Utrecht University

Model Fit and Cross Validation

Applied Data Analysis and Visualization I

Department of Methodology and Statistics
Javier Garcia-Bernardo, Ayoub Bagheri, Emmeke Aarts

Today

What	When
Data visualization	Week 2
Model fit and cross validation	Week 3
Linear regression for data science	Week 4
Classification	Week 5
Interactive visualizations with R shiny	Week 6
...	...

Main points for today

1. Why do we model data?
2. Which model is best?
   - Model performance
   - Estimating model performance using new observations
   - Bias-variance trade-off; underfitting vs overfitting
   - General paradigms: Training/validation/test & cross validation
3. Conclusion

Part 1: Why model?

What is a model?

A simplified representation of a (complex) system, usually: $Y = f(X) + \epsilon$

- Y: variable to predict (observed outcome)
- $f(X)$: a function of observed predictors (independent variables)
- ϵ is a random error with mean zero

Helps us *understand* and *predict* the behavior of a system

Statistical learning: learn f using data

Regression

Suppose we have $i = 1, \ldots, n$ observations:

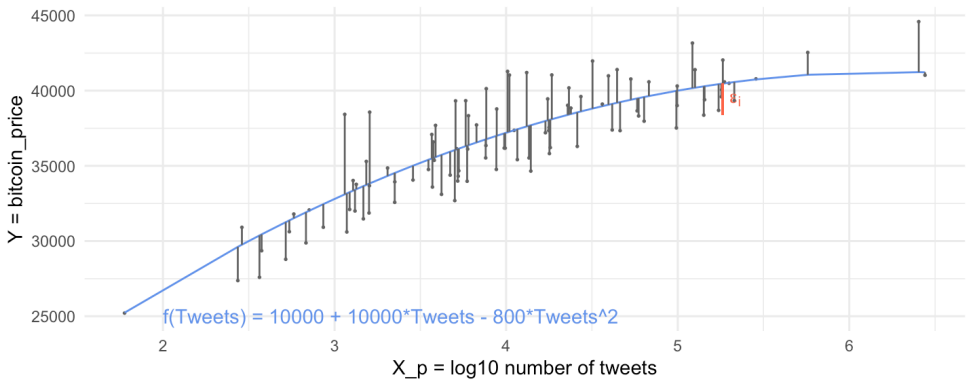
- Y_i: variable to predict (observed outcome)
- $X_i = (X_{i1}, \ldots, X_{ip})$: p predictors (independent variables)

We believe that there is a relationship between Y and at least one of the X's.

- We can model the relationship as $Y_i = f(X_i) + \epsilon_i$
- Where f is an unknown function and ϵ is a random error with mean zero.

$$y = f(x) + \epsilon$$

- y: Observed outcome – (e.g. price of bitcoin)
- x: Observed predictor(s) – (e.g. number of tweets about bitcoin)
- f(x): Prediction function, **unknown, to be estimated**
- ϵ: Unobserved residuals = **"irreducible error"**, $\epsilon = y - f(x)$
    - The higher the variance of ϵ (the more noise there is), the less we can predict.



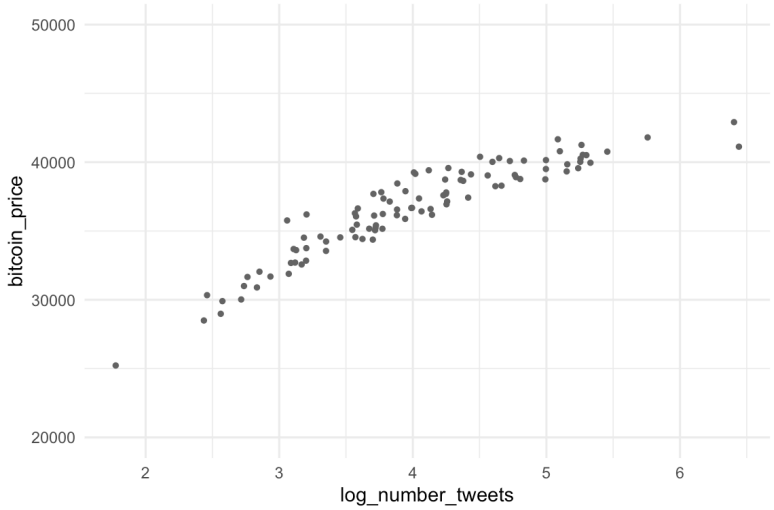
Why do we estimate f?

- The term statistical learning refers to using the data to “learn” f.
- There are two reasons for estimating f:
  - **Prediction**: If we can produce a good estimate for f (and the variance of ϵ is not too large) we can make accurate predictions for the response, Y, based on a new value of X.
  - **Inference**: Alternatively, we may also be interested in how the X’s (the predictors) affect the Y (the outcome)
    - Which particular predictors actually affect the response?
    - Is the relationship positive or negative?
    - Is the relationship a simple linear one or is it more complicated etc.?

How do we estimate f?

- We will assume we have observed a set of training data
  - $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$
- We use the **training data** and a **statistical method** to estimate f.
- e.g. $Price = f(Tweets) + \epsilon$
- Statistical Learning Methods:
  - **Parametric** Methods: Make some assumption about the functional form of f.
    - e.g. $f(Tweets) = \beta_0 + \beta_1 \cdot Tweets$
  - **Non-parametric** Methods: They do not make explicit assumptions about the functional form of f.
    - e.g. The price of bitcoin is the average of the 3 points in our dataset with the closest number of Tweets.

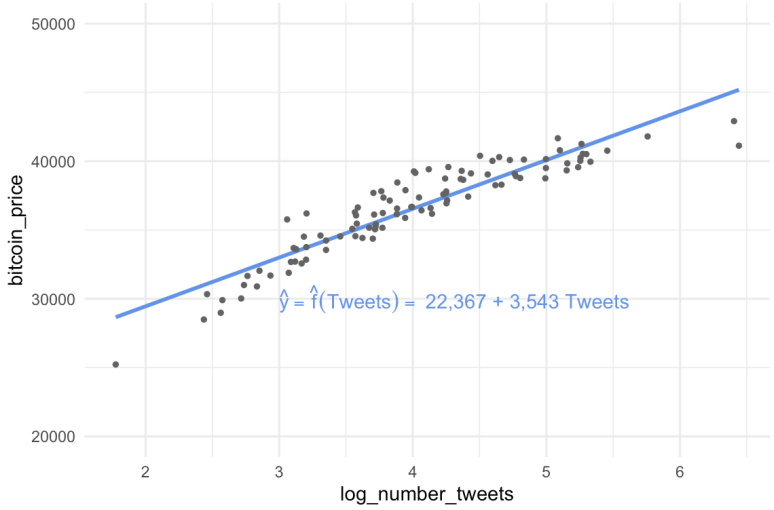
Estimation in example



Estimation in practice (parametric model)

$$y = f(x) + \epsilon$$

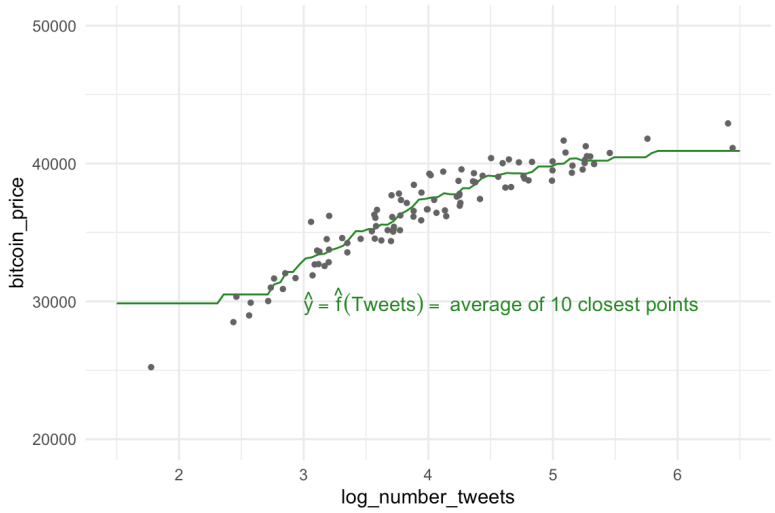
$$\hat{f}(Tweets) = \beta_0 + \beta_1 \cdot Tweets$$



Estimation in practice (non parametric model)

$\hat{f}(Tweets) = 1/k \sum_{k \in K} y_k,$

where K = closest k points to the evaluation point



Some types of models

Linear regression:

$$y_i = b_0 + b_1 x_i + \epsilon_i$$

Linear regression with quadratic term:

$$y_i = b_0 + b_1 x_i + b_2 x_i^2 + \epsilon_i$$

Linear regression higher-order polynomials (up to the power of 10):

$$y_i = b_0 + b_1 x_i + b_2 x_i^2 + \ldots + + b_{10} x_i^{10} + \epsilon_i$$

Nonparametric regression (LOESS):

y_i predicted from a “local” regression within a window of its nearest neighbors

Nonparametric regression (K-NN Regression):

y_i predicted from the value of the closest neighbors

What affects our ability to estimate f?

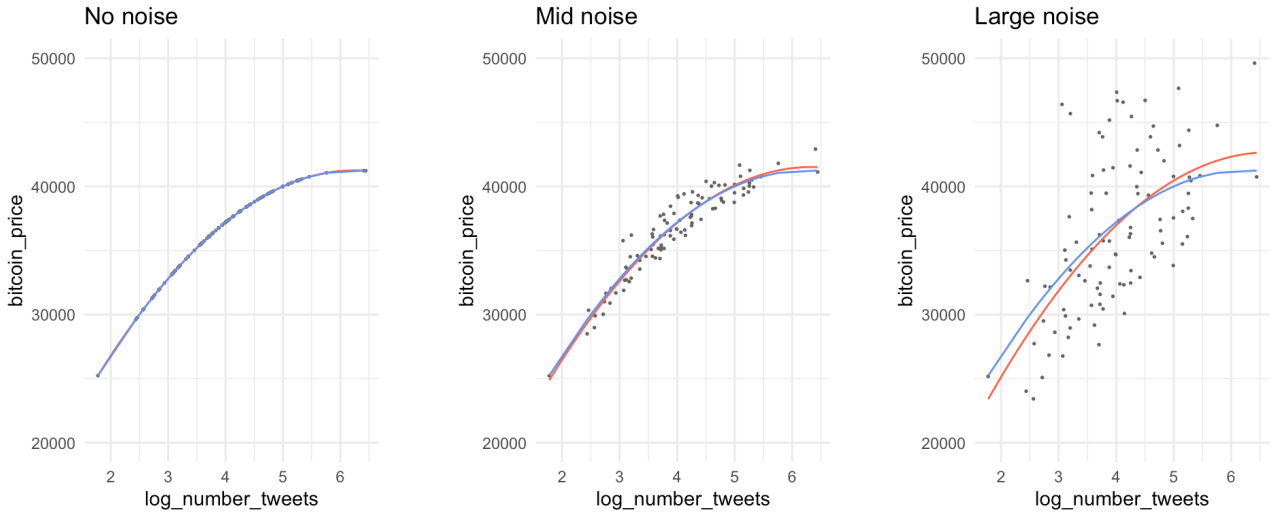
Irreducible error

Sample size

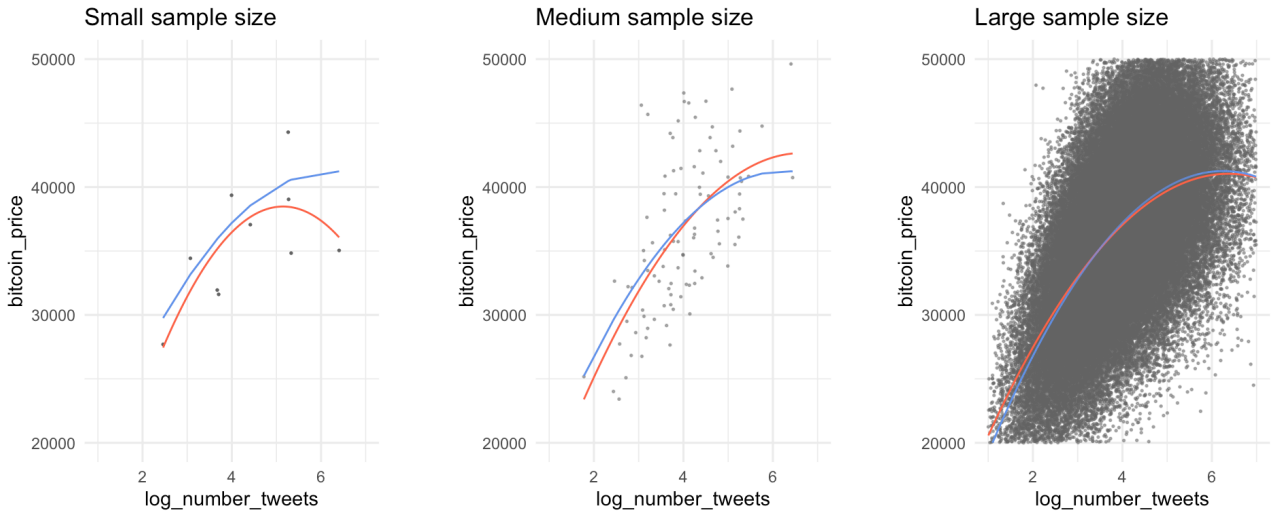
The balance between model & task complexity

The noise, $Var(\epsilon)$, affects our estimate of f

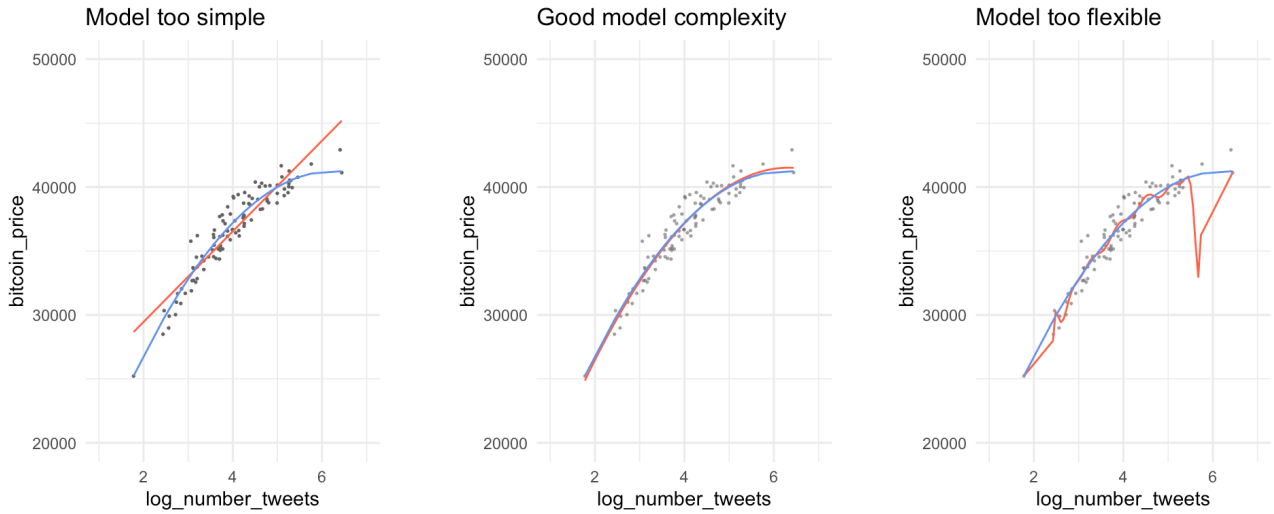
- The difficulty of estimating f will depend on the standard deviation of ϵ.
  - $\hat{f}$ (red); real f (blue, in practice unknown)



The sample size affects our estimate of f



The balance between model and task complexity

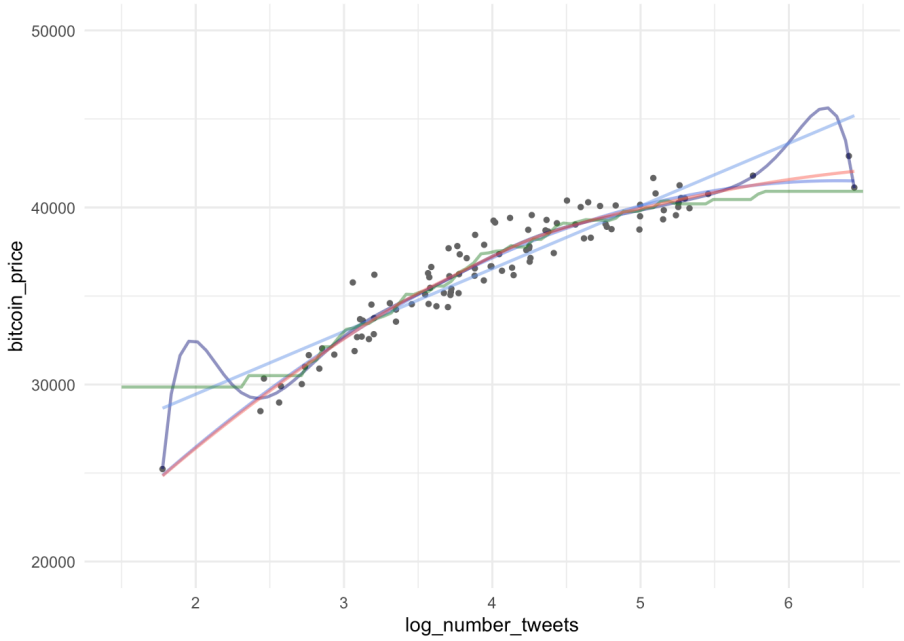


Which model ($\hat{f}$) is best? In real-life applications we do not know f (blue line)

To explore this interactively: javier.science/panel_bias_variance/

Model performance

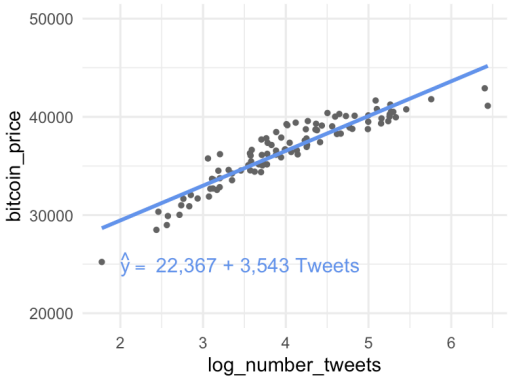
Which model fits better?

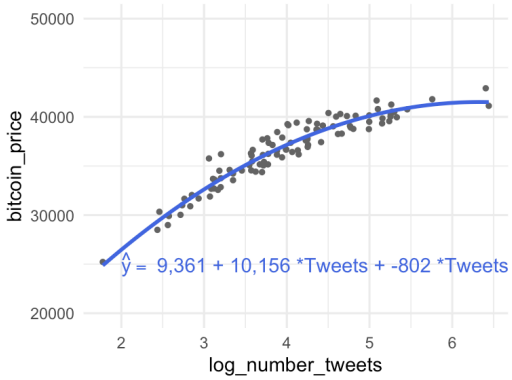


Model performance

- Often multiple methods available to describe/predict the same data, which one is best?
- When many predictors are available, which set of predictors is best?
- In more complex models, certain parameters have to be ‘tuned’. Which value for these tuned parameters is best?
- Answer: We can compare models using **model performance**

Which model is best: wooclap.com/ADAV2024





Why is best?

Left is underfitting (model too simple for the data)

How would you define model performance in terms of y (observed value of bitcoin price) and $\hat{y}$ (predicted value by the model)?

Model performance

Model performance:

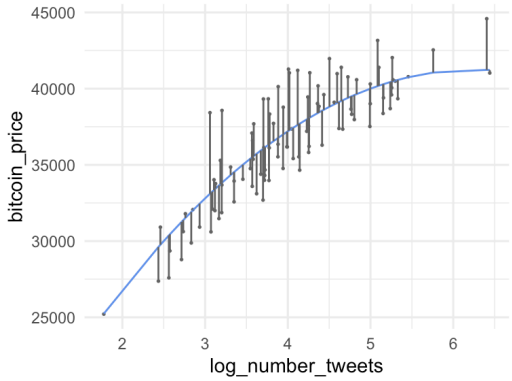
- The predictions $\hat{y} = \hat{f}(x)$ differ from the true y;
- We can evaluate how much this happens 'on average'.

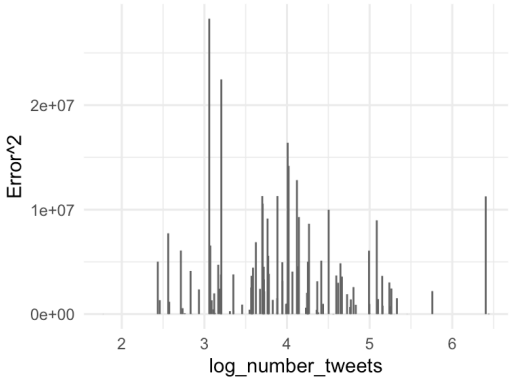
Different options:

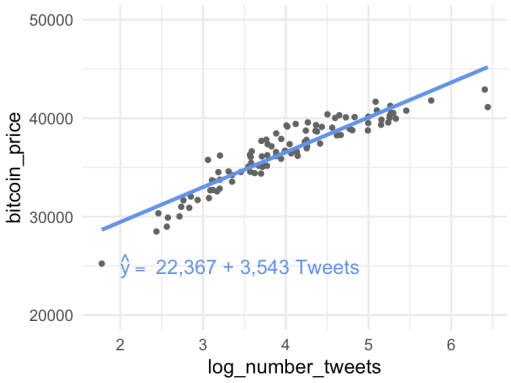
- **Mean squared error (MSE):** $MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2$
- Root mean squared error (RMSE): $\sqrt{MSE}$
- Mean absolute error (MAE): $MAE = \frac{1}{n}\sum_{i=1}^{n}|y - \hat{y}|$
- Median absolute error (mAE): $mAE = median|y - \hat{y}|$
- Proportion of variance explained (R^2): $R^2 = correlation(y, \hat{y})^2$

Remember that $y = f(x) + \epsilon$

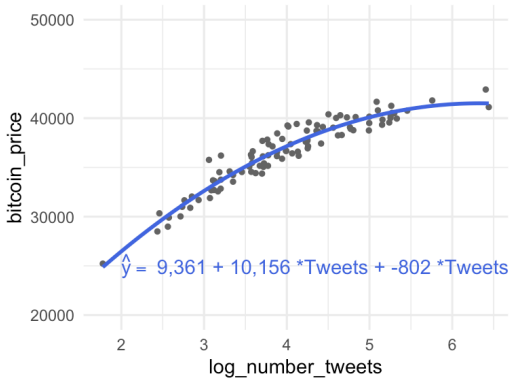
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$








```
## [1] "MSE:  1,638,400"
```

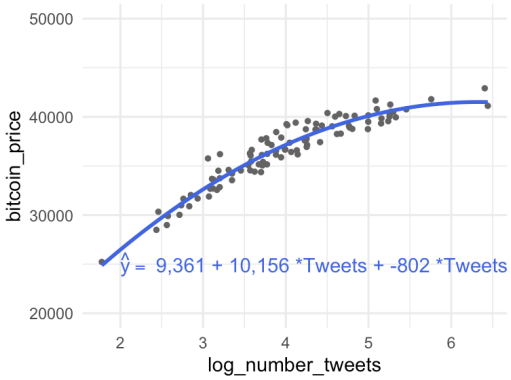



```
## [1] "MSE:  907,983"
```

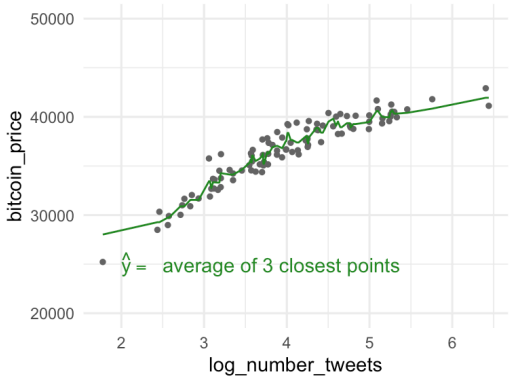
So far

- We can estimate f(x) using different models
- We can use MSE to understand which model is better

Which model is best: wooclap.com/ADAV2024




```
## [1] "MSE:  907,983"
```

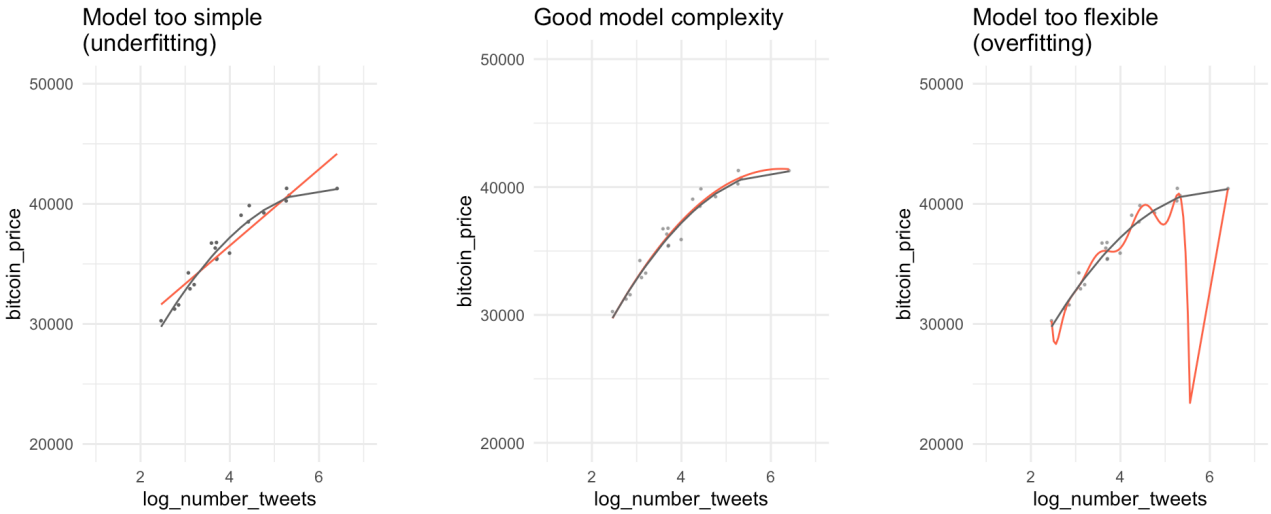



```
## [1] "MSE:  790,971"
```

Why?

Overfitting! (model too complex/flexible for the task)

Underfitting vs overfitting



What is happening?

MSE in the way we defined is problematic. What we care about is the generalization error $E(MSE)$

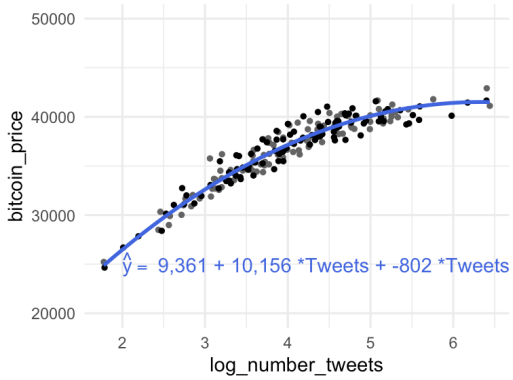
Important: We can approximate the generalization error $E(MSE)$ by calculating MSE using new data —i.e., data not used to train (fit) the model

- Why do we want to calculate MSE on unseen data?
  - Conceptually: We often want to understand/predict a general phenomena, not just the observations we already have
  - Pragmatically: It allows to understand if our model is overfitting the data. Idea = the model can't overfit data if it doesn't see that data.

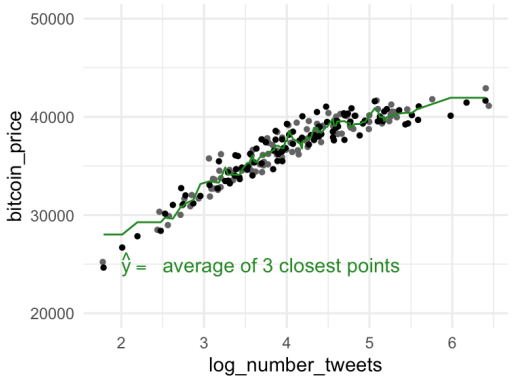
The “training/validation/test data” paradigm

- E(MSE)s must be estimated using a new dataset.
  - Ideally this is a different dataset (coming the intended prediction situation)
  - Often you only have one dataset, which we split in two or three: training + validation + test
- Datasets:
  - Training: Train the model
  - Validation: Estimate E(MSE) to compare between models, tune models, select features
  - Test: Estimate E(MSE) of the final model

MSE of our models in a new dataset



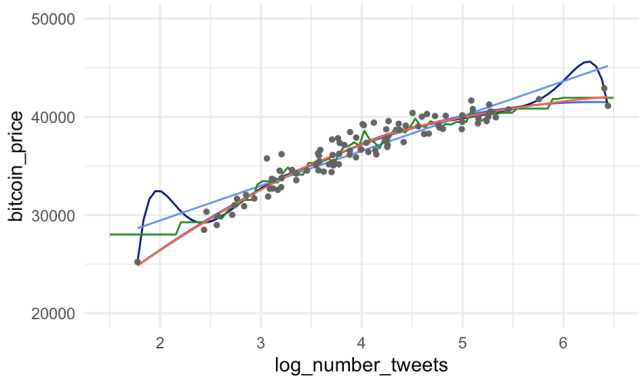

```
## [1] "MSE train:  907,983"
## [1] "MSE test:  923,395"
```




```
## [1] "MSE train:  790,971"
```

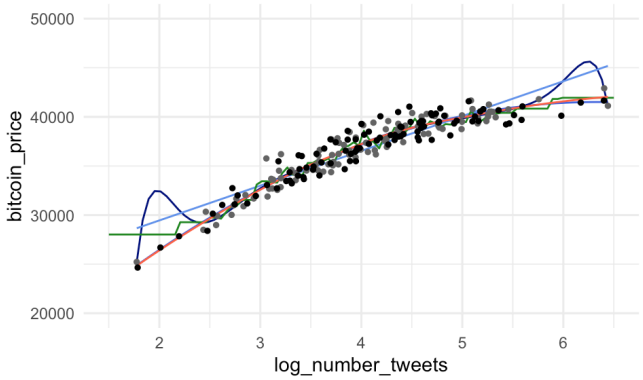
```
## [1] "MSE test:  1,204,628"
```

Training dataset




```
## [1] "Minimum error (Var(epsilon)):  924,921"
## [1] "Error of linear regression:  1,638,400"
## [1] "Error of quadratic regression:  907,983"
## [1] "Error of regression up to power 10:  851,925"
## [1] "Error of KNN regression:  790,971"
```

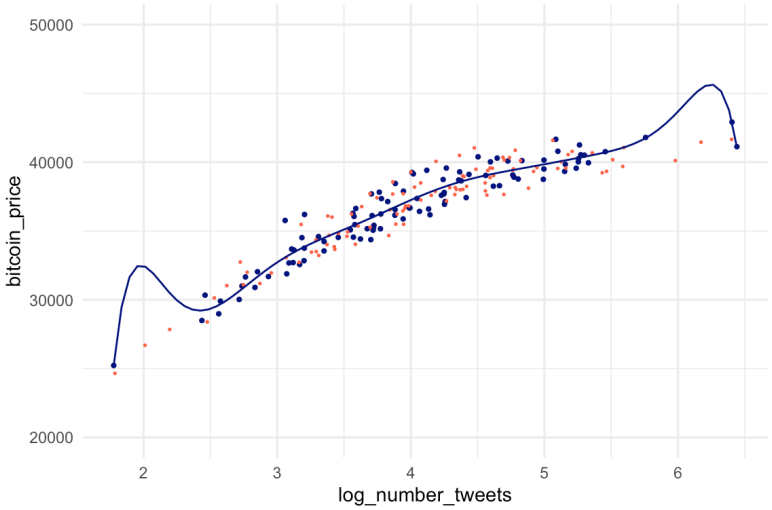
Validation (test) dataset




```
## [1] "Minimum error (Var(epsilon)):  909,786"
## [1] "Error of linear regression:  1,972,790"
## [1] "Error of quadratic regression:  923,395"
## [1] "Error of regression up to power 10:  1,611,194"
## [1] "Error of KNN regression:  1,204,628"
```

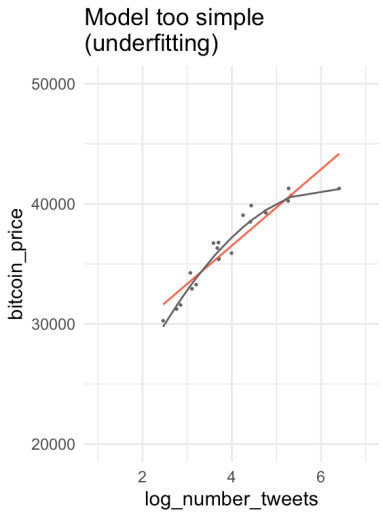
Which is the right model? Which models overfit and underfit?

What is happening

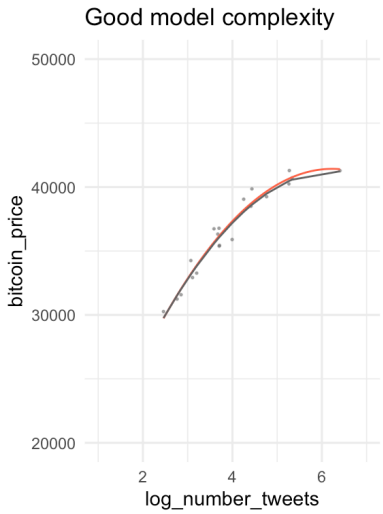


- Navy: Train dataset and fitted data
- Tomato: Validation (test) dataset
- See how the navy line is very close to the navy dots in both extremes, but far from the red ones

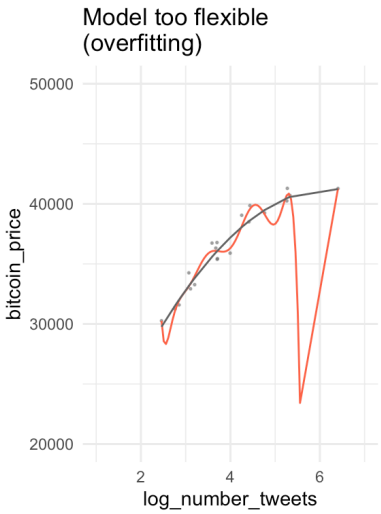
In general: Underfitting vs overfitting



High train and test MSE



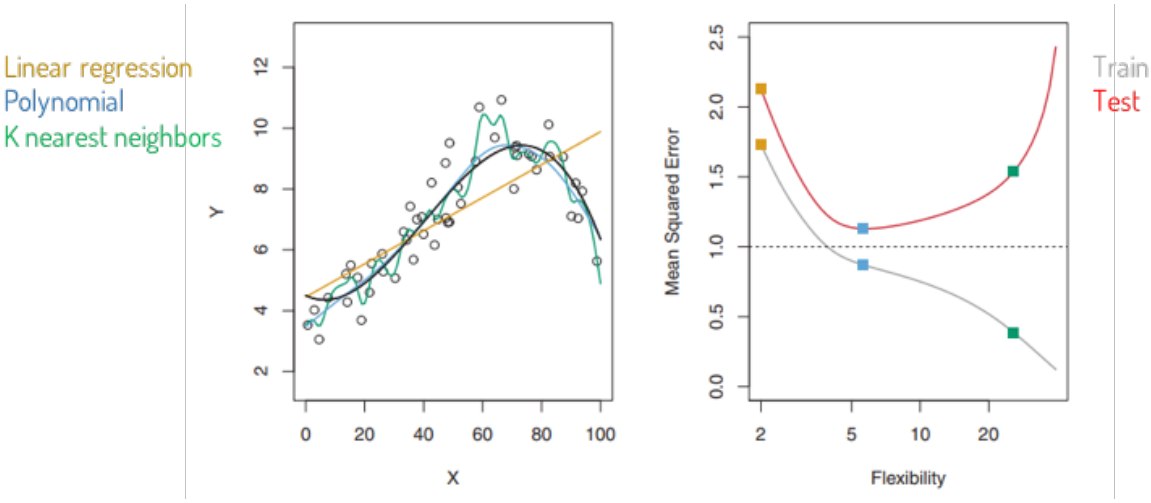
Low train MSE and low test MSE



Low train MSE and high test MSE

Model flexibility

In general, more flexible (complex) models will fit the training data more closely



Complexity

Possible definitions of complexity:

- Amount of information in data absorbed into model;
- Amount of compression performed on data by model;
- Number of effective parameters, relative to effective degrees of freedom in data.

For example:

- More predictors, more complexity;
- Higher-order polynomial, more complexity (x, x^2, x^3, $x_1 \times x_2$, etc.);

10 minute break

- After the break, we will wee why this relationship between complexity and E(MSE) happens: bias-variance trade off

Recap

- We want to learn a model of our data
- We evaluate the performance of a model via MSE on the test data:
  - We don't want to underfit (will have high MSE)
  - We don't want to overfit (will have high MSE)
- Now
  - MSE and the bias-variance trade-off
  - Model complexity and the bias-variance trade-off
  - Tuning and selecting models:
    - Train / validation / test paradigm
    - K-fold Cross-validation

MSE

Remember that $y = f(x) + \epsilon$

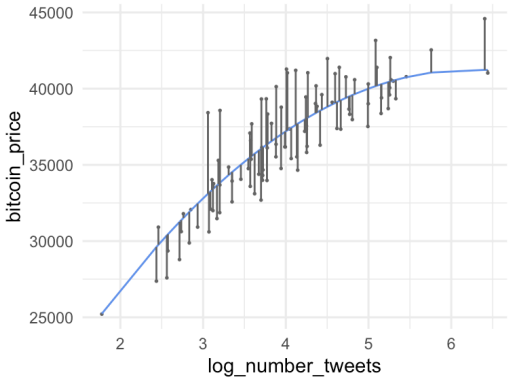
If we would know the real f(x)

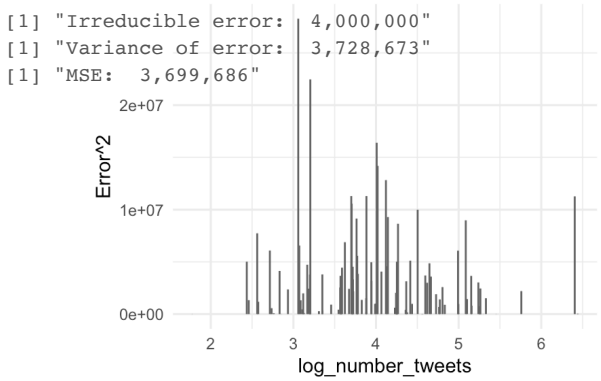
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\epsilon_i^2 = Variance(\epsilon) = Noise$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\epsilon_i^2 = Variance(\epsilon) = Noise$$

If we would know the real f, MSE would indicate the irreducible error





But we do not know the real f(x), only $\hat{f}(x)$

$$E(MSE) = E(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2) =$$

(The $E(.)$ means 'on average over samples from the target population')

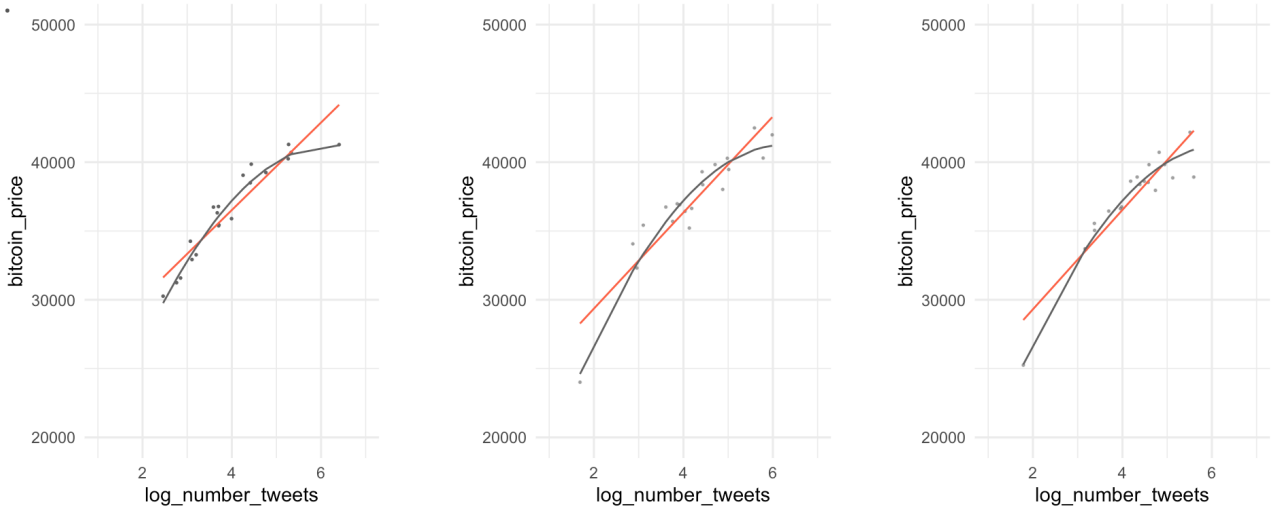
$$E(\frac{1}{n}\sum_{i=1}^{n}(outcome_i - predicted_i)^2) =$$

$$Bias^2(model) + Variance(model) + Variance(\epsilon)$$

- **Bias**: How wrong the predictions of the model are (on average). Usually comes from underfitting.
- **Variance**: How variable the predictions of the model are when fitted in different samples. Comes from overfitting.
- **Variance(**
  ϵ**)**: Irreducible error

- **Bias**: It measures how wrong are predictions from the average many models trained on different samples (or one model with large sample sizes) from the real f. The error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model. High bias often results from underfitting.
- **Variance**: It measures how different are the predictions of models that are fit on different samples. Variance refers the sensitiviy of our model to small fluctuations in the training dataset. Since the training data are used to fit the statistical learning method, different training data sets will result in a different $\hat{f}$. High variance often comes from overfitting.
- **Variance(**
  ϵ**)**: Irreducible error

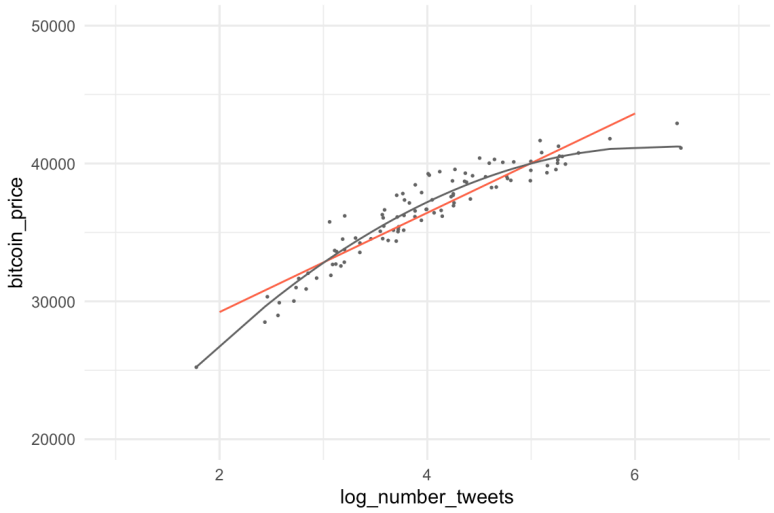
Variance of simple model



Changing the dataset doesn't change the model = **low variance**

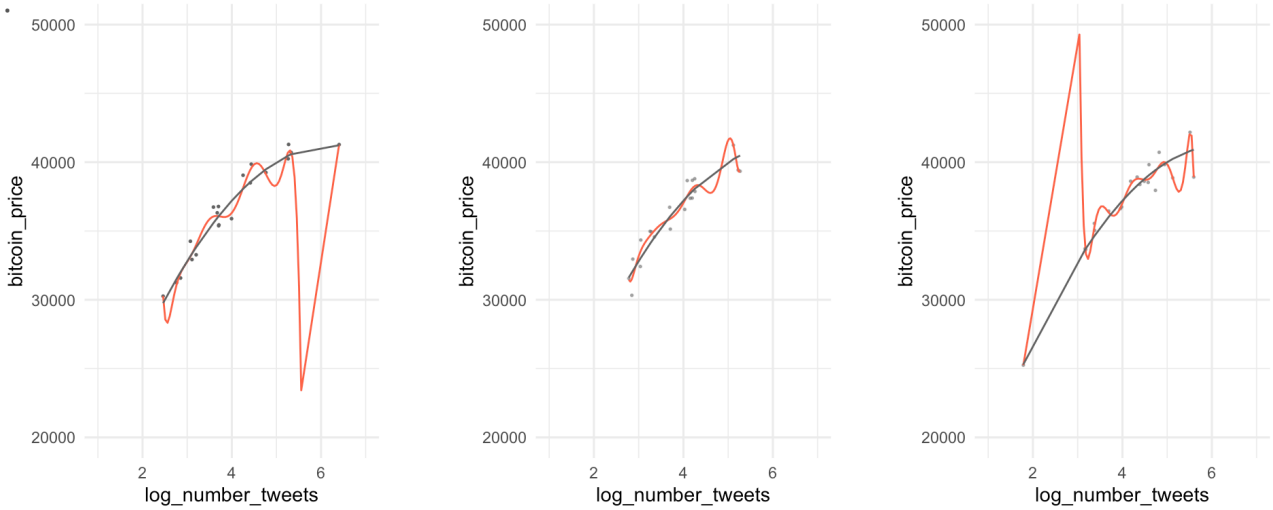
Bias of linear model (average 100 reps)

Average prediction for 100 models, trained in 100 datasets.



- The fitted model is always far from the points = **high bias**

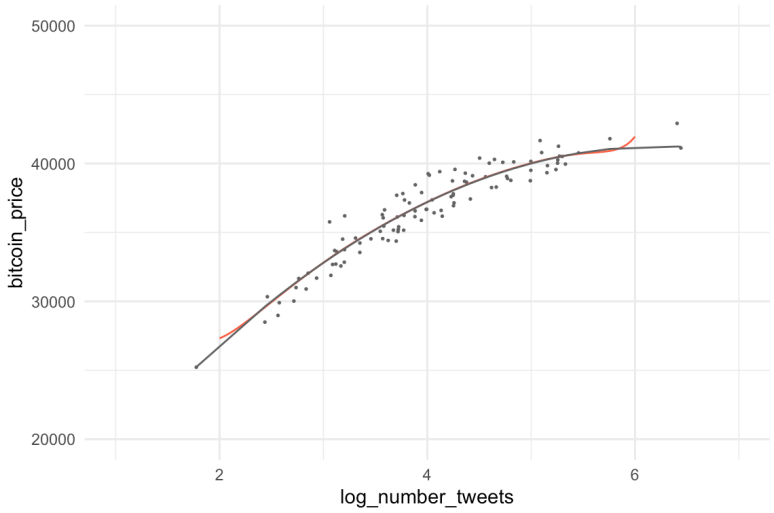
Variance of complex model



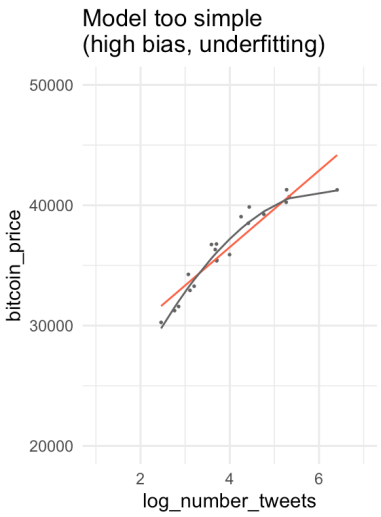
Changing the data a bit changes the fitted model a lot = **high variance**

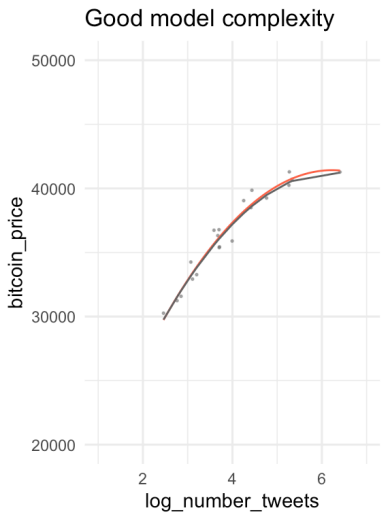
Bias of complex model (average 100 reps.)

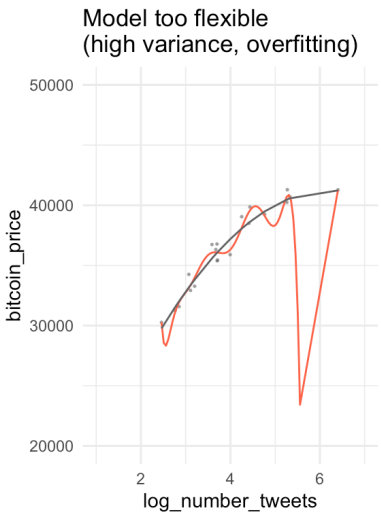
Average prediction for 100 models, trained in 100 datasets.



- The fitted model is **on average** close from the points = **low bias**


Model too simple
(high bias, underfitting)
50000
40000
30000
20000
bitcoin_price
2
4
6
log_number_tweets


Good model complexity
50000
40000
30000
20000
bitcoin_price
2
4
6
log_number_tweets


Model too flexible
(high variance, overfitting)
50000
40000
30000
20000
bitcoin_price
2
4
6
log_number_tweets

Bias - Variance trade-off

$$E(MSE) = Bias^2(model) + Variance(model) + Variance(\epsilon)$$

So, to have the best model we need to minimize bias and variance.

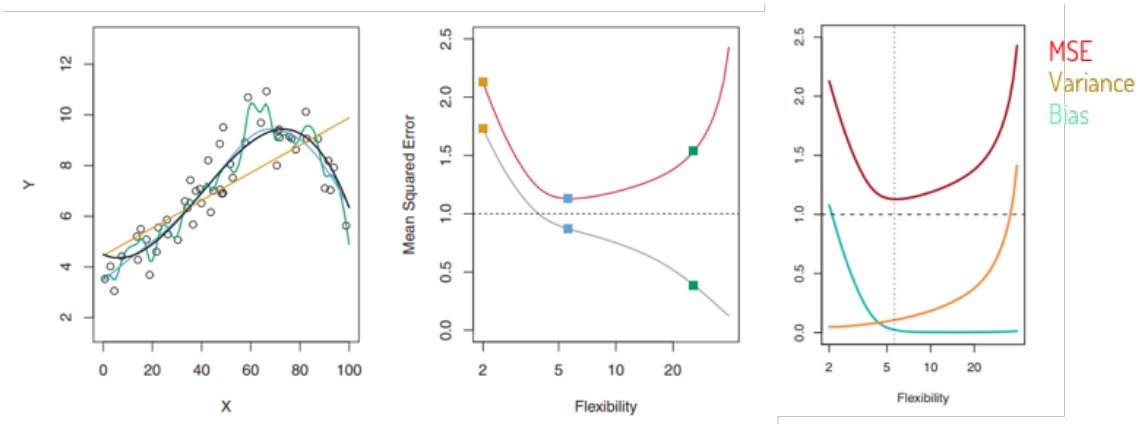
Problem? There is a trade-off between both, which depends on the **complexity** of the model

- Model is too simple: It cannot capture the complexity of the data (high bias)
- Model is too complex: It easily overfits accidental patterns (high variance)

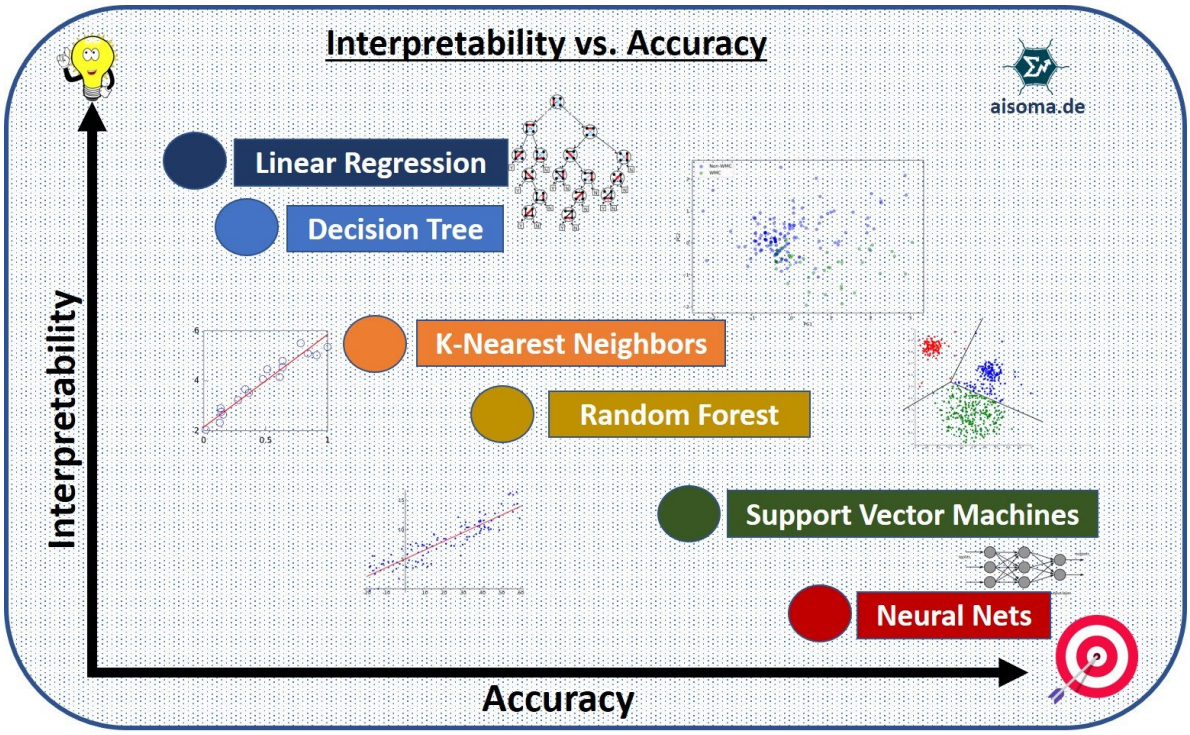
wooclap.com/ADAV2024

Model complexity and bias-variance trade-off

The mean squared error of the test set is a combination of bias and variance:



Model complexity vs interpretability



Recap

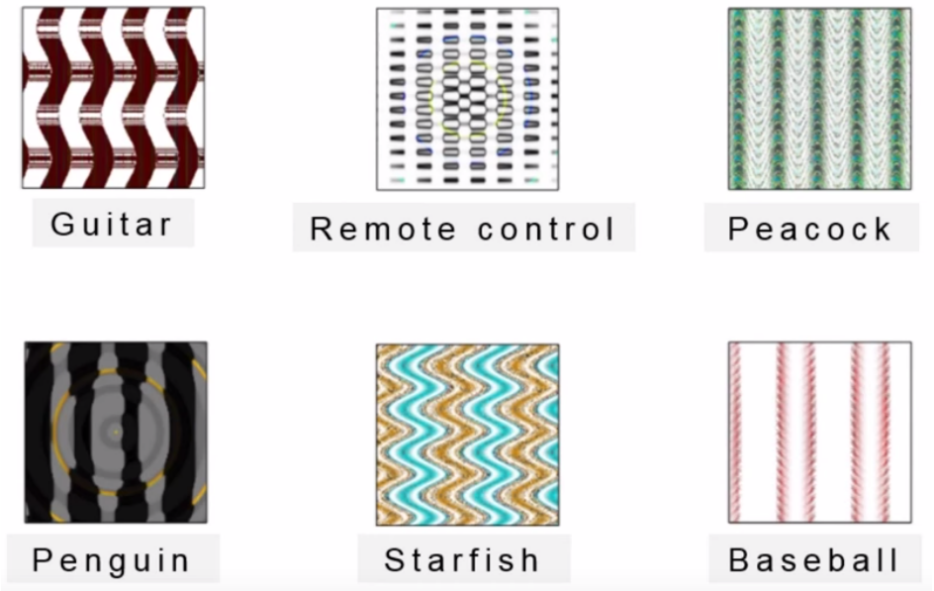
- We want to learn a model of our data
- We evaluate the performance of a model via MSE on the validation or test data:
  - We don't want to underfit (will have high MSE –> Because of high bias)
  - We don't want to overfit (will have high MSE –> Because of high variance)
- We choose the correct complexity estimating E(MSE) in the validation dataset
- We can estimate E(MSE) using the test dataset
  - Allows us to understand how the results of our model generalize to unseen data
  - Done only for the best model (sometimes best k models)
- Keep in mind that complex models are often less interpretable

Estimating E(MSE) using a test/validation dataset

Considerations with the test dataset

- The idea is that the MSE_{test} is a good estimate of the $E(MSE)$ (generalization error)
- This is only true if the test data is similar to the prediction data!

Example where this went wrong: ‘Neural networks are easily fooled’



https://www.youtube.com/watch?v=M2IebCN9Ht4

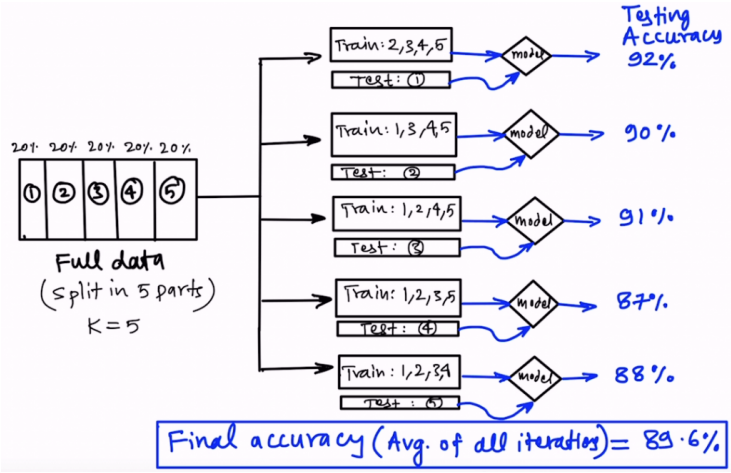
Primer on K-fold cross validation

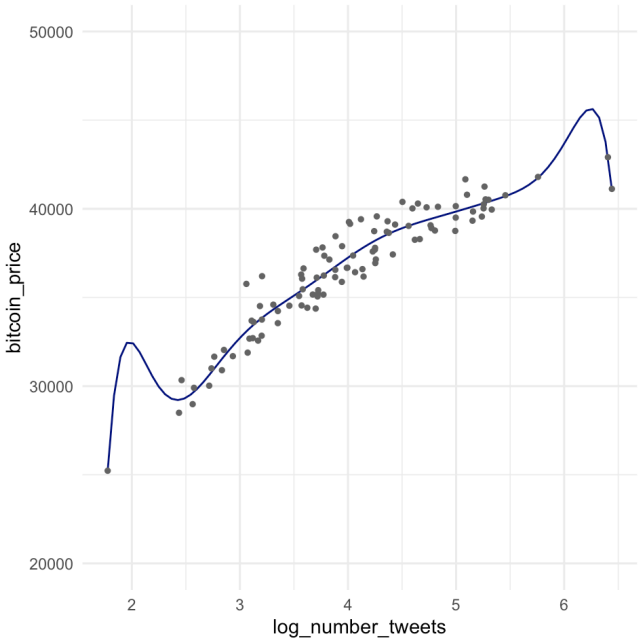
Test datasets are necessary to estimate E(MSE). They should be similar to the intended prediction scenario.

Usually ~10-20% of the data is used to estimate E(MSE)

But… what if we are unluckly and the test dataset is not representative?

A solution is to use K-fold cross validation





MSE:

```
## [1] "root squared MSE: 5,909"
## [1] "root squared MSE: 1,216"
## [1] "root squared MSE: 1,245"
## [1] "root squared MSE: 1,519"
## [1] "root squared MSE: 2,418"
## [1] "root squared MSE: 5,479"
## [1] "root squared MSE: 215,102"
## [1] "root squared MSE: 18,775"
## [1] "root squared MSE: 1,344"
## [1] "root squared MSE: 1,207"
## [1] "root squared MSE: 1,274"
## [1] "root squared MSE: 1,528"
## [1] "root squared MSE: 1,231"
## [1] "root squared MSE: 4,986"
## [1] "root squared MSE: 100,318"
## [1] "root squared MSE: 1,180,232"
## [1] "root squared MSE: 1,635"
## [1] "root squared MSE: 15,018"
## [1] "root squared MSE: 80,396"
## [1] "root squared MSE: 1,468"
## [1] "CV root squared MSE: 46,037"
```

Tuning and selecting models

Tuning and selecting models

Often we want to tune models before evaluating the performance of the best model

- Which type of model?
- Which features do we want to include?
- Hyperparameters (e.g. number of neighbors in KNN, regularization of the model)

Which observations should we use?

wooclap.com/ADAV2024

- Why not the test dataset? –> We would be using it for comparing models and for estimating E(MSE) (could lead to underestimating E(MSE))
- Why not the train dataset? –> We would be using it for training and comparing models
- Ideally a new dataset, but often we don't have that luxury

Training - validation - test paradigm

- **Training data**:
  Observations used to fit $\hat{f}(x)$

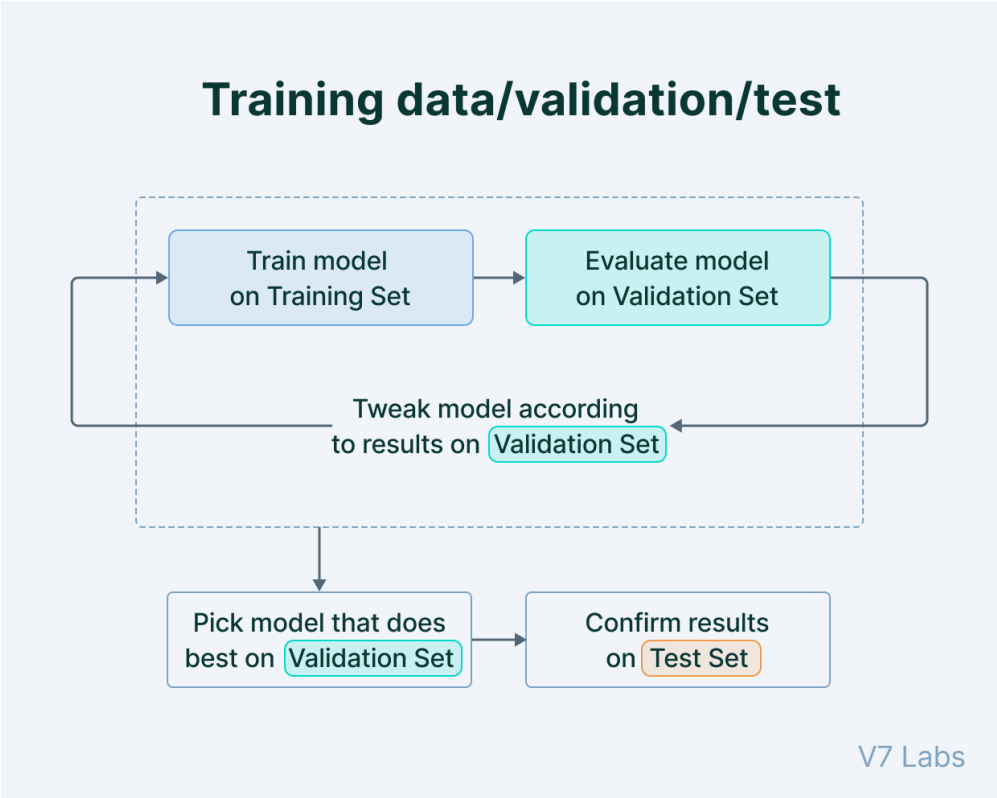
- **Validation data** (or 'dev' data):
  New observations used several times to select model between models Often people also called these data "test data"

- **Test data**:
  New observations used **once** to evaluate E(MSE) for your final model

Selecting model complexity

- These factors together determine what works best:
  - The amount of irreducible variance ($Var(\epsilon)$).
  - The sample size (n).
  - The complexity of the model (p/df or equivalent).
  - How close the functional form of $\hat{f}(x)$ is to the true $f(x)$.
- Sometimes a wrong model is better than a true model
- If you do not believe in true models: sometimes a simple model is better than a more complex one.

In practice


Training data/validation/test
Train model on Training Set
Evaluate model on Validation Set
Tweak model according to results on Validation Set
Pick model that does best on Validation Set
Confirm results on Test Set
V7 Labs

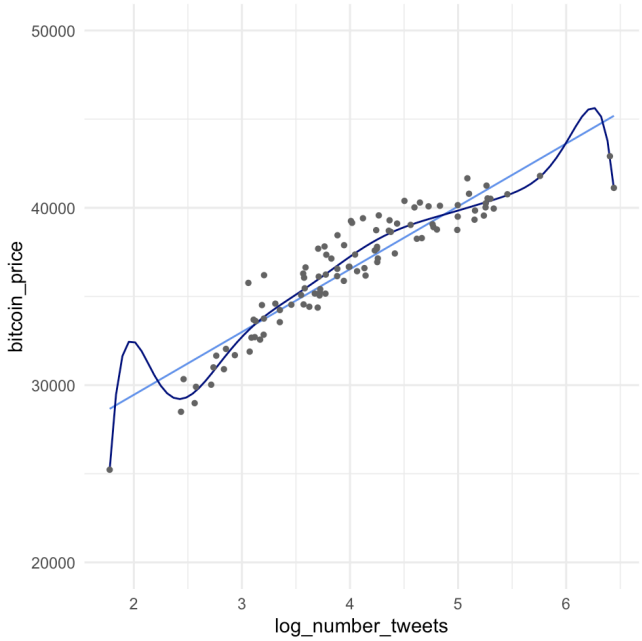
Example in practice

- We have one dataset:
  - Shuffle the observations
  - Divide into training (85%) and testing (15%)
  - Training –> Divide into training (70%) and validation (15%)
    - Tune models, evaluating models using MSE on the validation set
    - Select the best model
  - Use the training + validation set to retrain the final model
  - Evaluate the model using the test data

Drawbacks of train/dev/test

- Only a subset of the observations are used to fit the model.
- MSEs calculated in the validation data can be highly variable.

E.g. we train the models on the training data and we calculate MSE in the validation dataset



MSE:

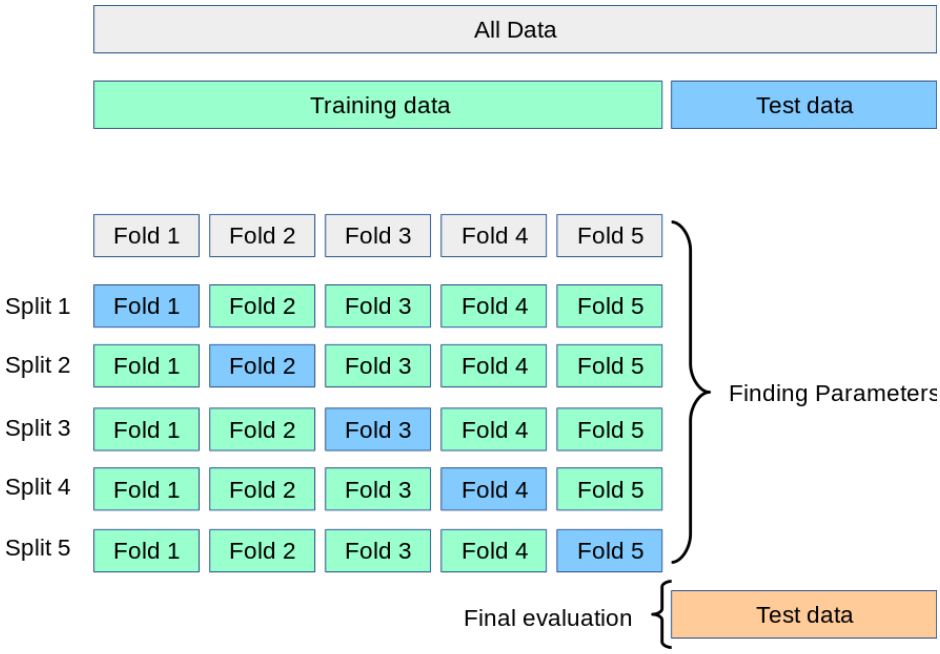
```
## [1] "MSE (linear/order 10):  1,972,790  /  1,611,194"
## [1] "MSE (linear/order 10):  2,316,250  /  34,912,977"
## [1] "MSE (linear/order 10):  1,541,603  /  1,478,344"
## [1] "MSE (linear/order 10):  2,060,496  /  1,551,094"
## [1] "MSE (linear/order 10):  2,095,906  /  2,308,450"
## [1] "MSE (linear/order 10):  2,017,427  /  5,847,994"
## [1] "MSE (linear/order 10):  2,406,703  /  30,016,685"
## [1] "MSE (linear/order 10):  3,374,062  /  46,268,769,597"
## [1] "MSE (linear/order 10):  2,099,708  /  352,512,504"
## [1] "MSE (linear/order 10):  1,786,325  /  1,807,504"
## [1] "MSE (linear/order 10):  1,894,595  /  1,456,931"
## [1] "MSE (linear/order 10):  1,561,332  /  1,623,019"
## [1] "MSE (linear/order 10):  1,989,382  /  2,335,140"
## [1] "MSE (linear/order 10):  1,642,818  /  1,515,588"
## [1] "MSE (linear/order 10):  2,049,801  /  24,863,926"
## [1] "MSE (linear/order 10):  2,152,156  /  10,063,705,111"
## [1] "MSE (linear/order 10):  3,755,307  /  1,392,947,401,3
## [1] "MSE (linear/order 10):  2,006,548  /  2,672,228"
## [1] "MSE (linear/order 10):  2,111,963  /  225,546,563"
## [1] "MSE (linear/order 10):  2,507,906  /  6,463,451,737"
```

Which model is best?

K-fold cross validation

- ‘Cross validation’ often used to replace single validation approach;
- Instead of dividing one time the training dataset (into train/validation), do it many times.
- Perform the train/validation split several times, and average the result.
- Usually K = 5 or K = 10.
- When K = N, ‘leave-one-out’;

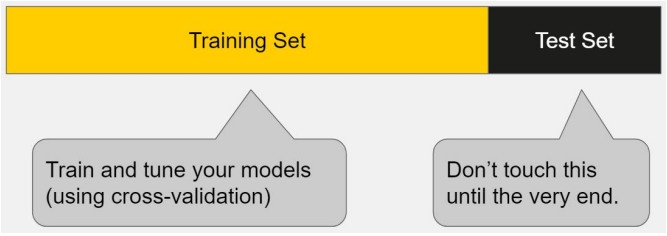
K-fold crossvalidation (K = 5 here)



Source: Scikit-learn website

Difference between validation and test datasets

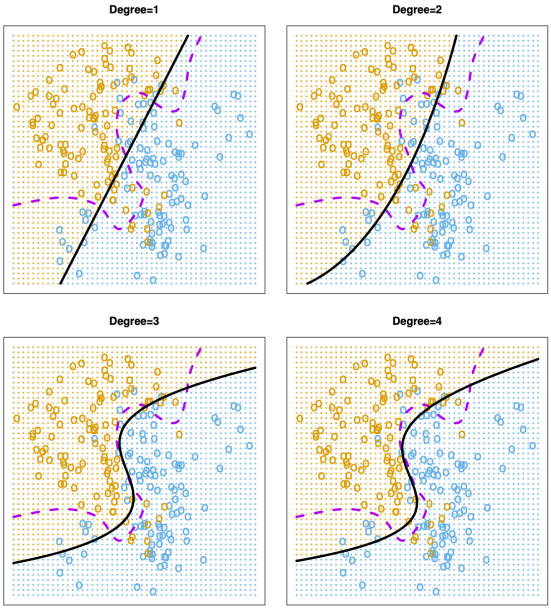
wooclap.com/ADAV2024



Source: Elite Data Science

What about classification?

- We use a different messure of the error: e.g. accuracy, precision, recall, F1-score, etc
- Same idea as before



Source: Scikit-learn website

How to split the datasets?

- Training: Bulk of observations (~50-80%)
- Validation and testing: Smaller subsets (~20-50%) –> Should be representative in order to estimate E(MSE) accurately.
- e.g. without cross-validation
  - Training: 50-70%
  - Validation: 15-25%
  - Test: 15-25%
- e.g. with cross-validation
  - Training: 70-80% + 5-10 fold cross-validation to separate into training/validation
  - Test: 20-30%

Conclusion

- We want to learn a model of our data
  - We don't want to underfit (will have high MSE –> Because of high bias)
  - We don't want to overfit (will have high MSE –> Because of high variance)
- More complex models tend to have higher variance and lower bias
- We choose the correct complexity calculating MSE in the validation dataset
  - Compare between models, tune the hyperparameters of the model or select features
  - Or even better, use cross-validation
  - Keep in mind that complex models are often less interpretable
- We can estimate E(MSE) using the test dataset
  - Allows us to understand how the results of our model generalize to unseen data
  - Done only for the best model (sometimes best k models, can be problematic)
  - Getting good test data is difficult, unsolved problem

Next class

Videos from the authors of ISLR: https://www.statlearning.com/online-course

Next week

Linear regression for data science.

Have a nice day!