fit on hold out data

Ridge vs. Lasso

- Ridge regularization "pushes" all coefficients towards being smaller in magnitude
- Lasso regularization leads to many coefficients being zero ("sparse")

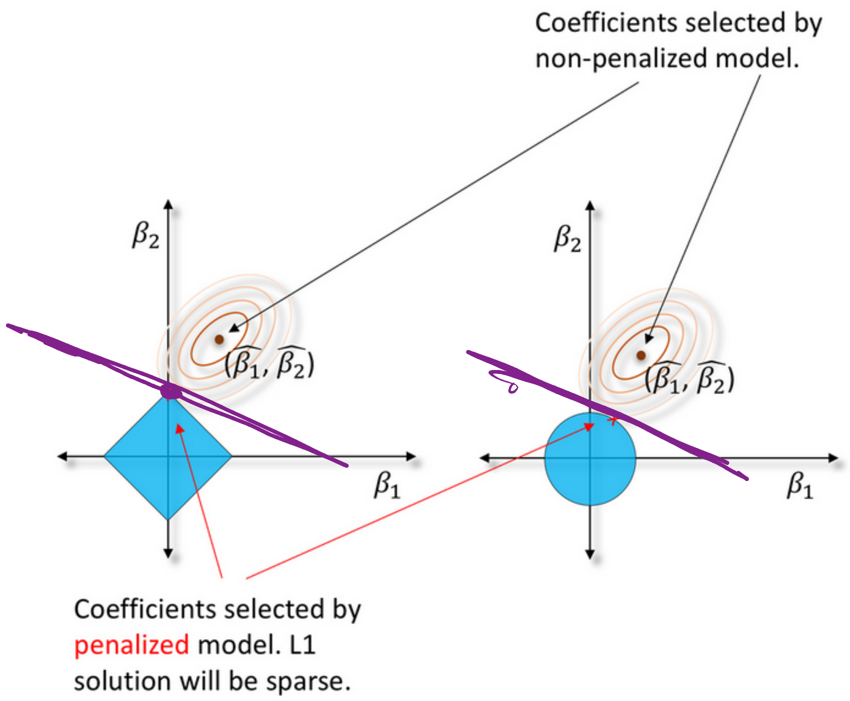
Ridge grid search results:

```
intercept: 0.4346572934973638
power 1 coefficient: -0.0129753549478808
power 2 coefficient: -0.08434943396805886
power 3 coefficient: -0.0851423769885607
power 4 coefficient: -0.10469223460010854
power 5 coefficient: -0.09366847362199286
power 6 coefficient: -0.04518720423782911
power 7 coefficient: 0.01826070818326474
power 8 coefficient: 0.07289582646401085
power 9 coefficient: 0.10486448766681324
power 10 coefficient: 0.11048929238255435
power 11 coefficient: 0.09298614102881175
power 12 coefficient: 0.05916306944812212
power 13 coefficient: 0.017084071073672723
power 14 coefficient: -0.025325007552068
power 15 coefficient: -0.06099981563023141
power 16 coefficient: -0.08404395397316082
power 17 coefficient: -0.08979736687220886
power 18 coefficient: -0.07477881652222561
power 19 coefficient: -0.03656845365621458
power 20 coefficient: 0.02633235061247559
power 21 coefficient: 0.11464387670477226
```

Lasso grid search results:

```
intercept: 0.4346572934973638
power 1 coefficient: -0.08938396133613612
power 2 coefficient: -0.08374627427768395
power 3 coefficient: -0.0
power 4 coefficient: -0.0
power 5 coefficient: -0.0
power 6 coefficient: -0.0
power 7 coefficient: -0.0
power 8 coefficient: -0.0
power 9 coefficient: -0.0
power 10 coefficient: -0.0
power 11 coefficient: -0.0
power 12 coefficient: -0.0
power 13 coefficient: -0.0
power 14 coefficient: -0.0
power 15 coefficient: -0.0
power 16 coefficient: -0.0
power 17 coefficient: -0.0
power 18 coefficient: -0.0
power 19 coefficient: -0.0
power 20 coefficient: 0.0
power 21 coefficient: 0.0
```

Visualization of the phenomenon:

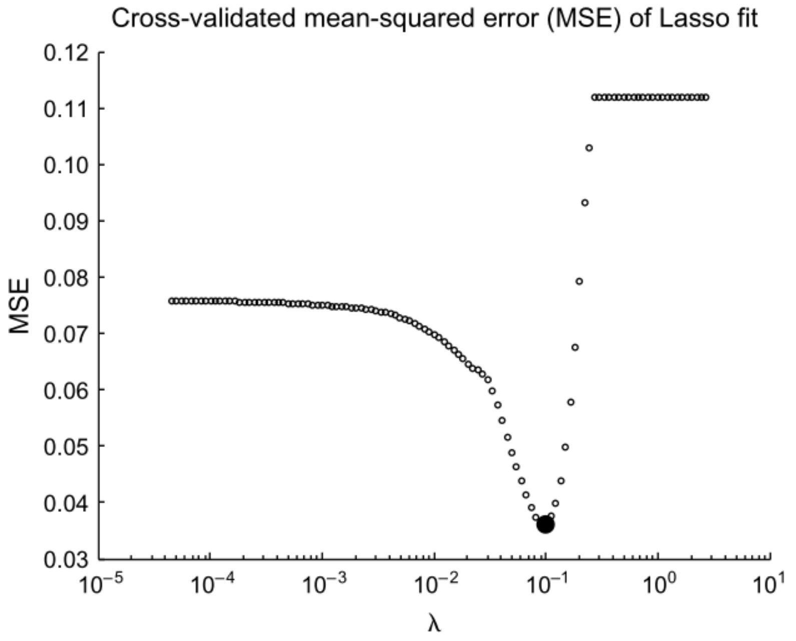


Source: Patrick Hall, Tomas Nykodym, and the H2O.ai team

Regularization is used widely!

- Example from estimating cross-product price elasticities:

Figure 1 An Example of the Result of Fivefold Cross-Validation



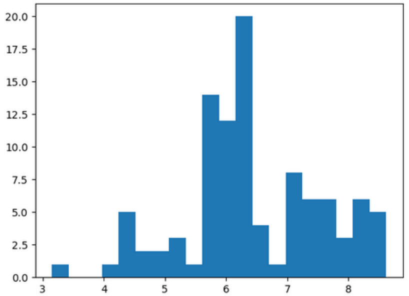
Notes. The value of λ highlighted with the large dot gives the lowest cross-validation error. Large values of λ (to the right) heavily penalize nonzero entries, resulting in the zero vector as the solution, which does not fit the data well. As λ is lowered, we begin to get some nonzero entries in the solution, providing a better fit of the data. However, as λ becomes even smaller, past the value marked with the large dot, we obtain dense solutions that tend to overfit, resulting in a higher cross-validation error.

Source: Li et al. (2015), The value of field experiments. *Management Science.*

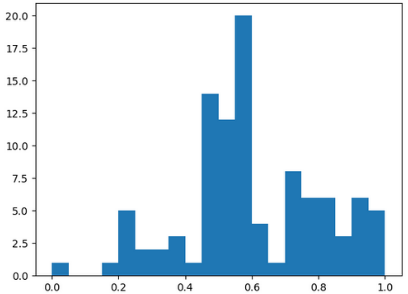
On normalizing and standardizing feature data

- *Normalizing*: rescale the feature to have minimum of 0 and maximum of 1
    - This is what `sklearn.preprocessing.MinMaxScaler` does .
- *Standardizing*: rescale the feature to have mean of 0 and standard deviation of 1
    - This is what `sklearn.preprocessing.StandardScaler` does.

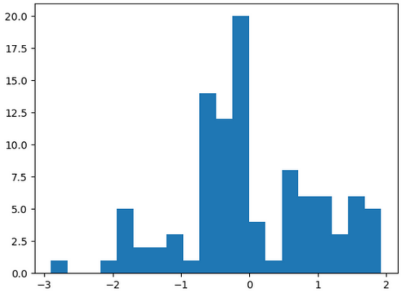
Log(distance) on New Taipei City example:



Unscaled data



Normalized data



Standardized data

- Why bother with this?
    1. This increases *numerical stability* of algorithms -> better performance
    2. This increases *interpretability* in that different feature coefficients are more comparable

- Which should you do? It depends – standardizing assumes data is (roughly) normally distributed. Discussion from Jason Brownlee (with Python examples) here.

Cross-validation is useful for selecting models

- Split the data into “test” and “training” sets and use cross-validation on the training set to find the best model parameter (e.g., regularization strength)

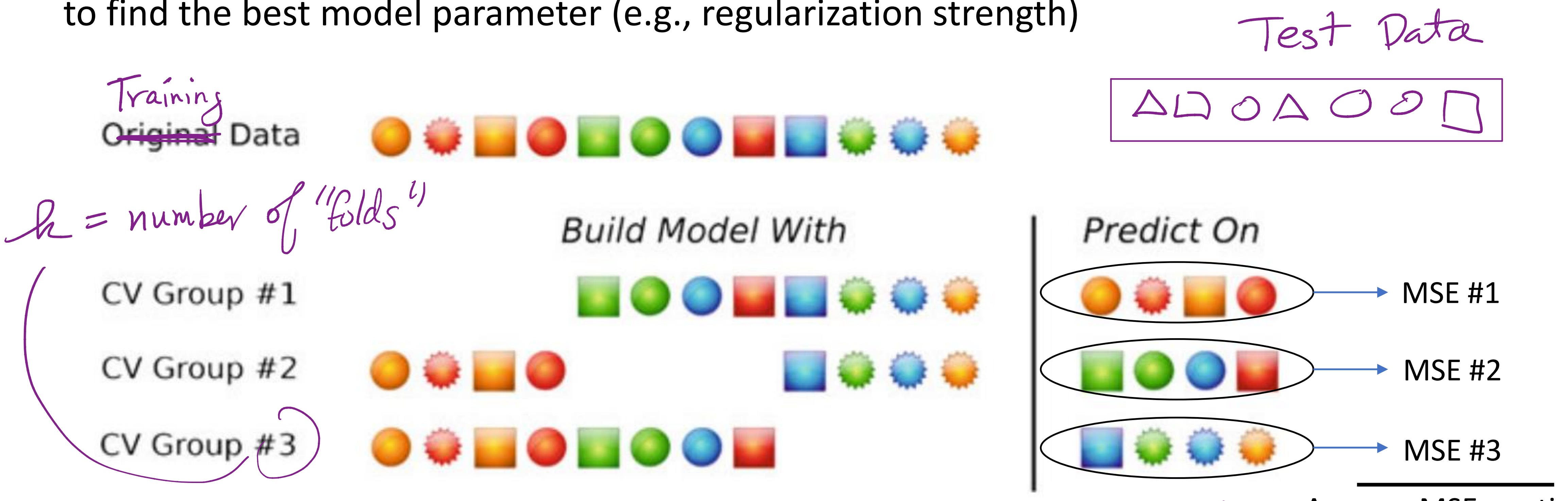


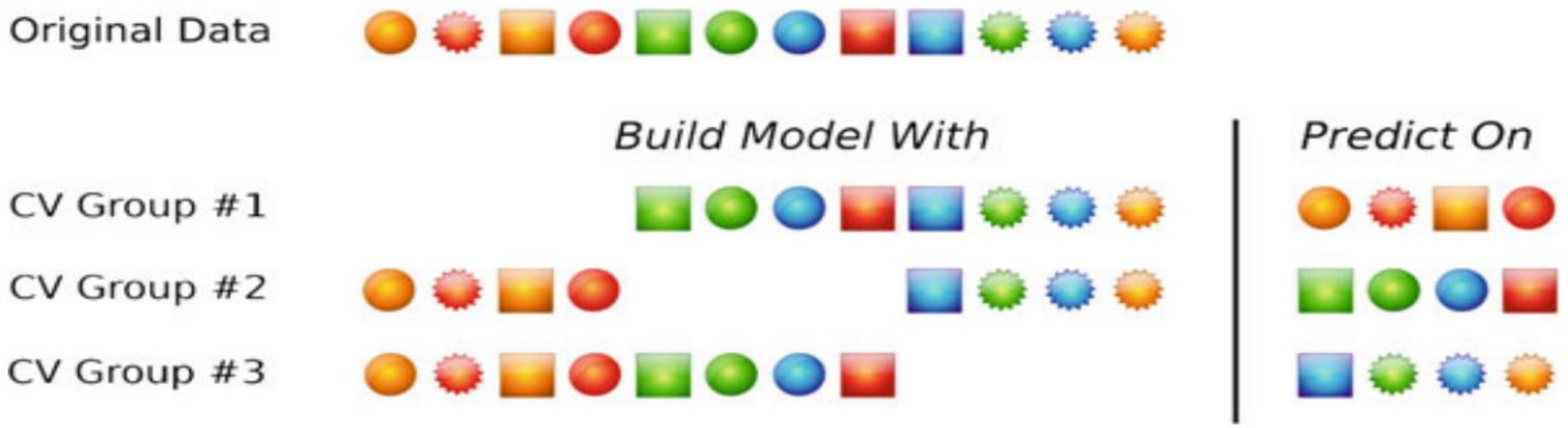
Illustration of 3-fold cross validation. *Source:* Kuhn and Johnson (2013)

For each α: Average MSE = estimate of model performance

- After CV, fit model with best parameter on full training set and evaluate on test set
- There are *many* variations of this idea (random resampling, “stratified” versions, nested versions, bootstrapping, …)

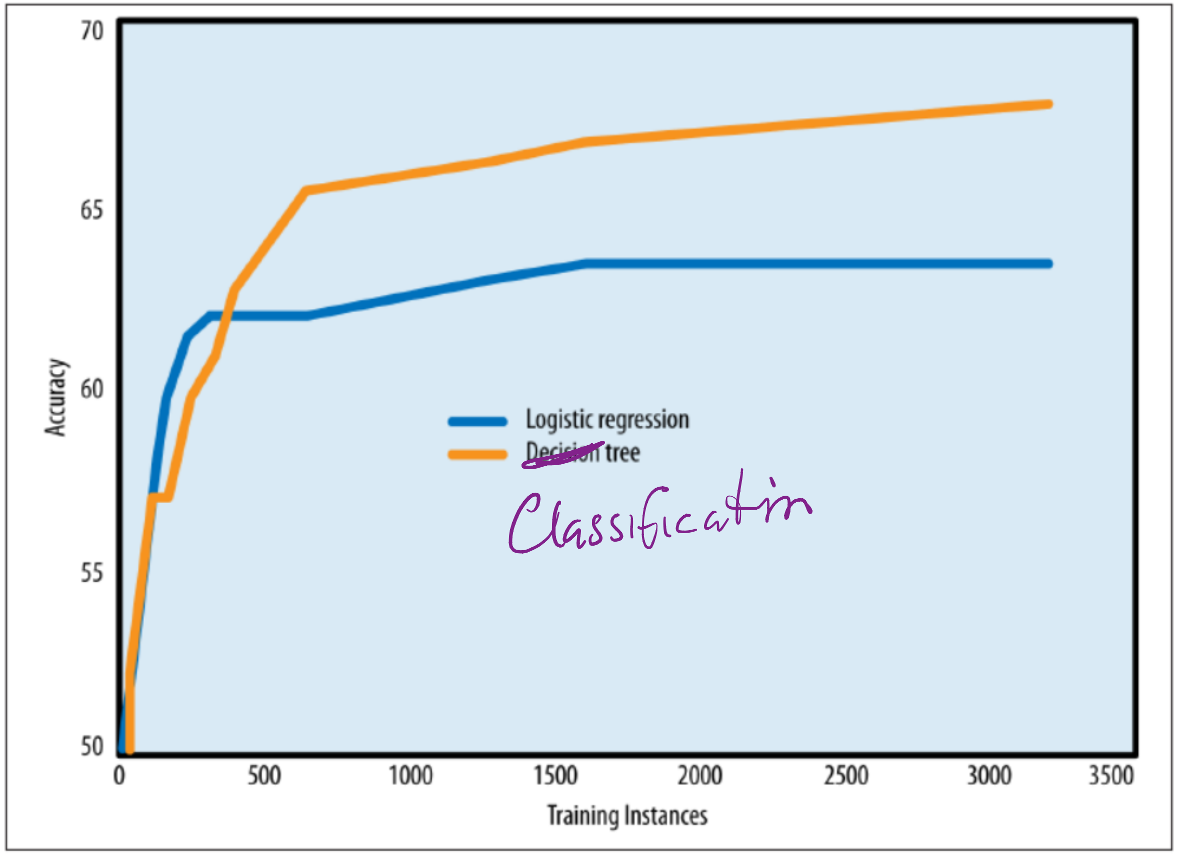
Choosing the k in k-fold cross validation

- There is usually an implicit (and subtle) bias-variance tradeoff in the choice of k:
  - Smaller k corresponds to smaller training folds and performance estimates are more biased
  - Larger k corresponds to more overlap in training folds and performance estimates vary more



- Practical advice:
  - The choice k = 10 is very common and many empirical studies suggest the range of k = 5 to 10 has robust performance
  - If data set is "small," the choice of k can matter… for large data sets, the choice of k is less sensitive

Learning curves display model performance as more training data becomes available



Performance on test data

Source: Provost and Fawcett (2013)

Size of training data

Summary

- Building predictive models that generalize well to new data is the core problem in data analytics!
- Tradeoff: more complex models allow us to capture more structure but are prone to capturing noise (overfitting)
- Regularization is a principled approach to designing regression models that balance this tradeoff
  - Ridge regression tends to make all coefficients smaller in magnitude
  - Lasso regression tends to make many coefficients equal to zero
- Cross-validation is a widely used method to perform model selection; we split the data up in different ways and fit to part of it and evaluate on the rest of it
  - Often used to select the strength of regularization, but can be used more broadly

Looking ahead to next time (Class 3)

- Homework 2 due at 11:59pm on Monday
  - Main goals: practice your understanding of regularization, Lasso, and model selection
  - TA support available over the weekend!

- Class 3:
  - Classification algorithms:
    - k-NN, Naïve Bayes, support vector machines
  - More on evaluating model performance

Finally … a joke for the weekend