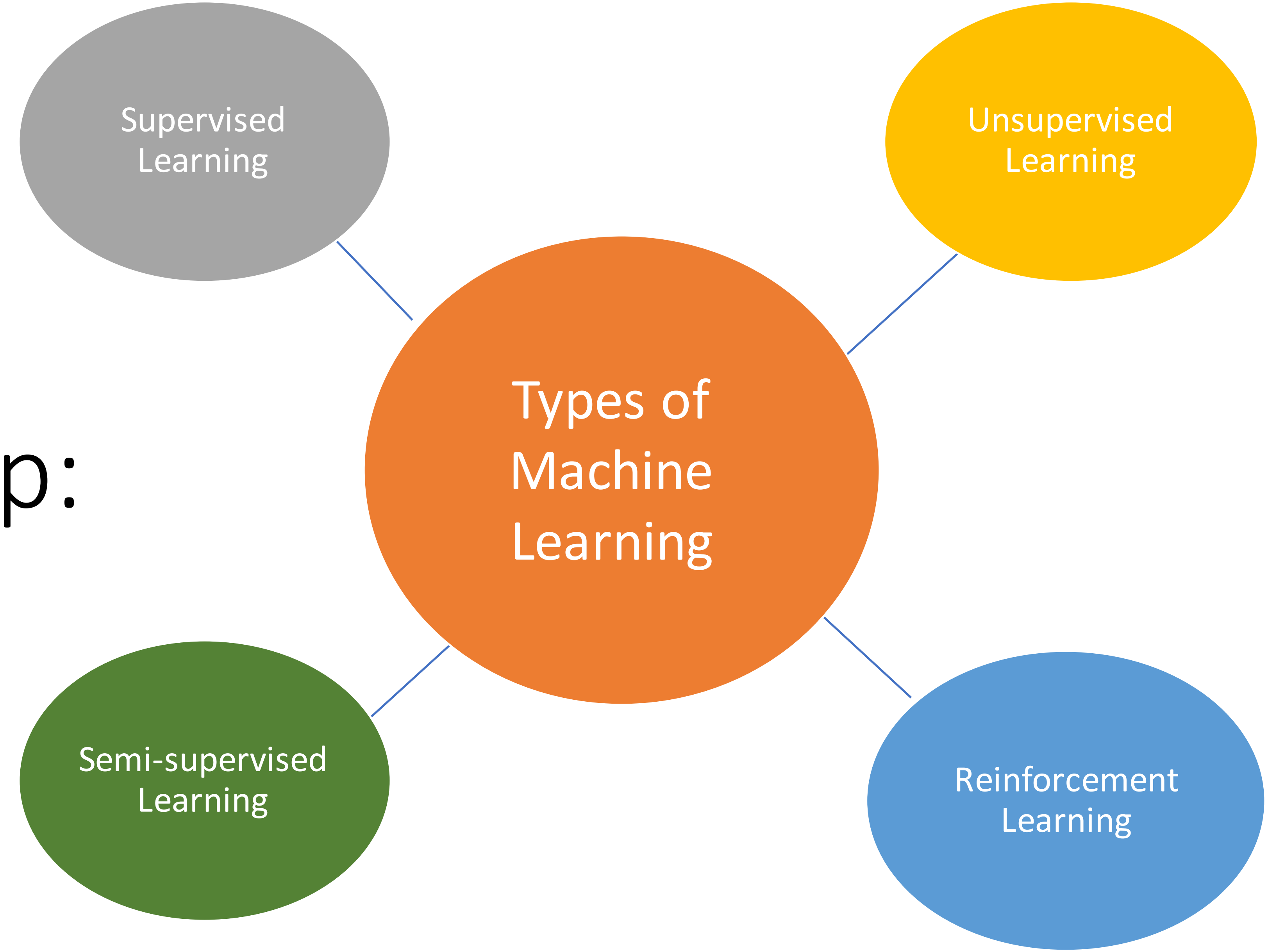
Ahmed Ibrahim

ECE 9039/9309 MACHINE LEARNING

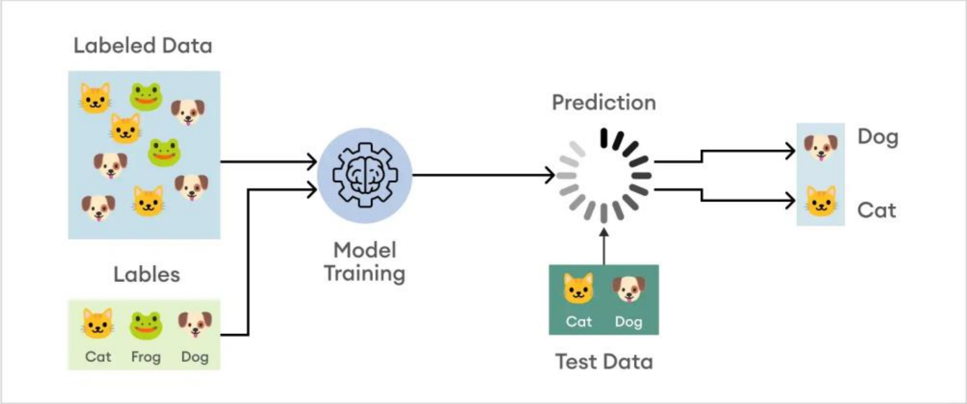
Outline

- Recap
- Multiple Linear Regression
  - Loss Functions
  - Evaluating Model Performance
  - Sources of Error
- Feature Construction, Manipulation and Selection

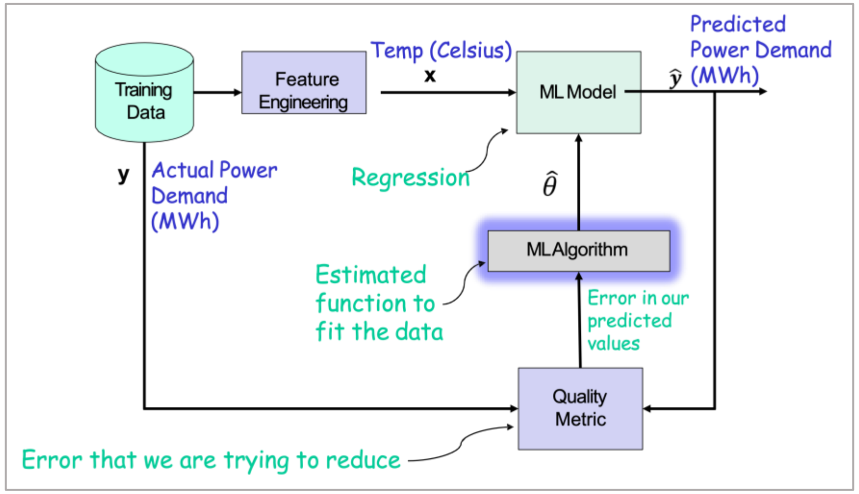
Recap:


Types of Machine Learning
Supervised Learning
Unsupervised Learning
Semi-supervised Learning
Reinforcement Learning

Recap: Types of Supervised Learning



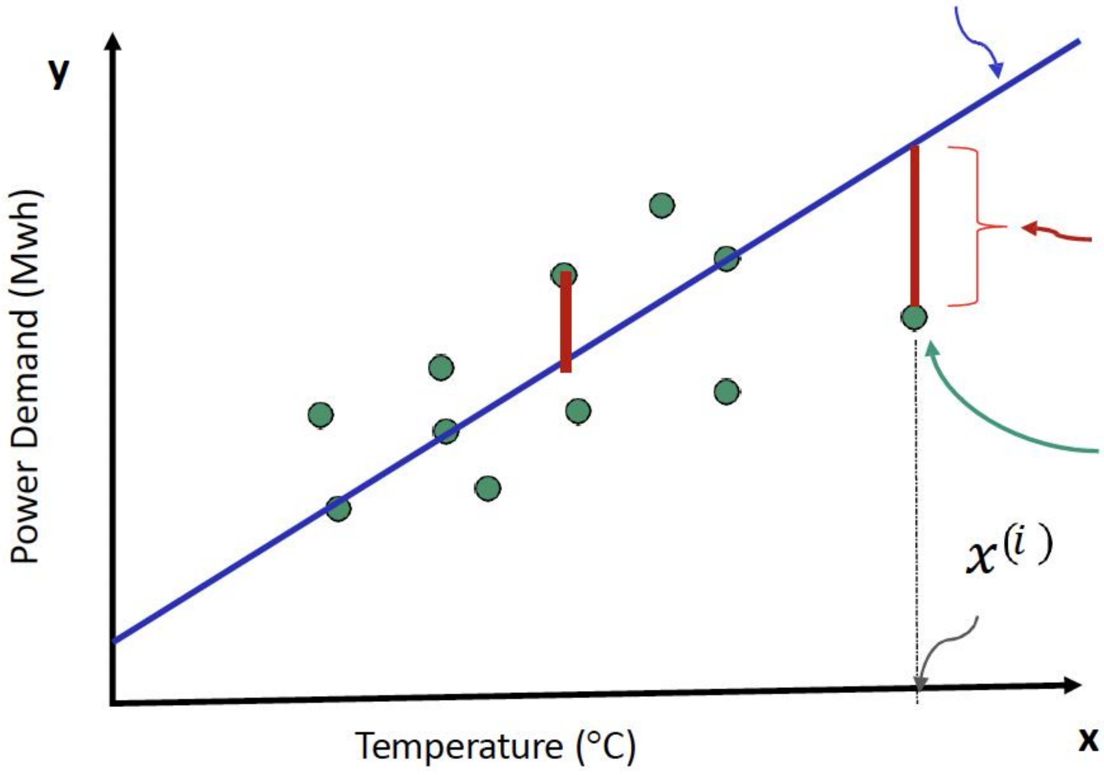
Source: https://www.superannotate.com/blog/image-classification-basics



- **Classification** example: Image recognition
- **Task**: given an image, predict the class label
  - Training data: millions of labeled images
  - Target: the image class

- **Regression** example: Car price prediction
- Task: given a car, predict its price
  - Training data: many examples of cars, including their features (mileage, age, brand, etc.) and their labels (price)
  - Target: price of car

Recap: Simple Linear Regression



Power Generation Example

The estimate line of y (power demand)

$$\hat{y} = f(x) = \theta_0 + \theta_1 x$$

The error:

$$\epsilon^{(i)} = y^{(i)} - \hat{y}^{(i)}$$

The true observation:

$$y^{(i)} = \hat{y}^{(i)} + \epsilon^{(i)}$$
$$= \theta_0 + \theta_1 x^{(i)} + \epsilon^{(i)}$$

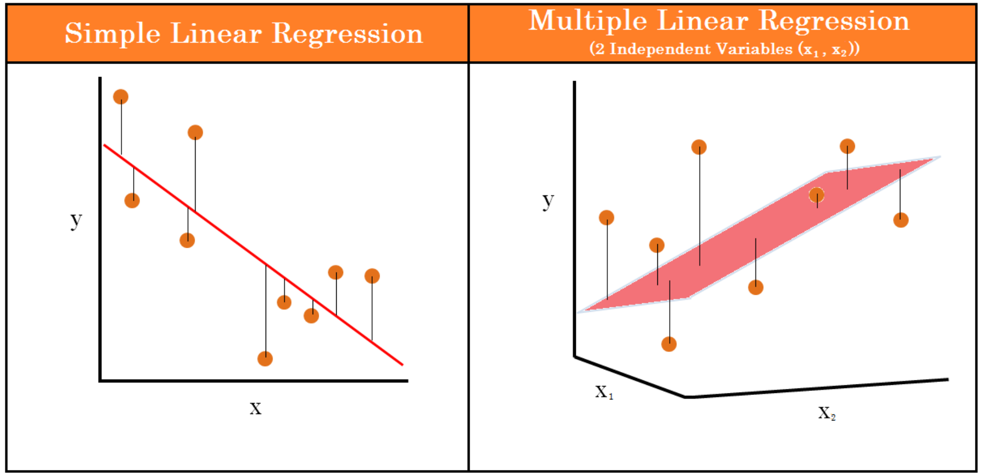
Recap: Linear Regression with Multiple Features

- ❑ The goal of linear regression with multiple features is to find a hyperplane that fits the target variable y based on ***n*** input variables\features
- ❑ Mathematically: $f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
- ❑ (x^i, y^i): i^{th} training set

  $y^{(i)}$: target variable (Scaler)

  $x^{(i)}$: input variables (Vector)

$$x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \\ . \\ . \\ . \\ x_n^{(i)} \end{bmatrix}$$



$f(x) = \theta_0 + \theta_1 x_1$

$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

Source: https://www.analyticsvidhya.com/blog/2021/11/startups-profit-prediction-using-multiple-linear-regression/

Question #1

In the context of supervised learning, what is the primary difference between classification and regression?

A. Classification predicts a continuous value, whereas regression predicts a discrete class label.

B. Regression is used for predicting behaviors, while classification is not.

C. Classification predicts a discrete class label, whereas regression predicts a continuous value.

D. Regression uses labeled examples, while classification does not.

https://forms.gle/nX7XwAw6zmJUeZTZ7

Question #2

In the gradient descent algorithm, what is the primary challenge of using a static learning rate as opposed to a dynamic (decreasing) learning rate?

A. A static learning rate may fail to converge to the global minimum due to constant step sizes.

B. Static learning rates typically lead to higher computational costs and slower convergence.

C. Dynamic learning rates increase the likelihood of overshooting the minimum.

D. Static rates cannot be used with convex optimization problems.

https://forms.gle/zQ9pvMtNT6ZYr9JZA

1.5: Loss Functions

Loss functions in machine learning are used to quantify how well a model's predictions match the actual data.

Loss Functions

- Absolute Error Loss Function:

$$L(\theta) = L\big(y, f_{\hat{\theta}}(x)\big) = \big|y - f_{\hat{\theta}}(x)\big| = \sum_{i=1}^{m} \big|y^{(i)} - \widehat{y^{(i)}}\big|$$

Observed values

Predicted values

- Minimize the sum of magnitudes (absolute values)
- LAD -> **Least Absolute Deviation**

- Squared Error Loss Function

$$L(\theta) = L\big(y, f_{\hat{\theta}}(x)\big) = \big(y - f_{\hat{\theta}}(x)\big)^2 = \sum_{i=1}^{m} \big(y^{(i)} - \widehat{y^{(i)}}\big)^2$$

- Minimize the sum of squared residuals
- OLS -> **Ordinary Least Squares**

Ordinary Least Squares

- OLS is the most commonly used Loss Function
  - Solutions can be easily computed
  - Big errors count relatively more than small errors

How do you decide between L1 and L2 Loss Functions?

- Generally, L2 Loss Function is preferred in most of the cases. However, when the **outliers** are present in the dataset, then the L2 Loss Function does not perform well.
  - The reason behind this bad performance is that if the dataset has outliers, then because of the consideration of the squared differences, it leads to a much larger error. Hence, the L2 Loss Function is not useful here.
- Prefer the L1 Loss Function as it is not affected by the outliers or remove the outliers and then use the L2 Loss Function.

Mean Square Error (MSE)

- MSE is calculated by the sum of the square of prediction error, which is real output minus predicted output, and then divided by the number of data points.
- The Mean Absolute Error (MAE) Loss Function, on the other hand, is an extension of the Absolute Error Loss.
- It calculates the average absolute difference between the predicted values and the actual target values across a dataset.

Number of data points

Observed values

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Predicted values

Mean Absolute Error (MAE)

- MAE is like MSE. However, instead of the sum of the square of error in MSE, MAE takes the sum of the absolute value of error.

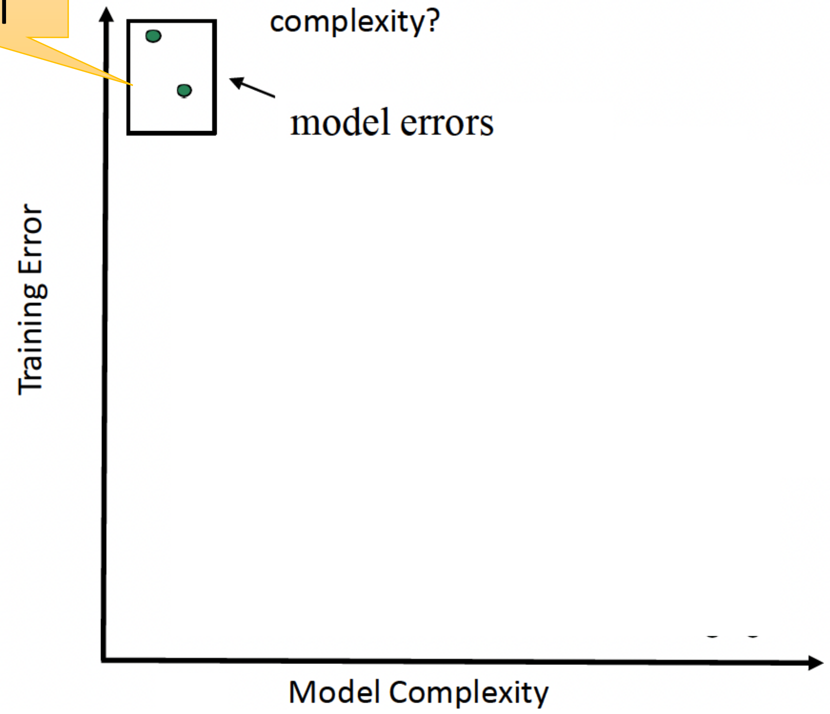
$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

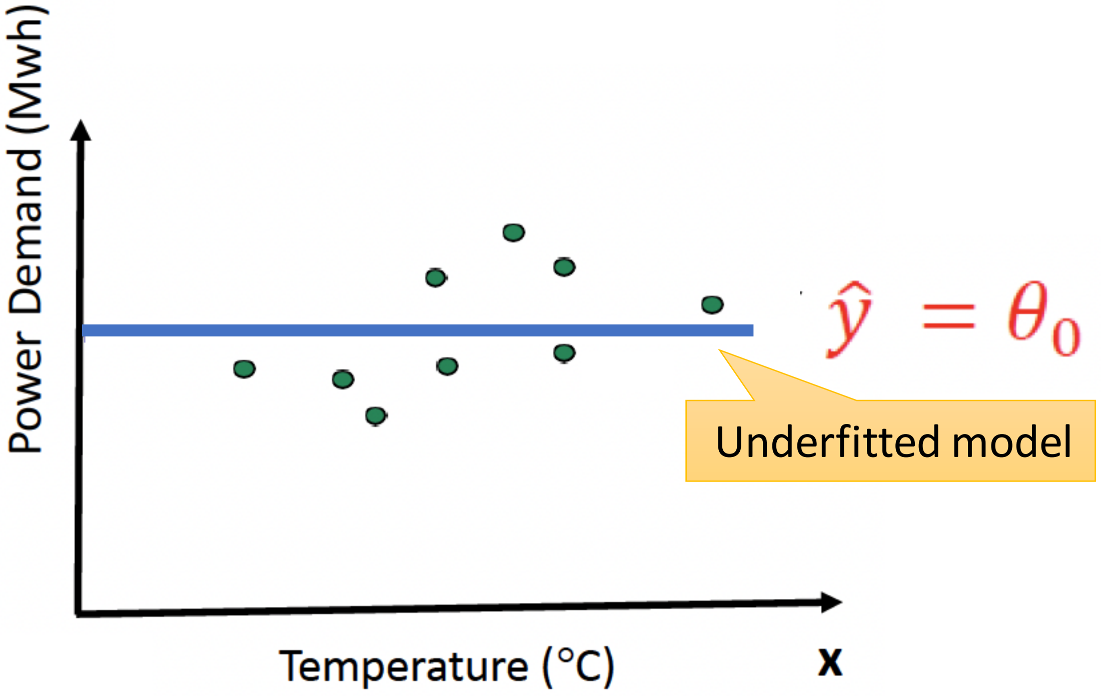
- P.S. MSE penalizes big prediction errors by squaring them, while MAE treats all errors the same.

Training Error vs. Model Complexity

Constant Model

How training errors behave with model complexity?

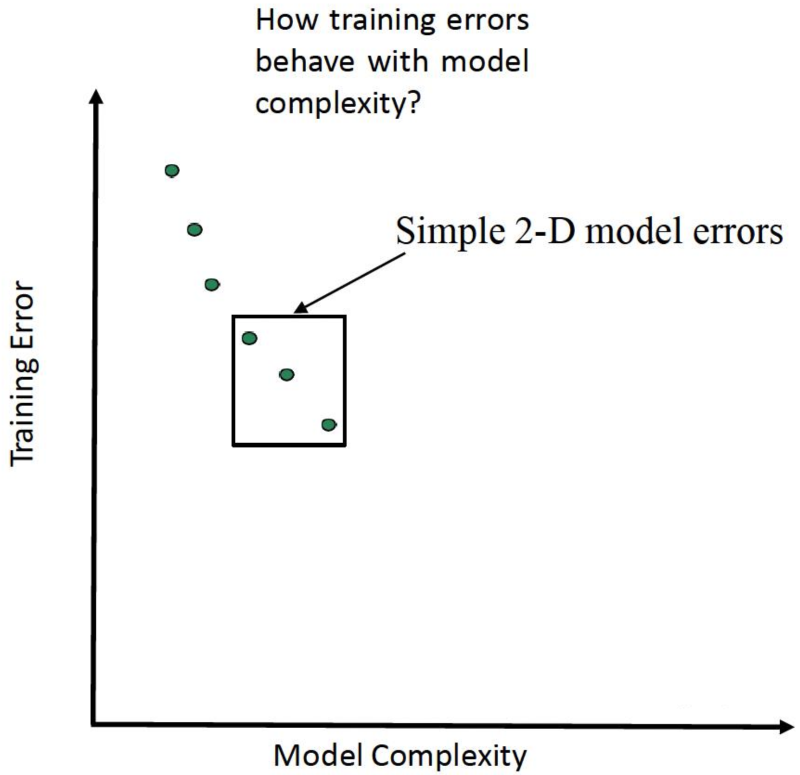


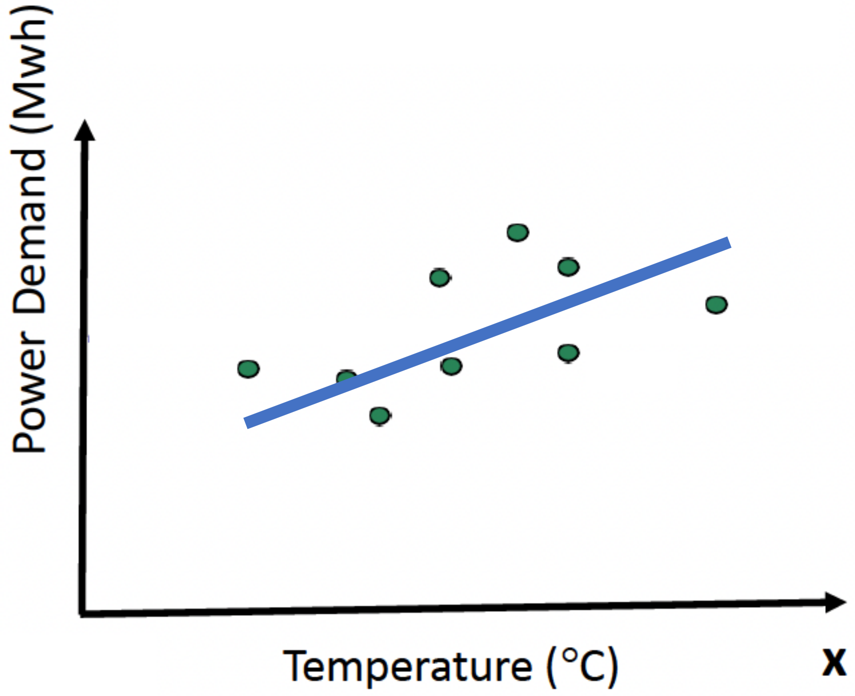


$\hat{y} = \theta_0$

Underfitted model

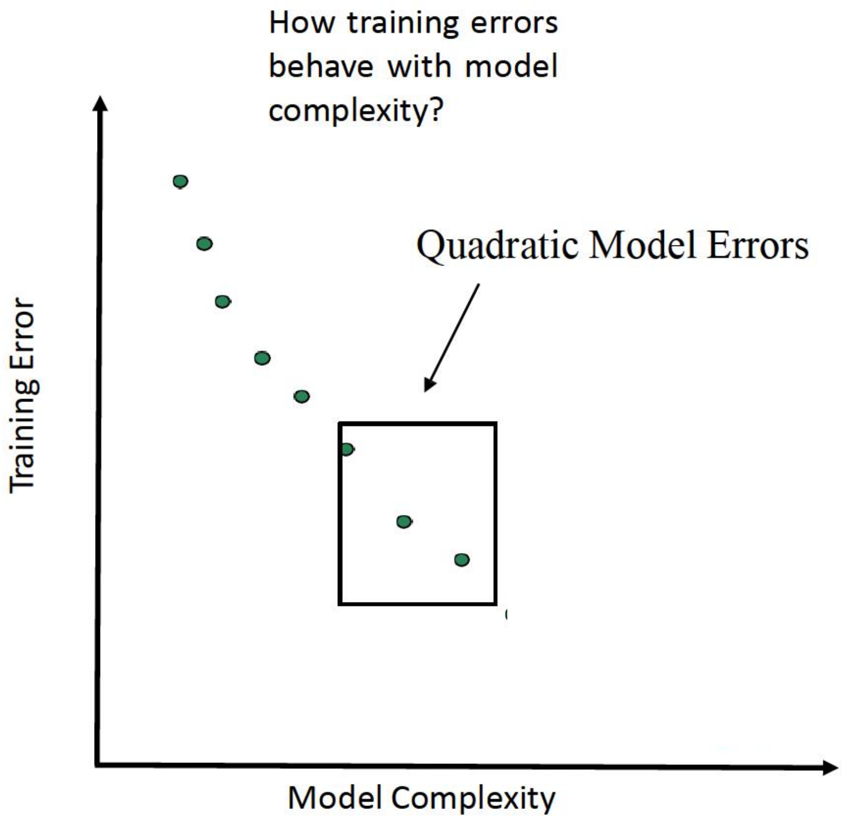
Training Error vs. Model Complexity (cont.)

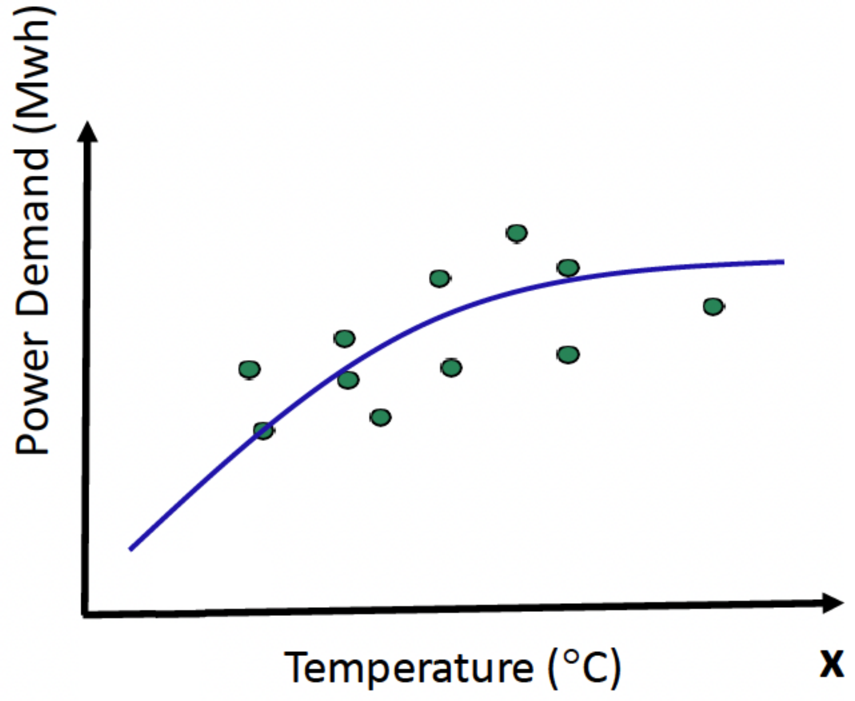




$$\hat{y} = \theta_0 + \theta_1 x_1$$

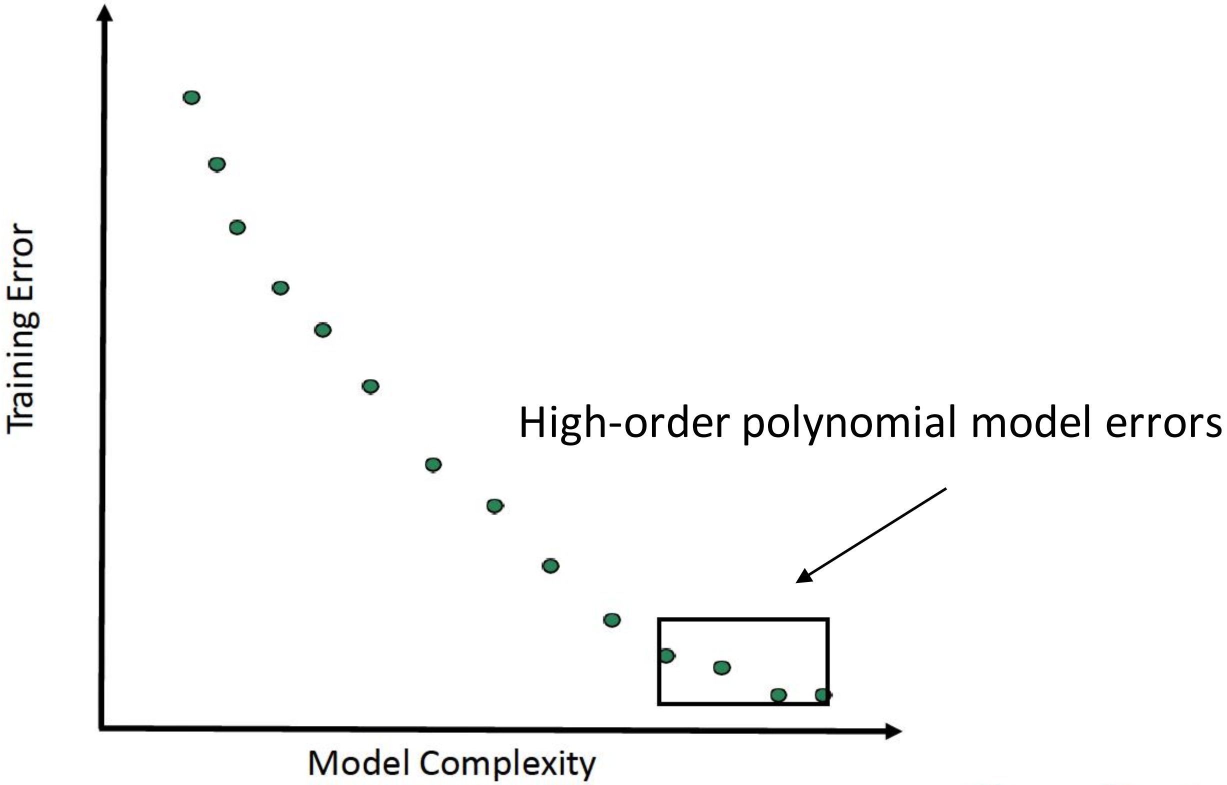
Training Error vs. Model Complexity (cont.)

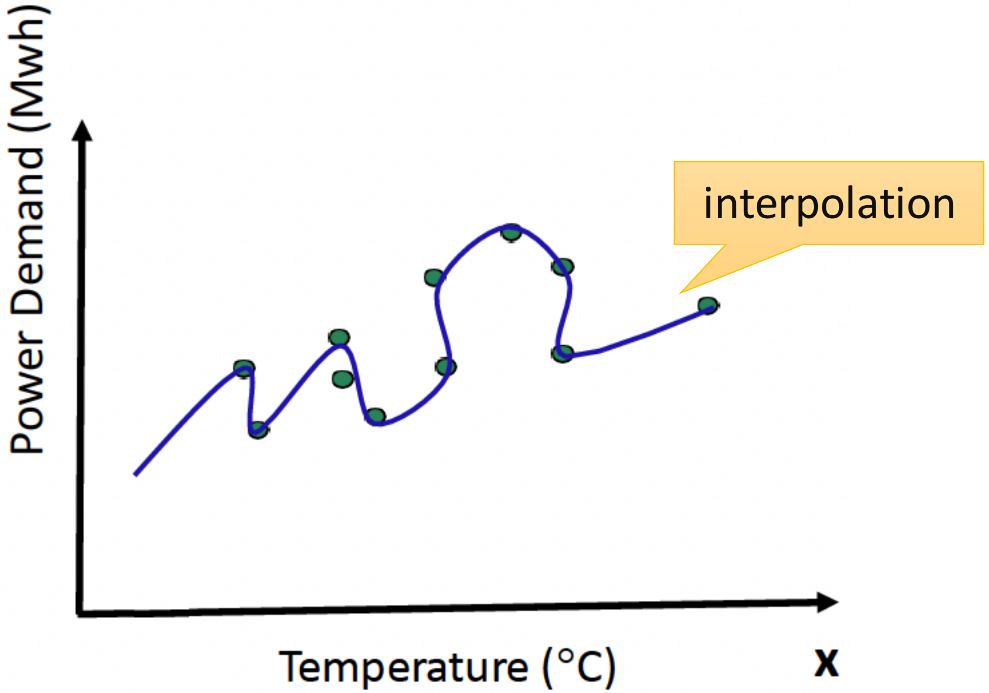




$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2$$

Training Error vs. Model Complexity (cont.)



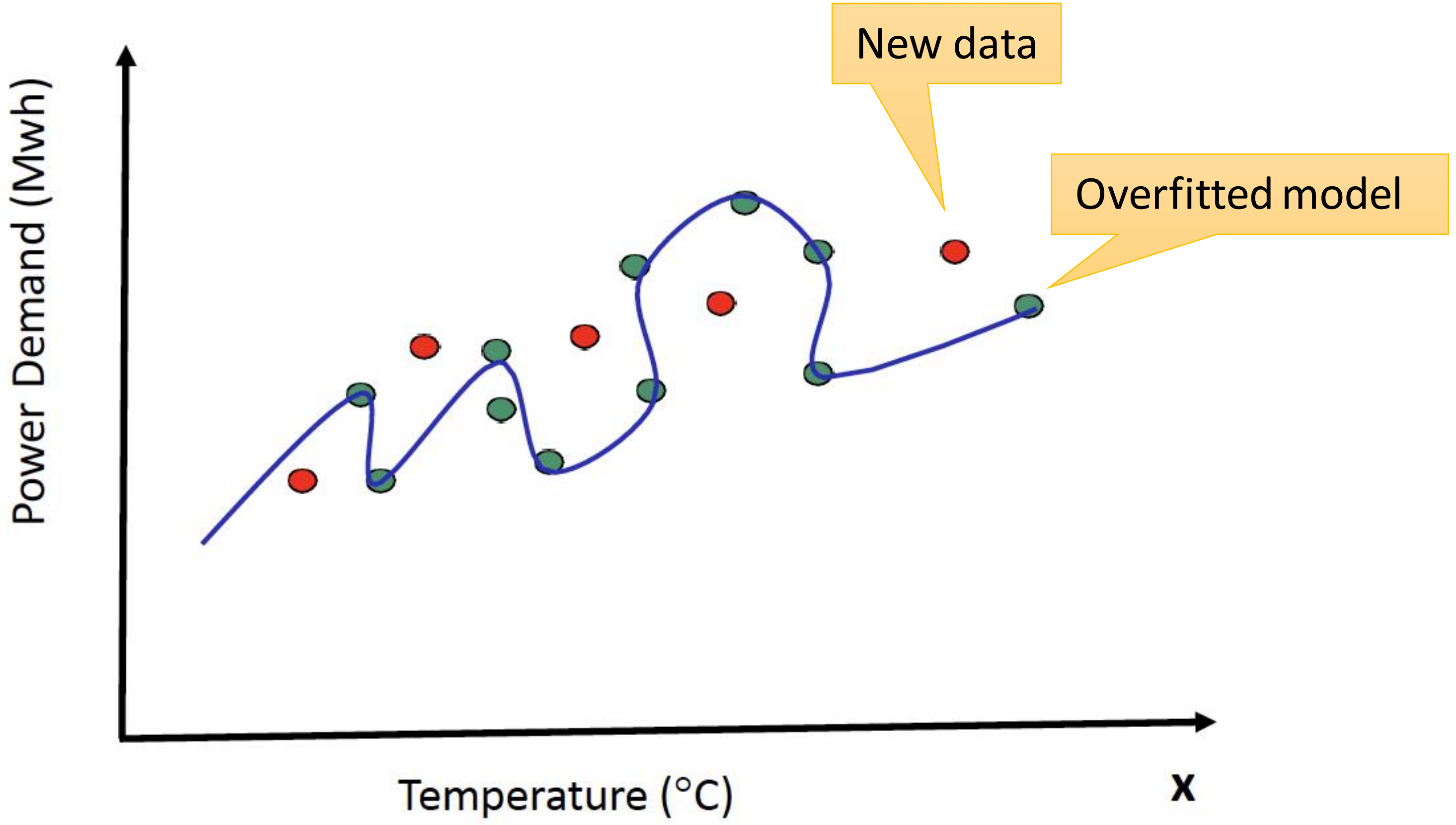


$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^7$$

Is **training error** a good measure of predictive performance?

Overfitted Models

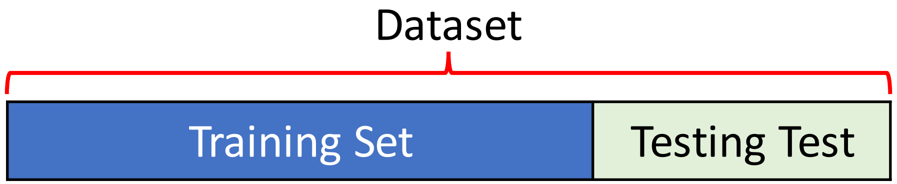
- How do we expect to perform on new data examples?
- Training error is overly optimistic.
- $\hat{\theta}$ were fit to the training data.



1.6: Evaluating Model Performance

Dataset Split: Training/Test

- Splitting a dataset is crucial to accurately assess your regression model's performance. The most common approach is to use a train-test split or a cross-validation technique (will see later…).



Too few → poorly estimated

Too few → test error gives be approximation of generalization error

- Typically, just enough test points to form a reasonable estimate of generalization error.
- Common split ratios are 70-30, 80-20, or 90-10, depending on the size of your dataset.
- The test set should be kept in a "vault," and be brought out only at the evaluation stage.
- Ensure that the split is done randomly to avoid any bias (will addressed later…) in the selection of data points.
- In case the data set is large, or the probability distribution is unknown, use **Random Walk Sampling.**

Testing Error

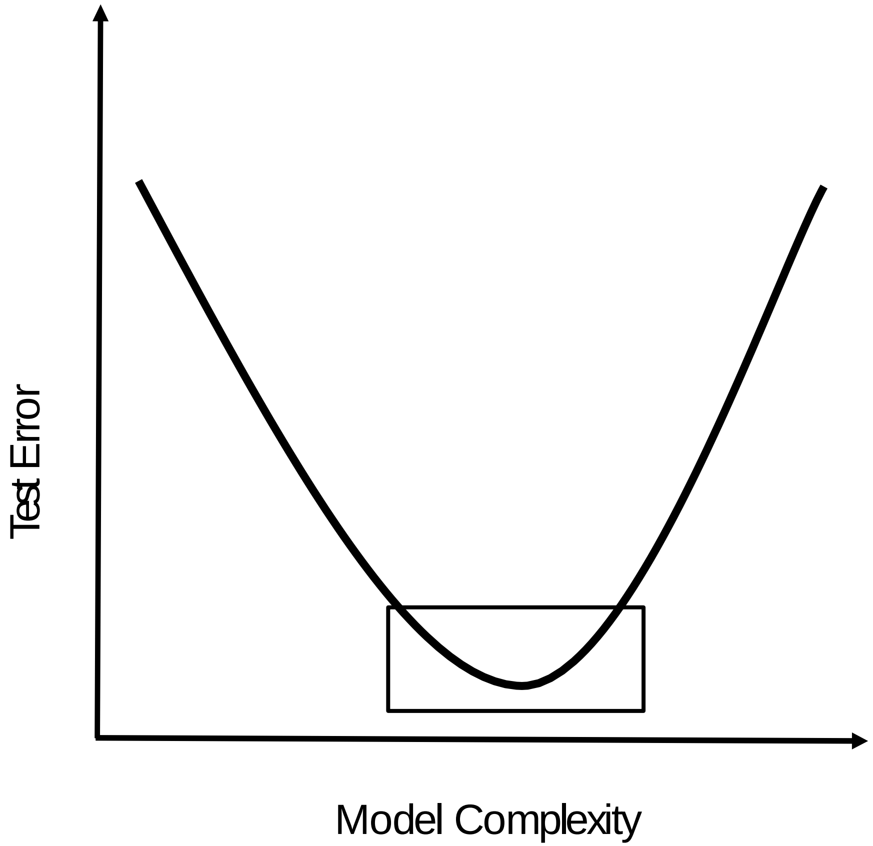
- Test error = average loss in the test set

$$= \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} L\big(y, f_{\hat{\theta}}(\boldsymbol{x})\big) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \left(y^{(i)} - f_{\hat{\theta}}(\boldsymbol{x}^{(i)})\right)^2$$

# test points examples

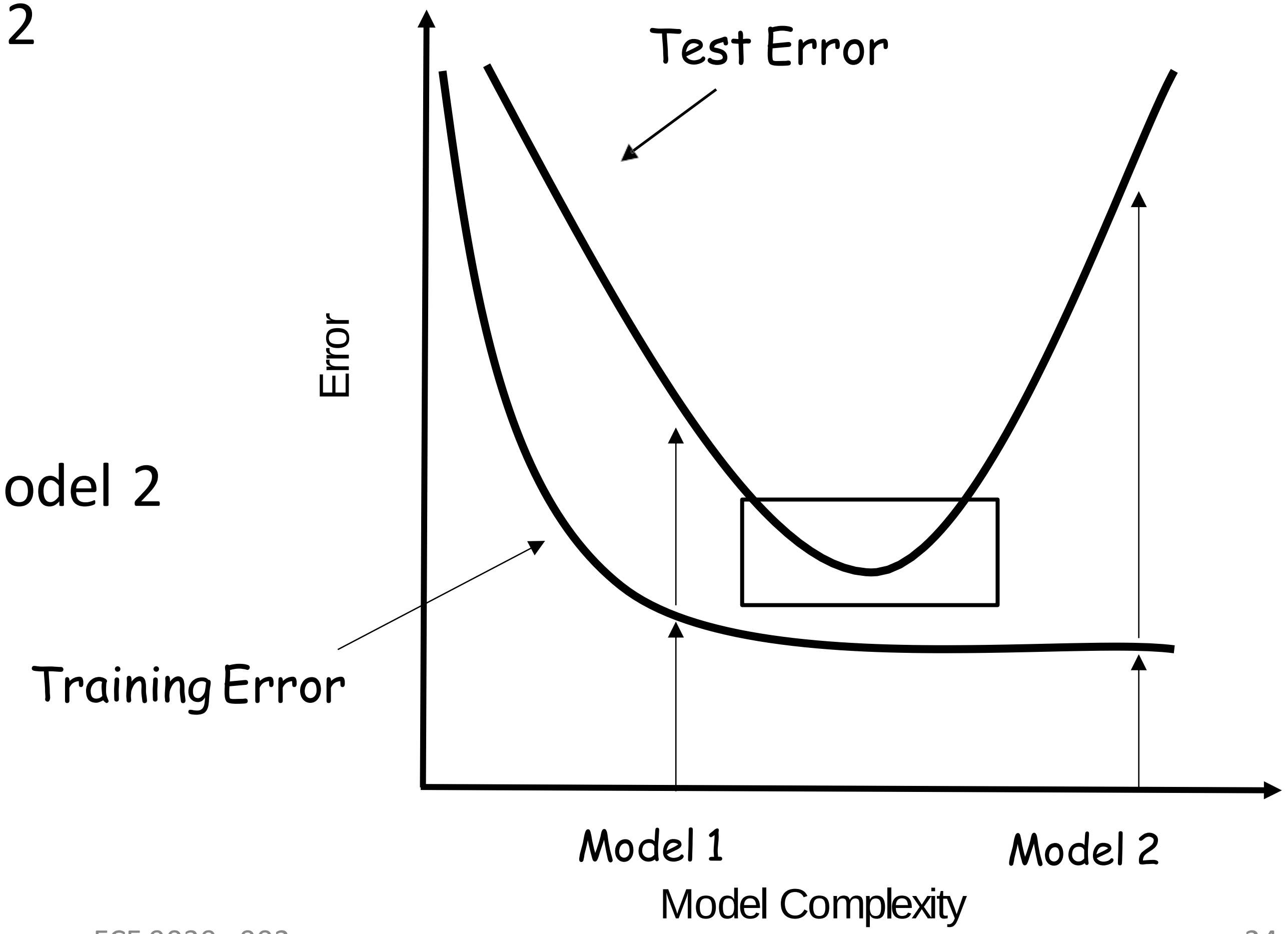
fit using training data

- has never seen test data!

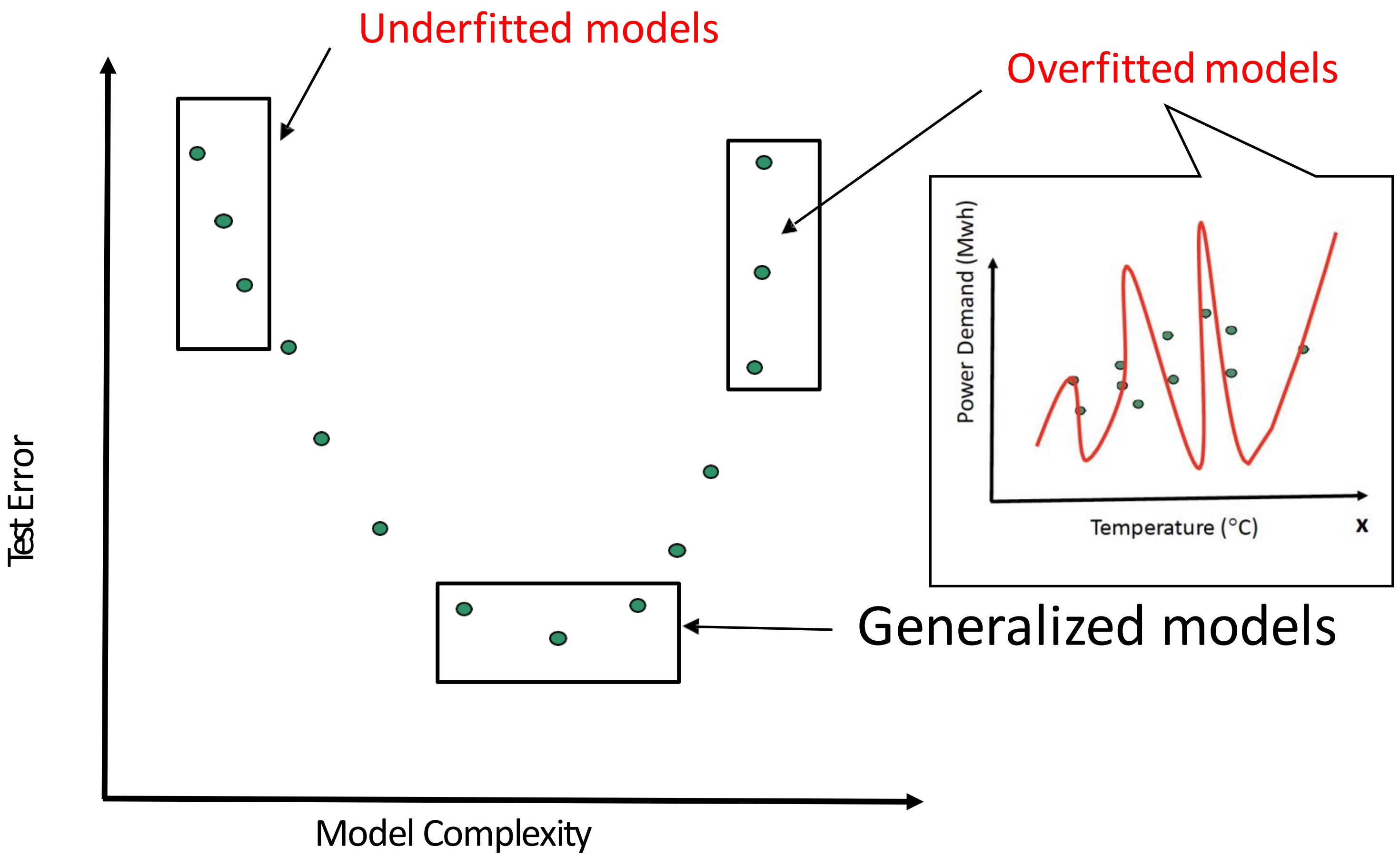


Testing Error (cont.)

- Training errors: Model 1 > Model2
- Test errors: Model 2 < Model1
- Overall: Model1 is better than Model 2

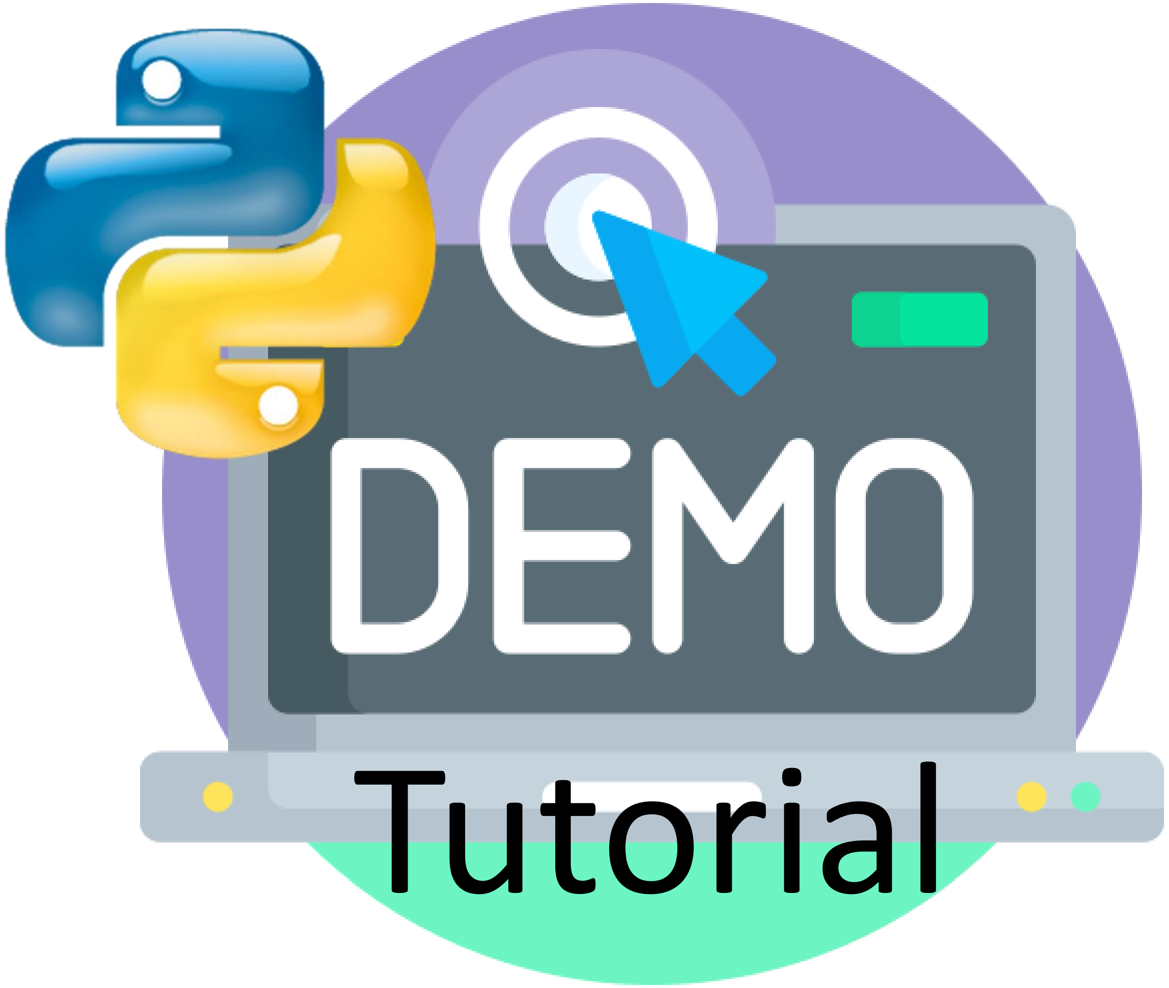


Generalization



Attendance

You can use the provided link if you don't have a mobile phone or if your phone lacks a QR-Code reader – https://rebrand.ly/ECEJan23



Tutorial

Sources of Error

In prediction models, there are 3 sources of error:

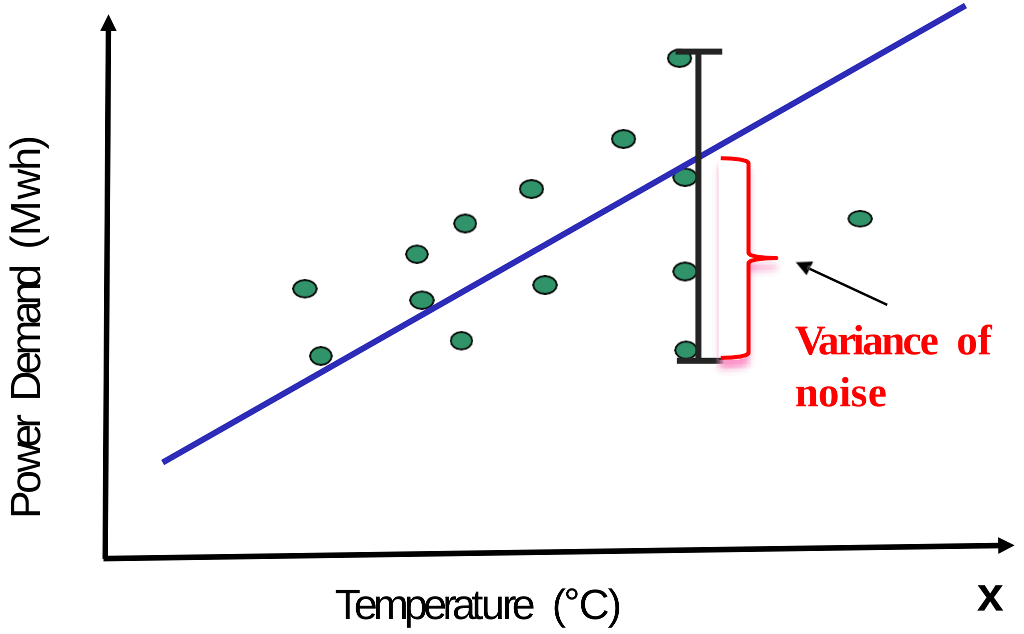
- **Irreducible Error (Noise)** – Irreducible error, also referred to as noise, represents the inherent randomness and unpredictability in the data.
  - No matter how complex or well-fitted a model is, there will always be some level of irreducible error present.
- **Bias** – in the context of prediction models, refers to the error introduced by overly simplistic assumptions in the modeling process.
- **Variance** – can be described as the degree of deviation of the model's predictions from the mean or average prediction.

Noise: Data is Inherently Noisy

- There are a lot of other contributing factors to the power demands, including other attributes that are not included, just temperature or how a person feels when they go in and use electricity (turn on a dishwasher, a washing machine, ..) or a personal social event (people gathering → more use of electricity).
- We have no control over this. It is a property of the data → Irreducible error; nothing we can do to reduce it → Unexplained Error
- Where

  $E[\epsilon]$: The expected value or mean of the error ϵ

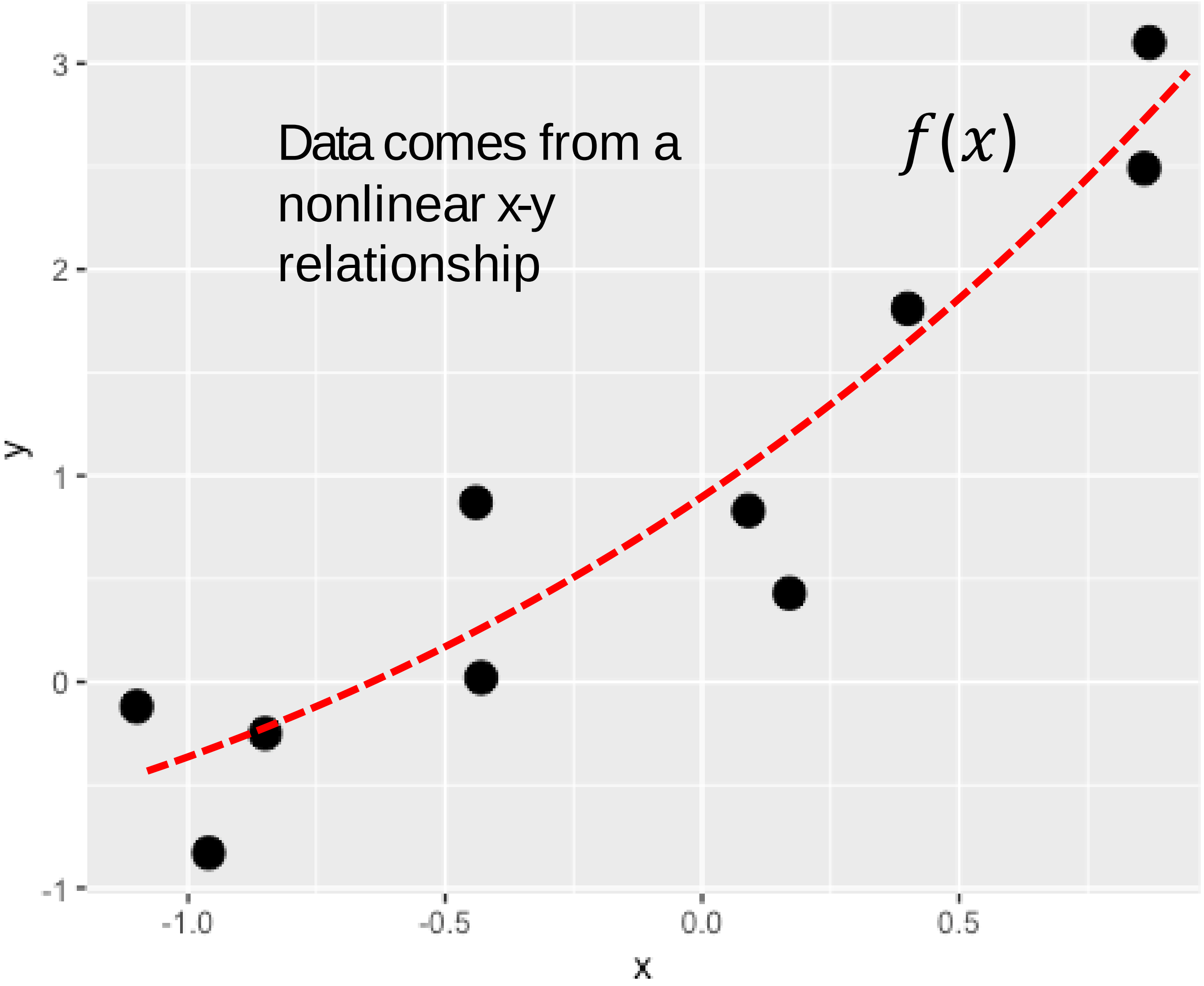
$$y_i = f(x_i) + \epsilon$$



$$E[\epsilon] = 0, Var[\epsilon] = \sigma_\epsilon^2$$

Bias

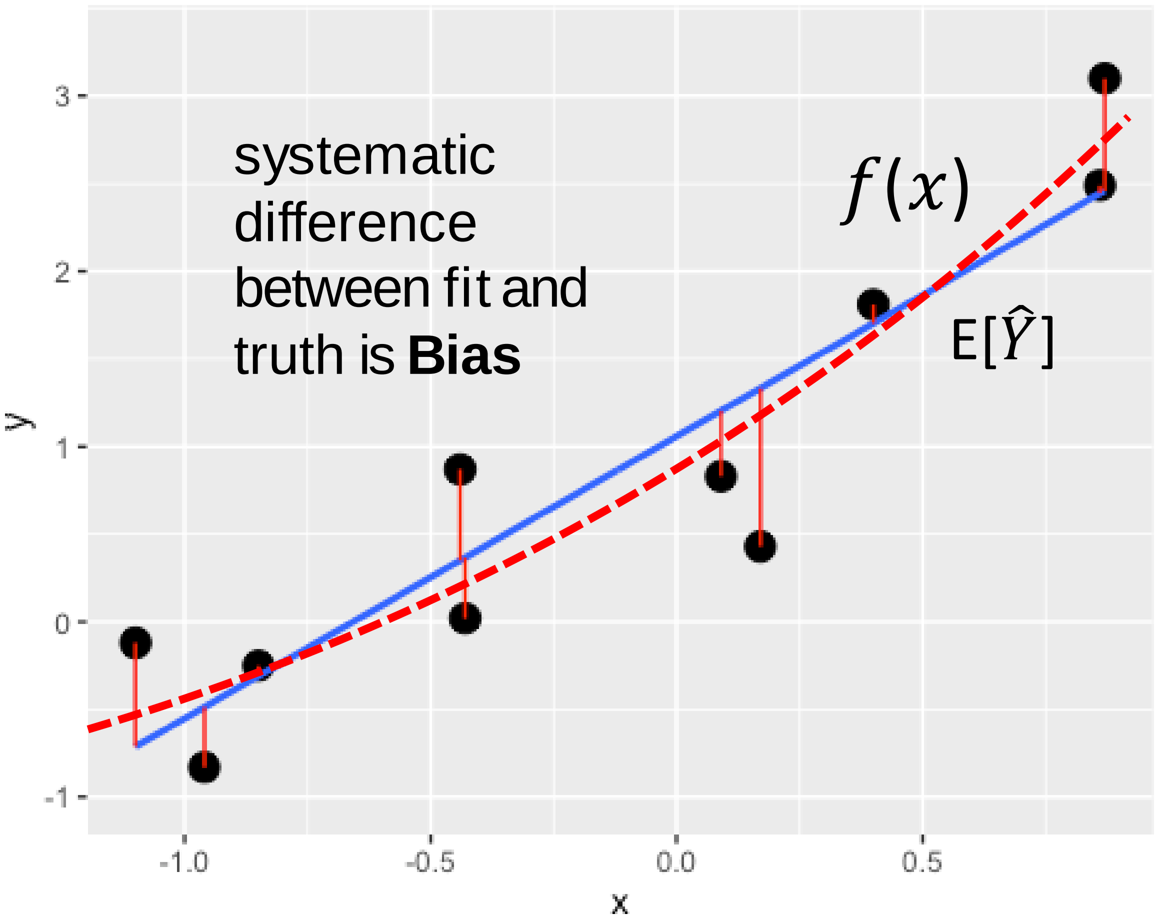
- The **bias** here refers to the error introduced by overly simplistic assumptions in the modeling process.
- This type of bias is different from the bias in a dataset (which is about the **representativeness of data**).



Bias

- Bias – The difference between the true relationship *(f(x))* and the best-fitted model.
- High bias can result in models that are too simplistic and unable to capture the underlying patterns in the data, **leading to poor predictions**.

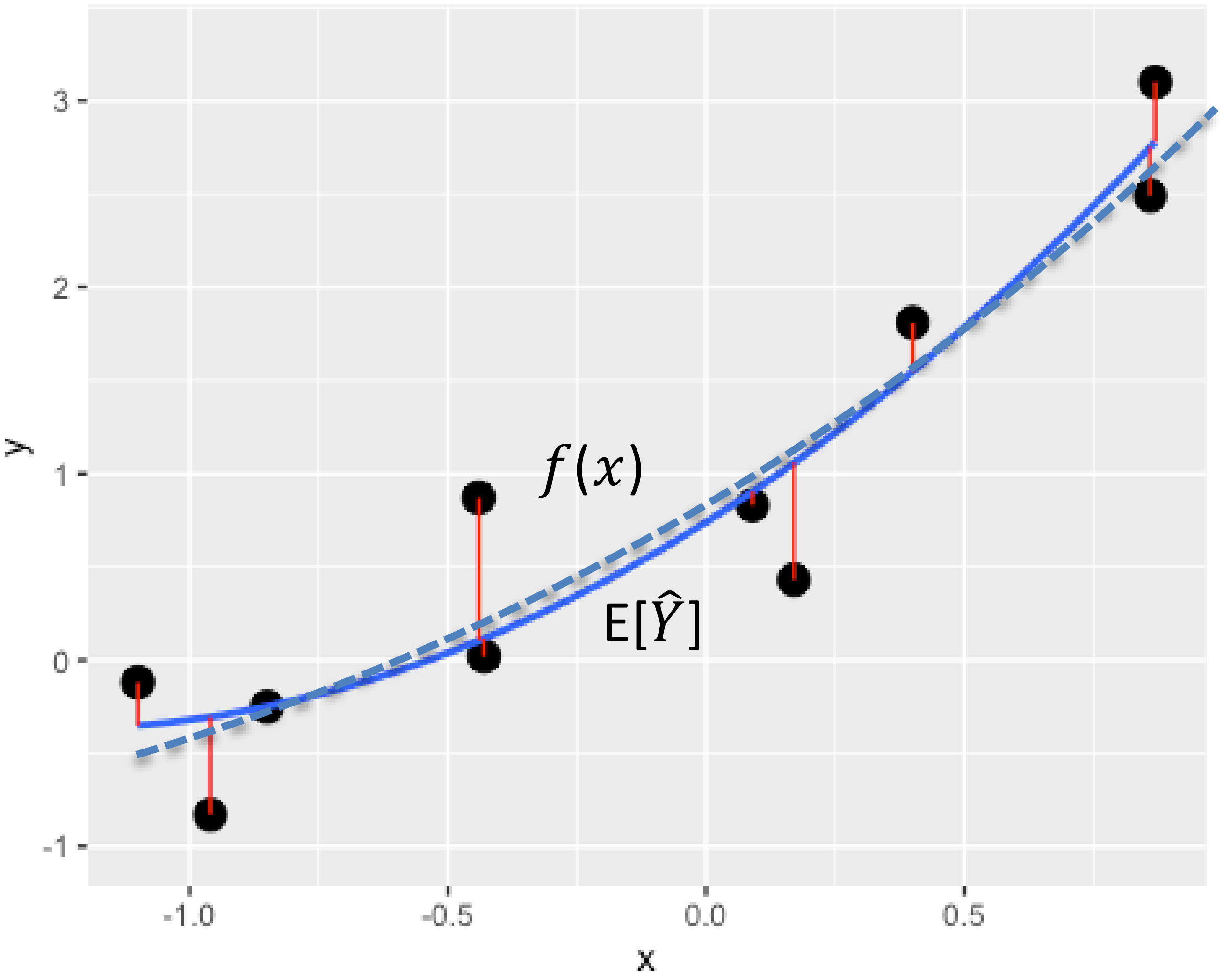
$$Bias = E\left[\hat{f}(x_i)\right] - f(x_i)$$



systematic difference between fit and truth is **Bias**

Bias

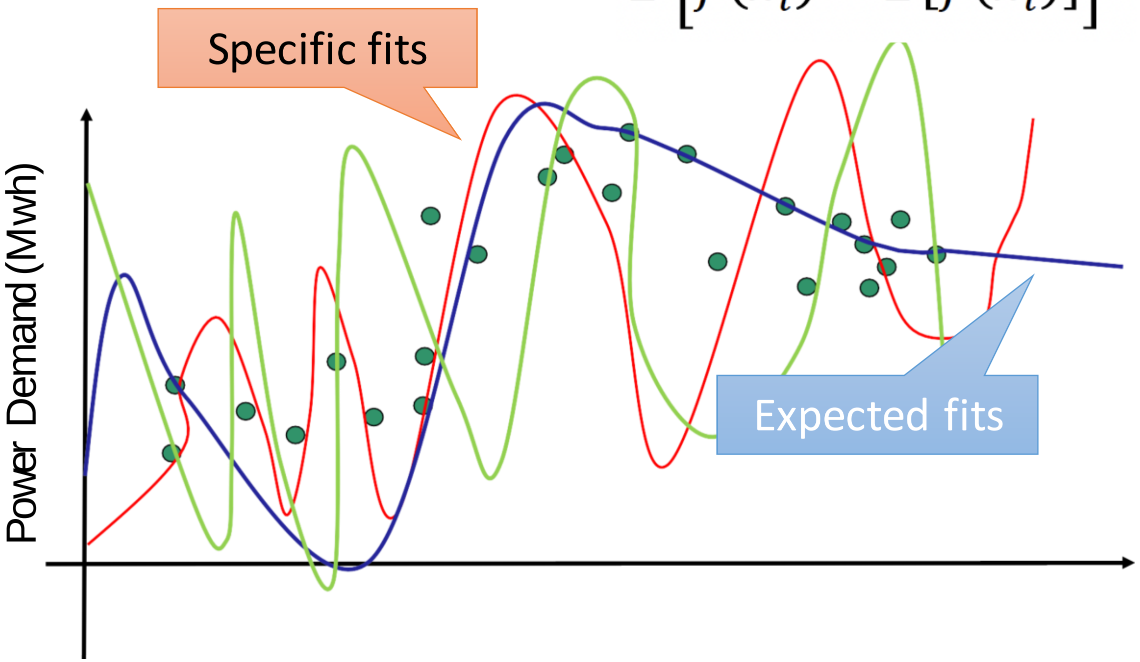
- As the model becomes **complex enough** to model f (x), the bias disappears.



Variance

- How much do **specific fits** vary from the **expected fits**?
- Variance can be described as the degree of **deviation** of the model's predictions from the <u>mean or average prediction</u>.
- High-variance models are overly complex and tend to fit the training data **too closely**, capturing not only the underlying patterns but also the disturbance.
- This can lead to poor generalization of new, unseen data, as <u>the model may become too specific to the training set</u>.

$$E\left[\hat{f}(x_i) - E[f(x_i)]\right]^2$$



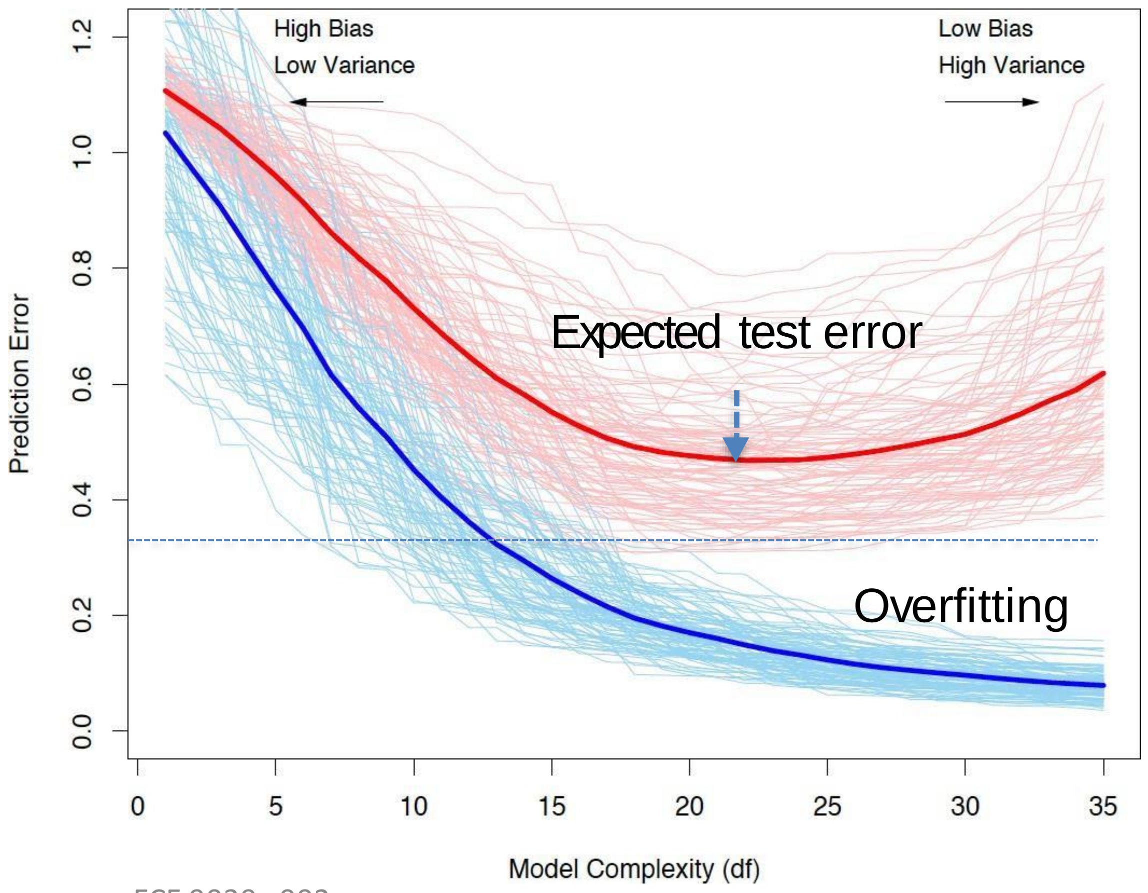
If the model is too complex, the fit becomes very variable

Bias/Variance Trade-off

- Low Bias, High Variance:
  - Highly complex models with many parameters have low bias but high variance. They excel at fitting the training data but <u>may struggle to generalize due to memorizing it</u>.
- High Bias, Low Variance:
  - Simple models with few parameters show high bias and low variance.
  - They may miss the <u>true patterns</u> in the data.

Bias

Variance

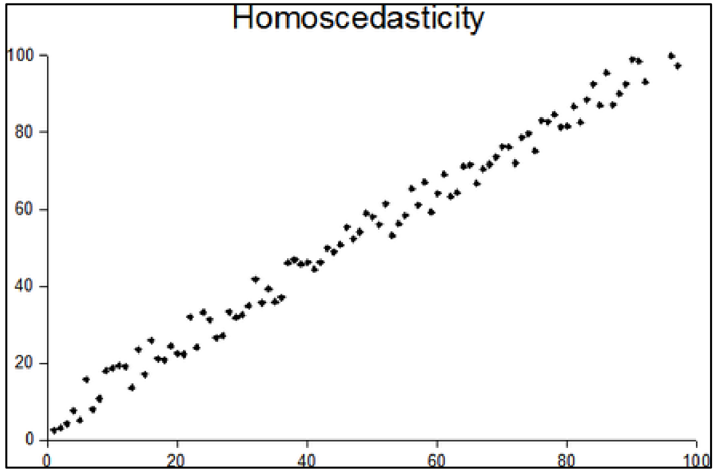


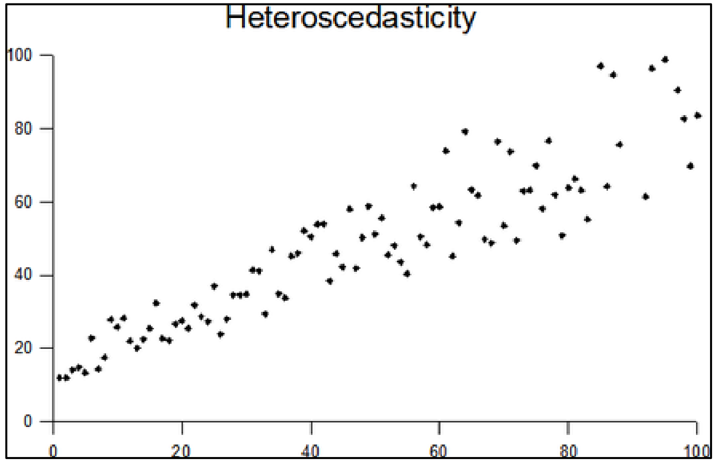
Test error

Training error

Homo vs. Hetero (scedasticity)

- In the context of linear regression, we assume that the residuals are distributed evenly for all values of the independent variables. This assumption is known as **homoscedasticity**.
- When this assumption is violated, and the scatter or spread of the residuals varies at different levels of the independent variable(s), the situation is referred to as **heteroscedasticity**.
- Heteroscedasticity can affect the variance of the estimator in a regression model.
  - Applying transformations (like logarithmic) can stabilize the variance of residuals.





R^2 score

- R^2, also known as the R^2 score, is a statistical metric employed in the context of linear regression analysis. It quantifies the percentage of variance in the dependent variable that can be explained by the independent variables.
- The R^2 value ranges from 0 to 1. A higher R^2 value suggests a more effective alignment between the model and the observed data.
- For instance, an R^2 value of 0.70 indicates that the model explains 70% of the variation observed in the dependent variable, attributed to the independent variables.

- The formula to calculate R^2 in the context of a simple linear regression model is:

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

Where:

- SS_{res} is the sum of squares of residuals.
- SS_{tot} is the total sum of squares (proportional to the variance of the data).

$$SS_{\text{tot}} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Where $\bar{y}$ is the mean of the actual values y_i

2.1: Feature Construction

Feature Construction

- Creating, manipulating, ranking, and selecting relevant features from raw data is a crucial step in the process of building effective machine learning models.
- **Feature Creation**
  - **Domain Knowledge** – Start by gaining a deep understanding of the problem domain. Domain knowledge can help you create or find new features that are meaningful and relevant.
  - **Feature Engineering** – Generate new features from the existing ones. This could involve mathematical transformations, or combining features. For example, you might calculate ratios, differences, or percentages between variables.

House Price Example

- **Objective –** Build a model to predict the **price** of a house. A set of inputs have been given: the square footage of the house, the size of the lot, the number of rooms, how much was sold in the past, the location, and the number of concrete blocks in the driveway, etc.
- How do we know what features to use? Or what makes even a good feature?

Example cont.

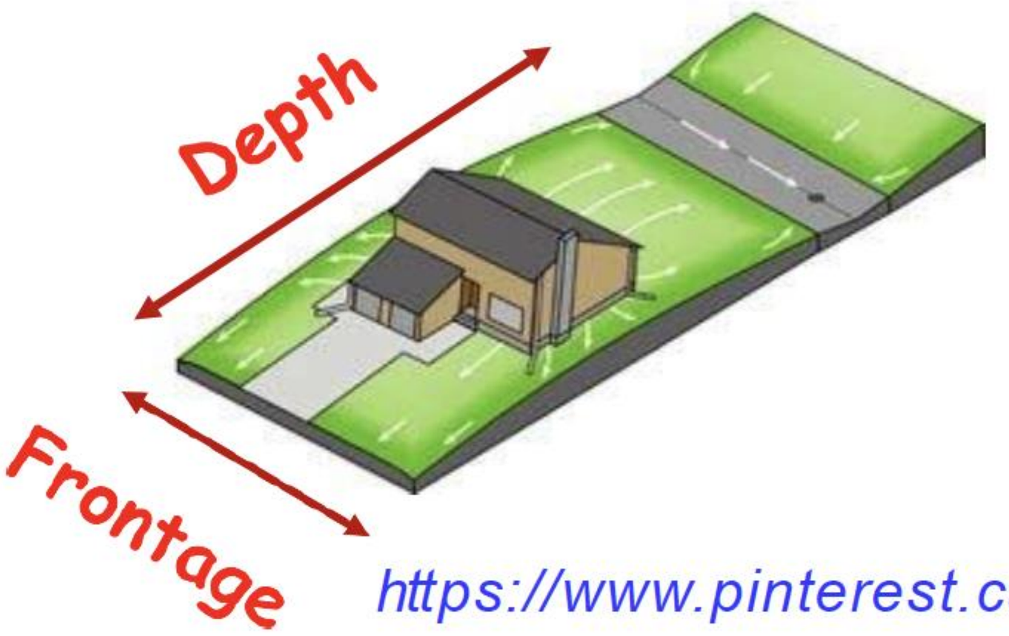
- Suppose that we are given two features. The frontage of the house and the depth of the house.
- Linear Regression Model:

$$\hat{y}_i = \theta_0 + \theta_1 Frontage_i + \theta_2 Depth_i$$

- Construct a new feature:

$$Area_i = Frontage_i \times Depth_i$$

- Now, the linear regression model will be $\hat{y}_i = \theta_0 + \theta_1 Area_i$



https://www.pinterest.ca/

- **Multicollinearity** in regression analysis refers to a situation where two or more predictor variables (independent variables) are highly correlated. In other words, one predictor variable in a regression model can be linearly predicted from the others with a substantial degree of accuracy.

Feature Manipulation: Scaling and Normalization

- Standardize or normalize numerical features to have similar scales.
- This can help algorithms converge faster and make the model less sensitive to the magnitude of the features.

Min-Max Scaling

- Suppose you have a dataset with two numerical features: "Age" and "Income." The "Age" values range from 0 to 100, while the "Income" values range from 20,000 to 200,000.
- Min-Max Scaling (Normalization):
  - Min-Max scaling scales the data to a specific range, typically [0, 1].
  - The formula to perform Min-Max scaling on a feature is: $X_N = (X - X_{min}) / (X_{max} - X_{min})$
- Example:

Age	Income
25	50000
35	75000
45	100000
60	150000
30	80000

Before

Age (Normalized)	Income (Normalized)
0.0	0.0
0.25	0.25
0.5	0.5
0.75	0.75
0.2	0.3

After

Z-Score Standardization

- **Z-Score Standardization –** Z-score standardization transforms data to have a mean (average) of 0 and a standard deviation of 1.
- For a random variable X with a dataset of values {x_1, x_2, x_3, ..., x_n}, the Z-score (standardized score) of a specific value x_i is calculated as: $Z_i = \frac{x_i - \bar{X}}{\sigma}$
- Where:
  - Z_i is the Z-score for the *i*-th data point.
  - x_i is the value of the *i*-th data point.
  - $\bar{x}$ is the mean (average) of the dataset,

  calculated as: $\bar{X} = \frac{1}{n} \sum_{i=1}^{n} x_i$
  - σ is the standard deviation of the dataset,

  calculated as: $\sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{X})^2}$

Example:

Age	Income
25	50000
35	75000
45	100000
60	150000
30	80000

Before

Age (Standardized)	Income (Standardized)
-1.161895	-1.161895
-0.387298	-0.387298
0.387298	0.387298
1.549193	1.549193
-0.387298	-0.387298

After

Feature Manipulation: Missing Values and Encoding

- **Handling Missing Values** – Address missing data by imputing missing values, removing rows or columns with excessive missing data, or using advanced imputation techniques.
- **Encoding Categorical Variables** – Convert categorical variables into numerical format through techniques like one-hot encoding or label encoding.

Example

- **Handling Missing Values**
  - **Imputation** – We can fill missing values with the mean or median of the respective columns.

Age	Income	Marital Status
25	50000	married
35	NaN	Not married
45	100000	married
NaN	150000	married
30	80000	Not married

Age	Income	Marital Status
25	50000	married
35	95000	Not married
45	100000	married
34	150000	married
30	80000	Not married

Age	Income	Married	Not married
25	50000	1	0
35	95000	0	1
45	100000	1	0
34	150000	1	0
30	80000	0	1

  - **Encoding Categorical Variables** – To use the "Marital Status" variable in a machine learning model, we need to convert it into **numerical** format. One common approach is **one-hot encoding**, where we create binary columns for each category.
- We handled missing values in the "Income" and "Age" columns through imputation (filling with mean values), and we encoded the "Marital Status" categorical variable into binary columns using one-hot encoding.

Feature Manipulation: Feature Ranking

- Features with high **absolute correlations** are often good candidates for inclusion.
- Correlation analysis could be performed by computing the **Pearson correlation coefficient**, commonly referred to as **Pearson's r**. This coefficient assists in quantifying the linear association between two numerical variables.
- The formula for the Pearson correlation coefficient (r) is

as follows:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

- Where
  - X_i and Y_i are the individual data points for "Age" and "Income.”
  - $\bar{X}$ and $\bar{Y}$ are the means of "Age" and "Income," respectively.
  - *n* is the number of data points.

- A positive value indicates a positive correlation, while a negative value indicates a negative correlation.
- The closer the absolute value of *r* is to 1, the stronger the correlation.

2.2: Feature Selection

Feature Selection

- Lot size
- House type
- Year built
- Last sold price
- Last sale price
- Finished sqft
- Unfinished sqft
- Finished basement sqft
- # of floors
- Flooring types
- Parking type
- Parking amount
- Cooling
- Roof type
- Exterior materials
- Structure style
- Dishwasher
- Garbage disposal
- Microwave
- Sprinkler System
- Refrigerator
- Washer
- Dryer
- Laundry location
- Heating type
- Jetted Tub
- Deck
- Fenced Yard
- Lawn
- Garden

Feature Selection

- Why might you want to perform feature selection?
  - Some or many of the features used in a multiple regression model are, in fact, not associated with the response.
  - Irrelevant features lead to unnecessary complexity in the resulting model.
- Which features are relevant to the prediction? Interpretability
- In the context of linear regression, "interpretability" refers to the ability to understand and make meaningful sense of the relationship between the predictor variables (independent variables) and the target variable (dependent variable) within the model.

Feature Selection

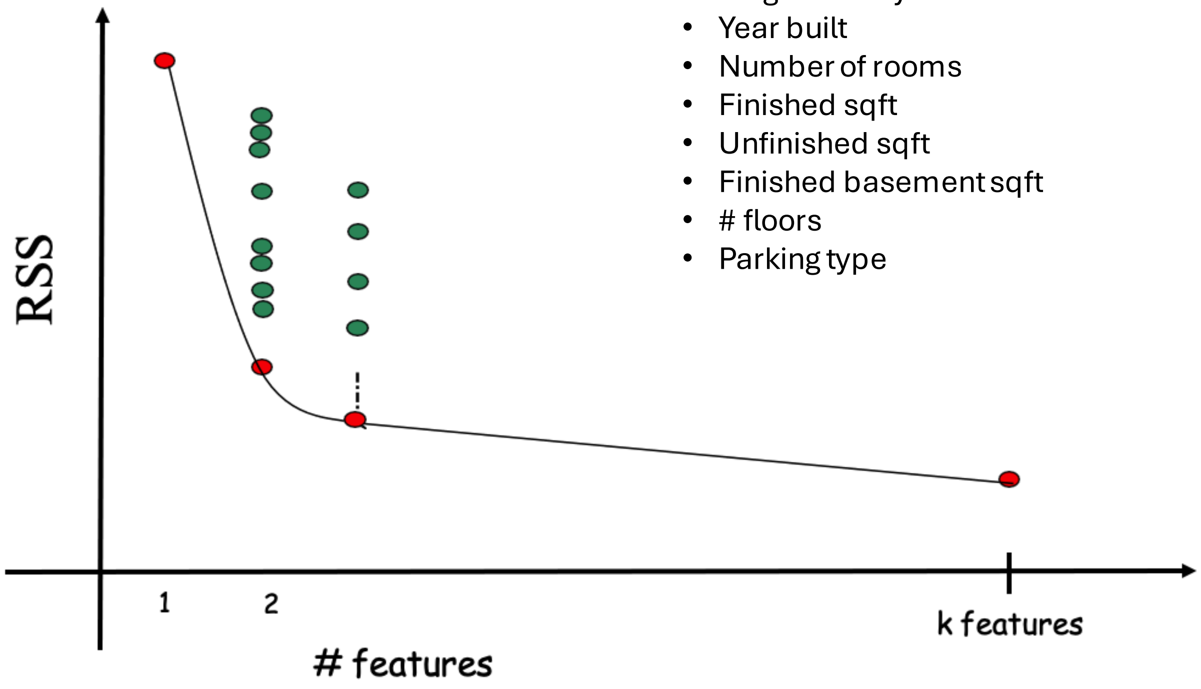
- **Subset Selection** – This approach involves identifying a subset of *k* predictors that we believe to be related to the response.
  - **Best Subset Selection** – Search over every possible combination of features we might want to include in our model and look at the performance of each of those models
  - **Stepwise Selections** – Stepwise methods explore a far more restricted set of models: attractive alternatives to best subset selection: **Forward Stepwise Selection**, **Backward Stepwise Selection**
  - Both consider a much smaller set of models compared to the <u>best subset selection</u>
- **Dimension Reduction** – This approach involves projecting *k* predictors into a *M* dimensional subspace, where *M* < *k*

Best Subset Selection

- To perform best subset selection, we fit a separate least squares regression best subset for each possible combination of the *k* predictors.
- The set of all *n*-combinations of a set of size *k*
- So with *K* features, we have 2^K possibilities of combining them.

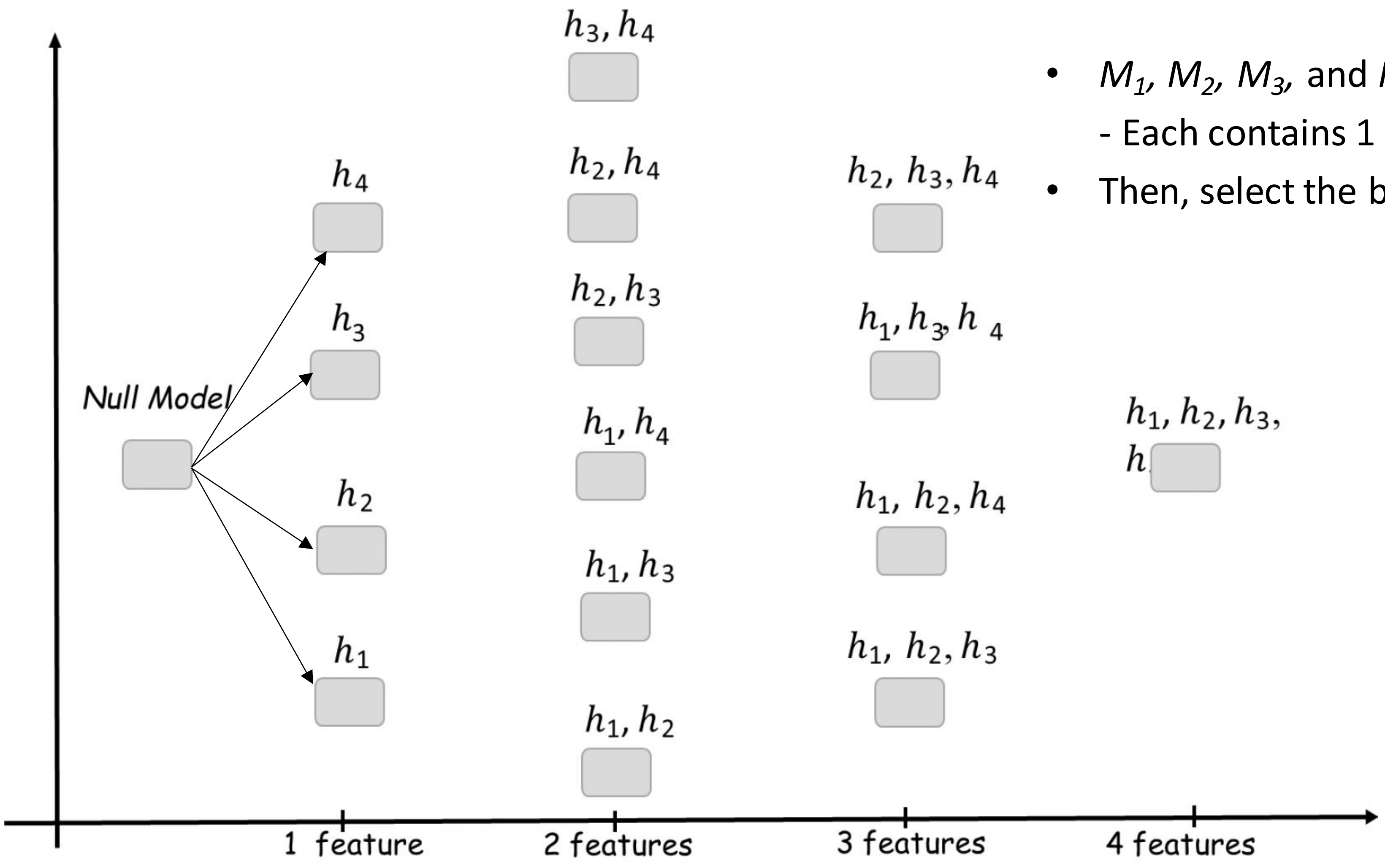
$$\binom{k}{n} = \frac{k!}{n!\,(k-n)!}$$

- Lot size
- Single-Family
- Year built
- Number of rooms
- Finished sqft
- Unfinished sqft
- Finished basement sqft
- # floors
- Parking type

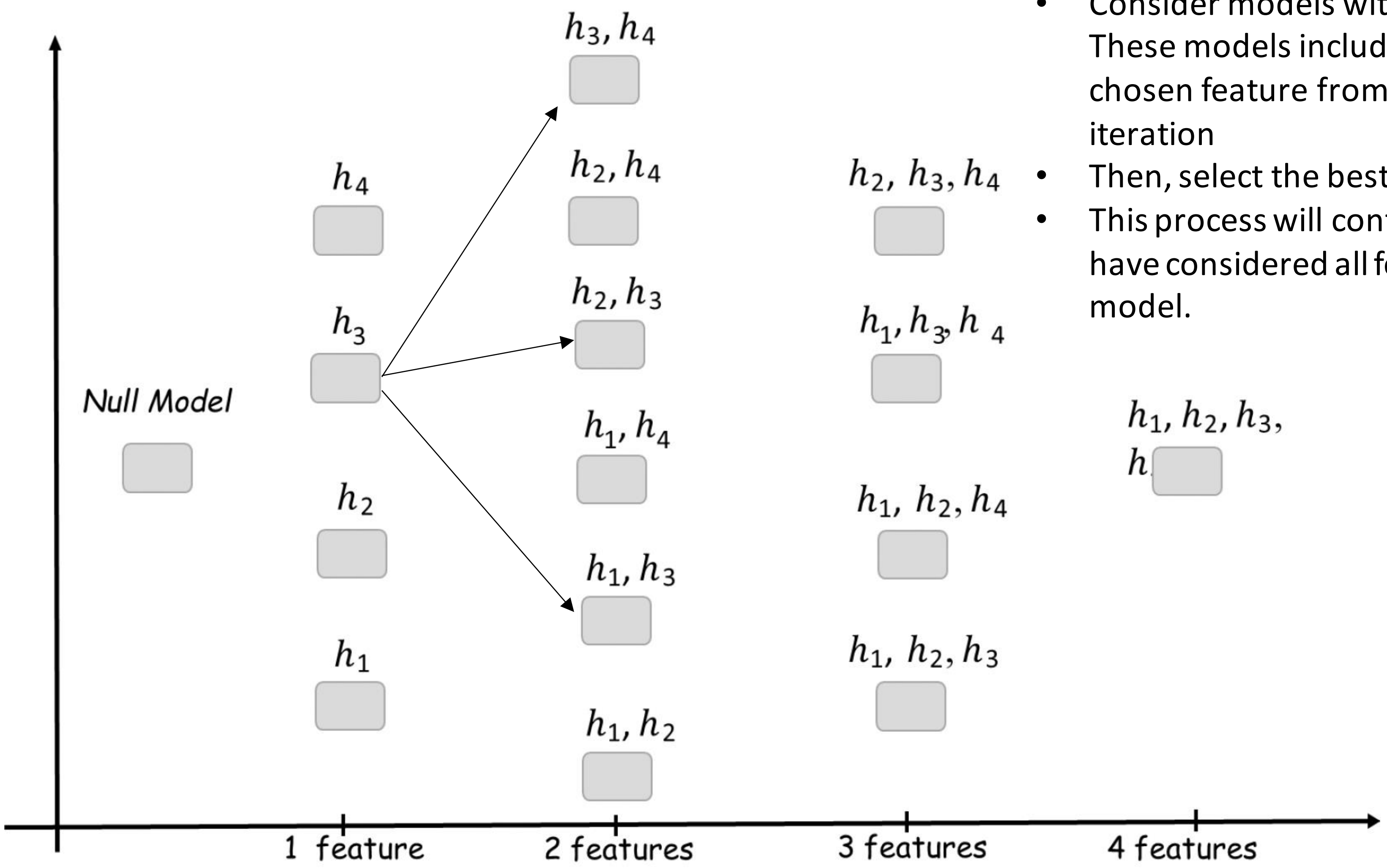


Forward Stepwise Selection

- Forward stepwise selection begins with a model containing one feature and then adds features to the model, one at a time, until all of the features are in the model.
- At each step, the feature that gives the greatest additional improvement to the fit is added to the model



- M_1, M_2, M_3, and M_4 models
  - Each contains 1 feature
- Then, select the best model



- Consider models with 2 features - These models include only the chosen feature from the previous iteration
- Then, select the best model
- This process will continue until you have considered all features in your model.

Backward Stepwise Selection

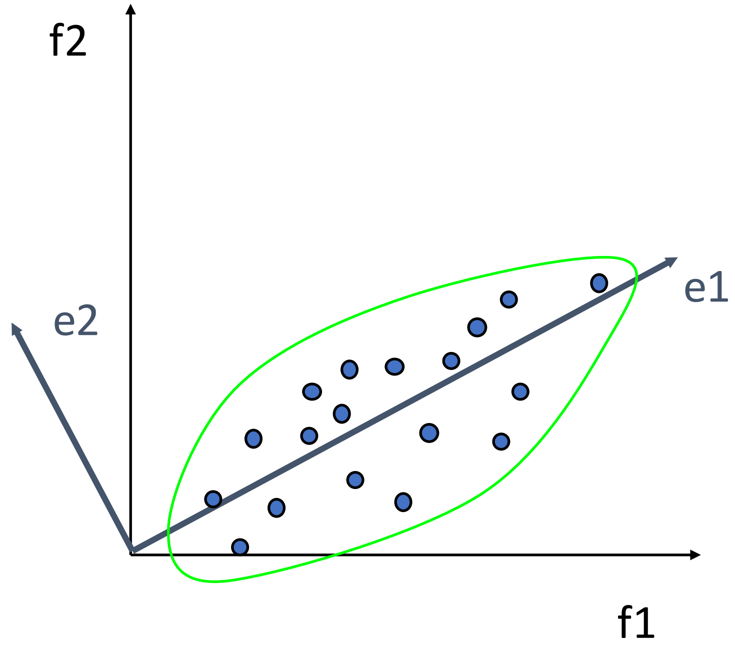
- In backward stepwise selection, you start with the full model (containing all available predictor variables or features) and progressively eliminate the least valuable feature one at a time.
- You retain the best-performing models throughout this process.

Dimension Reduction

- Dimension reduction involves reducing the number of **predictors** while retaining the essential information needed to make accurate predictions.
- One commonly used approach in dimension reduction for linear regression is projecting the original *k* predictors into a lower-dimensional subspace M, where M < *k*.
- This process can be achieved through techniques like Principal Component Analysis (PCA).

Principal Components Analysis (PCA)

- Principal components analysis (PCA) is a technique that linearly transforms and chooses a new coordinate system for the dataset such that the greatest variance by any projection of the dataset comes to lie on the first axis (then called the first principal component), the second greatest variance on the second axis, and so on.
- PCA can be used for reducing dimensionality by eliminating the later principal components.



Principal Components Analysis (PCA)

- To apply PCA on a dataset, you need to follow the following steps:
    1. Standardize the Data – PCA is affected by scale, so it's important to scale the features in the data before applying PCA.
    2. Compute the Covariance Matrix – This matrix represents the covariance between each <u>pair of features</u> in the data.
    3. Calculate the Eigenvalues and Eigenvectors of the Covariance Matrix – These will determine the principal components.
    4. Sort Eigenvalues and Eigenvectors – Sort the eigenvalues and their corresponding eigenvectors in descending order. The eigenvectors with the highest eigenvalues are the principal components.
    5. Project the Data Onto the Principal Components – This will result in a new dataset of possibly lower dimensions.

Next

Thank you