Learning Machine Learning with Kaggle Challenges

(2) Classification

Qiyang Hu
UCLA IDRE
October 31, 2019
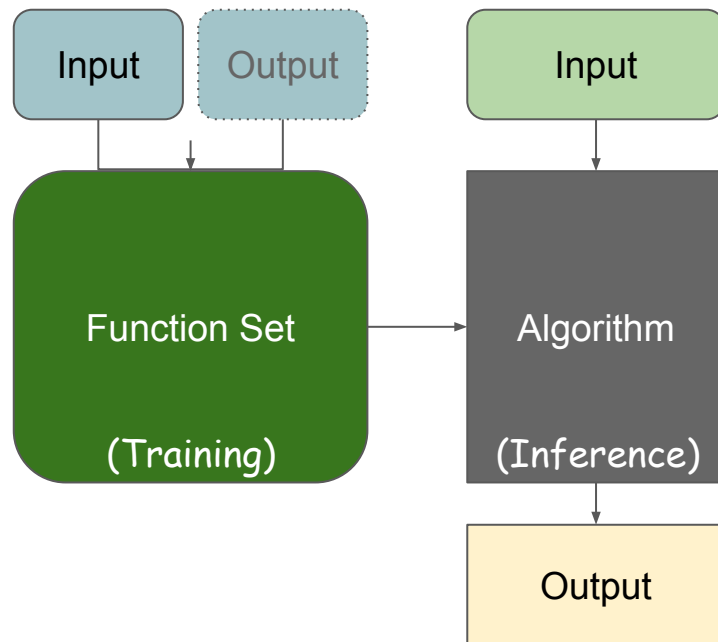
Quick Recap

The Series's Github Repo

https://github.com/huqy/
idre_learning_machine_learning

Machine Learning

Input   Output   Input

Function Set

(Training)

Algorithm

(Inference)

Output

Before running the colab demos in this series

1. Register a Kaggle account
    a. Kaggle.com → "Register"
    b. You can email me your kaggle username so that I can add you to a team for challenges.

2. Create Kaggle API token and download json file
    a. Sign in → Your Profile → "My Account" → "Create New API Token"

3. Join the 2 competitions → "Join Competition"
    a. Titantic Challenge
    b. Dogs-vs-Cats Challenge

4. Get/run the colab files
   a. Just follow the links in the github repo readme page.

Titanic Kaggle Challenge

To predict the survival or the death of a given passenger in Titanic

- Titanic Facts:
  - Survivors
    - 492 passagers
    - 214 crews
  - Victims
    - 832 passagers
    - 685 crews
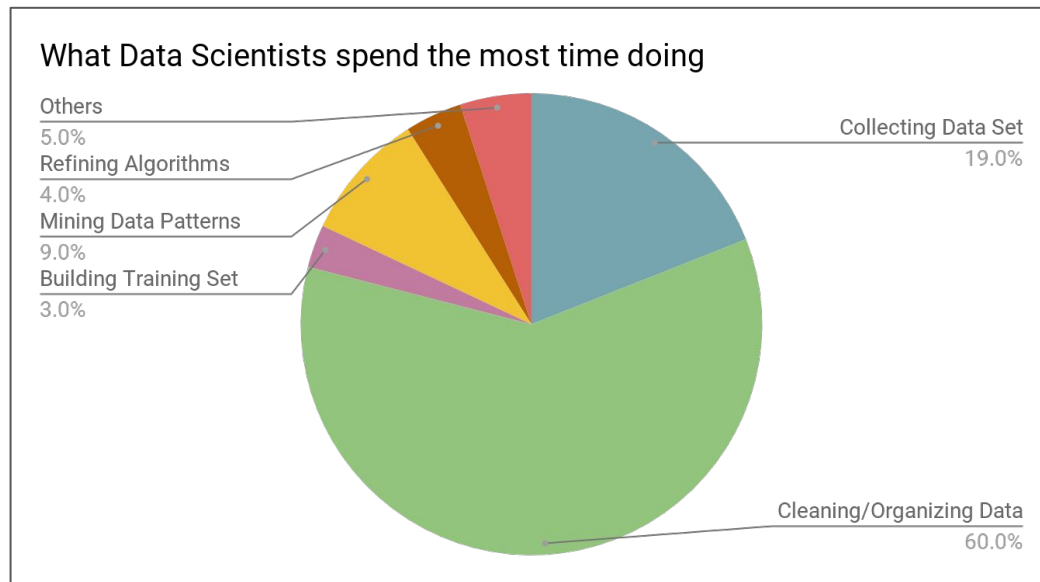   - Death causes: drowning, hypothermia, injury, suicide, …
    - List of deaths

Qiyang Hu

Workflow for a machine learning project

Step 1 — Step 2 — Step 3 — Step 4 — Step 5

Qiyang Hu
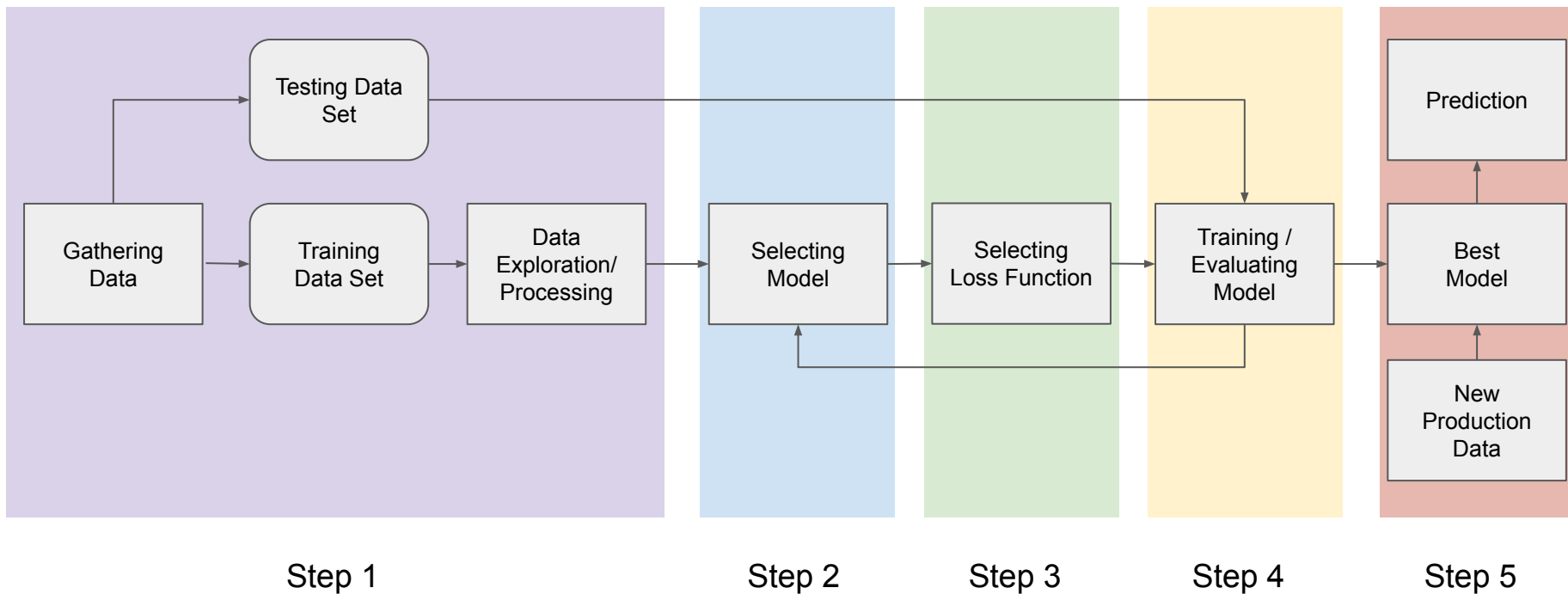
Step 1. Data Prep:

- The most time-consuming but the most creative job
    - Take > 80% time
    - Require experience
    - May need domain expertise

- Determines the upper limit for the goodness of ML
    - Models/Algorithms:
      just approach the upper limit

What Data Scientists spend the most time doing

Others
5.0%
Refining Algorithms
4.0%
Mining Data Patterns
9.0%
Building Training Set
3.0%

Collecting Data Set
19.0%

Cleaning/Organizing Data
60.0%

Survey from Forbes in 2017 (Data Source)

What we will do in Step 1

1. Get the data
2. Know the data (Exploratory Data Analysis)
3. Feature Engineering
4. Feature Selection (use the feature importance property of the model)

In Kaggle's training dataset: 891 passengers, 12 columns (features)

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Qiyang Hu

Feature Engineering

- Transforming raw data into features with a good representation

- Techniques used in our work:

  - Imputation (almost every column)

  - One-hot encoding (embarked, cabin, pclass, ticket, etc)

  - Label binarization (sex)

  - Binning and grouping (family info)

  - Scaling: standardization and normalization (age, fare, family size)

  - Splitting the features (title from name, etc)

Qiyang Hu

Workflow for a machine learning project

Step 1   Step 2   Step 3   Step 4   Step 5
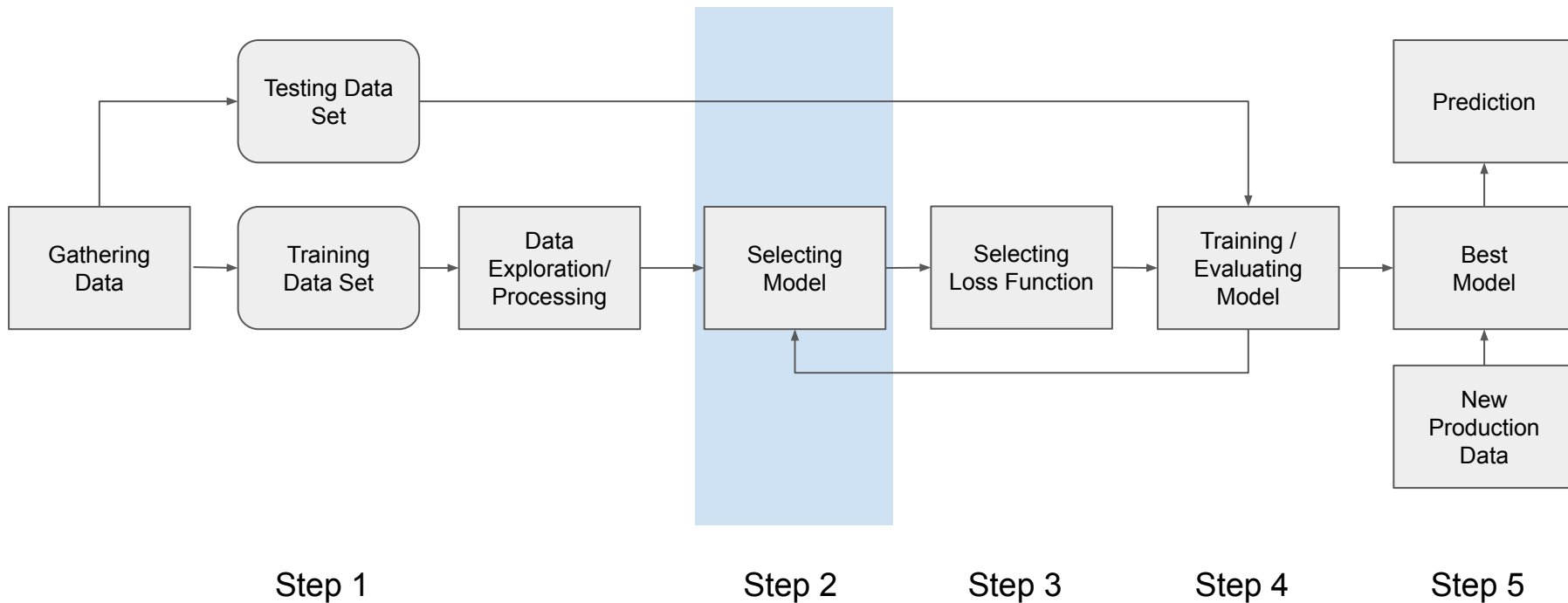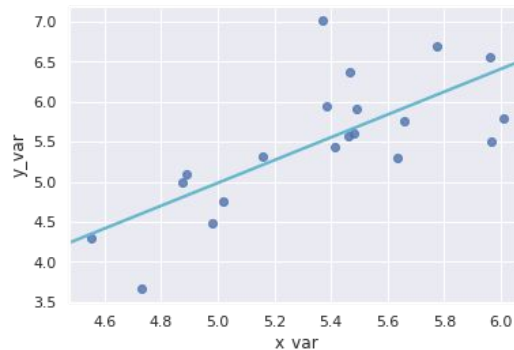
Regression: linear models

- Regression: model output as continuous variable(s)
  - For estimating or predicting a response
  - Real problems: predicting prices, controlling self-driving car, etc.

- Linear Regression:
  - Fitting the data into a straight line $y = \sum_i w_i x_i + b$
  - In Scikit learn:
    - sklearn.linear_model.LinearRegression

- Polynomial Regression:
  - Treated as linear regression
  - In Scikit Learn:
    - sklearn.preprocessing.PolynomialFeatures
    - sklearn.pipeline.make_pipeline

Classification: linear models from logistic regression

- Classification: model output as discrete variable (class labels)
  - For identifying group membership(s)
  - Examples: handwritten recognition, spam filtering, disease diagnosis
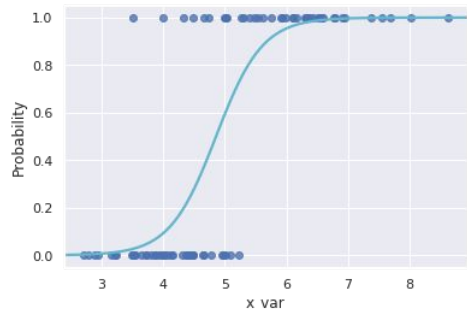
- Logistic Regression:
  - Fitting the data linearly separable
  - Linear model + Sigmoid function

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} = \sigma\left(\sum_i w_i x_i + b\right)$$

 - Scikit Learn:
    - sklearn.linear_model.LogisticRegression

- Binary classification: defining a threshold to map logistic regression
  - Logistic regression give probability

Qiyang Hu

Multi-class Classification

- Multinomial logistic regression (Softmax Regression):

  - Softmax to amplify

  - Different from Sigmoid

  - Mathematically it gives the posterior probabilities that represents a categorical distribution.

  - Commonly used as the last layer in neural networks

$$y_i = \frac{e^{z^{(i)}}}{\sum_{j=0}^{K} e^{z^{(j)}}}$$
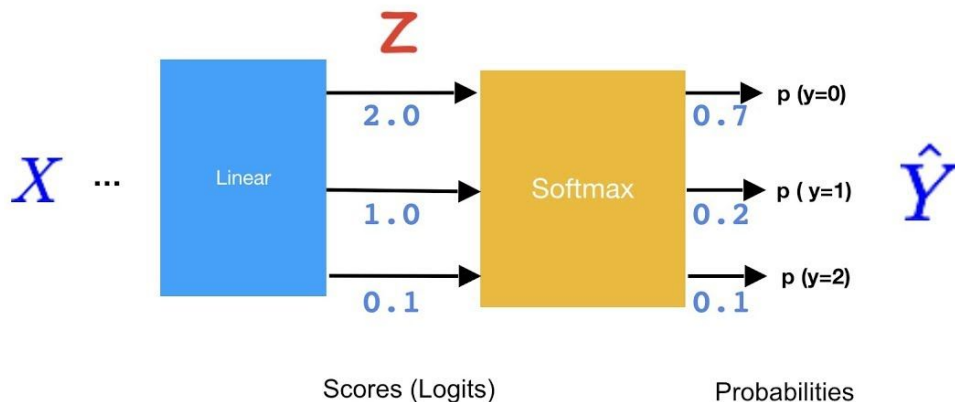
Screenshot From the youtube clip
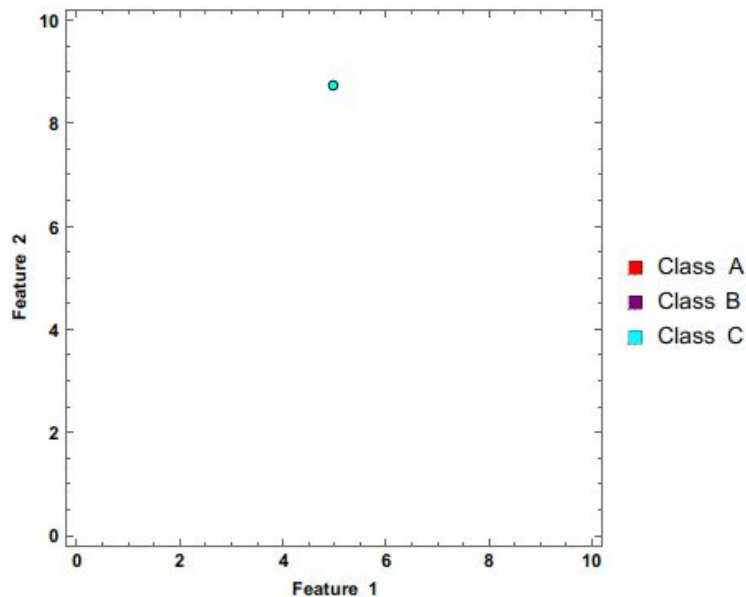
Other Classification methods

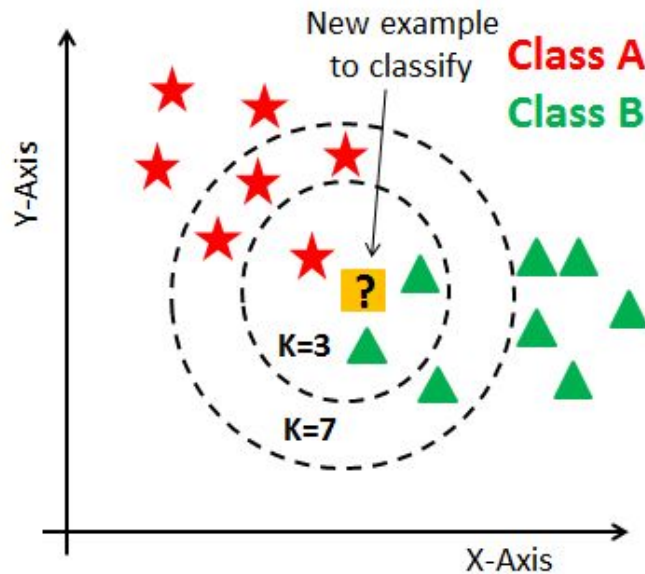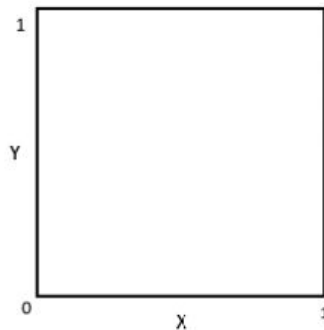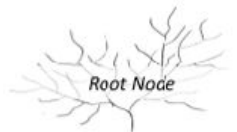- Naive Bayes

Figure from Wikimedia

- K Nearest Neighbor

Figure from post

Qiyang Hu

Other Classification methods

- ## Decision Tree

Root Node

For more tutorials: annalyzin.wordpress.com

Figure source

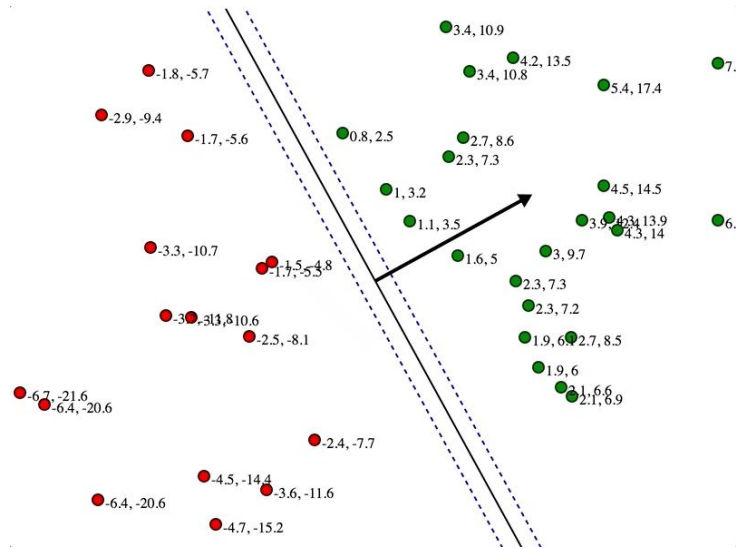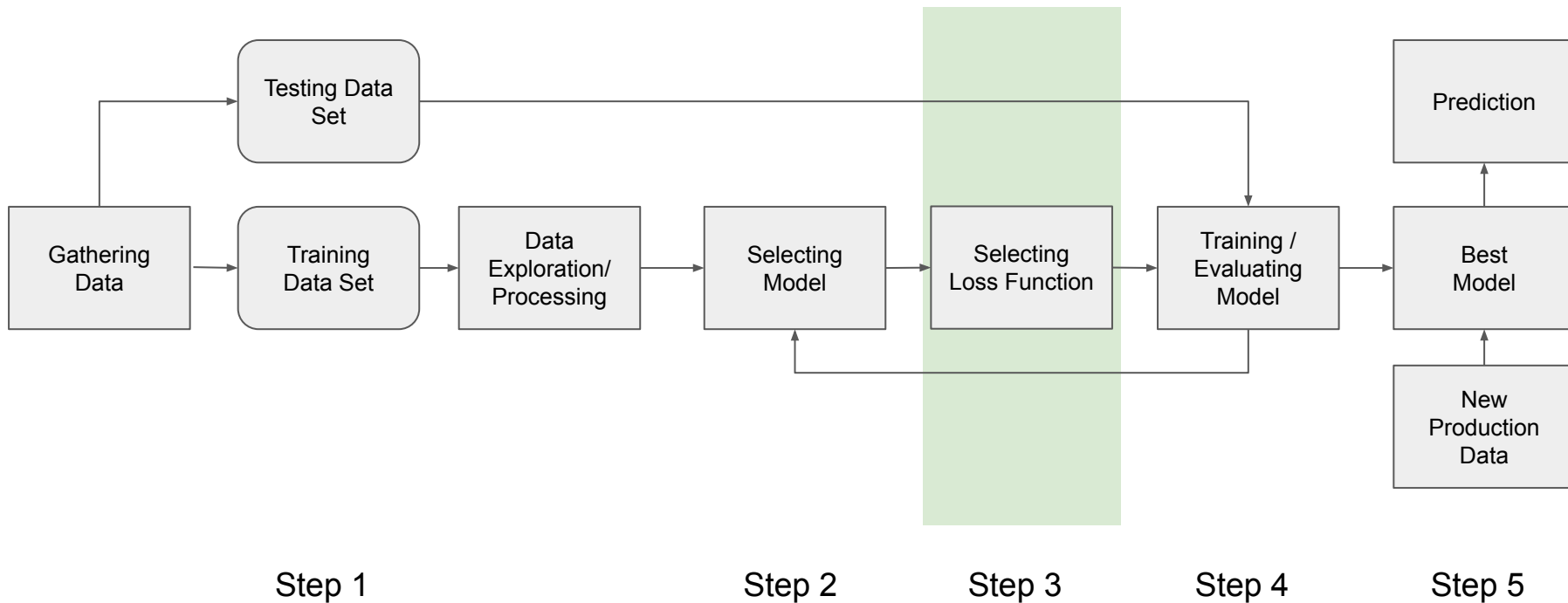- ## Support Vector Machine

Figure source

Workflow for a machine learning project

Qiyang Hu

How to measure the performance of the model?

- General name: objective function

- Measure the misfit of the model as a function of parameters
  - Criterion is to *minimize* the error functions
  - Loss Function: for a single training example
  - Cost Function: over the entire or mini-batch training set

- Evaluate the probability of generating training set
  - Criterion is to *maximize* the distribution likelihood as a function of parameters
  - Maximum (log)-likelihood estimation

- Regression losses and classification losses

Common loss functions

- ## Regression Loss
  - Mean Square Error / Quadratic Loss / L2 Loss: $L_{MSE} = \dfrac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$

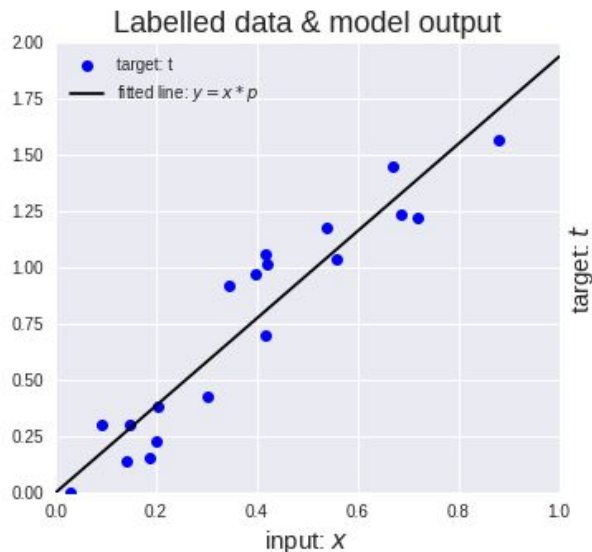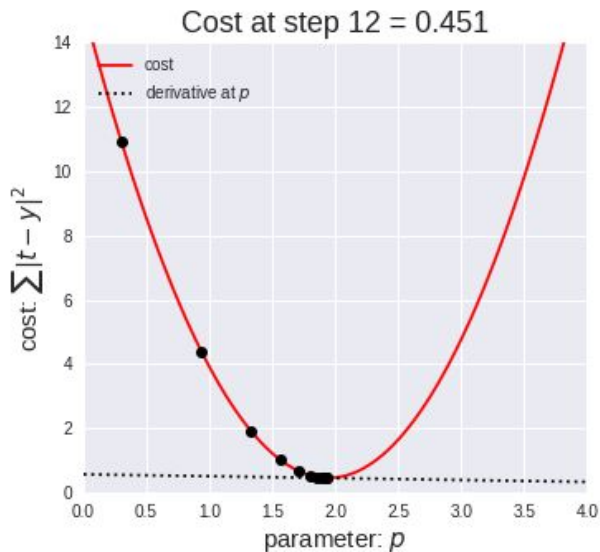 - Mean Absolute Error / L1 Loss: $L_{MAE} = \dfrac{\Sigma_{i=1}^{n}|y_i - \hat{y}_i|}{n}$

- ## Classification Loss
  - Hinge Loss / Multi class SVM Loss: $L_{Hinge} = \Sigma_{j \neq y_i} \max(0, 1 + s_j - s_{y_i})$

  - Cross Entropy Loss / Log Loss / Negative Log Likelihood:

$$L_{CEL} = -[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

Qiyang Hu

ML Optimization Algorithm

- Gradient Descent (a 1st-order approach) $\theta \longleftarrow \theta - \alpha \nabla J(\theta)$
  - Most popular algorithm
    - Pros: simple and fast
    - Cons: sometimes hard to tune

Many Gradient Descent Variations

- Stochastic GD / Mini-Batch GD

- Classical Momentum (CM)

- Nesterov's Accelerated Gradient (NAG)

- AdaGrad

- AdaDelta

- RMSprop   (animation source) →

- ADAM (as default in many libs)

- AdaBound/AmsBound (ICLR 2019)
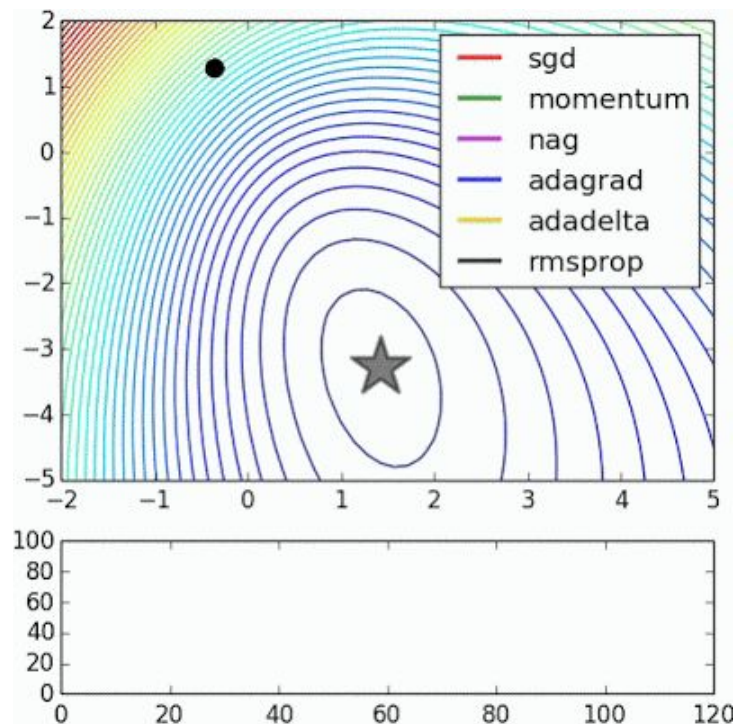
Higher Order Optimization Algorithm

- Newton-like methods (2nd-order methods)

$$\theta \longleftarrow \theta - \frac{\ell'(\theta)}{\ell''(\theta)}$$

  - Prons:
    - Fewer iterations (quadratic convergence)
    - Fewer hyperparameters
  - Cons:
    - Much more **costly** in each iteration
    - Need more storing
  - BFGS/L-BFGS: a quasi-newton one
    - Good for low dimensional models
  - CG (Conjugate gradient)
    - moderately high dimensional models

Figure from Wikipedia
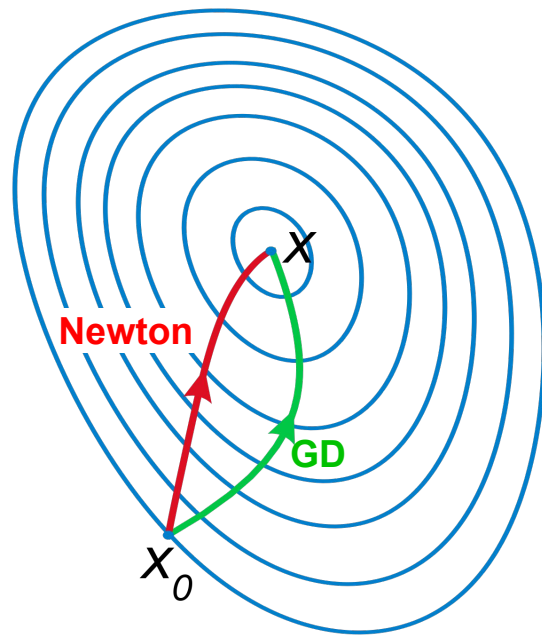
Workflow for a machine learning project

Qiyang Hu

Figure Source	**Underfitting**	**Just Right**	**Overfitting**
Regression illustration			
Classification illustration			

Qiyang Hu

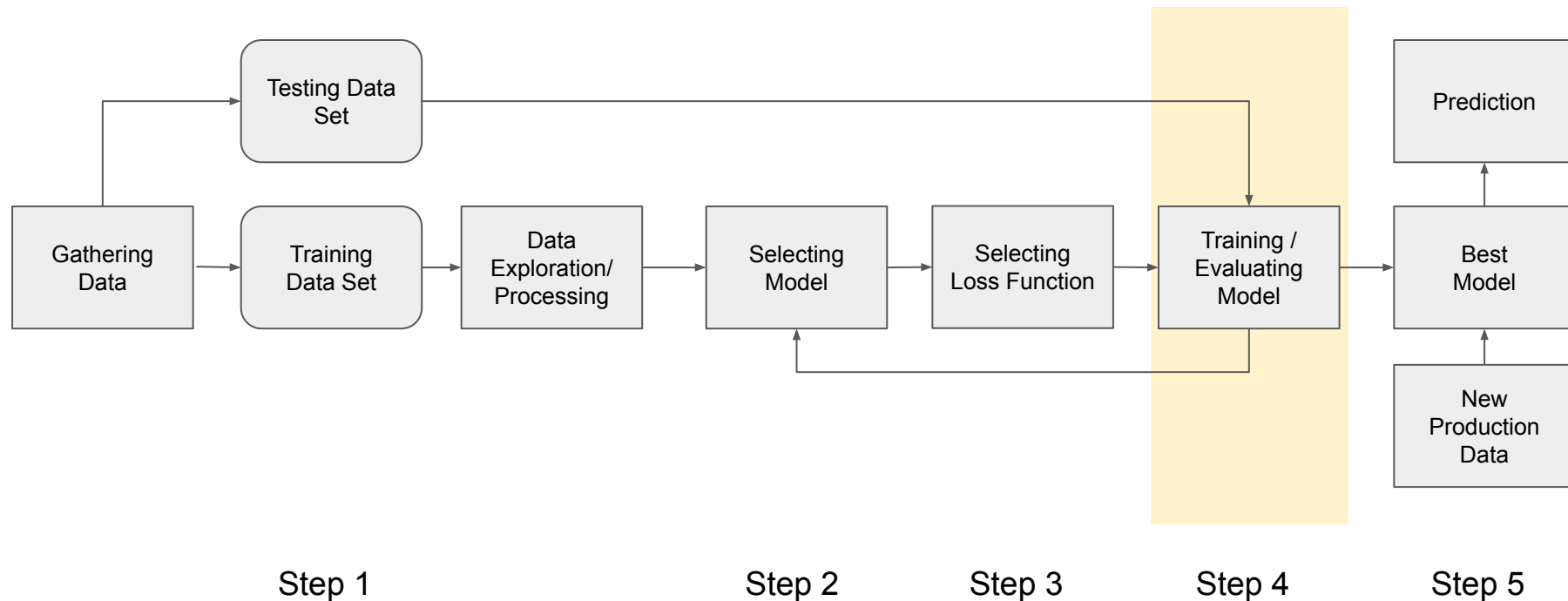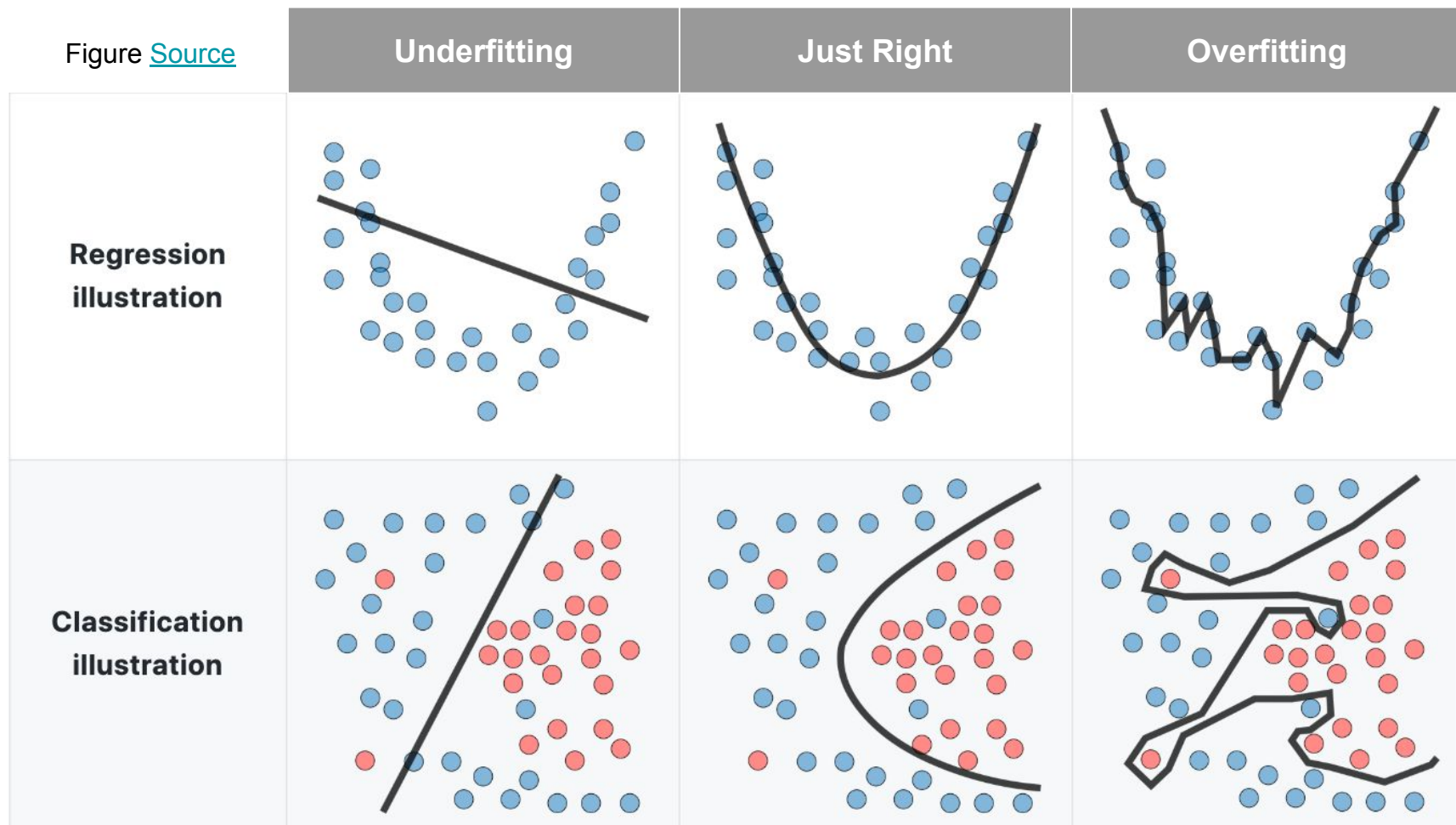# Underfitting and Overfitting

- Underfitting: model too simple:
  - Diagnose:
    - cannot even fit the training data
    - training error ~ testing error
  - Ignore the variance in training data
  - Higher prediction bias

- Overfitting: model too complex
  - Diagnose:
    - well-fit for training data
    - large error for testing data
  - Over-interpret training data
  - More deviation from new data
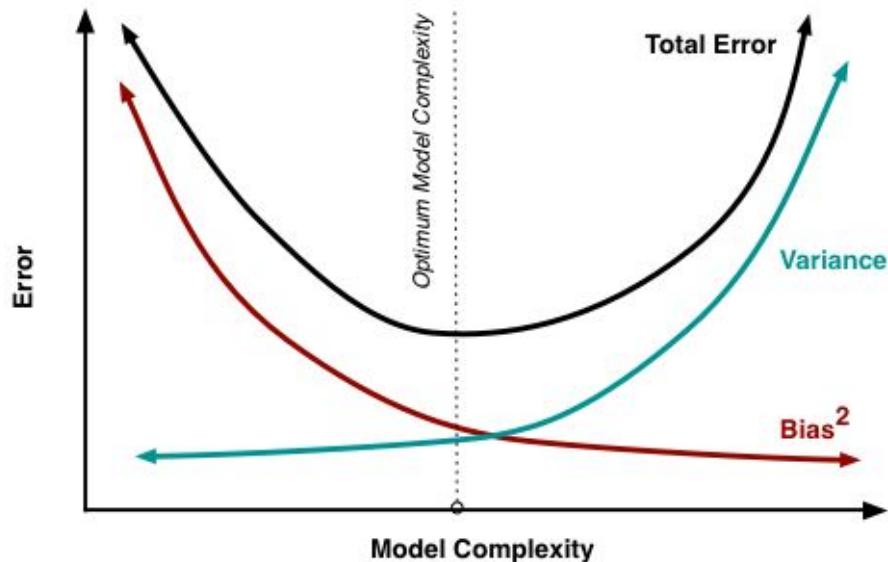
How to prevent underfitting?

- Redesign the model

- Increase model's complexity

- Add more features as input

- Training longer

- More data will <u>not</u> help

How to prevent overfitting?

- Get more data
    - Collect more data
    - Data augmentation

- Reduce the model's complexity

- Regularization
    - Weight Regularization to make the model smoother (L1, L2, Elastic net)
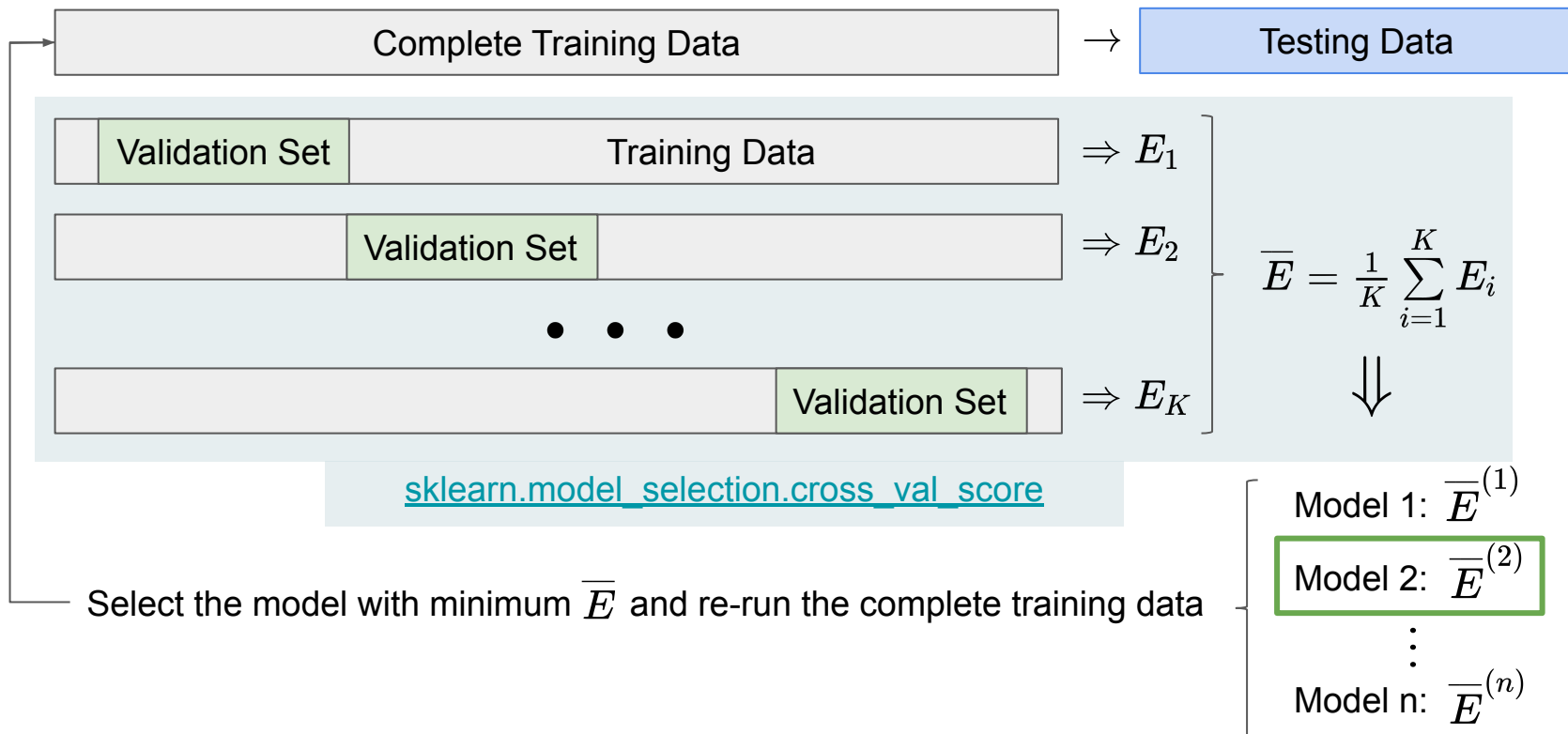
$$\hat{L}(x, y) = L(x, y) + \lambda \sum_{i=1}^{n} \theta_i^2$$

    - Dropout
    - Early stopping

Qiyang Hu

Model Selection: K-fold Cross Validation

Complete Training Data	→	Testing Data

Validation Set — Training Data $\Rightarrow E_1$

Validation Set $\Rightarrow E_2$

• • •

Validation Set $\Rightarrow E_K$

$$\overline{E} = \frac{1}{K} \sum_{i=1}^{K} E_i$$

$$\Downarrow$$

sklearn.model_selection.cross_val_score

Select the model with minimum $\overline{E}$ and re-run the complete training data

Model 1: $\overline{E}^{(1)}$

Model 2: $\overline{E}^{(2)}$

⋮

Model n: $\overline{E}^{(n)}$

Errors/scores in practice

			Public	Private
Training Set	Validation Set		Testing Set	Testing Set

Error: $E^{val} \quad < \quad E^{Pub} \quad < \quad E^{Pri}$

Score: $S^{val} \quad > \quad S^{Pub} \quad > \quad S^{Pri}$

Ensemble Methods: *wisdom of crowd*

- Bagging (bootstrap aggregating)
  - To decrease the bias to avoid <u>overfitting</u> for "strong" models
    - Decision tree -> Random Forest
  - Partitioning the data randomly to have *parallel* ensemble: each model is built independently
  - Out-of-Bag evaluation for validation

- Boosting (hypothesis boosting)
  - To decrease the bias to avoid <u>underfitting</u> for "weak" models
  - *Sequential* ensemble: try to add new models that do well where previous models lack
  - 3 major methods:
    - AdaBoost, Gradient Boost, XGBoost

- Stacking
  - Add a higher-level of classifier to decide weights between strong and weak models

Qiyang Hu

Don't forget to

- Sign in your info to the class
  - To get the email notifications

- Email me your Kaggle team name
  - To merge to the IDRE_LML team

- Contact me
  for questions or discussions
  - huqy@idre.ucla.edu
  - Office: Math Sci #3330
  - Phone: 310-825-2011

- Fill out the survey for comments:
  - https://forms.gle/t3f8CztFQpeFFksy6

Qiyang Hu