Data Analytics (PETR 6397)

Regression

Dr. Ahmad Sakhaee-Pour

Petroleum Engineering Department

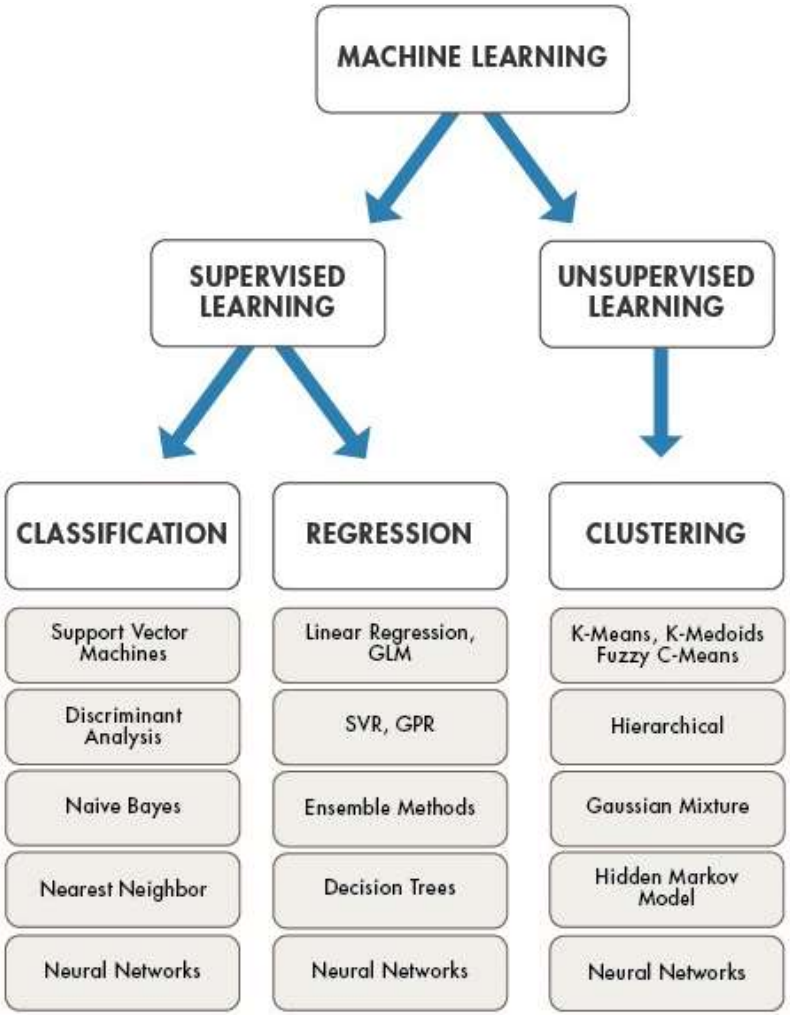
University of Houston

Spring 2025

Discussed topics

- Regression
- Linear regression
- Cost functions
- Gradient Descent
- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-batch Gradient Descent
- Regularized: Ridge regression, Lasso regression, Elastic Net regression
- *K*-nearest Neighbors regressor
- *K*-fold cross-validation
- Logistic regression

Regression


MACHINE LEARNING
SUPERVISED LEARNING
UNSUPERVISED LEARNING
CLASSIFICATION
REGRESSION
CLUSTERING
Support Vector Machines
Discriminant Analysis
Naive Bayes
Nearest Neighbor
Neural Networks
Linear Regression, GLM
SVR, GPR
Ensemble Methods
Decision Trees
Neural Networks
K-Means, K-Medoids Fuzzy C-Means
Hierarchical
Gaussian Mixture
Hidden Markov Model
Neural Networks

Regression analysis

- Regression is a statistical process to estimate the relationships among variables
  - Dependent variables
  - Independent variables
- There are various types of regression algorithms, and each is suitable for different purposes:
  - Linear regression
  - Ridge regression
  - Lasso regression
  - ….

Linear regression

- Best straight line with the lowest error

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

$\hat{y}$: predicted value

n: number of features (input variables)

x_i: feature value

θ_i: model parameter (tuning parameter)

θ_0: bias

θ_1, θ_2,…, θ_n: weights

Linear regression (vector)

$$\hat{y} = h_{\theta}(\mathbf{x}) = \boldsymbol{\theta}.\mathbf{x}$$

h_{θ}: hypothesis function (definition)

$$\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \ldots, \theta_n]$$

$$\mathbf{x} = [x_0, x_1, x_2, \ldots, x_n]$$

Q: what is the value of x_0?

Cost function (or loss function)

- Cost function (or evaluation metrics) quantitively determines the performance of the model
- **1. Mean absolute error (MAE):**
  - Average of the absolute value of the difference between actual and predicted values

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\widehat{y}_i - y_i|$$

n: number of samples

y_i: actual value

$\hat{y}_i$: predicted value

Cost function (continued)

- **Mean squared error (MSE):**

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

- **Root mean squared error (RMSE):**
  - RMSE is basically the square root of MSE

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

- It is easier to minimize MSE than RMSE
- Minimizing MSE or RMSE leads to similar tuning parameters
- MSE is used more often

Minimizing the cost function leads to an optimal scenario

- There is an analytical solution for minimizing the MSE

$$\mathrm{MSE}(\theta) = \frac{1}{n}\sum(\hat{y}_i - y_i)^2$$

$$\mathrm{MSE}(\theta) = \frac{1}{n}(\mathbf{\theta x} - \boldsymbol{y})^T(\mathbf{\theta x} - \boldsymbol{y})$$

$$\mathrm{MSE}(\theta) = \frac{1}{n}\left(\mathbf{\theta^T x^T} - \boldsymbol{y^T}\right)(\mathbf{\theta x} - \boldsymbol{y})$$

$$\mathrm{MSE}(\theta) = \frac{1}{n}\left(\mathbf{\theta^T x^T \theta x} - \mathbf{\theta^T x^T}\boldsymbol{y} - \boldsymbol{y^T}\mathbf{\theta x} + \boldsymbol{y^T y}\right)$$

$$\mathrm{MSE}(\theta) = \frac{1}{n}\left(\mathbf{\theta^T x^T \theta x} - \mathbf{2\theta^T x^T}\boldsymbol{y} + \boldsymbol{y^T y}\right)$$

$$\frac{\partial}{\partial\theta}\mathrm{MSE}(\theta) = \frac{1}{n}\left(\mathbf{x^T \theta x} + \mathbf{x^T \theta x} - \mathbf{2x^T}\boldsymbol{y} + \mathbf{0}\right) = \frac{2}{n}\mathbf{x^T}(\mathbf{\theta x} - \boldsymbol{y}) = 0$$

Normal equation minimizes the error, but it is not always applicable

$$\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}\ (\mathbf{minimum\ error}) = \left(\mathbf{x}^{\mathbf{T}}\mathbf{x}\right)^{-\mathbf{1}}\mathbf{x}^{\mathbf{T}}\boldsymbol{y}$$

$\boldsymbol{y}$ Includes target values

Reasons for inapplicability:

1. $\mathbf{x}^{\mathbf{T}}\mathbf{x}$ may not be invertible
2. Matrix inversion is computationally expensive

Singular value decomposition addresses the shortcomings to some extent

Singular value decomposition (SVD) solution for minimizing MSE

Decomposition:

$$\mathbf{x} = \mathbf{V}\Sigma\mathbf{U}^{\mathbf{T}}$$

$\mathbf{V}$: orthogonal matrix

Σ: diagonal matrix

$\mathbf{U}^{\mathbf{T}}$: orthogonal matrix

$$\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}\ (\mathbf{minimum\ error}) = \mathbf{V}(\Sigma^{\mathbf{T}}\Sigma)^{-\mathbf{1}}\Sigma^{\mathbf{T}}\mathbf{U}^{\mathbf{T}}\mathbf{y}$$

Computational cost prevents us from using the Normal equation when there are many input variables

- If there are *n* variables:
  - Normal equation: ~ O(n^2.4) to O(n^3)
  - Singular value decomposition: O(n^2)

- SVD is better than the Normal equation, but its cost is still prohibitive
- n:10,000
  - Normal equation: 3.98e-9 to 1e12
  - SVD: 1e8

Gradient Descent

- It initializes the tuning parameters of the regression model randomly
- Then, it finds the gradient of the cost function with respect to the variables
- It descends (goes downward) to lower the cost
- It iterates this updating to minimize the cost

$$\boldsymbol{\theta} \coloneqq \boldsymbol{\theta} - \eta \boldsymbol{\nabla} \mathrm{MSE}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \eta \boldsymbol{\nabla} \mathbf{J}(\boldsymbol{\theta})$$

$\boldsymbol{\theta}$ includes the tuning parameters (vector)

η Is the learning rate (scalar)

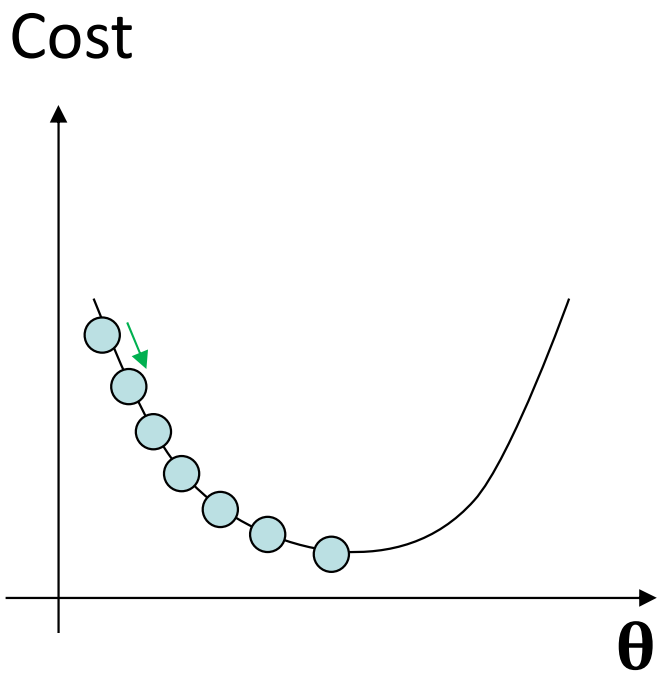
Expansion of the gradient term

- $$\nabla \mathbf{J}(\boldsymbol{\theta}) = \nabla \mathrm{MSE}(\boldsymbol{\theta}) = \begin{Bmatrix} \frac{\partial}{\partial \theta_0} \mathrm{MSE}(\theta_0, \theta_1, \theta_2, \dots, \theta_n) \\ \frac{\partial}{\partial \theta_1} \mathrm{MSE}(\theta_0, \theta_1, \theta_2, \dots, \theta_n) \\ \frac{\partial}{\partial \theta_2} \mathrm{MSE}(\theta_0, \theta_1, \theta_2, \dots, \theta_n) \\ \vdots \\ \vdots \\ \frac{\partial}{\partial \theta_n} \mathrm{MSE}(\theta_0, \theta_1, \theta_2, \dots, \theta_n) \end{Bmatrix}$$
- Gradient shows upward direction, so we use its negative
- The learning rate controls pace of going downward

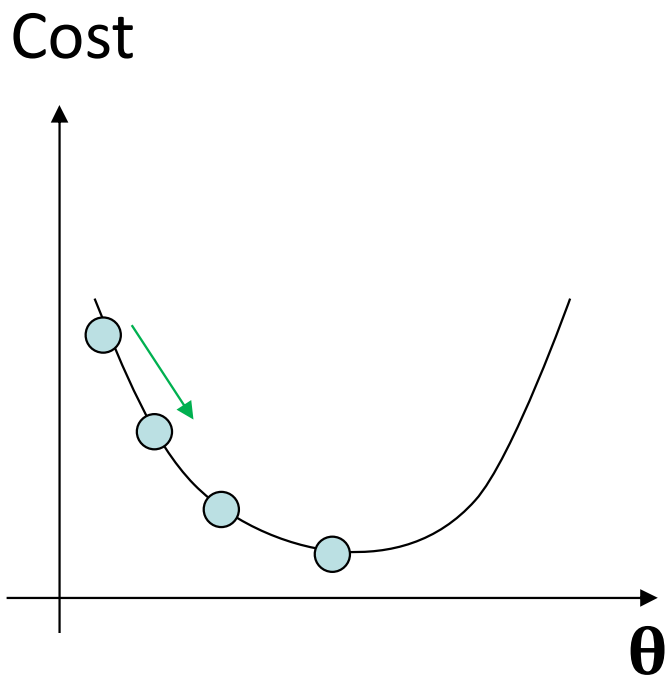
$$\boldsymbol{\theta} \coloneqq \boldsymbol{\theta} - \eta \nabla \mathrm{MSE}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \eta \nabla \mathbf{J}(\boldsymbol{\theta})$$

Effects of the learning rate on the Gradient Descent

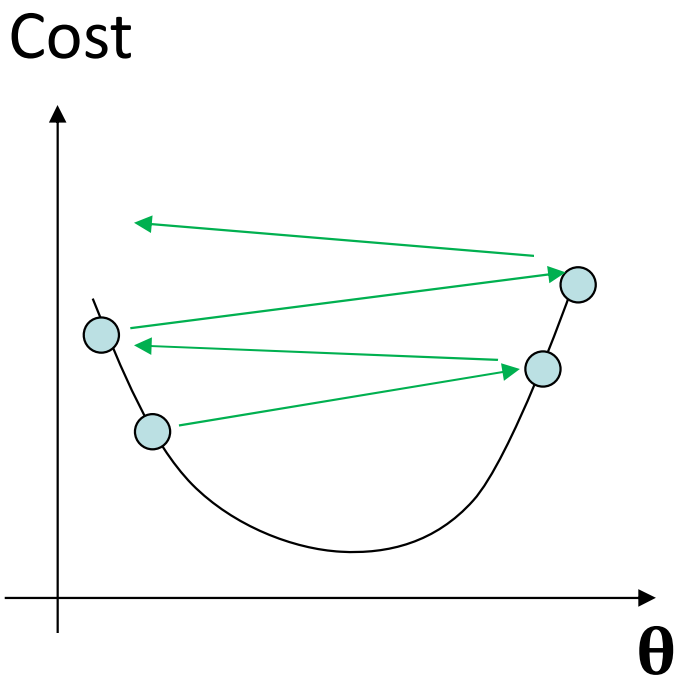
- Step size depends on the learning rate



Small rate



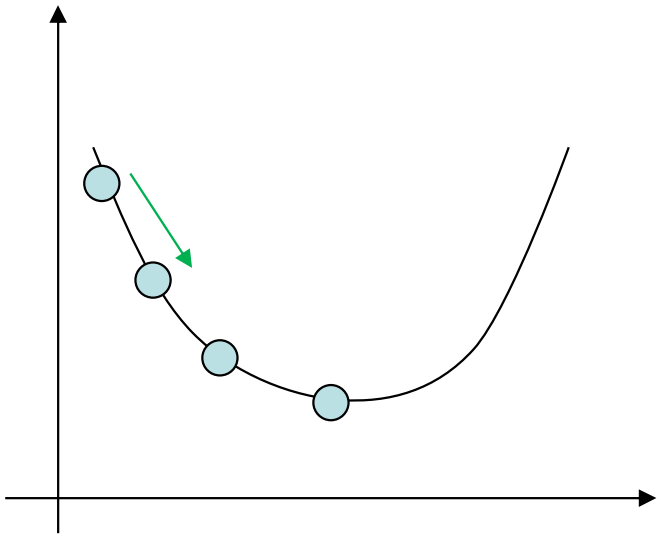
Medium rate



Large rate

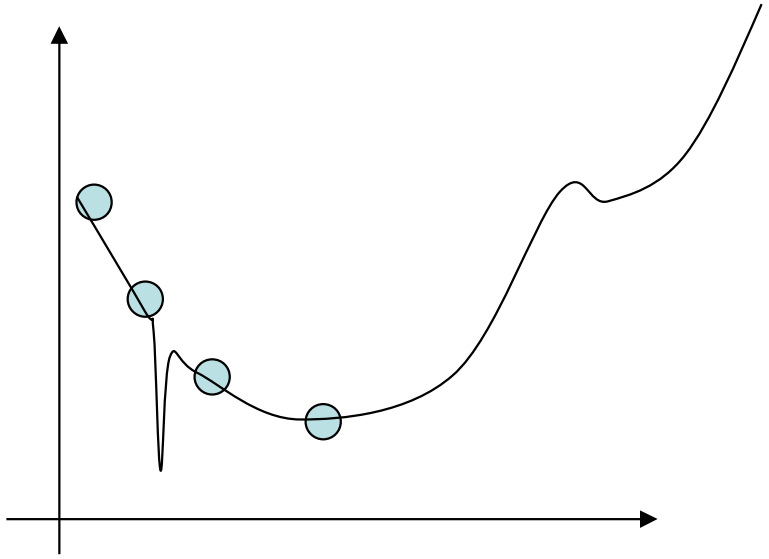
Cost function (MSE) of linear regression is convex, unlike many other regression models

Cost



Linear regression model

Cost



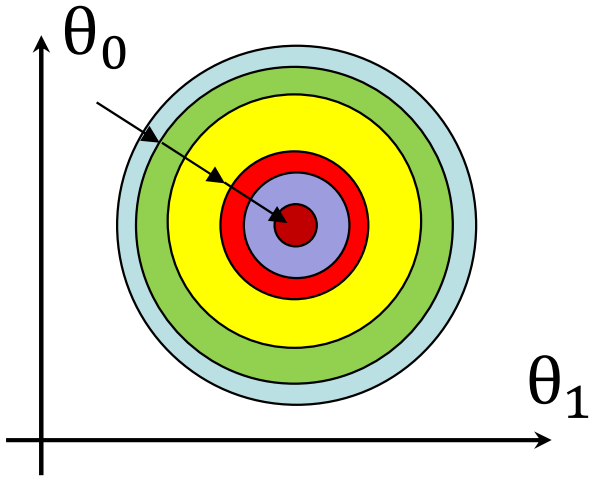
More complex model

Cost of Gradient Descent

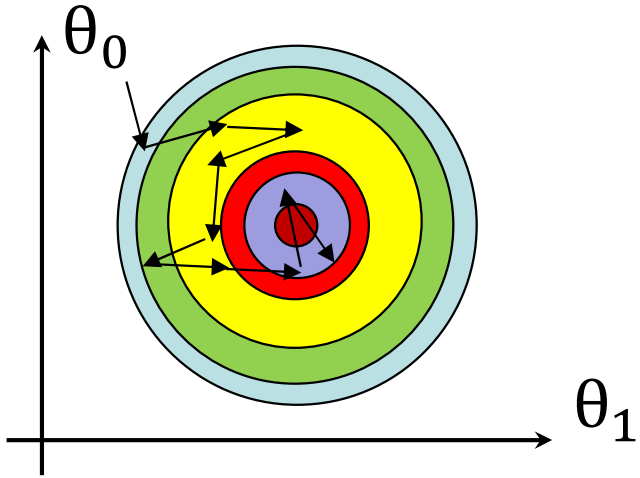
- If we have *n* variables and *m* tuning parameters in the model, its computation cost is ~ O(*m.n*) for each iteration
- If you have 10,000 observations (*n*) and plan to find the best line with 10 variables (m): 10^4.10=10^5
- Gradient Descent is faster than the Normal equation but still slow because it uses all the training data with *n* variables
- Solution: choose a random set (obviously, this is smaller than *n*).

Stochastic Gradient Descent

- It uses a random set of observations (training set) to minimize the cost function at every step
- It behaves stochastically (less regular) compared to the Gradient Descent
- Cost function is shown using contours



Gradient Descent



Stochastic Gradient Descent

Main features of Stochastic Gradient Descent

- Its results are good (not necessarily optimal)
- It is likely to skip the local minimum in its search to minimize the cost function
- Note: always shuffle the data, especially when you use Stochastic Gradient Descent because your solution is based on stochasticity (randomness)

Batch versus mini-batch

- Batch: the entire set of data
- Mini-batch: a small set of data (for instance, 100 samples from 10000 observations)
- At each iteration:
  - Stochastic Gradient Descent uses one random sample
  - Batch Gradient Descent uses the entire data at each to minimize the
  - Mini-batch Gradient Descent uses a random set

Epoch versus iteration

- Each iteration corresponds to a single update of the model weights (single update)

- Each epoch corresponds to going through the entire data once (full pass)

- Q1: Suppose we have 10,000 observations (data) and use a Mini-Batch Gradient Descent with a mini-batch size of 500. What is the relation between iteration and epochs?

- Q2: What if we use Stochastic Gradient Descent?

Polynomial regression

- We first define new features based on the original features
- We then apply linear regression using the new features
- Let us suppose we have one input variable (x) and like to apply polynomial regression of order 3:

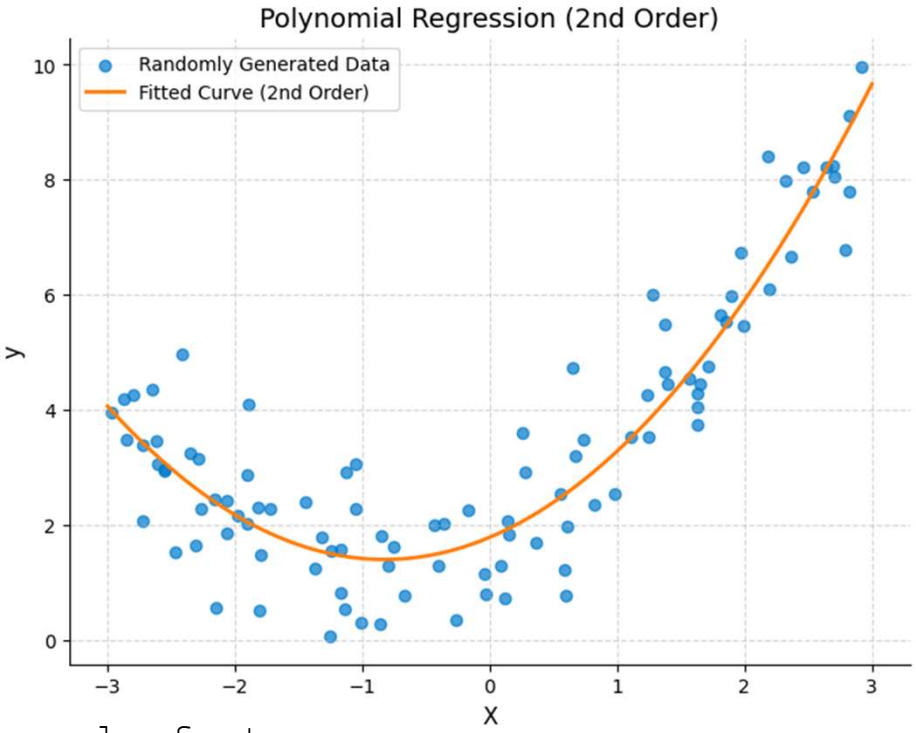
  $z_1 = x$

  $z_2 = x^2$

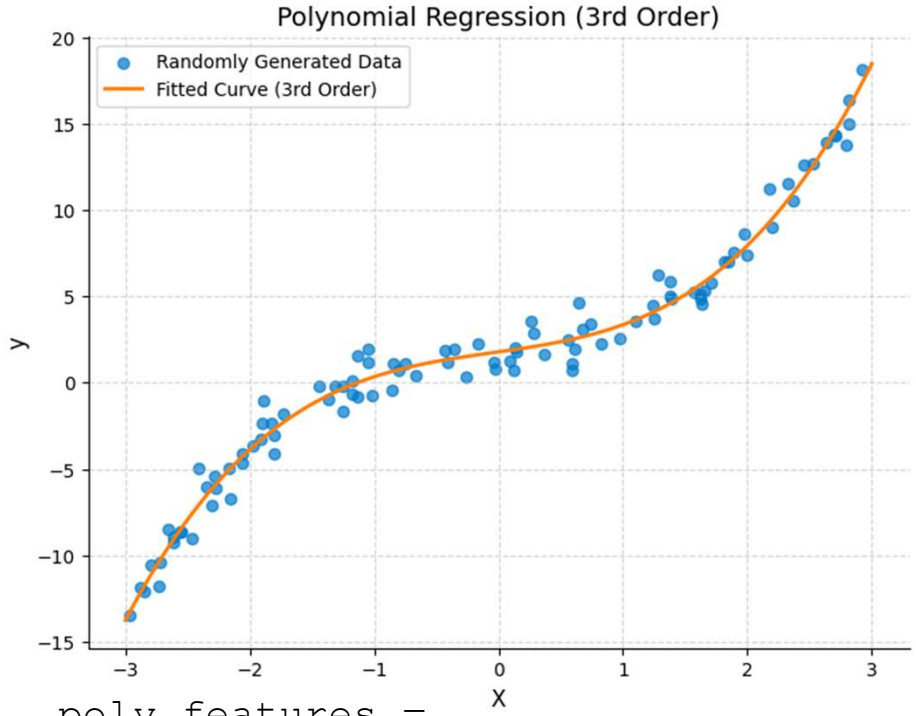
  $z_3 = x^3$

- New model: y= $\theta_0 + \theta_1 \times z_1 + \theta_2 \times z_2 + \theta_3 \times z_3$
- What if you have two variables (x_1, x_2) and like to apply the polynomial regression of order 2?

Examples of polynomial regression




```
poly_features =
PolynomialFeatures(degree=2,
include_bias=False)
X_poly = poly_features.fit_transform(X)
```




```
poly_features =
PolynomialFeatures(degree=3,
include_bias=False)
X_poly = poly_features.fit_transform(X)
```

Transform input features for polynomial regression

Linear regression based on other parameters

- Is it possible to use other variables in linear regression instead of polynomials?
- How about the following?

  z_1 =sin(x_1)

  z_2 =cos(x_2)

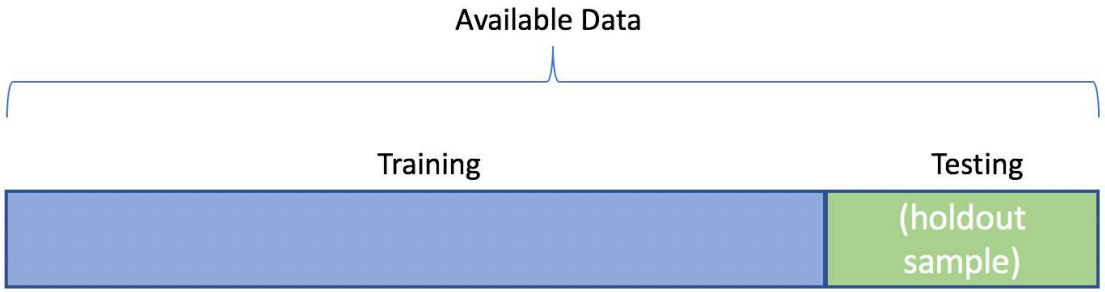
  z_3 =sin(x_1)cos(x_2)
- How does linear regression find the coefficients?

Data division for training and testing the model

- We usually divide the data (100%) into training (~80%) and test (~ 20%) sets
- We tune the parameters by minimizing the loss using the training data
- We then check the model performance using the test data
- We may divide the data into training, validation, and test data
- Validation is used to check the model performance during the training

Training-testing divide

	Predictor 1	Predictor 2	Predictor 3	Predictor 4	Predictor 5	Target
Training	Observation1					
	.					
	.					
	.					
Testing	.					
	.					
	Observation1000					



Original data

Gamma Ray	Resistivity	Poisson ratio	Density	Velocity	Rock Type
73.215	0.25	0.4126	137.81	7256.3675	Sandstone
69.152	0.2	0.4118	138.06	7243.2276	Shale
65.965	0.21	0.4109	138.06	7243.2276	Siltstone
68.215	0.27	0.4104	137.31	7282.7908	Siltstone
70.84	0.36	0.4096	136.31	7336.2189	Sandstone
62.262	0.52	0.4039	134.31	7445.462	Shale
61.637	0.7	0.3927	127.78	7825.9509	Shale

Example of training-test data

Predictors (X1,X2,….Xn)	Y
train_X	train_Y
test_X	test_Y

Overfitting versus underfitting

- Underfitting is a scenario where the model is not complex enough for your problem
- Overfitting happens when your model is too complex for your problem
- Main features
  - Underfitting: Cost function remains high for training and test data
  - Overfitting: The model performs well on the training data (low cost) but does not generalize when assessed on the test data (high cost)
  - There is a sweet spot that you should find by checking the cost functions (RMSE, MSE,..) for the training and test data

Ideal outcome

- Cost functions of training and test data are small (model is not underfitting)
- Cost functions of training and test data are close (model is not overfitting)
- The cost function of test data is slightly higher than the cost function of the training data because the model has seen the training data to tune its parameters

Bias versus variance trade-off

- What are the definitions of bias and variance?
- How do these terms relate to overfitting and underfitting?
- What are their governing equations?

Regularized linear models

- Regularization constrains the weights of the model to prevent overfitting
- Ridge regression:

$$J_{Ridge}(\theta) = \mathrm{MSE}(\theta) + \frac{\alpha}{m}\sum \theta_i^2$$

- Lasso regression:

$$J_{Lasso}(\theta) = \mathrm{MSE}(\theta) + 2\alpha \sum |\theta_i|$$

- Elastic Net regression:

$$J_{Elastic\ Net}(\theta) = \mathrm{MSE}(\theta) + r\left(2\alpha \sum |\theta_i|\right) + (1-r)(\frac{\alpha}{m}\sum \theta_i^2)$$

- Note: θ_0 is not regularized ($i > 0$ in all equations)

Ridge regression uses L2 norm squared

- The model minimizes not only the MSE but also the weights

$$J_{Ridge}(\theta) = \mathrm{MSE}(\theta) + \frac{\alpha}{m}\sum \theta_i^2$$

- α is a hyperparameter
- α=0 simplifies the model to linear regression
- Large α makes weights close to zero
- Increasing α flattens the output of the model (less sensitive to the input variables)

Least Absolute Shrinkage and Section Operator (Lasso) regression

- This model is similar to the Ridge regression but uses the L1 norm in the regularization

$$J_{Lasso}(\theta) = \mathrm{MSE}(\theta) + 2\alpha \sum |\theta_i|$$

- Lasso regression sets the less-important weights equal to zero (this is similar to feature selection)

Elastic Net Regression

- This regression is between the Ridge and Lasso regressions (check the solution where r=0 and r=1)

$$J_{Elastic\ Net}(\theta) = \mathrm{MSE}(\theta) + r\left(2\alpha\sum|\theta_i|\right) + (1-r)(\frac{\alpha}{m}\sum\theta_i^2)$$

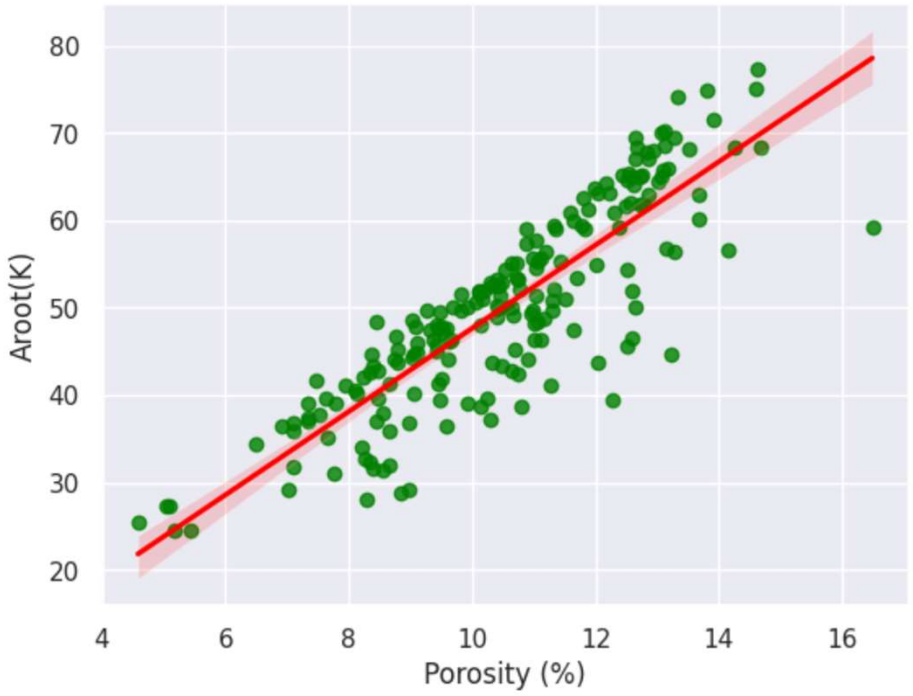
Which regression model works the best

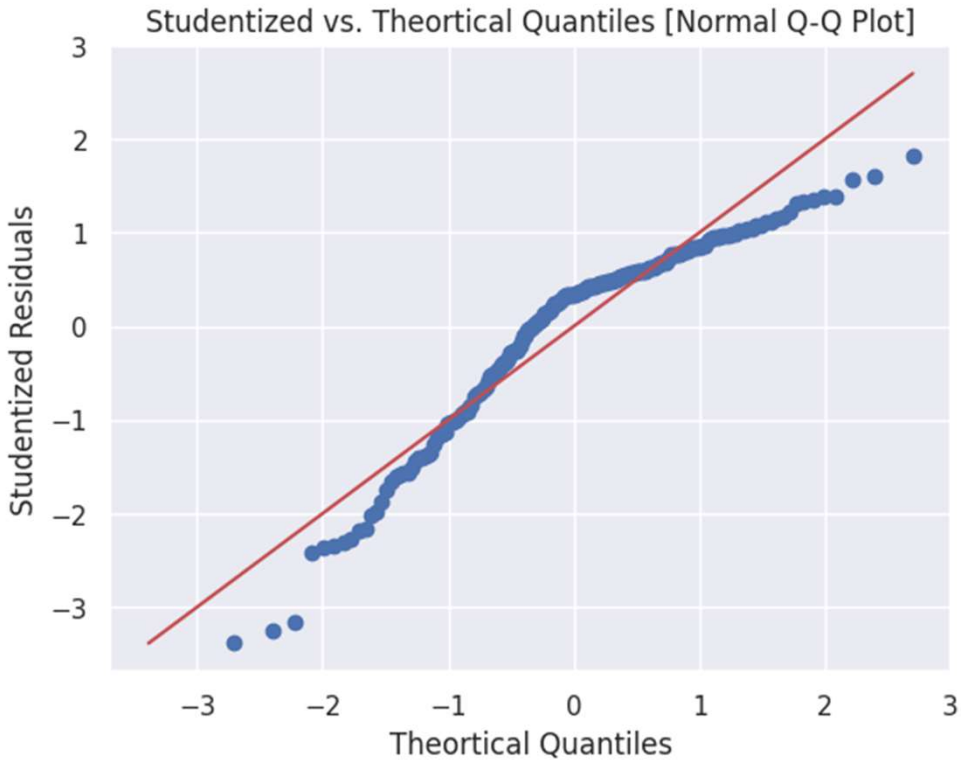
- Obviously, it depends!
- Ridge regression is usually better than the original regression because regularizations, in general, improve the performance
- Lasso and Elastic Net regressions are better than Ridge regression if you plan to check the effects of the number of tuning variables
- Lasso regression allows you to transition your model from Ridge to Lasso regression
- You should be able to compare them quickly using a few lines of codes

Code 1

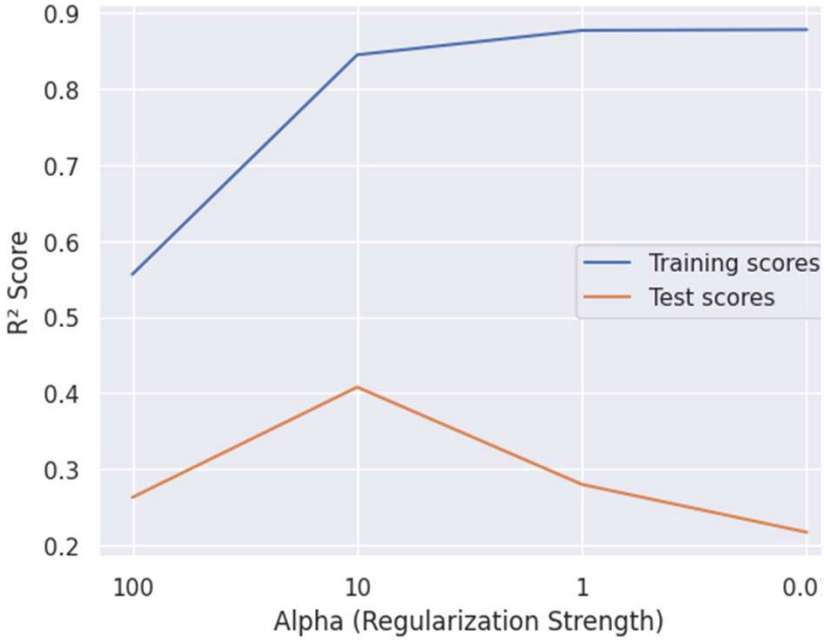
- Code_Linear_regression (actual code)
- data_3 (stored data)

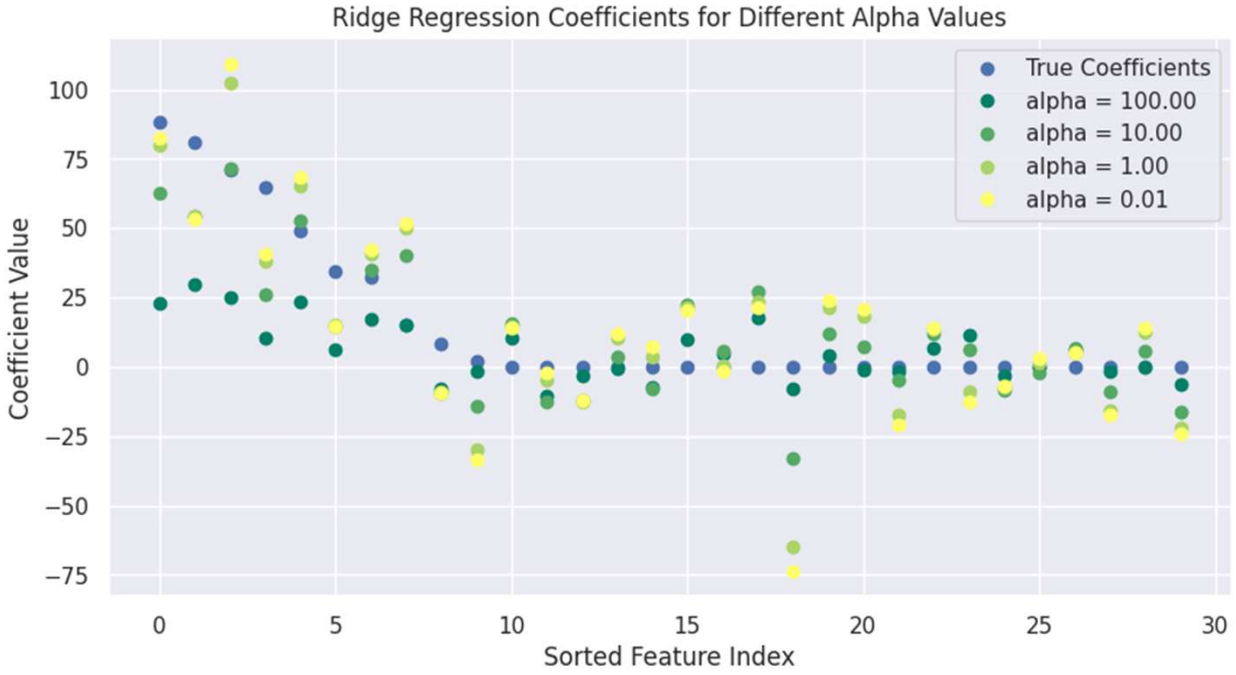
Code 1: Linear regression





Code 1: Ridge regression

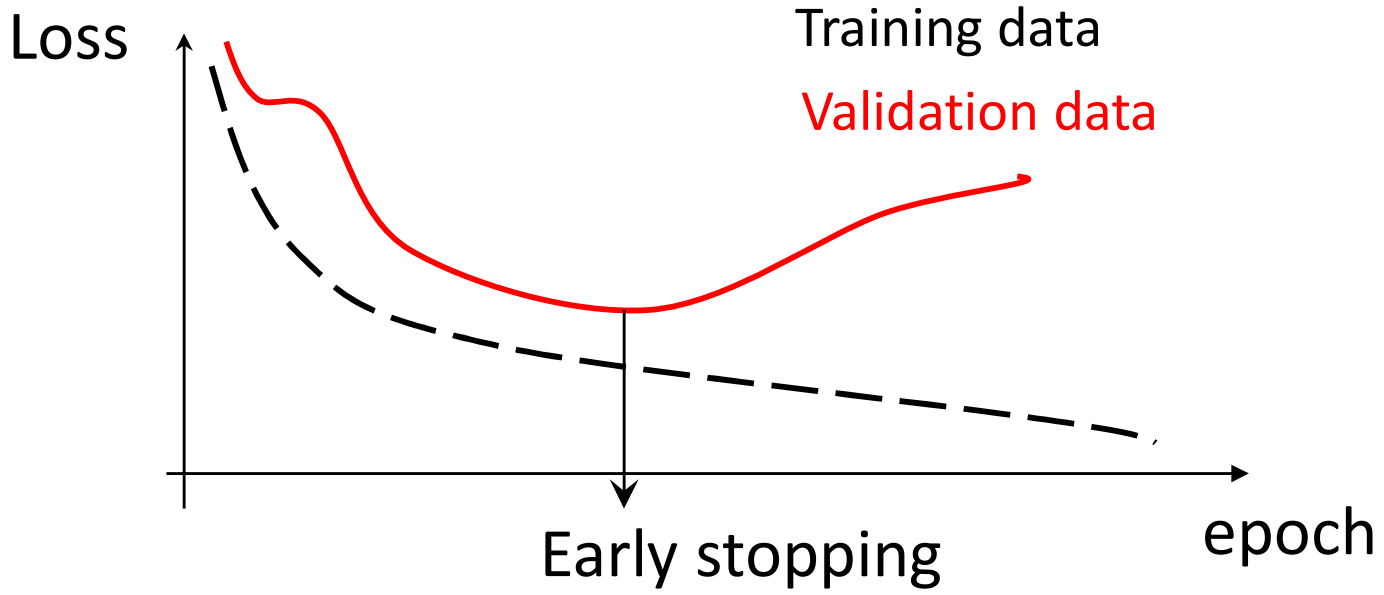




Code 1: Lasso Regression

Early stopping

- Early stopping helps us avoid overfitting
- The loss values of validation and training decrease initially as the model learns the underlying features
- We stop the training when the validation loss reaches a minimum (after this point, the model overfits the training but does not generalize)



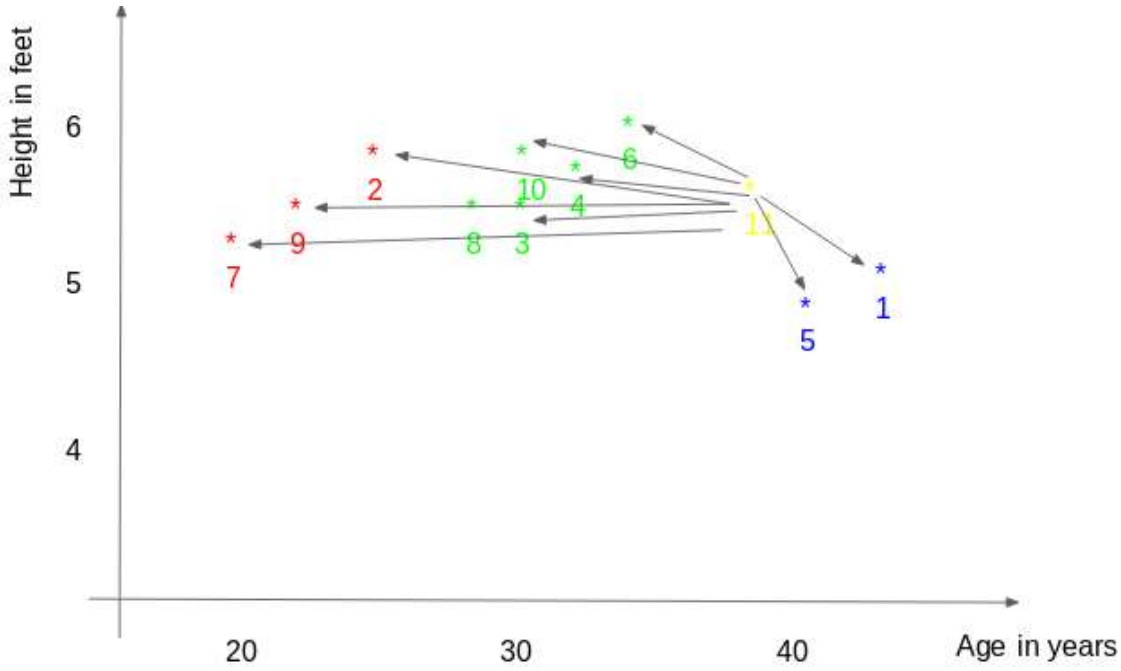
Another regression model

- *K*-nearest Neighbors Regressor

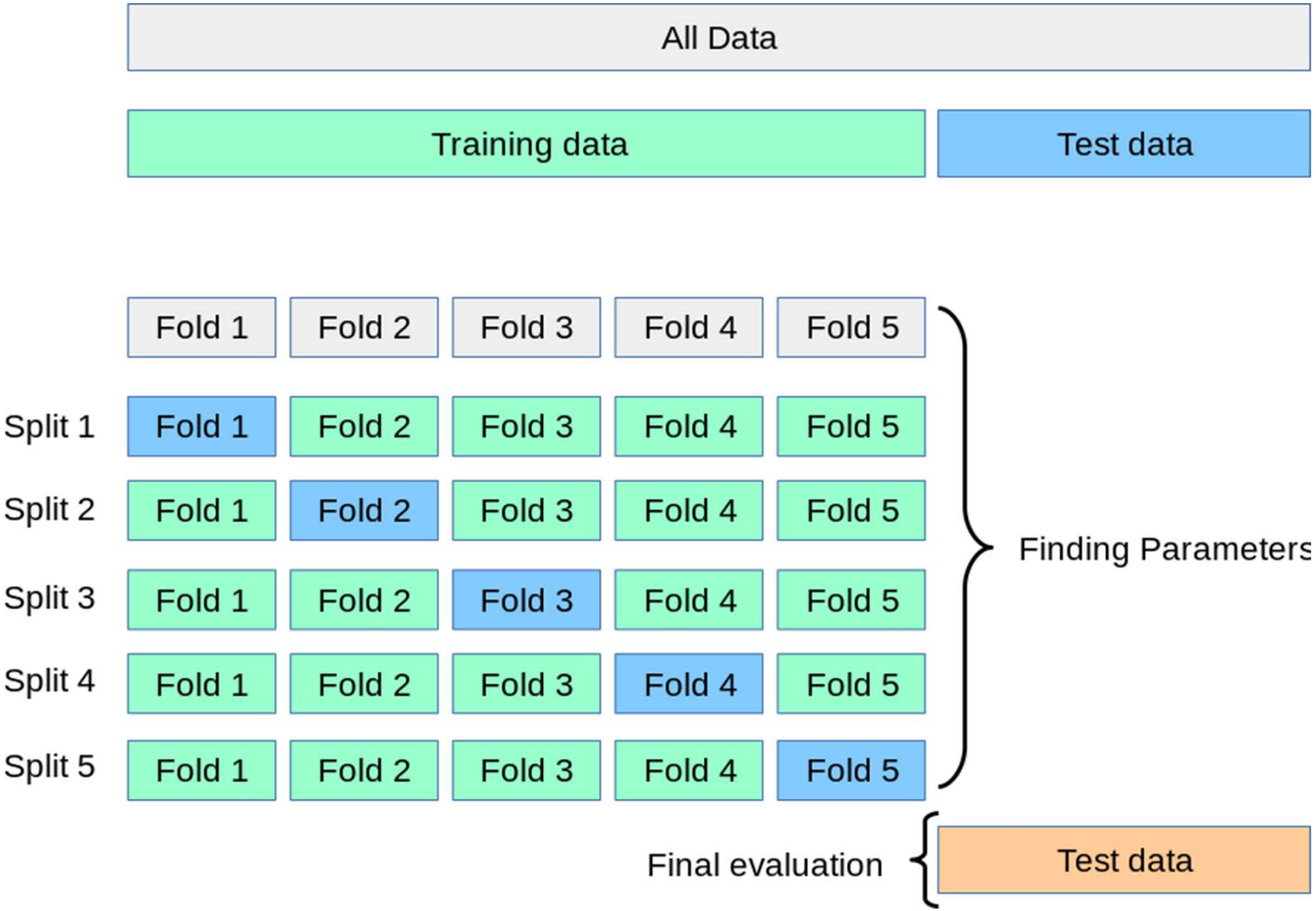
K-nearest Neighbors Regressor

- How does it work?

ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?



K-fold Cross-validation



Some comments on KNN

- It is not popular, but it is a good baseline
- It does not require defining new features because it can handle oscillation and nonlinearity
- KNN is more powerful than Linear Regression for interpolation, but **it usually fails to extrapolate**

Code 2

Code- run the k-nearest regressor code

- A. Upload the Notebook (KNN_Regressor)
- B. Upload its data (KNN_reg_data)
- C. Run the code once
- Answer the following questions:
- 1. What is the effect of `random_state=55`?
- 2. What is the division percentage (training versus testing)? What is a more common division in data analytics?
- 3. Change the number of neighbors and discuss its effect on the model performance
- 4. What is the best number of neighbors?

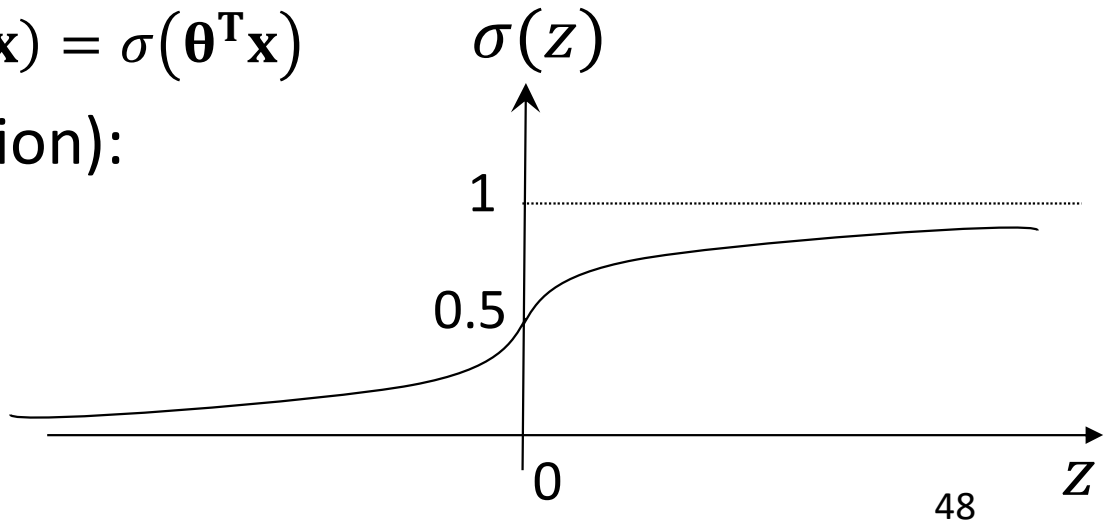
Logistic regression

- This regression determines the probability of a sample being part of a specific group:
  - What is the probability of a hydrocarbon reservoir being economically producible?
  - What is the probability of a patient having covid?
- It is a regression because it provides the probability and not just the class:

$$\hat{p} = h_\theta(\mathbf{x}) = \sigma(\boldsymbol{\theta}^{\mathbf{T}}\mathbf{x})$$

- Logistic function (definition):

$$\sigma(\mathrm{z}) = \frac{1}{1 + e^{-z}}$$



Derivative of the logistic function (sigmoid function)

Show: d/dz($\sigma(z)) = \sigma(z)(1 - \sigma(z))$

Cost function of Logistic regression and its partial derivative

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^i \log(\hat{p}^i) + (1 - y^i)\log(1 - (\hat{p}^i))]$$

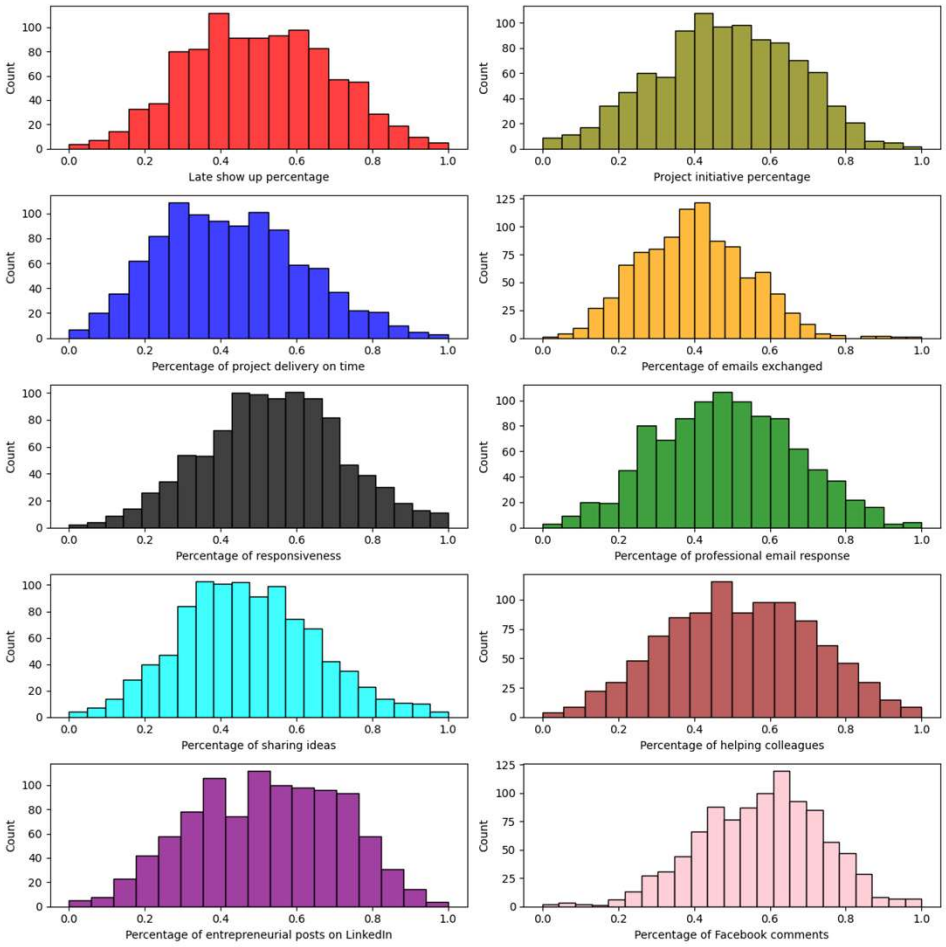
$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\sigma(\boldsymbol{\theta}^{\mathbf{T}} x^i) - y^i)\, x_j{}^i$$

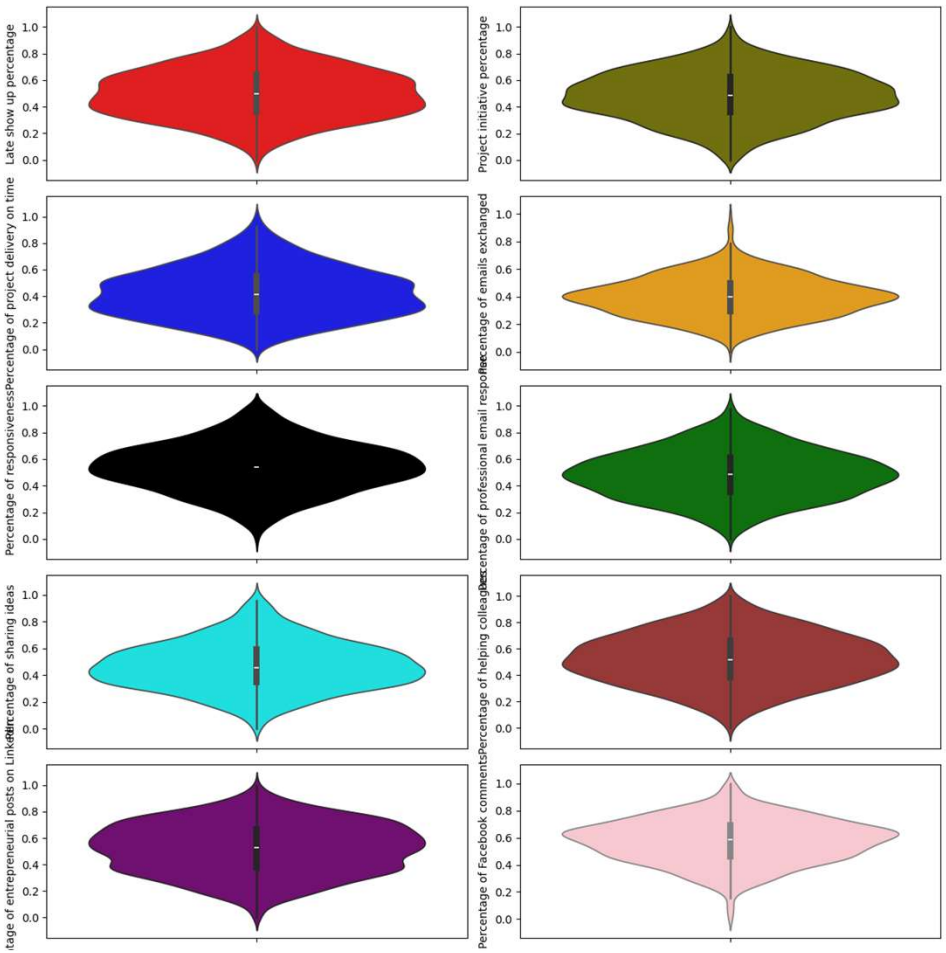
- We will discuss these in more detail later in the course

Code 3- upload its notebook (logistic regression) and dataset (HR_DataSet)

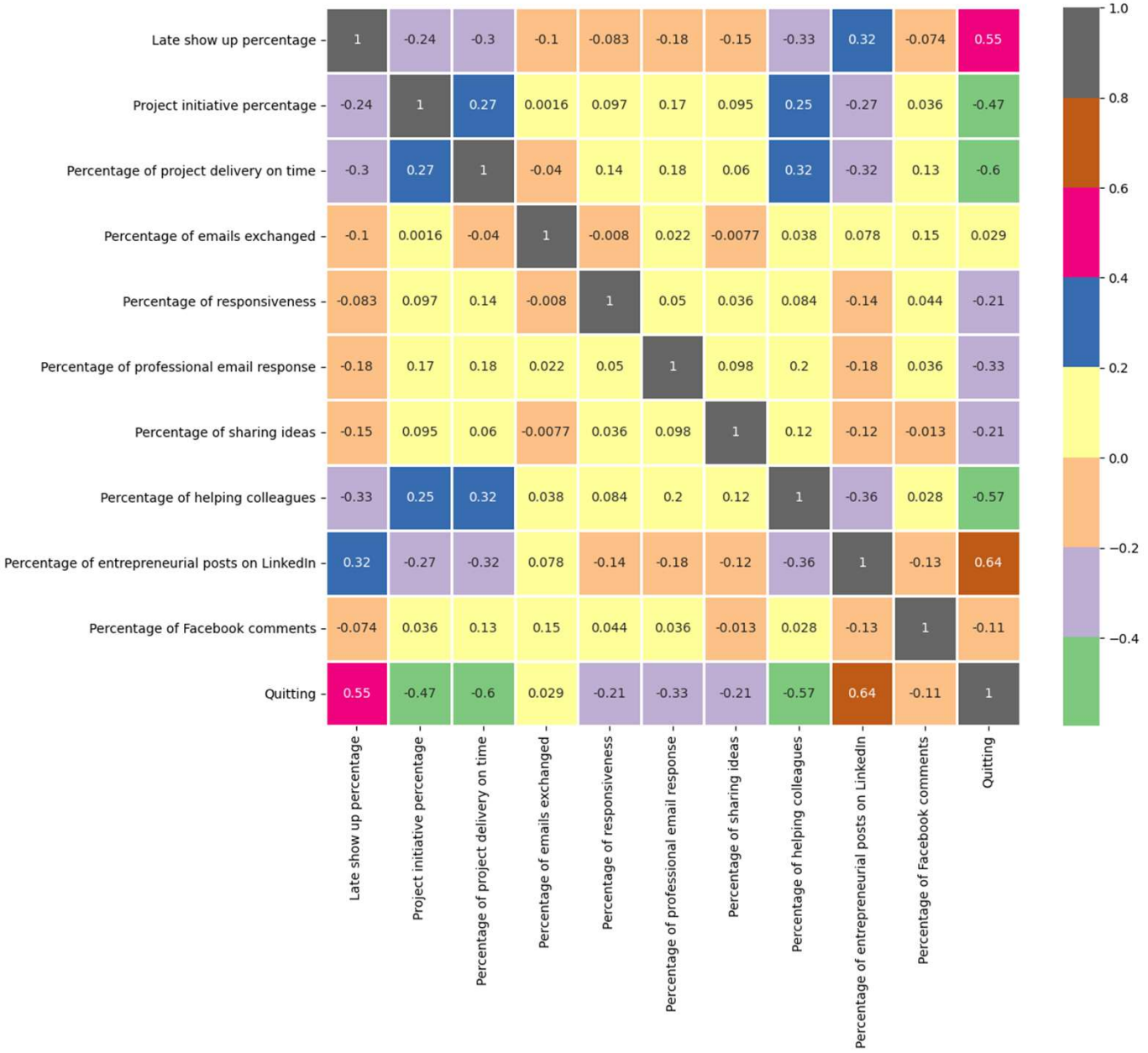
- **Data set:**
  - HR data set:
    - Late show-up percentage
    - Project initiative percentage
    - Percentage of project delivery on time
    - …
  - Target Feature
    - Quitting
- **Analytics Question:**
  - Using the HR data set, Can we create a classification logistic regression model to predict whether an employee is quitting or not?

Results from the second code





Logistic regression data – no obvious collinear relation



Confusion matrix