Introduction to Machine Learning

Amit Sethi, faculty, IIT Bombay
Sudhakar Kumar, student, IIT Bombay
Rishav Arjun, student, IIT Bombay

SHALA2020.GITHUB.IO
2020.04.24 :: 21:00 UTC+5:30

Learning objectives

- Define machine learning
- Distinguish between suitable and unsuitable problems for ML
- Classify ML problems into supervised and unsupervised
- Classify supervised ML problems into classification, regression, and others
- Write the difference between metric, loss, parameter, and hyperparameter
- Draw a basic picture that identifies underfitting and overfitting
- Write steps for hyper-parameter tuning

What is Machine Learning ?

The practice of developing algorithms that use data to discover a/an …

- Algorithms, or
- Models

Examples of bad and good problems for ML:

- Bad : Prediction model is well known, Too little data
- Good : Prediction model is not well known, Lots of data

Learning is improving task performance based on experience

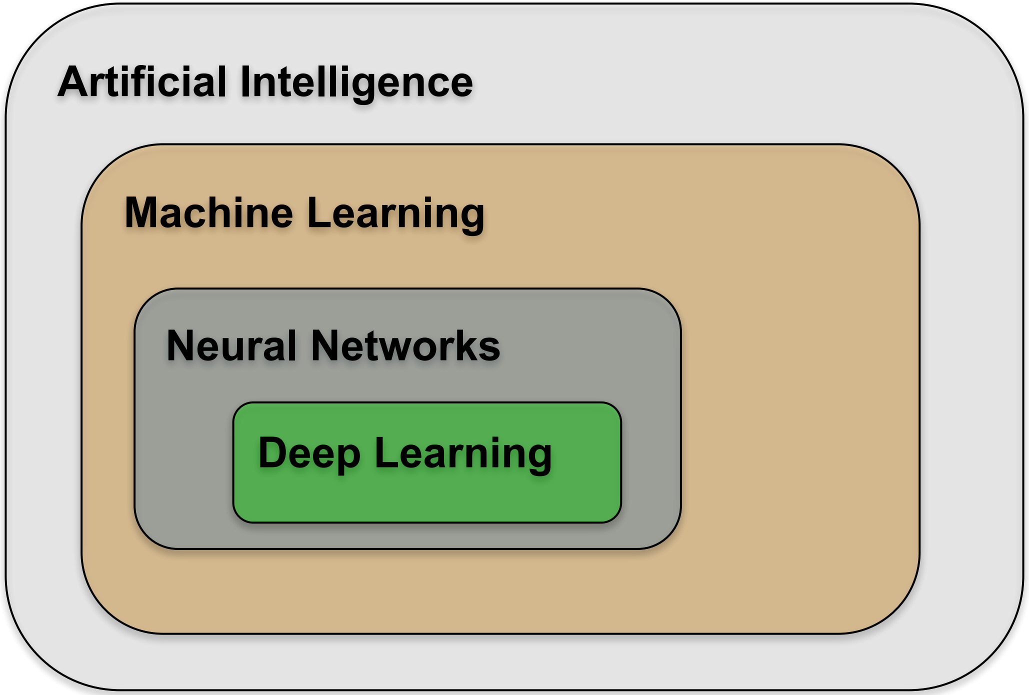
Example: Learning to ride a bicycle

- T: **Task** of learning to ride a bicycle
- P: **Performance** of balancing while moving
- E: **Experience** of riding in many situations

Steps for framing a machine learning problem

- Identify inputs
- Identify desired outputs
- Map problem to broad machine learning paradigms
- Identify performance criteria

Relation of ML to other fields



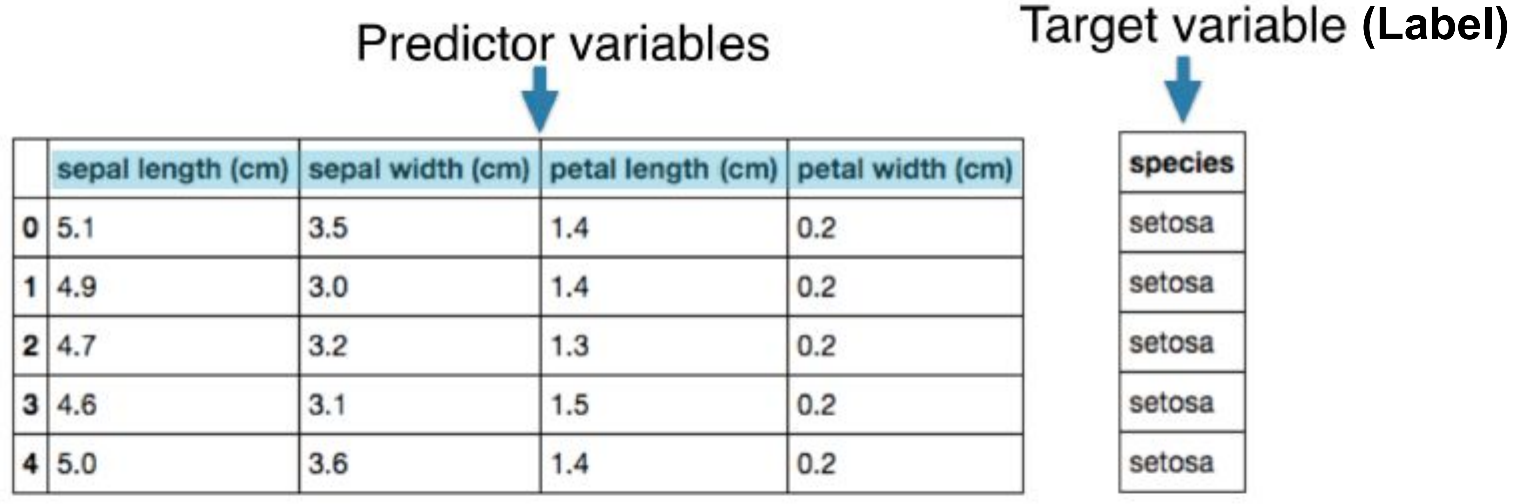
Relation of ML to other fields

Type of ML problems

- Supervised learning: uses labeled data
  - Classification: Labels are discrete
  - Regression: Labels are continuous
  - Ranking: Labels are ordinal
- Unsupervised learning: uses unlabeled data
  - Clustering: Divide data into discrete groups
  - Dimension reduction: Represent data with fewer numbers
- Somewhere in between: fewer labels than one per example
  - Semi-supervised learning: some examples are labeled
  - Weakly supervised learning: groups of examples are labeled
  - Reinforcement learning: Label (reward) is available after a sequence of steps

Supervised Learning

- Predictor variables/features and a target variable (label)
- Aim: Predict the target variable (label), given the predictor variables
  - **Classification**: Target variable (y) consists of categories
  - **Regression**: Target variable is continuous


Predictor variables
Target variable **(Label)**

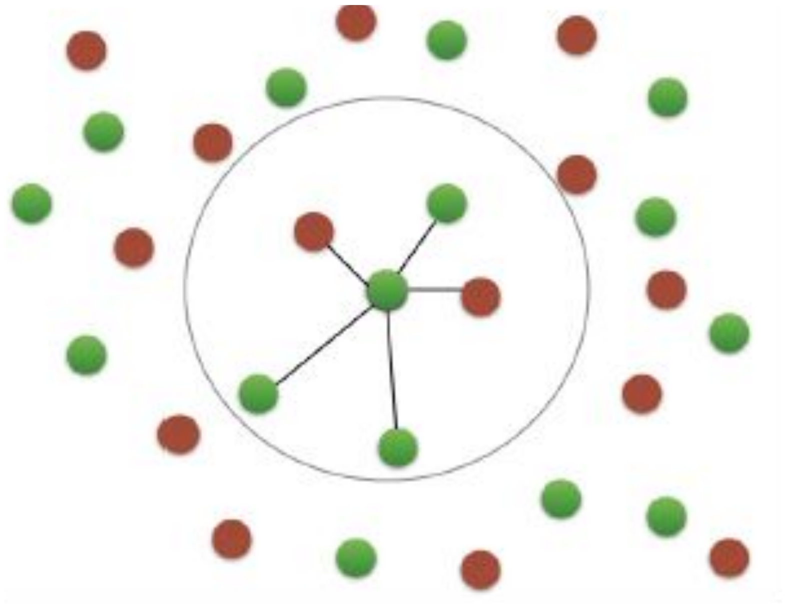
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

species
setosa
setosa
setosa
setosa
setosa

k-Nearest Neighbors

Basic idea: Predict the label of a data point by

- Looking at the 'k' closest labeled data points
- Taking a majority vote
- Works for both classification and regression



Packages in Python

- We can use scikit-learn/sklearn. It Integrates well with the SciPy stack.

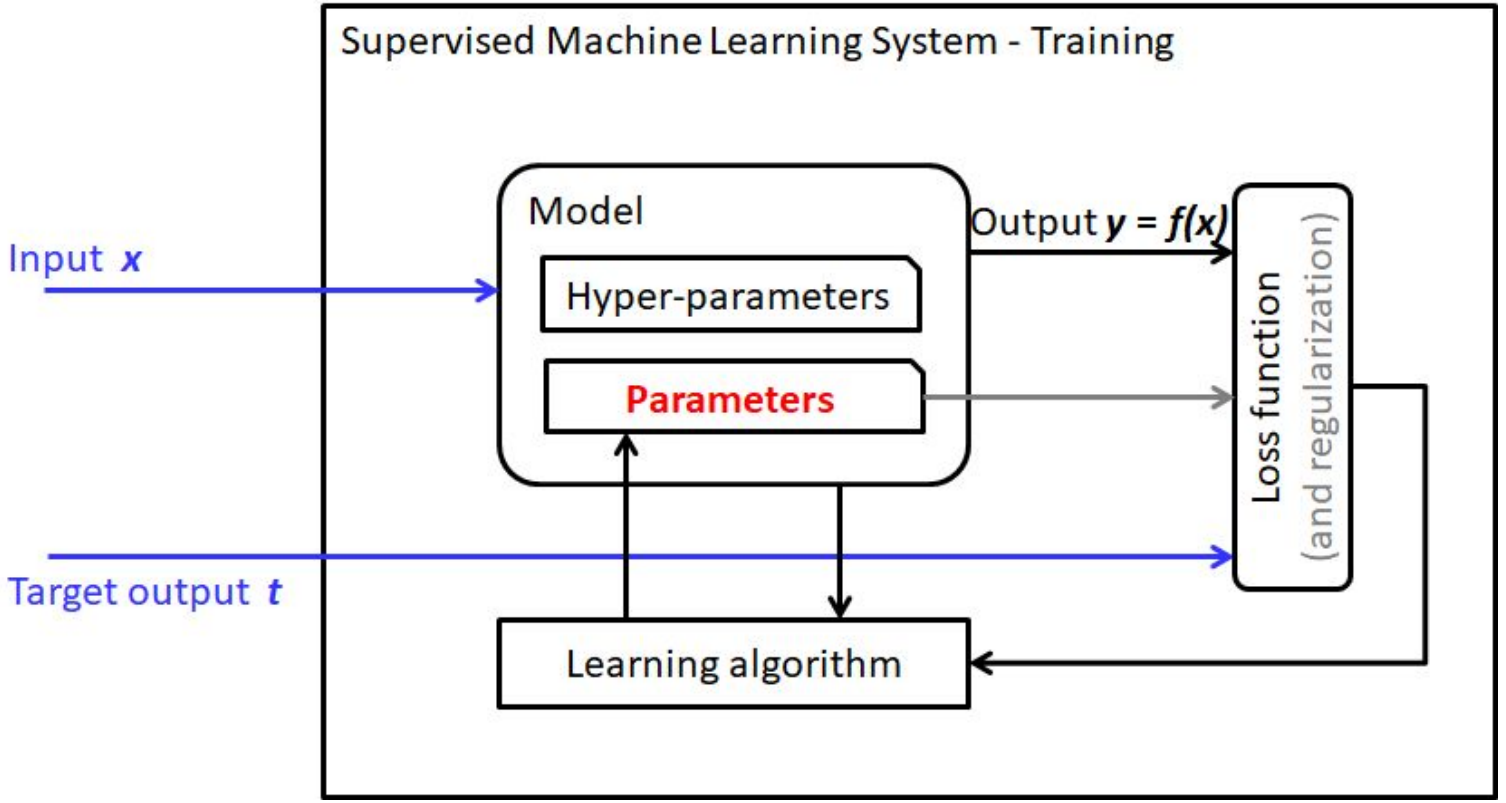
- Other libraries are TensorFlow and keras (mainly for neural networks and deep learning).

Parameters and Hyperparameters

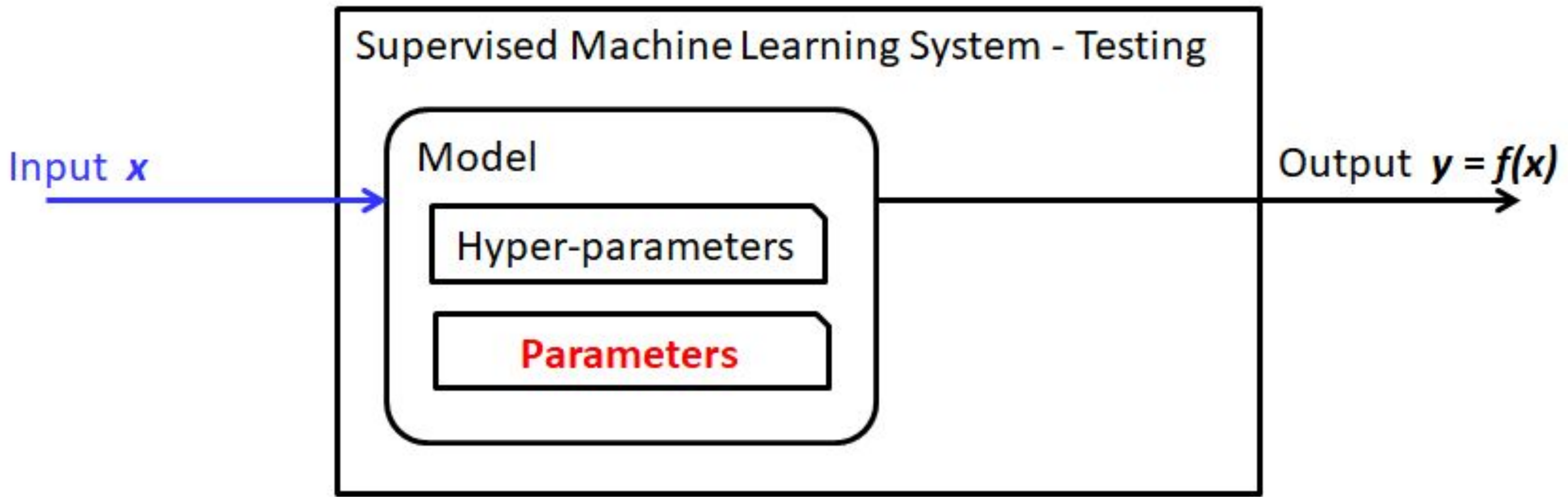
Key Concept

- Parameters: These are the variable whose values are updated during the training process of model.
  - Feature coefficient in regression model
  - Weights of a neural network
- Hyperparameters: These are the variables/ parameter whose values are fixed by model developer before the beginning of learning process.
  - Number of variables in a tree node
  - Height of a tree
  - Number of layers of a neural network

A simplified view of supervised learning

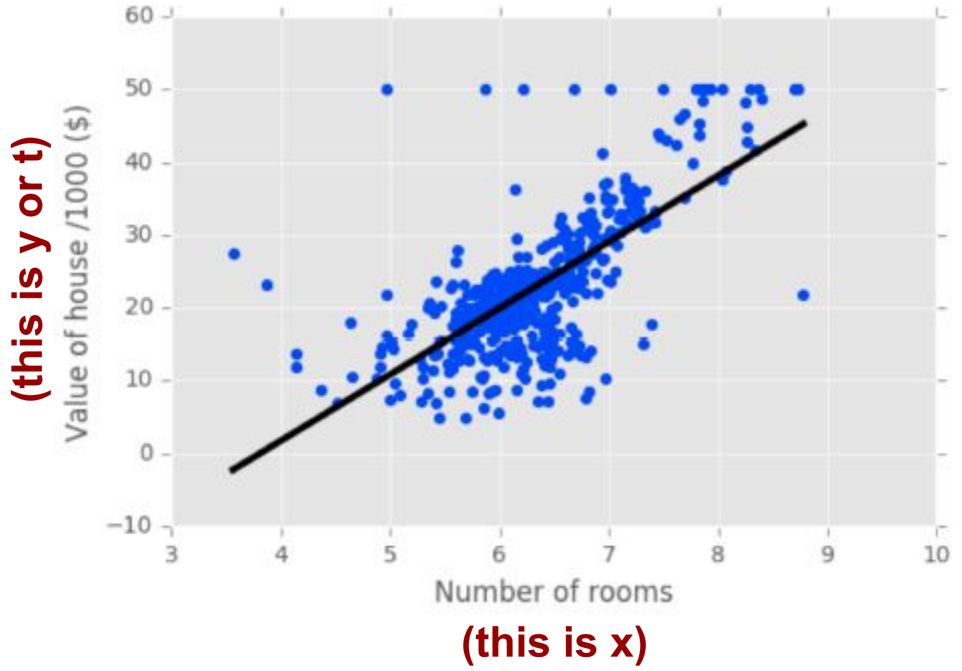


A trained model during testing (deployment)



Regression

- $y_i = f_\theta(x) = ax_i + b$
  - y_i = output
  - t_i = target (label)
  - x_i = single feature
  - a, b = parameters of model (θ)
- Slope changes with a, y-intercept changes with b (bias)
- How do we choose a and b? This is supervised learning problem
- Define an loss function and choose a line that minimizes that loss function
- We will use this loss function to update our model weights

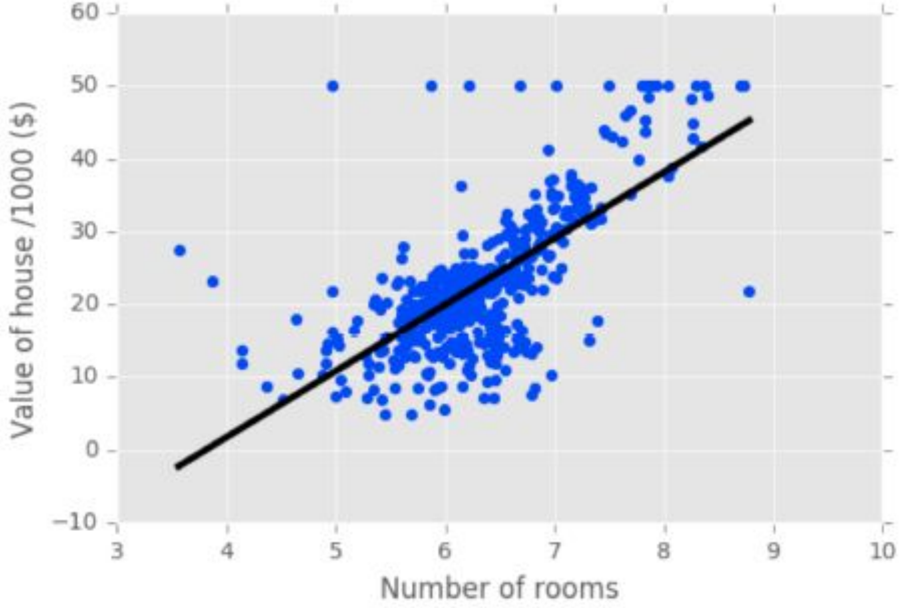


Loss: Sum of squared errors $\sum_i (y_i - t_i)^2$. The best fit line minimizes this term.

Regression

In higher dimensions, we must specify coefficient for each feature, and the bias

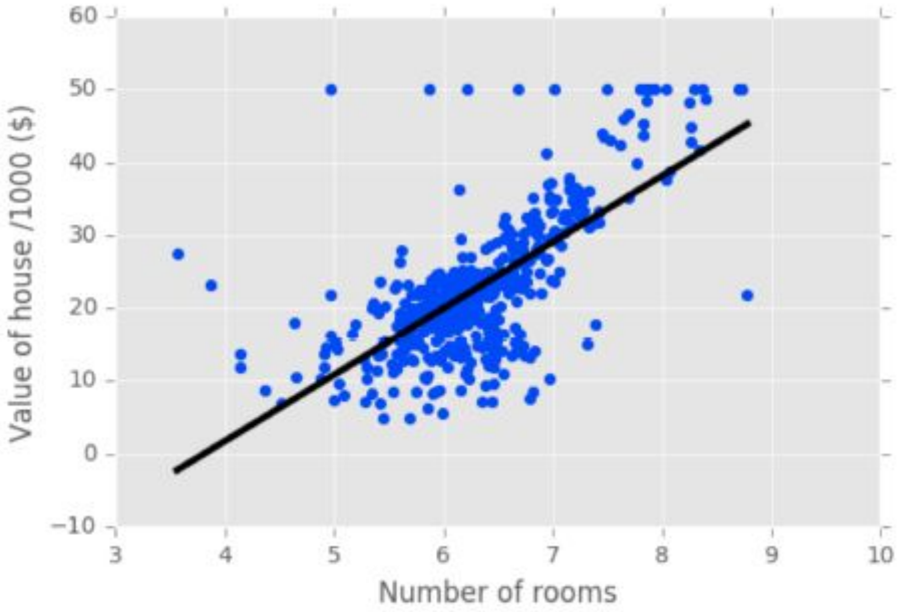
- $y = a_1x_1 + a_2x_2 + a_3x_3 + ... + a_nx_n +$ b



Regression

In higher dimensions, we must specify coefficient for each feature, and the bias

- $y = a_1x_1 + a_2x_2 + a_3x_3 + ... + a_nx_n + b$



Hyperparameter:

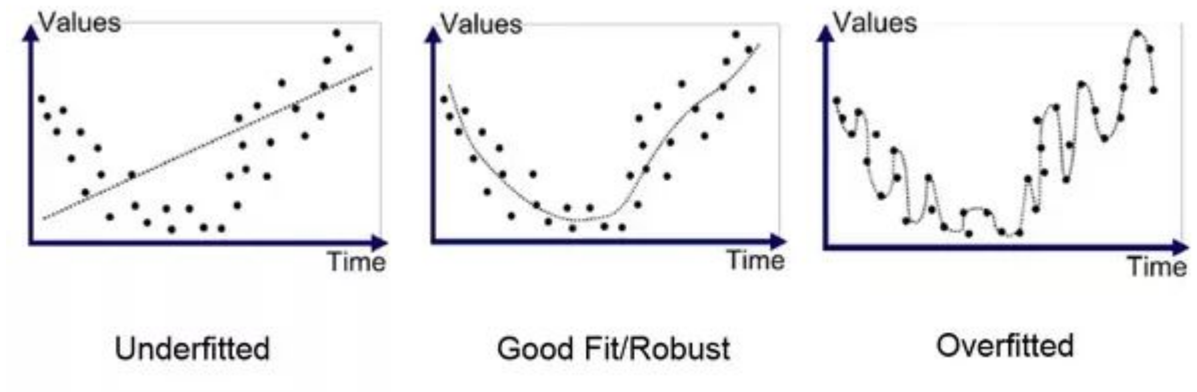
- Should we include x_i^p terms?
- What about product terms $x_i^p x_j^q$?

Regularized regression

Key Concept

Informal definition: Restrict the model from becoming too wild such that it over-fits

- Optimization minimizes a loss function
- For example, it finds an optimal coefficient for each variable that leads to minimum loss
- Large coefficients can lead to overfitting
- Penalizing large coefficients: Regularization



Original image source unknown

Ridge Regression

Loss function = OLS loss function + $\alpha * \sum_{i=1}^{n} {a_i}^2$

- Alpha is a hyperparameter and it controls model complexity
- Alpha = 0: We get back OLS (Can lead to over-fitting)

Lasso Regression

Loss function = OLS loss function + $\alpha * \sum_{i=1}^{n} |a_i|$

- Can be used to select important features of a dataset
- Shrinks the coefficients of less important features very close to 0

Types of dataset

Key Concept

- Training data: The sample of data used to fit the model
- Validation data: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.
- Testing data: The sample of data used to provide an unbiased evaluation of the final model.

Validation data should be representative of test situation (deployment). Otherwise, you are fooling yourself !!

Generally, we split data into training and testing (80/20 is indeed a good starting point). After that we can split the training data further into training and validation (again, 80/20 is a fair split).

Model Fit: Underfitting vs. Overfitting

Key Concept

- Underfitting: When the model performs poorly on the training data.
- Overfitting: When model perform well on the training data but but does not perform well on the evaluation da

Loss Function: $\frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n\}^2 + \frac{\lambda}{2}\|w\|^2$

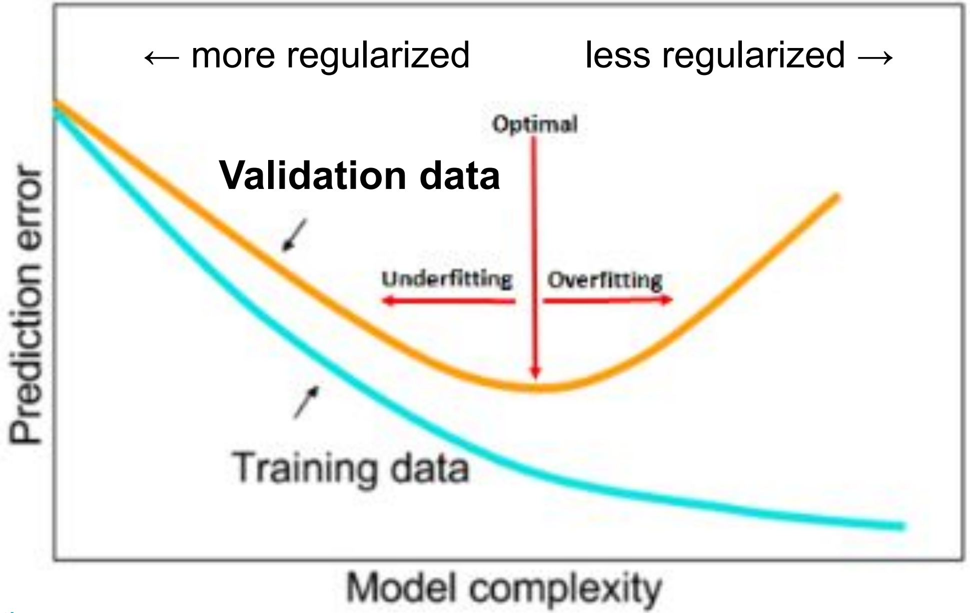
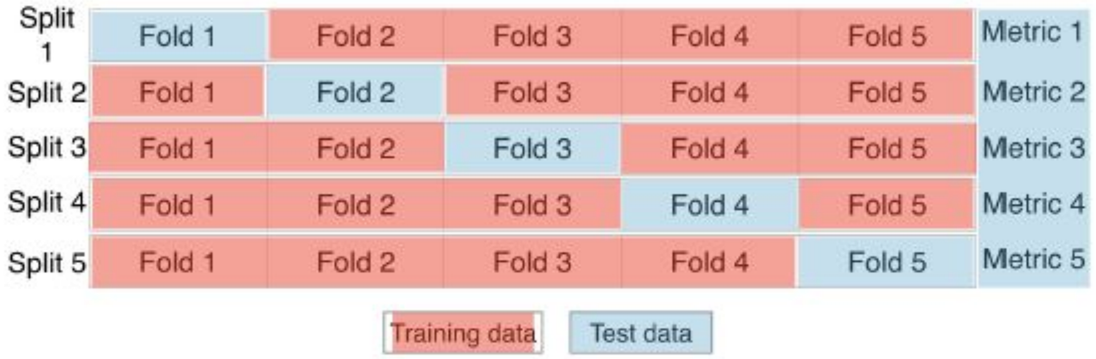


Image sources:
https://media.arxiv-vanity.com/render-output/2177140/under-overfitting.png

Cross-validation

- Model performance measurement is dependent on way the data is split
- Not representative of the model’s ability to generalize
- Solution: Cross-validation, especially when data is less


Split 1 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5 Metric 1
Split 2 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5 Metric 2
Split 3 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5 Metric 3
Split 4 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5 Metric 4
Split 5 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5 Metric 5
Training data
Test data

Cross-validation

- k folds = k-fold CV
- More folds = More computationally expensive

Final metric would be the average of all k metrics value.

Bias-variance trade-off

Generalization of model is bounded by the two undesirable outcomes high bias and high variance.

- Underfitting: High bias, Low variance
- Overfitting: Low bias, High variance

Bias occurs when an algorithm has limited flexibility to learn the true signal from the dataset. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

Variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

Loss versus performance metric

- Loss is a convenient expression used for guiding the learning (optimization)
- Loss is related to performance metric, **but** it is not the same
- Loss also includes regularization
- Performance metric is what is used to judge the model
- Performance metric on only the held-out (validation or test) data makes sense

Measuring model performance

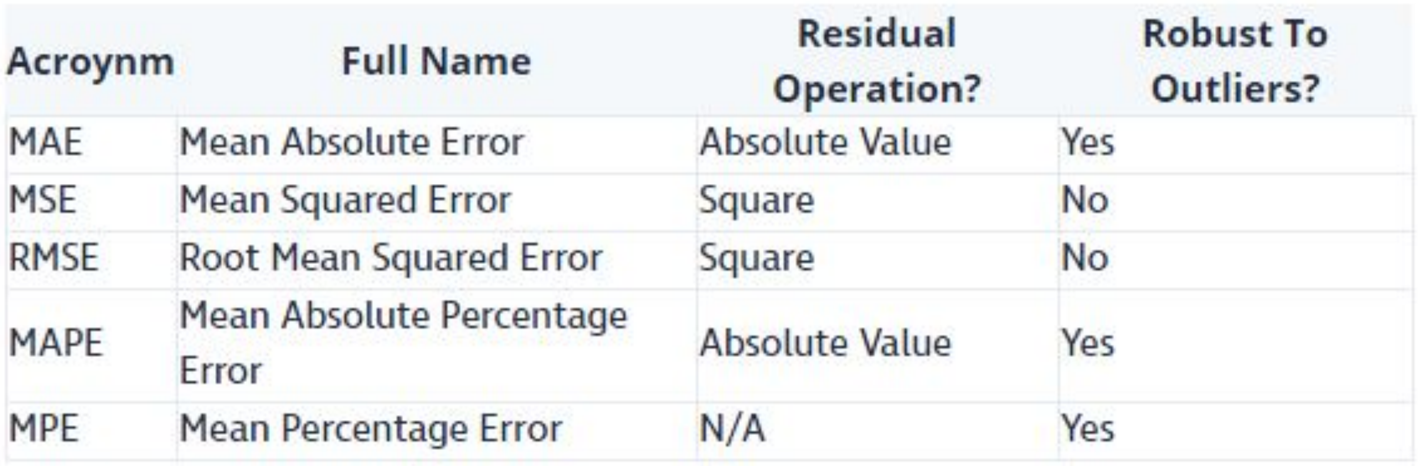
1. **Regression Metrics**

$$MAE = \overset{\text{Divide by the total number of data points}}{\boxed{\frac{1}{n}}}\ \underset{\text{Sum of}}{\Sigma}\ \underbrace{\left| \overset{\text{Actual output value}}{\boxed{y}} - \overset{\text{Predicted output value}}{\boxed{\hat{y}}} \right|}_{\text{The absolute value of the residual}}$$

$$MSE = \frac{1}{n}\,\Sigma\,\underbrace{\left(y - \hat{y}\right)}_{\text{The square of the difference between actual and predicted}}^{2}$$

$$MAPE = \frac{100\%}{n}\,\Sigma\,\left|\frac{y-\hat{y}}{y}\right|$$

https://www.dataquest.io/blog/understanding-regression-error-metrics/



Acroynm	Full Name	Residual Operation?	Robust To Outliers?
MAE	Mean Absolute Error	Absolute Value	Yes
MSE	Mean Squared Error	Square	No
RMSE	Root Mean Squared Error	Square	No
MAPE	Mean Absolute Percentage Error	Absolute Value	Yes
MPE	Mean Percentage Error	N/A	Yes

- The effect of the square term in the MSE equation is most apparent with the presence of outliers in our data. Model will be penalized more for making predictions that differ greatly from the corresponding actual value.
- The RMSE is analogous to the standard deviation (MSE to variance) and is a measure of how large your residuals are spread out.
- MAPE has a clear interpretation since percentages are easier for people to conceptualize. But it’s undefined for data points where the value is 0.

2. Classification Metrics

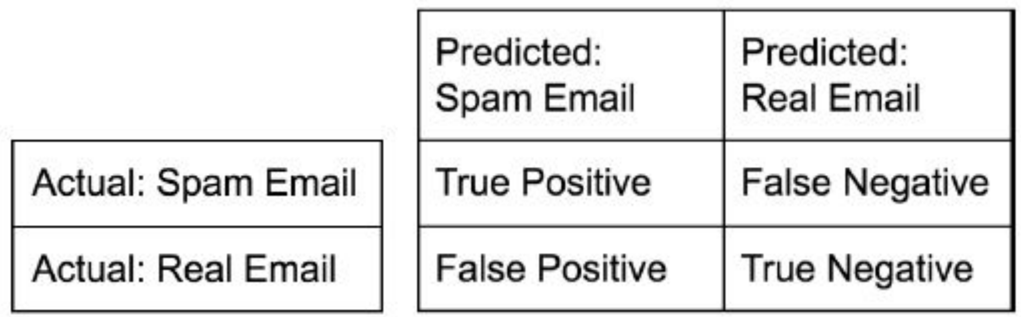
- **Accuracy** is a commonly used metric. It is fraction of correct predictions

  But it is not always a useful metric. For example in spam classification problem 99% of emails are real; 1% of emails are spam. We could build a model that predicts **all** emails as real and still it is 99% accurate !

- Other metrics can be
  - Precision
  - Recall
  - ROC (Receiver operating characteristic)

Confusion matrix

It is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one.



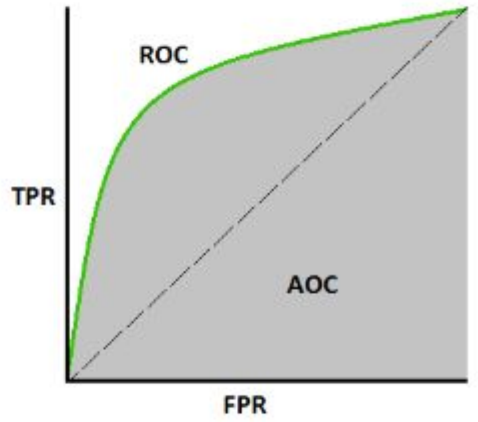
	Predicted: Spam Email	Predicted: Real Email
Actual: Spam Email	True Positive	False Negative
Actual: Real Email	False Positive	True Negative

- Accuracy is defined as $$\frac{tp + tn}{tp + tn + fp + fn}$$

Precision = $\frac{tp}{tp + fp}$

Recall = $\frac{tp}{tp + fn}$

- High precision: Not many real emails predicted as spam
- High recall: Predicted most spam emails correctly



ROC curve: The ROC curve is plotted with True Positive rate (TPR) against the FPR where TPR is on y-axis and FPR is on the x-axis.

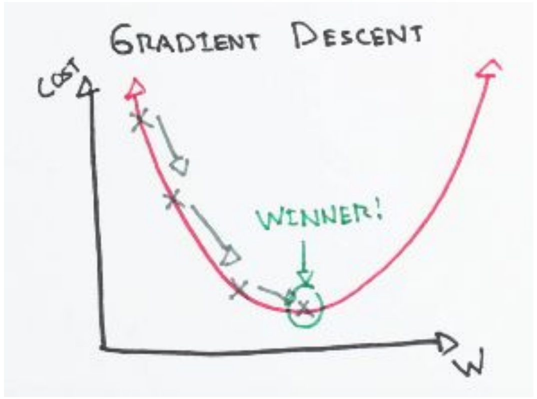
AUROC curve: AUC tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. An excellent model has AUC near to the 1.

$$\text{TPR / Recall / Sensitivity} = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{TN + FP}$$

https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Parameter optimization (weight update)

- **Gradient descent**: It is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.
- In machine learning, we use gradient descent to update the parameters of our model. Parameters refer to coefficients in Linear Regression.



https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

Parameter optimization (weight update)

- **Learning rate**: The size of these steps is called the learning rate. A low learning rate is desirable, but it requires more computation power.
- **Cost function (L)**: A Loss Functions tells us “how good” our model is at making predictions for a given set of parameters. The cost function has its own curve and its own gradients.

$$w_{\text{new}} = w_{\text{current}} - \eta \frac{\partial L}{\partial w_{\text{current}}}$$

Unsupervised learning

- Unsupervised learning finds patterns in data but without a specific prediction task in mind
- For example, clustering customers by their purchases
- Represent 50 genes by just two numbers (dimension reduction)

Feature normalization

- It is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units.
- If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

Source: https://www.geeksforgeeks.org/ml-feature-scaling-part-2/

Techniques to perform Feature Scaling

- Min-Max Normalization: This technique re-scales a feature or observation value with distribution value between 0 and 1.

- Standardization: It is a very effective technique which re-scales a feature value so that it has distribution with 0 mean value and variance equals to 1.