Sixth Session

Alireza Moradi

Multiple linear regression (MLR)

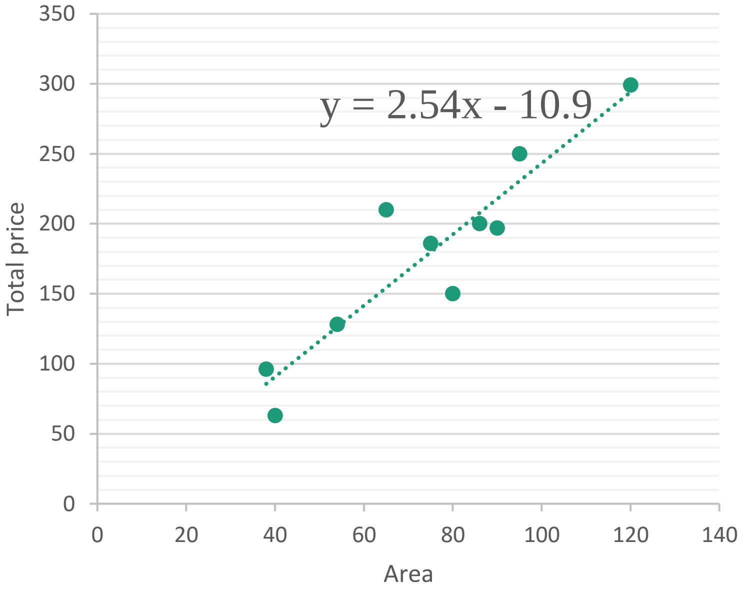
- Linear regression (discussed in last session) is a simple approach to supervised learning. It assumes that the relationship between Y and X is linear.

- Although it may seem overly simplistic, linear regression is extremely **useful both conceptually and practically**.

- We described linear regression with one independent variable. Recall that our model was :

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 \cdot X_i$$

- In Multiple linear regression (MLR), we have multiple independent variables.



Multiple linear regression

Example:

Observation	Area(m^2)	Number of bedrooms	Age(years)	Total Price($ K)
1	54	1	12	128
2	75	2	2	186
3	80	1	5	150
4	40	0	30	63
5	38	0	5	96
6	120	3	10	299
7	90	1	12	197
8	95	1	7	250
9	86	2	1	200
10	65	1	5	210

Multiple linear regression

- In Multiple Linear Regression, our model is :

$$y = \beta_0 + \beta_1 . X_1 + \beta_2 . X_2 + \cdots + \beta_p . X_p$$

- Given estimates for $\beta_0, \beta_1, \ldots, \beta_p$ we can make predictions using the following formula:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \cdot X_1 + \hat{\beta}_2 \cdot X_2 + \cdots + \hat{\beta}_P \cdot X_P$$

- For linear regression problems, two common methods for finding the line of best fit are **Ordinary Least Squares (OLS)** and **Gradient Descent**.

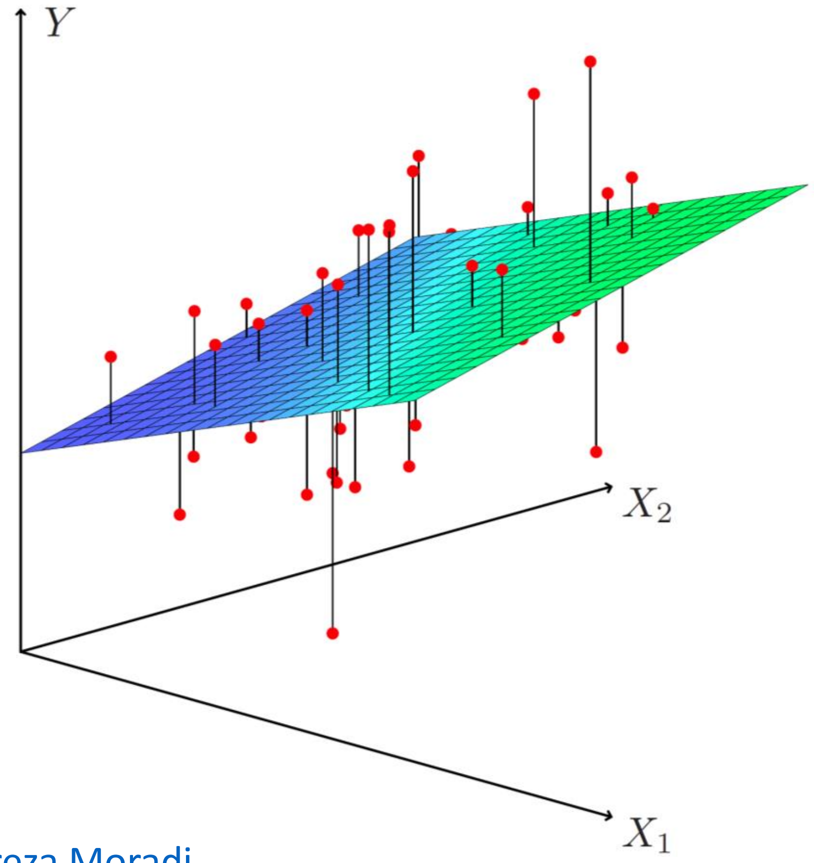
Multiple linear regression

- Linear regression with more than one features form a hyperplane.

- For example, with 2 features, we will have a **plane** as our model.

✓ A **plane** is a 2-dimensional hyperplane embedded in a 3-dimensional space.

✓ A **line** is a 1-dimensional hyperplane embedded in a 2-dimensional space.



MLR; Standard notation

- $x_j \Rightarrow j\text{-}th$ feature $\quad \forall\, j = 1,2,\cdots,p$

p = number of features

- X $\Rightarrow$ Vector of features:

$$X = [x_1, x_2, \ldots, x_p]$$

- $X^{(i)} \Rightarrow$ features of $i\text{-}th$ training example

$$\forall\, j = 1,2,\cdots,n$$

n = number of observations

- $x_j^{(i)} \Rightarrow$ value of feature j in $i\text{-}th$ training example

Observation	Area(m^2)	Number of bedrooms	Age(years)	Total Price($ K)
1	54	1	12	128
2	75	2	2	186
3	80	1	5	150
4	40	0	30	63
5	38	0	5	96
6	120	3	10	299
7	90	1	12	197
8	95	1	7	250
9	86	2	1	200
10	65	1	5	210

Vectorization

- In ML, **vectorization** is the process of converting data into numerical vectors. These vectors then become the input for ML algorithms.

- Without vectorization:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

- With vectorization:

$$y = \beta_0 + \beta \, . \, X$$

Dot product

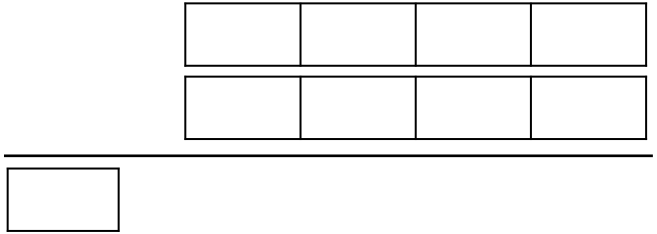
$$\beta = [\beta_1, \beta_2, \dots, \beta_p]$$
$$X = [x_1, x_2, \dots, x_p]$$

Vectorization

- Vectorization allows performing operations on multiple data elements simultaneously, which can lead to faster computations.

- For example:

$$y = 2 + 2.3 + 3.1 + 4.2 + 2.4$$



- This optimization leverages the parallel processing capabilities of modern CPUs and GPUs.

- In a nutshell, Vectorization can often enable **parallelized computations**, potentially leading to significantly **faster calculations** for specific operations on large datasets, leveraging the capabilities of modern processors.

Gradient Descent in MLR

- We can modify gradient decent algorithm from one feature to multiple features.

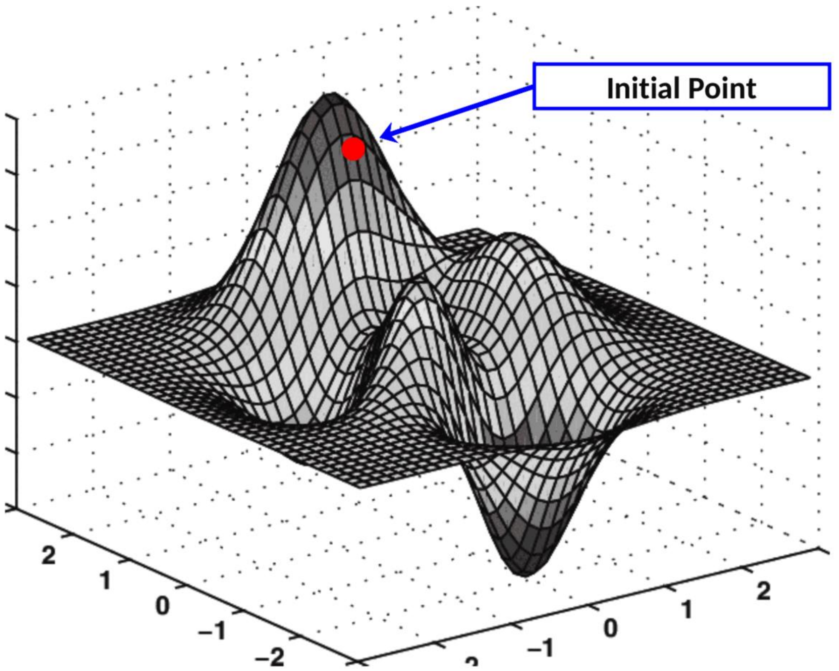
- The idea is to iteratively update each feature's weight until the convergence.

$Repeat\ untill\ convergence\{$

$$\beta_j = \beta_j - \alpha \frac{\partial J(\beta)}{\partial \beta_j}$$

$\}$

✓ What is interesting?



Gradient Descent in MLR

- In MLR our parameters are:

$$\beta_0, \beta_1, \beta_2, \ldots, \beta_p$$

- For convenience of notation, we define $x_0 = 1$ and rewrite the model as:

$$\hat{y} = \hat{\beta}_0 . X_0 + \hat{\beta}_1 \cdot X_1 + \hat{\beta}_2 \cdot X_2 + \cdots + \hat{\beta}_p \cdot X_p$$

- The cost function is:

$$J(\beta_0, \beta_1, \beta_2, \ldots, \beta_p) = \frac{1}{2n} \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)})^2$$

Gradient Descent in MLR

- As we said before the new algorithm for gradient descent in MLR will be:

$$Repeat\ untill\ convergence\{$$

$$\beta_{j,\,new} = \beta_j - \alpha \frac{\partial J(\beta)}{\partial \beta_j}$$

$$\}$$

- We should simultaneously update β_j for every $j = 0, 1, 2, \dots, p$ and the new algorithm will be:

$$Repeat\ untill\ convergence\{$$

$$\beta_{j,new} = \beta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} (\hat{y}^{(i)} - y^{(i)}) x_j^{(i)}$$

$$\}$$

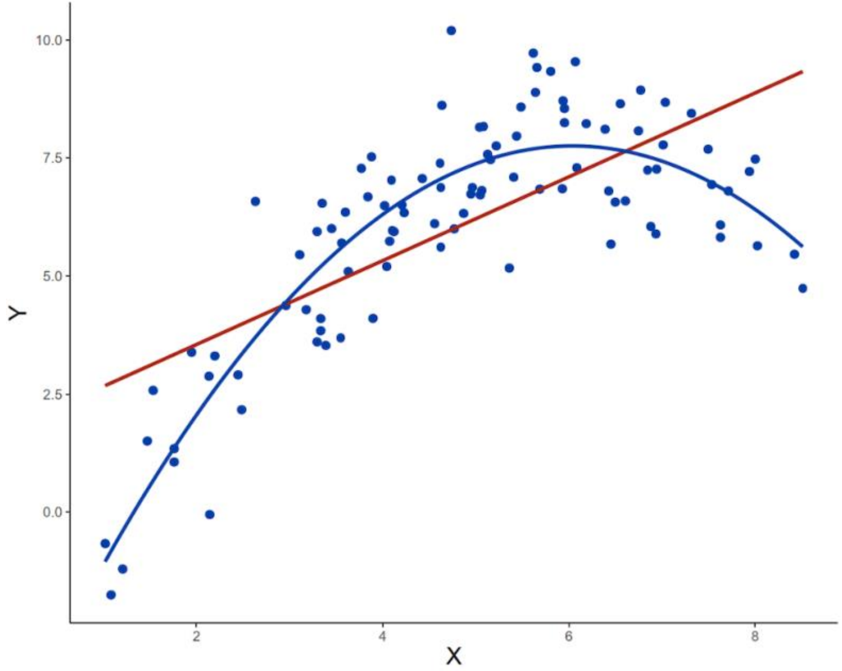
Polynomial regression

- The truth is almost never linear!

- A **polynomial regression** model is a machine learning model that can capture non-linear relationships between variables by fitting a non-linear regression line.

- Model with one feature:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \cdots + \beta_d x_1^d$$



Polynomial regression

- We can find the non-linear regression line by converting a polynomial regression to a multiple linear regression.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \cdots + \beta_d x_1^d$$

$\Rightarrow$	
x_1	x'_1
x_1^2	x'_2
...	...
x_1^d	x'_d

$$y = \beta_0 + \beta_1 x'_1 + \beta_2 x'_2 + \cdots + \beta_d x'_d$$

- We can use this trick with other terms. (E.g., $x'_1 = \sqrt{x_1 \cdot x_2}$)
- “Essentially, all models are wrong, but some are useful” by George Box

Interpretation

- We interpret β_j as the average effect on Y of a one unit increase in x_j, holding all other predictors fixed.

- For example:

$$y = 16.3 + 2.24\, x_1 + 5.3\, x_2 - 1.3\, x_3 \quad \begin{cases} x_1 \Rightarrow \text{Area}(m^2) \\ x_2 \Rightarrow \text{Number of bedrooms} \\ x_3 \Rightarrow \text{Age(years)} \end{cases}$$

- Claims of causality should be avoided for observational data.

  "Correlation does not imply causation" - Karl Pearson

- A regression coefficient β_j estimates the expected change in Y per unit change in x_j , with all other predictors held fixed. But **predictors usually change together**!

Interpretation

- The true way of interpreting => **partial derivatives**

- The partial derivative of y with respect to x_j (denoted as $\frac{\partial y}{\partial x_j}$) indicates how much y changes in response to a unit change in x_j , holding all other variables constant.

- For example:

$$y = 6x_1^2 + 3x_1 + x_2 - x_1.x_2$$

x_1

x_2

Interpretation

- Despite its simplicity, the linear model has distinct advantages in terms of its interpretability and often shows good predictive performance.

- The reason:

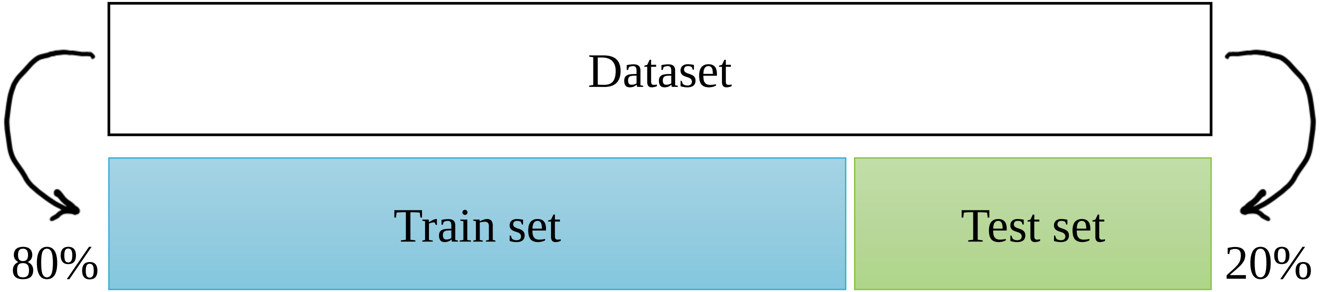
$$\frac{\partial y}{\partial x_j} = \beta_j$$

Train – test splitting

- The train-test split is a technique for **evaluating the performance** of a machine learning algorithm.

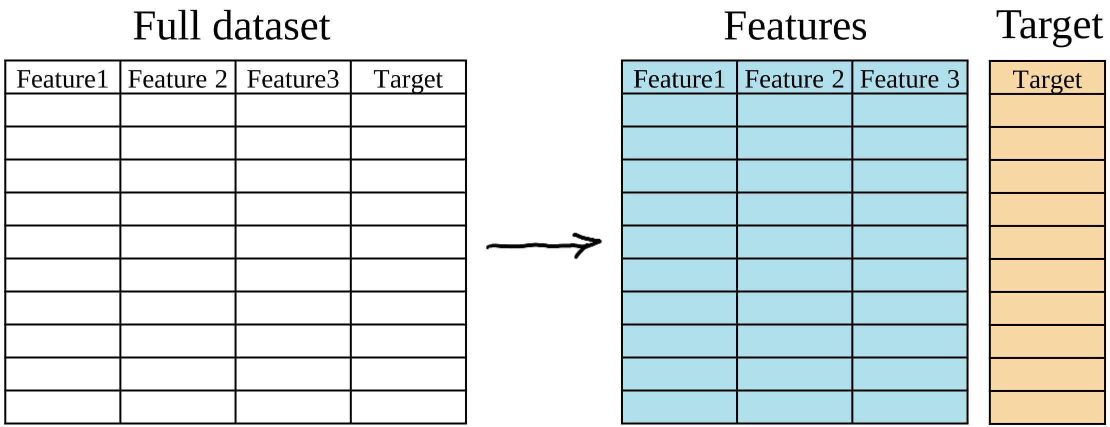
✓ The **training set** is used to train the model.
✓ The **test set** evaluates its performance on unseen data.

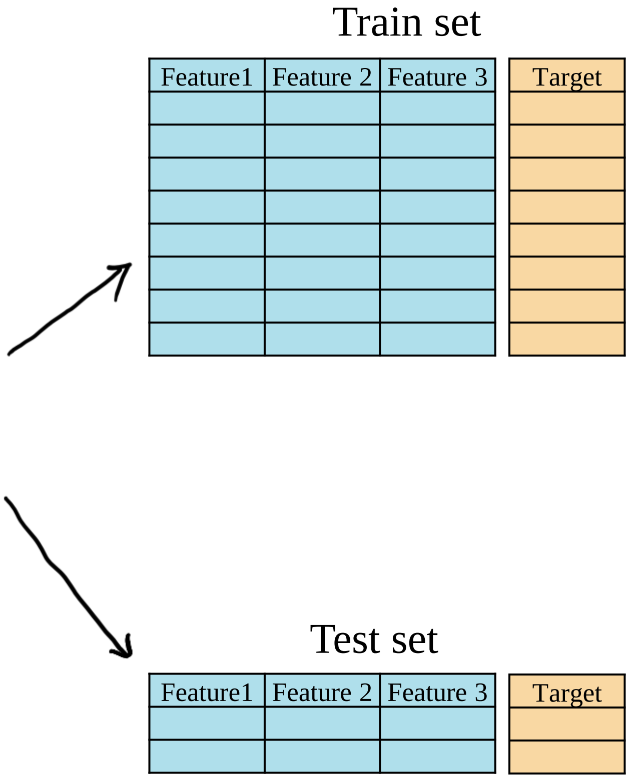
- A commonly used ratio is **80:20**, which means 80% of the data is for training and 20% for testing. Other ratios such as 70:30, 60:40, and even 50:50 are also used in practice.



Train – test splitting

- The process:





Train – test splitting

For our example:

Observation	Area(m^2)	Number of bedrooms	Age(years)	Total Price($ K)
1	54	1	12	128
2	75	2	2	186
3	80	1	5	150
4	40	0	30	63
5	38	0	5	96
6	120	3	10	299
7	90	1	12	197
8	95	1	7	250
9	86	2	1	200
10	65	1	5	210

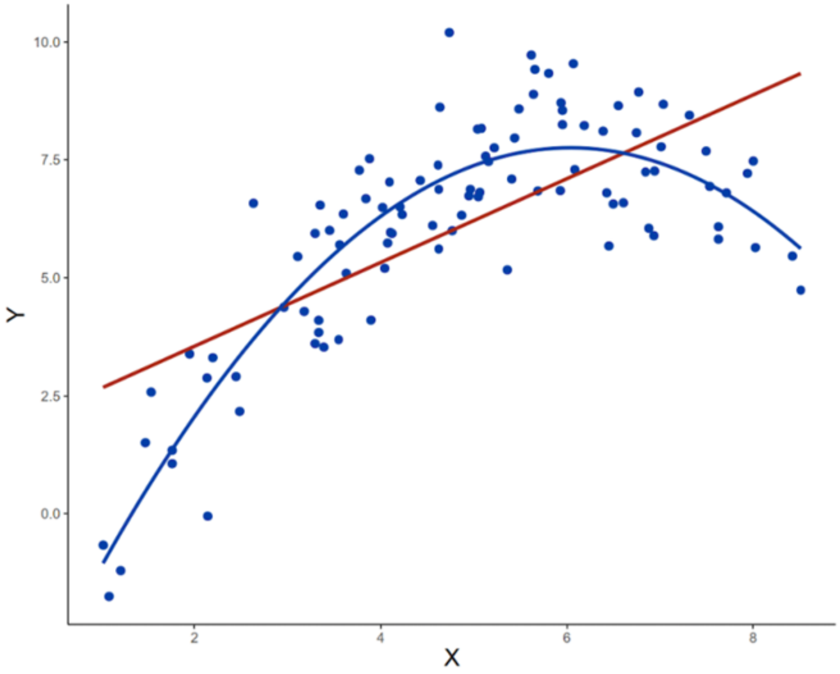
Observation	Area(m^2)	Number of bedrooms	Age(years)	Total Price($ K)
1	54	1	12	128
2	75	2	2	186
3	80	1	5	150
4	40	0	30	63
5	38	0	5	96
6	120	3	10	299
7	90	1	12	197
8	95	1	7	250

Observation	Area(m^2)	Number of bedrooms	Age(years)	Total Price($ K)
9	86	2	1	200
10	65	1	5	210

Evaluation metrics for regression

- An evaluation metric should allow us to compare two models directly.

- which model is better in this example?



Evaluation metrics for regression

- We want a metric that evaluates model performance using both training and test data.

- Regression models are often evaluated using MSE, RMSE, MAE, MAPE, R^2, adjusted R^2, AIC, BIC, and Cp.

✓ Mean Squared Error (MSE)
✓ Root Mean Squared Error (RMSE)
✓ Mean Absolute Error (MAE)
✓ Mean absolute percentage error (MAPE)
✓ Coefficient of determination (R^2 or COD)
✓ Akaike information criterion (AIC)
✓ Bayesian Information Criterion (BIC)
✓ Mallows' Cp (Cp)

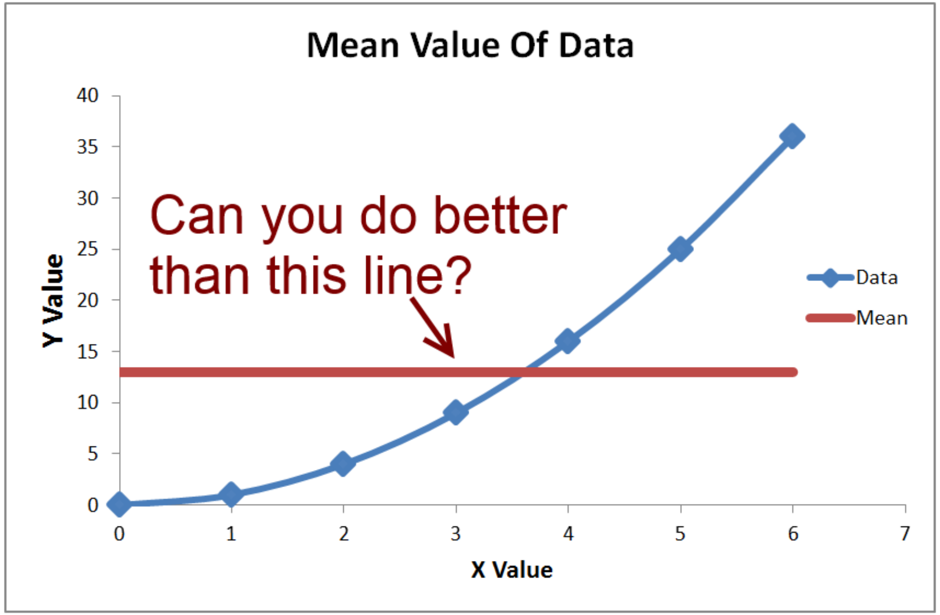
- In this course we introduce **MSE** and **R^2**.

R-squared (R²)

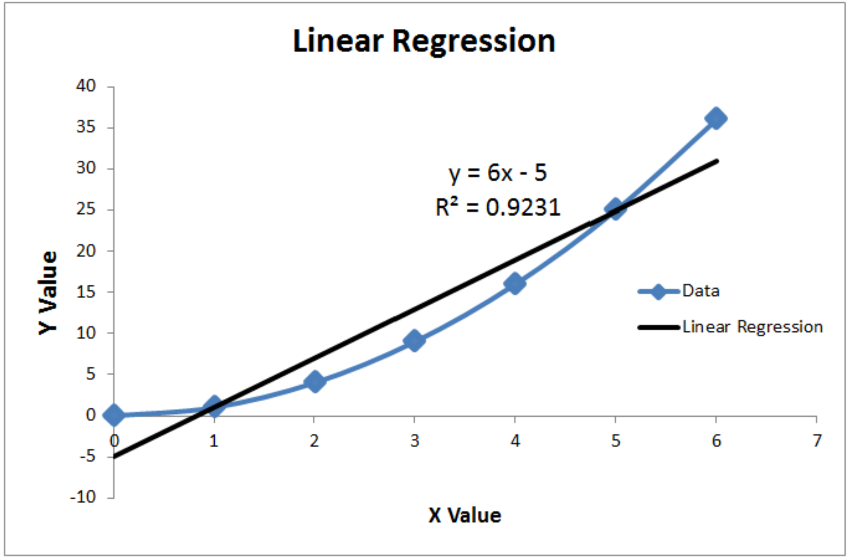
- R-Squared is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

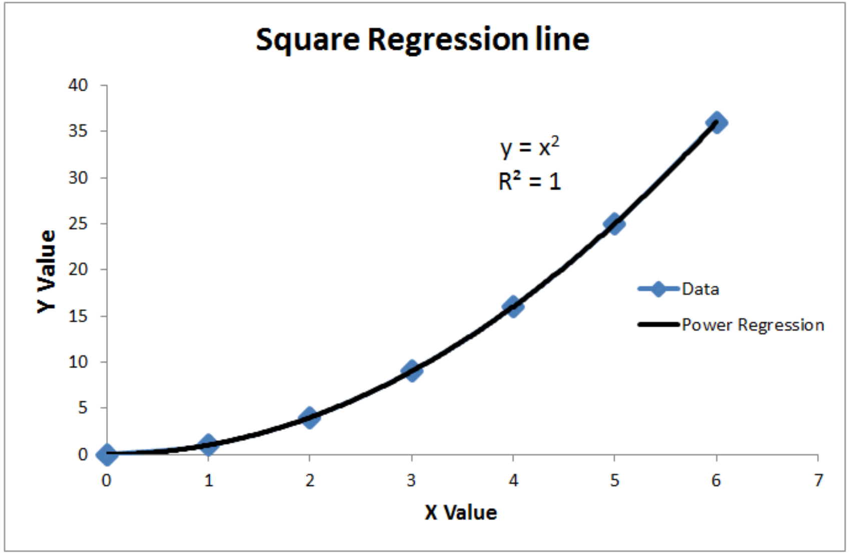
✓ What does $R^2 = 90\%$ means?



R-squared (R^2)



Linear Regression
y = 6x - 5
R² = 0.9231
Data
Linear Regression
Y Value
X Value
40
35
30
25
20
15
10
5
0
-5
-10
0
1
2
3
4
5
6
7



Square Regression line
y = x²
R² = 1
Data
Power Regression
Y Value
X Value
40
35
30
25
20
15
10
5
0
0
1
2
3
4
5
6
7

Mean Squared Error (MSE)

- Mean Squared Error (MSE) formula: $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

- This is the train set:

	Total Price	Prediction	$Error$	$Error^2$
1	128	120	8	64
2	186	180	6	36
3	150	160	-10	100
4	63	65	-2	4
5	96	100	-4	16
6	299	300	-1	1
7	197	197	0	0
8	250	250	0	0
Summation			-3	221

Mean Squared Error (MSE)

- Train set and test set:

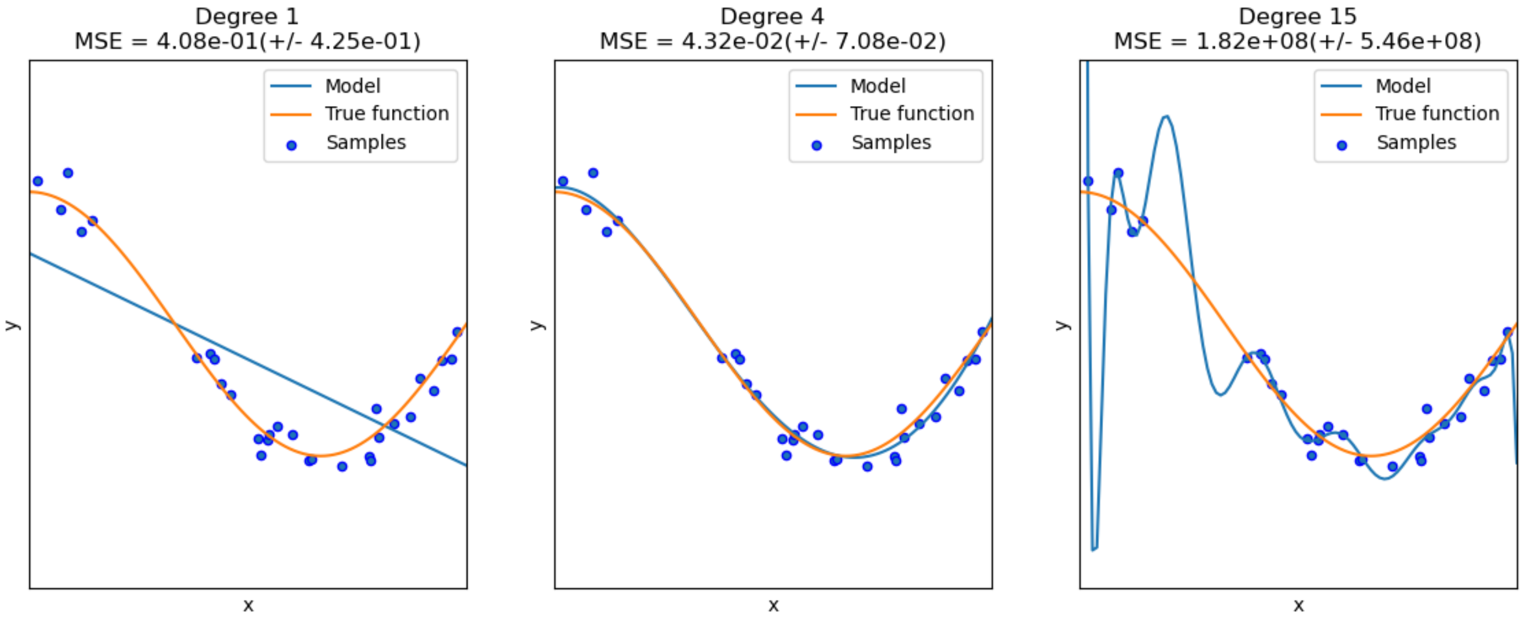
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

	Total Price	Prediction	$Error$	$Error^2$
1	128	120	8	64
2	186	180	6	36
3	150	160	-10	100
4	63	65	-2	4
5	96	100	-4	16
6	299	300	-1	1
7	197	197	0	0
8	250	250	0	0
Summation			-3	221

	Total Price	Prediction	$Error$	$Error^2$
9	200	207	-7	49
10	210	205	5	25
Summation			2	74

Bias-variance tradeoff

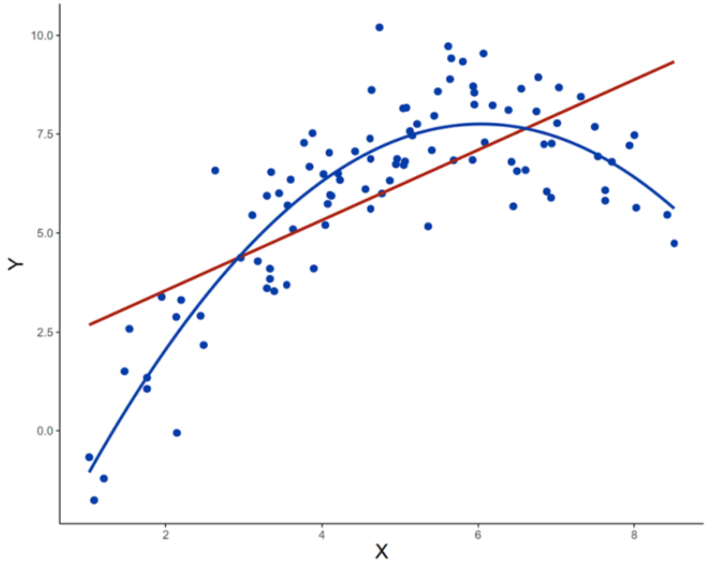
- One of the most essential notions in modern data science is the **bias-variance tradeoff.**

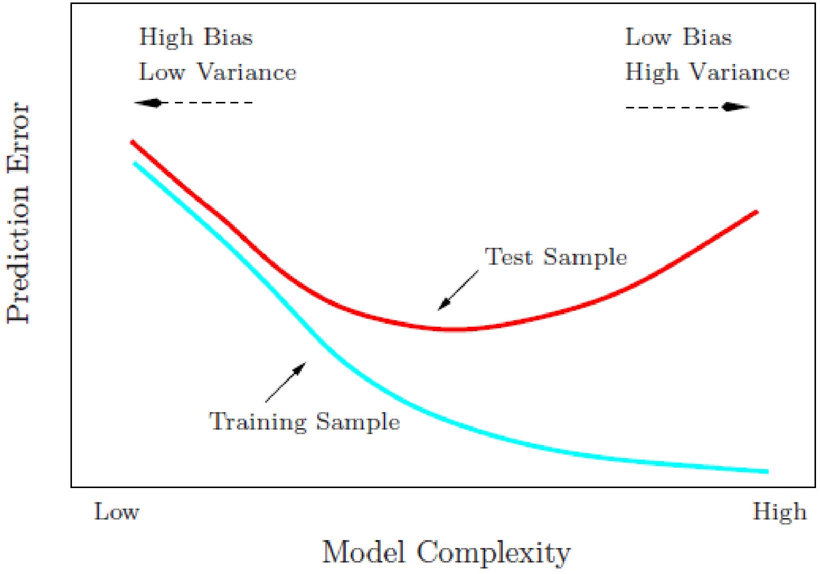


Bias-variance tradeoff

- Training error versus test error:

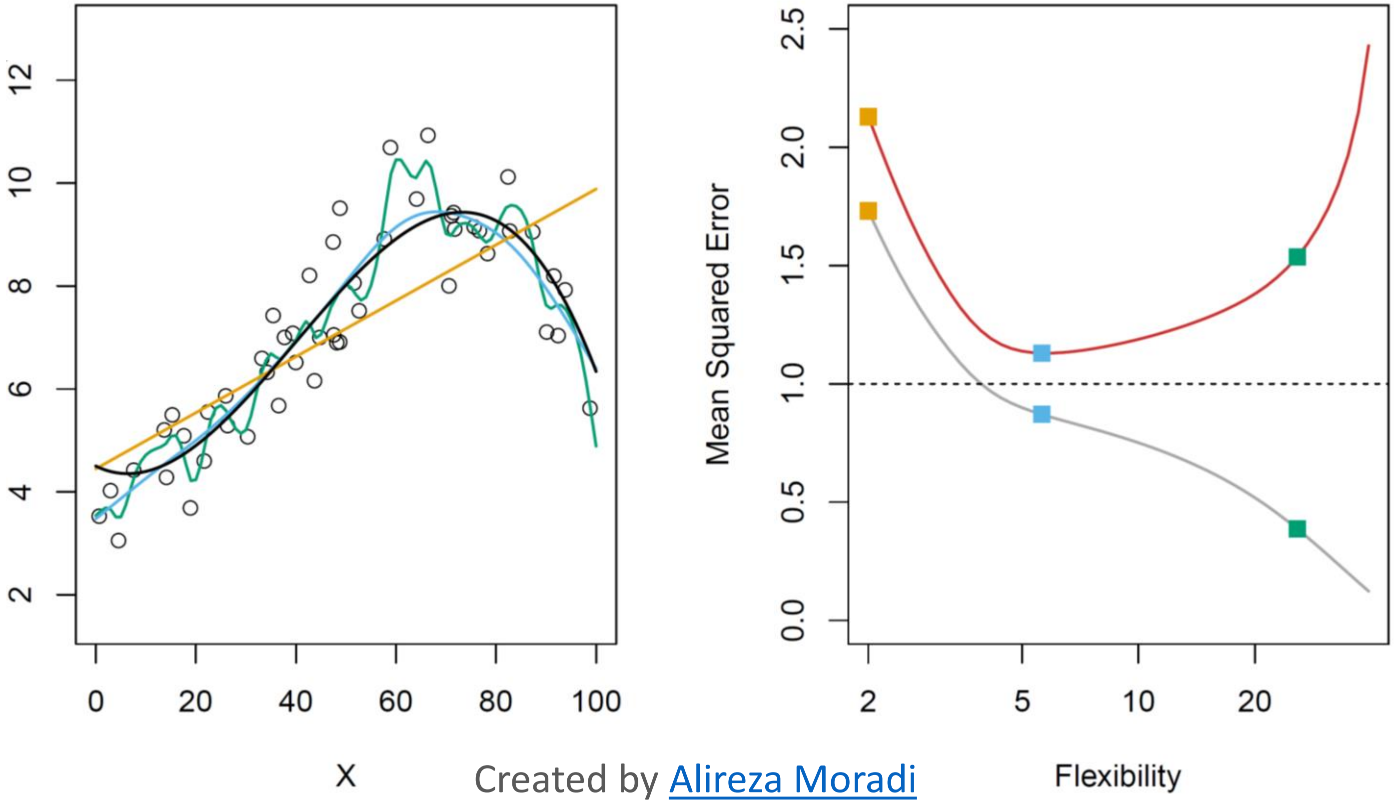
✓ What is model complexity?





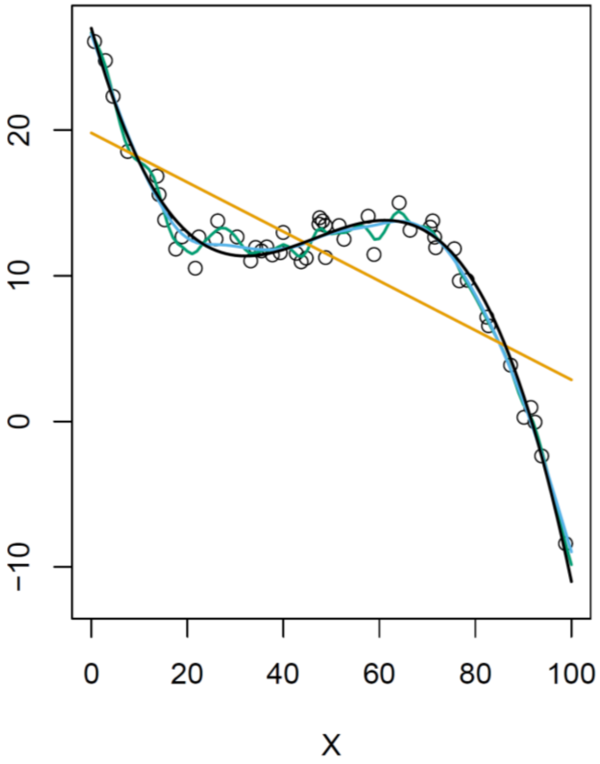
Bias-variance tradeoff

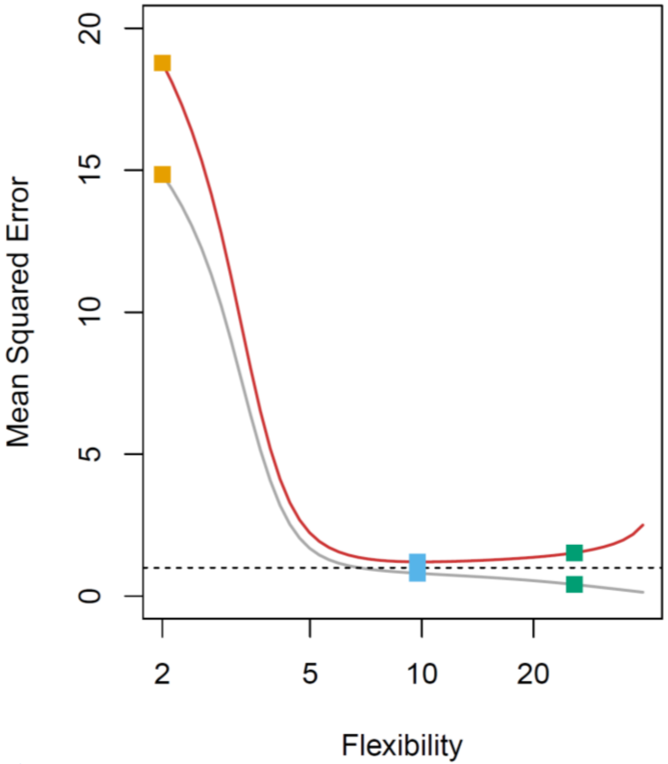
- Simulated data: **Black line**



Bias-variance tradeoff

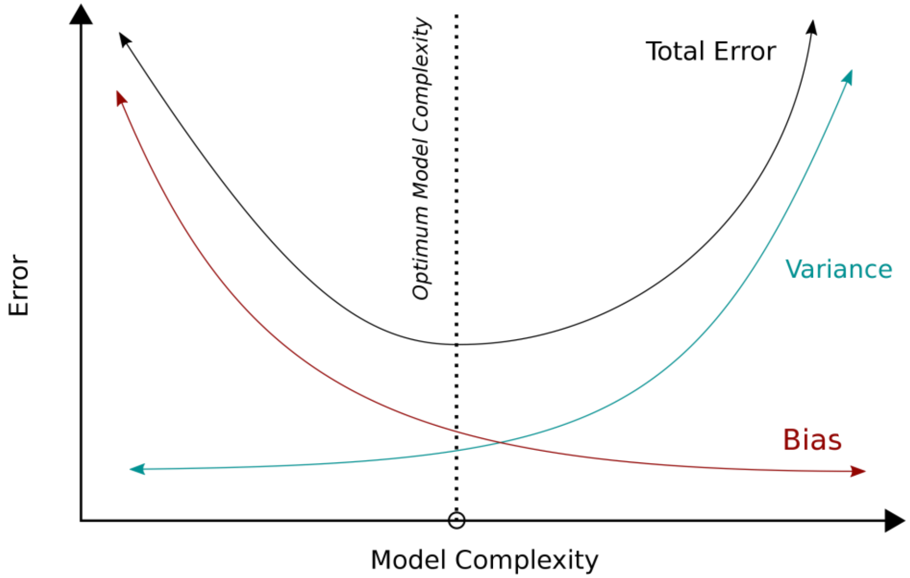
- Another example:





Bias-variance tradeoff

- The general idea is that developing models always is **a balance between** models that vary too much, and models which are too heavily biased.

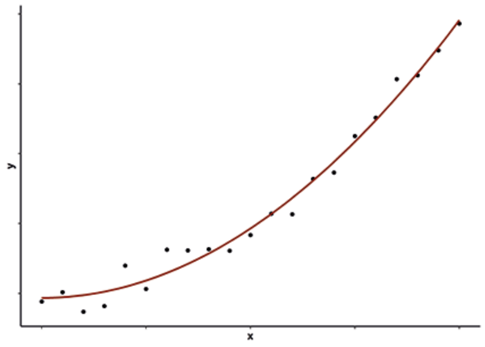


Underfit, fit and overfit

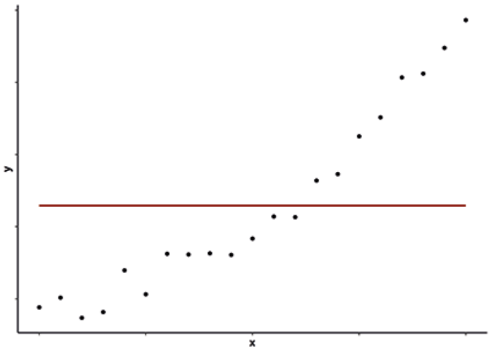
- You could in theory make a function that always equals some arbitrary number.(E.g., mean) So regardless of the values of inputs, we always get the same output.

- This is an example of a model that is completely biased and is one extreme of our bias variance tradeoff.

- Models that lean towards the bias extreme are experiencing what is called **underfitting**.



Good Fit

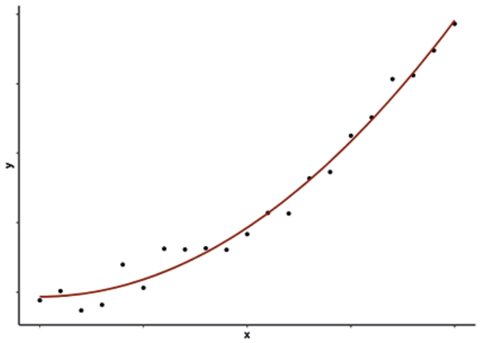


Underfitting

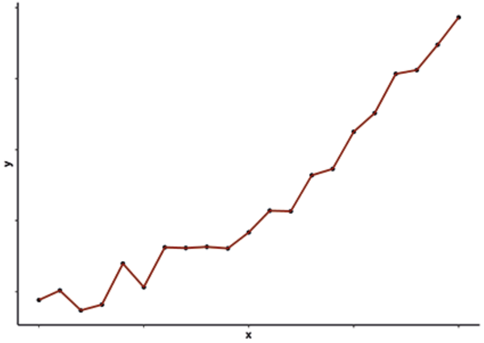
Underfit, fit and overfit

- The other end is a model which varies so much that it **doesn't generalize well**. (Test data shows this!)

- This is called **overfitting** and is when a model matches the data so closely that it fails to generalize to new data.



Good Fit



Overfitting