Introduction to Machine Learning

Presenters

Senior Data Scientist,
Personalization,
Customer Tech

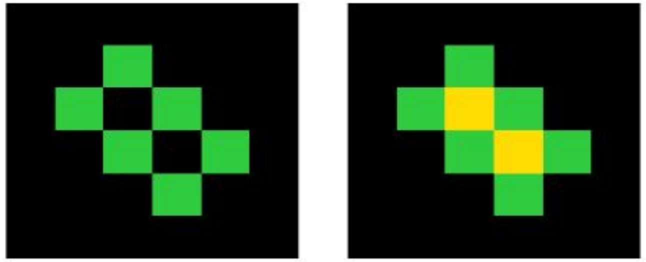
Data Scientist
Data Science Foundations,
Platform

Contents

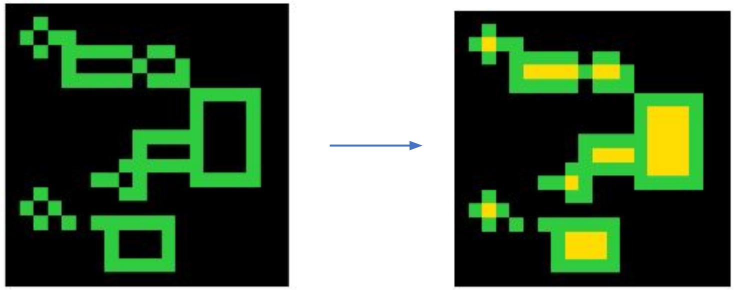
- AI, Machine Learning Overview
- Supervised Machine Learning
  - Regression
  - Classification
- Classification
  - kNN Classifier + code walkthrough
  - Logistic Regression Classifier + code walkthrough
- Data Science Modeling Pipeline
  - Train – Test Splits
  - Feature Pre-processing
  - Diagnosing and Fixing Underfitting/Overfitting

Humans are great at

Abstraction and Reasoning



Input Output



Humans are great at ...

Learning tasks.

Example: Humans learning to drive a car **vs** training a Self Driving Car.

Humans are good at

Planning:

- A birthday party
- A vacation
- Business strategies
- Navigating Traffic

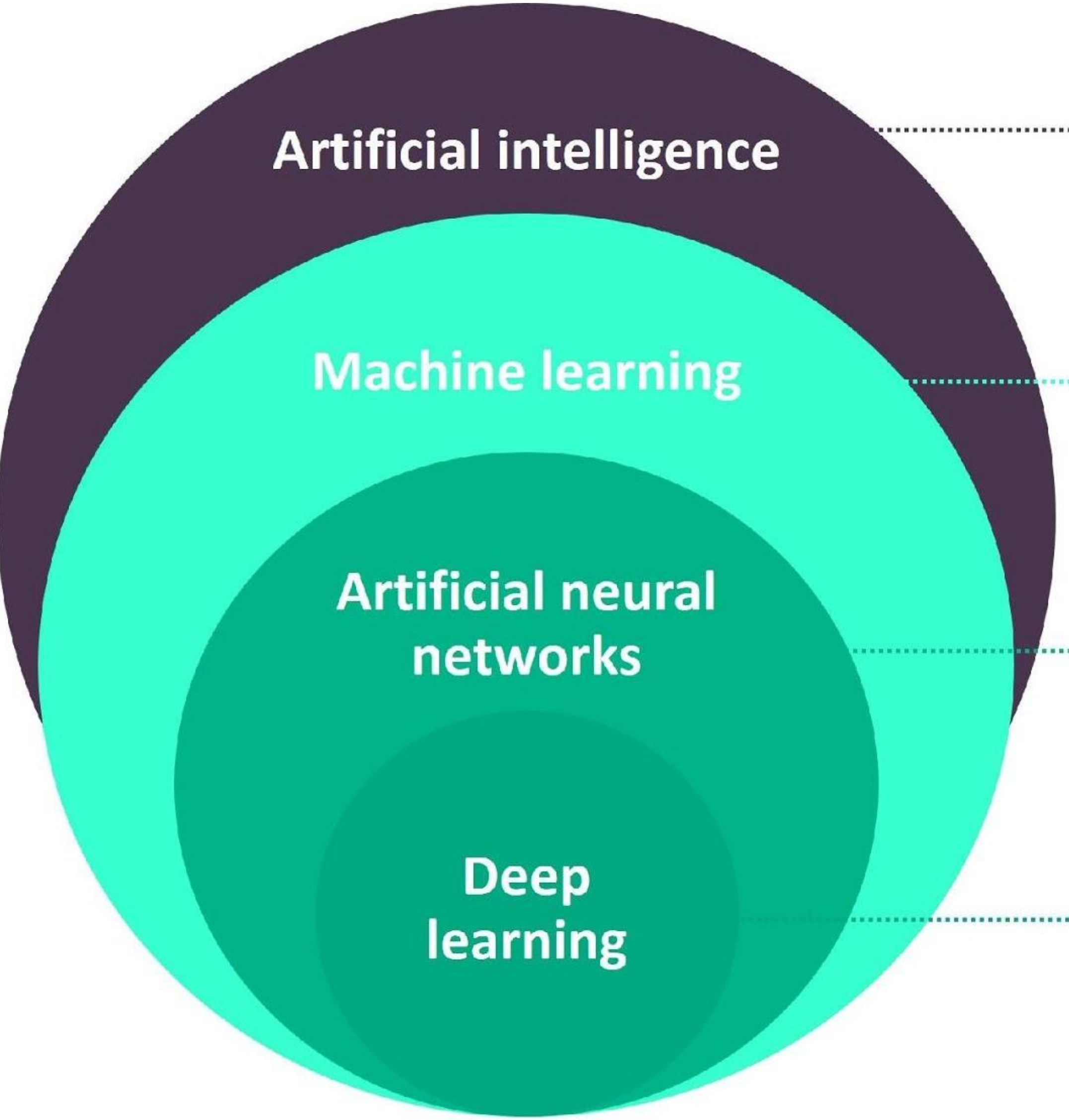
Natural Language Understanding

- Complex sentences (Legal)
- Short sentences (Twitter)
- Sarcasm
- Humor
- Emotion
- Innate understanding of the world

Visual Perception

- Scene Understanding
- Depth Perception
- Object Recognition

Artificial Intelligence



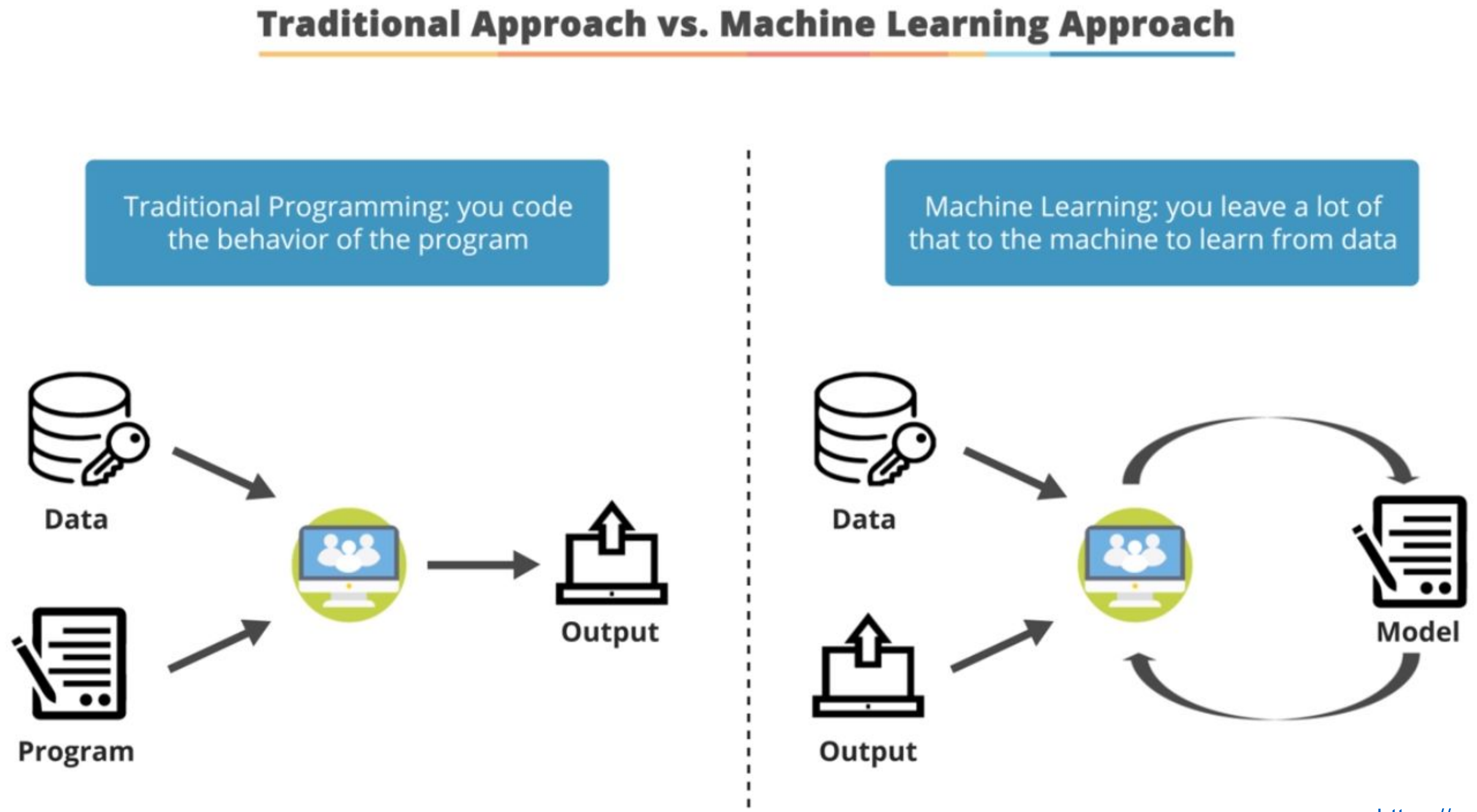
Artificial intelligence (AI)
Any techniques that enable machines to solve a task in a way like humans do

Machine learning (ML)
Algorithms that allow computers to learn from examples without being explicitly programmed

Artificial neural networks (ANN)
Brain-inspired machine learning models

Deep learning (DL)
A subset of ML which uses deep artificial neural networks as models and automatically builds a hierarchy of data representations

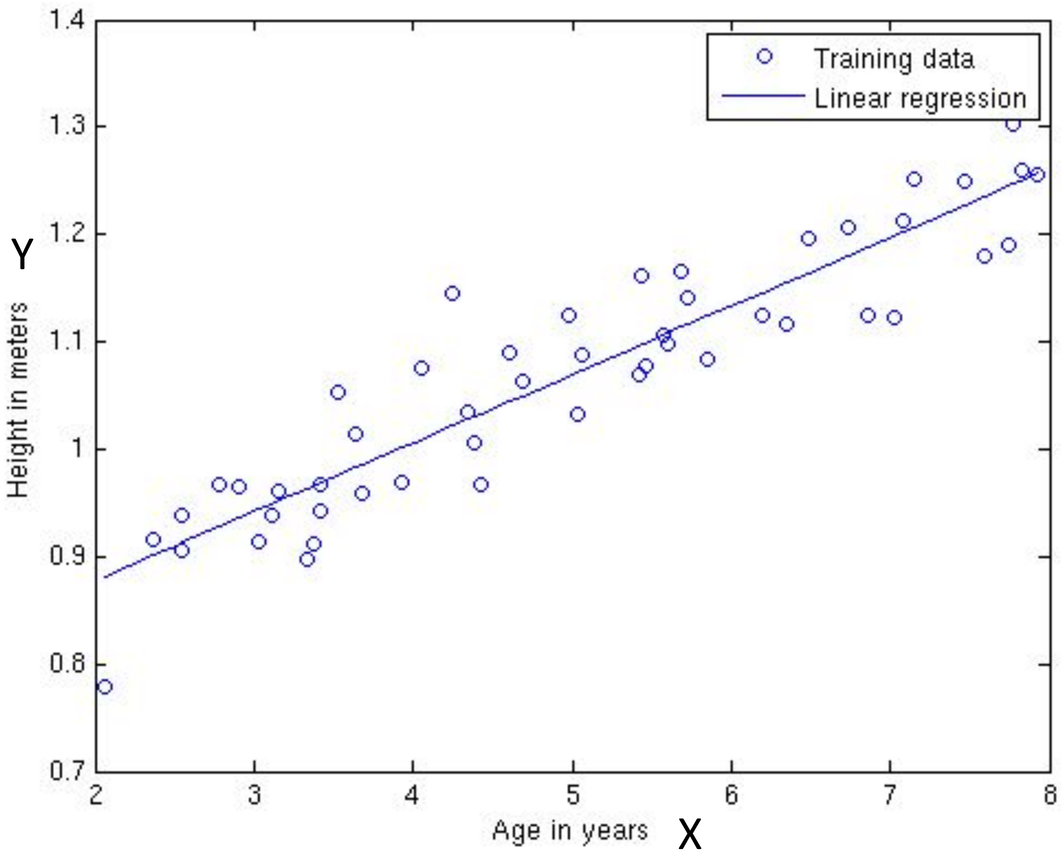
Supervised Machine Learning



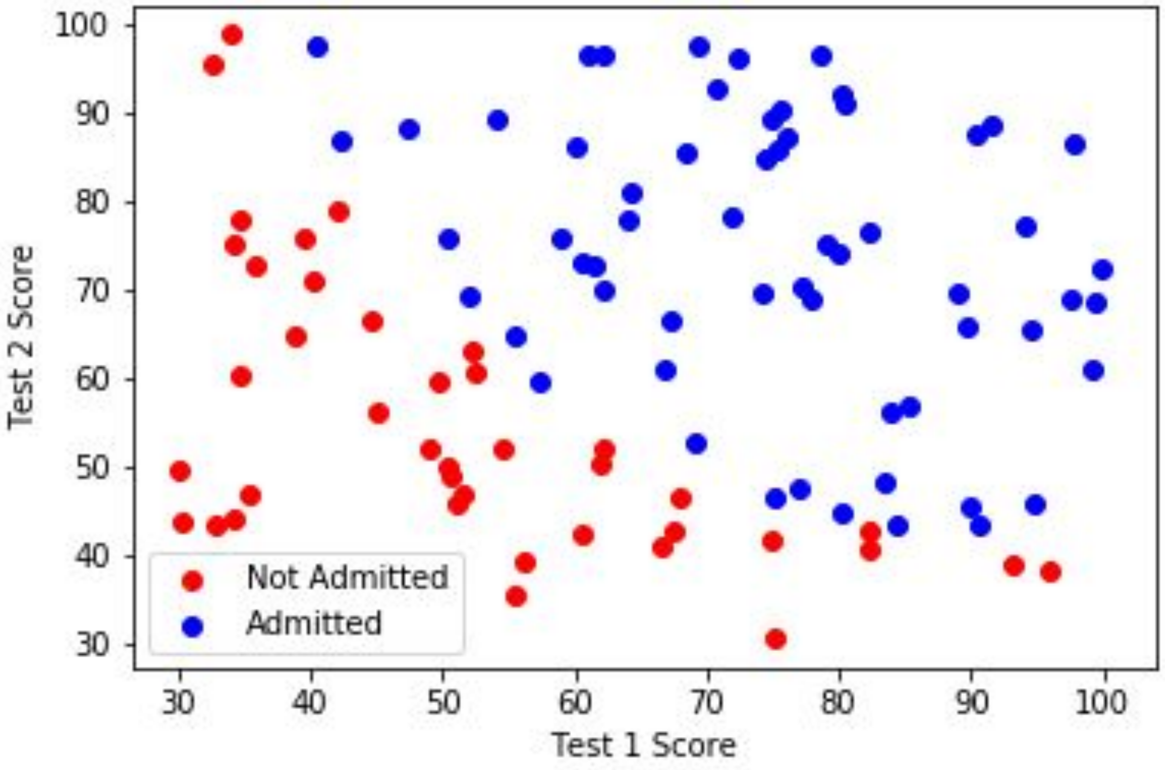
https://medium.com/machine-learning-with-pratik/introduction-to-ai-and-machine-learning-ml001-e-c4a994c6c612

Supervised Learning – Regression Problem

Continuous Valued Target

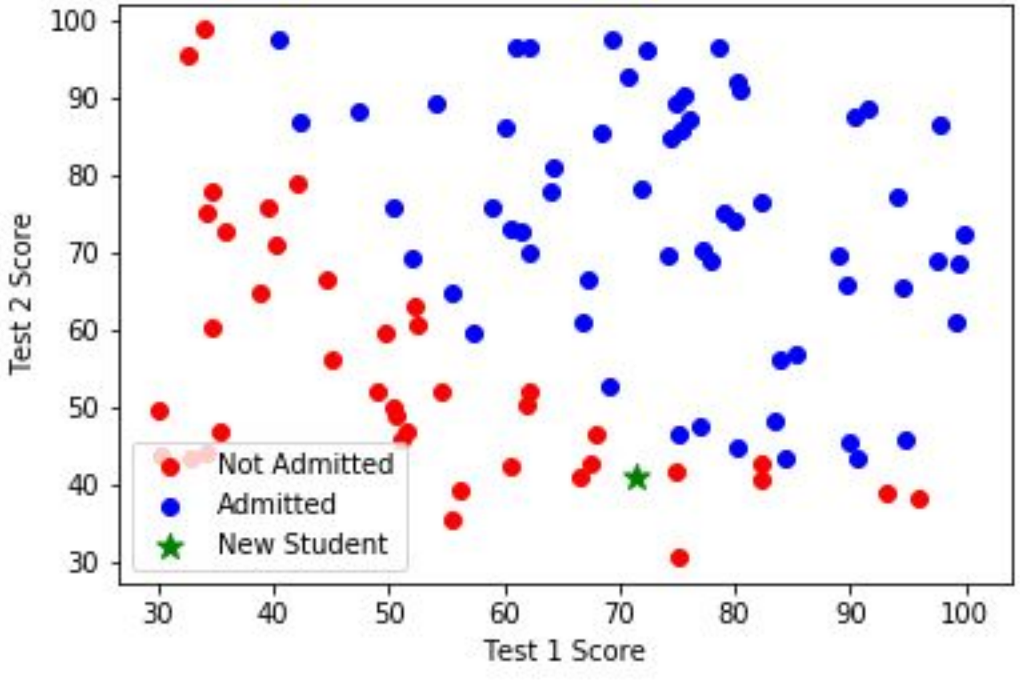


Predicting a Student’s Admission



Historical Data OR Training Data

Predicting a Student’s Admission

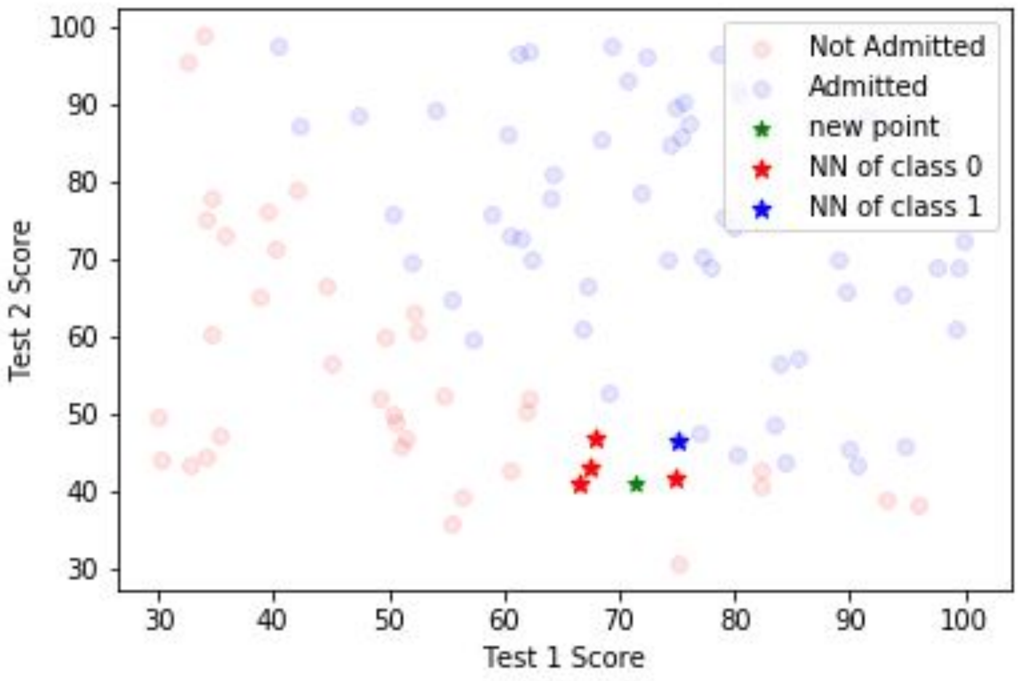


Binary Classification Task:

Predict whether a **new** student will secure admission.

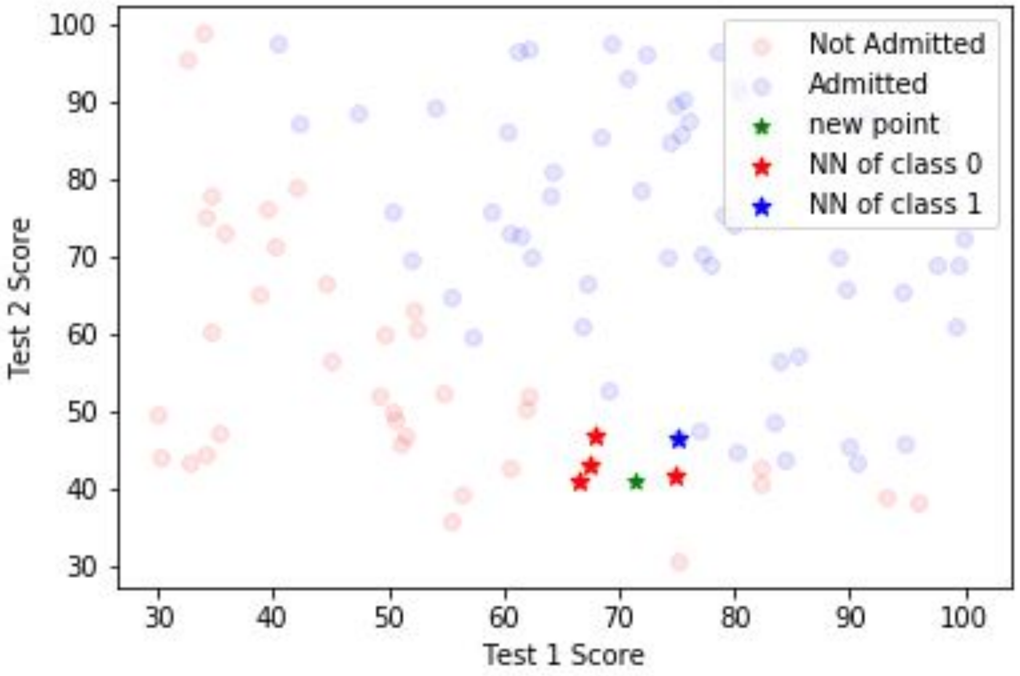
New student will not be present in historical data

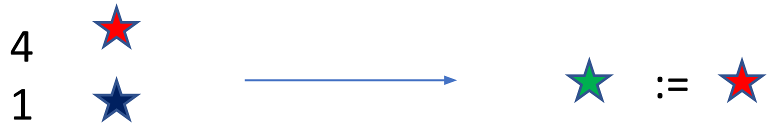
k-Nearest Neighbours (kNN)



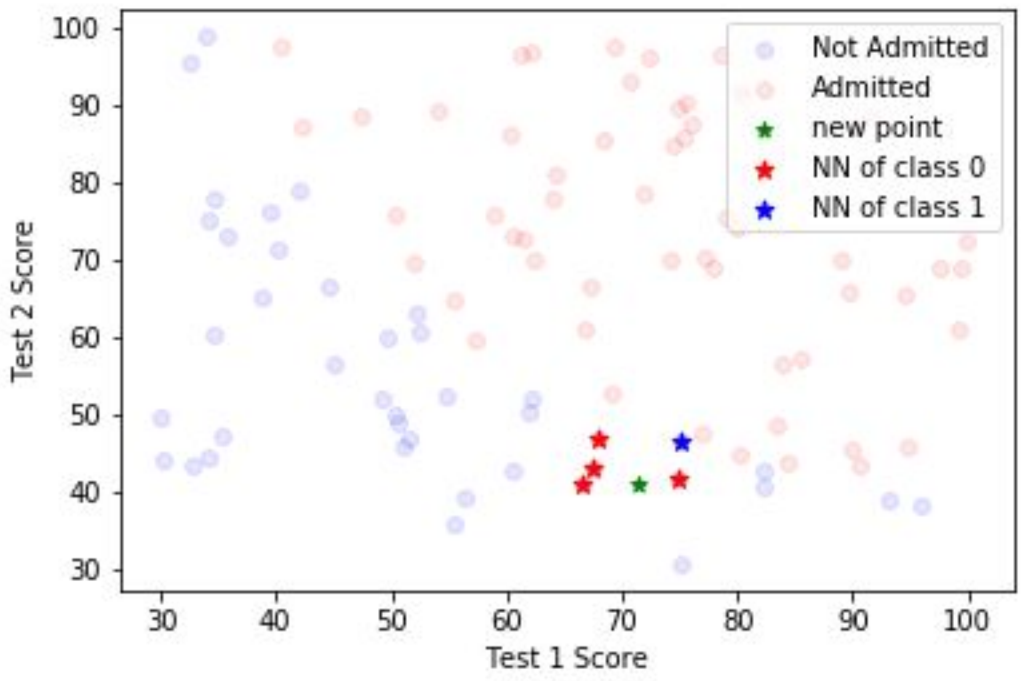
Indicate the closest points in historical data to the new point

kNN Classification – Majority Voting





kNN Classification – Probability Output



4

1

K = 5

Let $\hat{y}$**=1** be the event **"Student is Admitted"**

$$Prob(\hat{y} = 1 \mid x) = \frac{n_1}{K} = \frac{1}{5} = 0.2$$

Where n_1 is the number of nearest neighbors of class 1

kNN – The Math

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

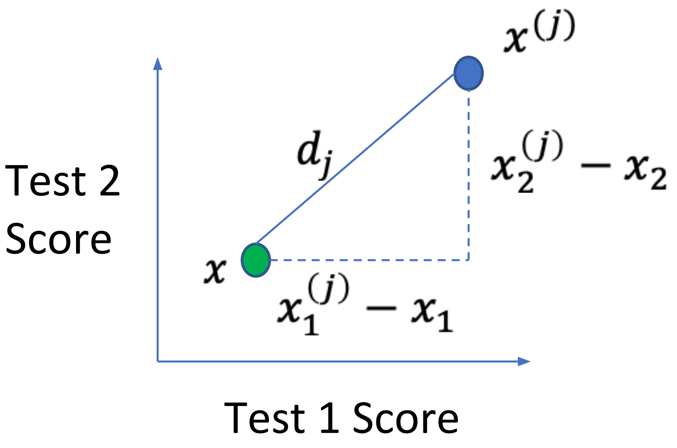
x is a new observation column vector

$$X = \begin{bmatrix} {x^{(1)}}^T \\ {x^{(2)}}^T \\ {x^{(M)}}^T \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ x_1^{(M)} & x_2^{(M)} \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(M)} \end{bmatrix}$$

Design Matrix

Where,

x_1 = Test 1 Score
x_2 = Test 2 Score



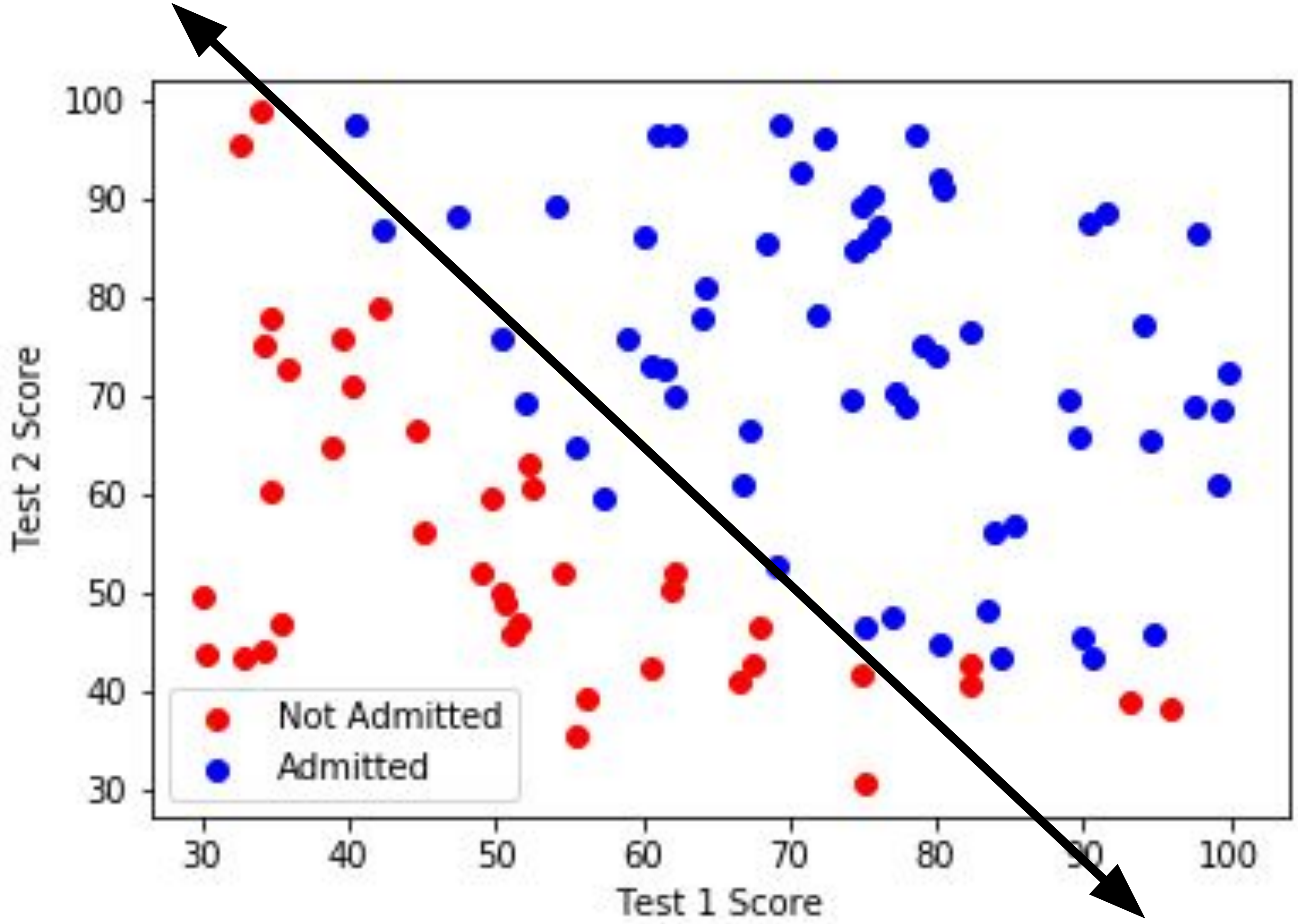
$$d_j = \sqrt{\left(x_1^{(j)} - x_1\right)^2 + \left(x_2^{(j)} - x_2\right)^2}$$

From Pythagoras' Theorem

$$NN(x, X) = \arg \min_{j \in \{1,M\}} d\left(x, x^{(j)}\right)$$

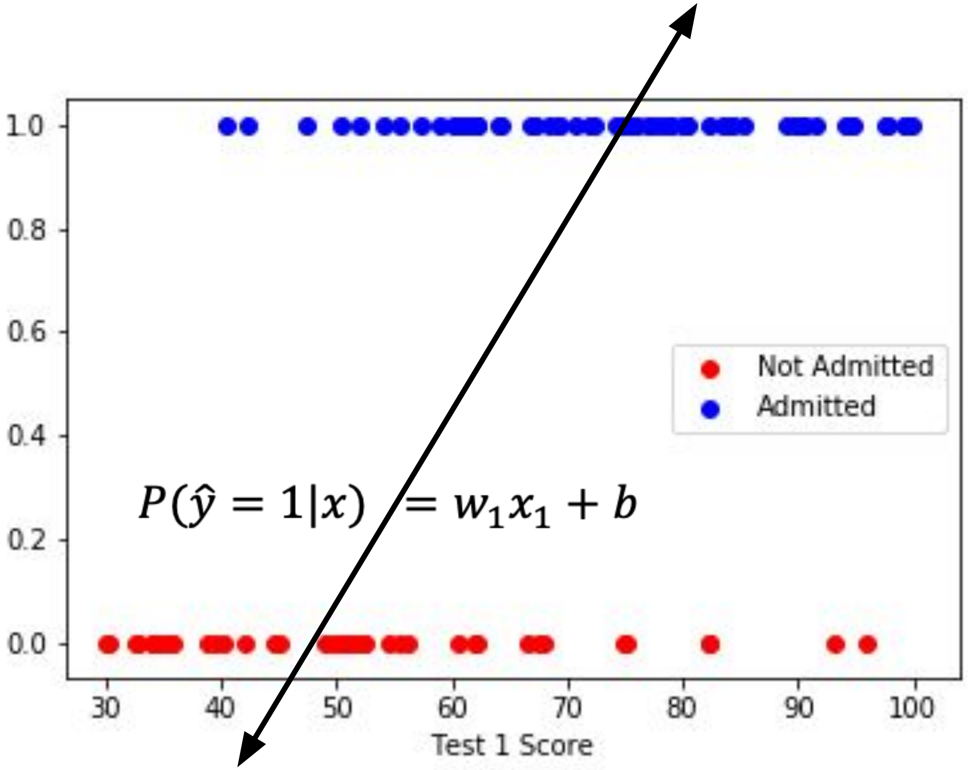
QnA & Code Walkthrough - kNN

Logistic Regression



A model which learns a linear boundary between the classes

First, a linear probability model



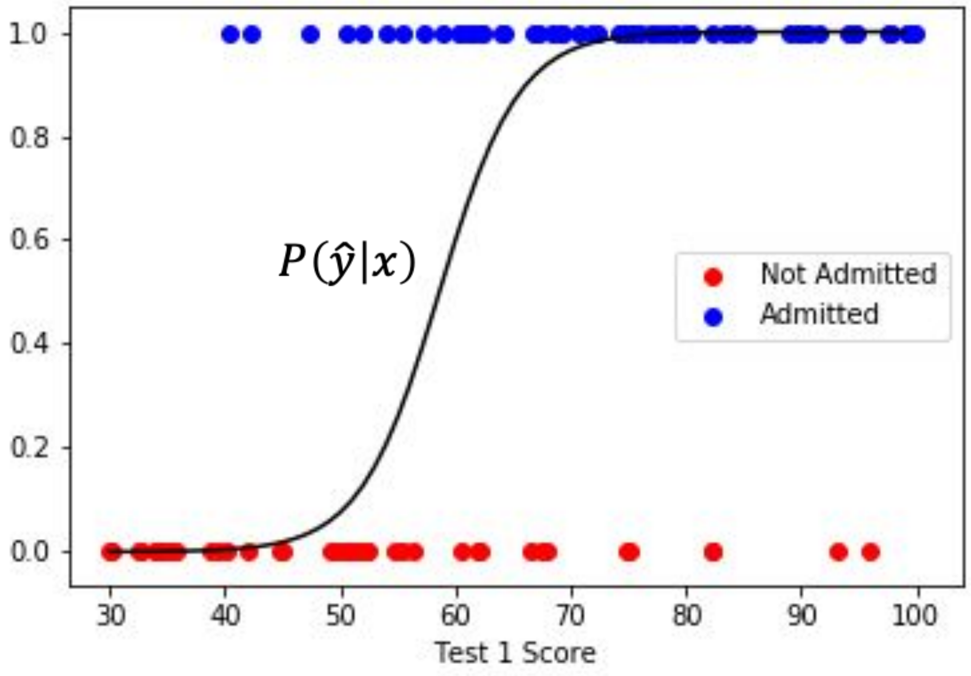
$$P(\hat{y} = 1|x) = w_1 x_1 + b$$

Estimate the coefficients w_1 and b through Least Squares Regression

$$w = (X^T X + \lambda I)^{-1} X^T Y$$

A Major Problem with this Model:
Non-sensical probabilities (P > 1 or P < 0)

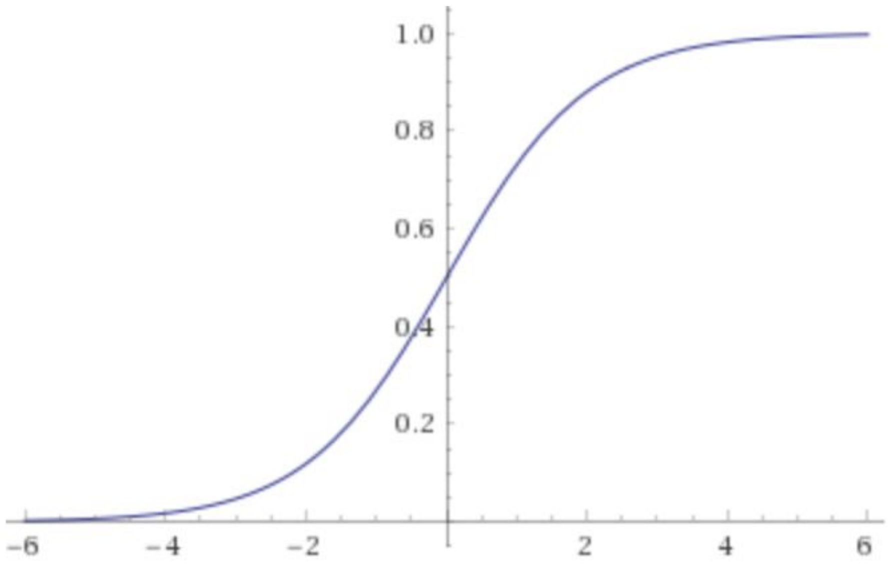
Would be nice, if …



Output bounded between 0 and 1.

Sigmoid/Logistic Function

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

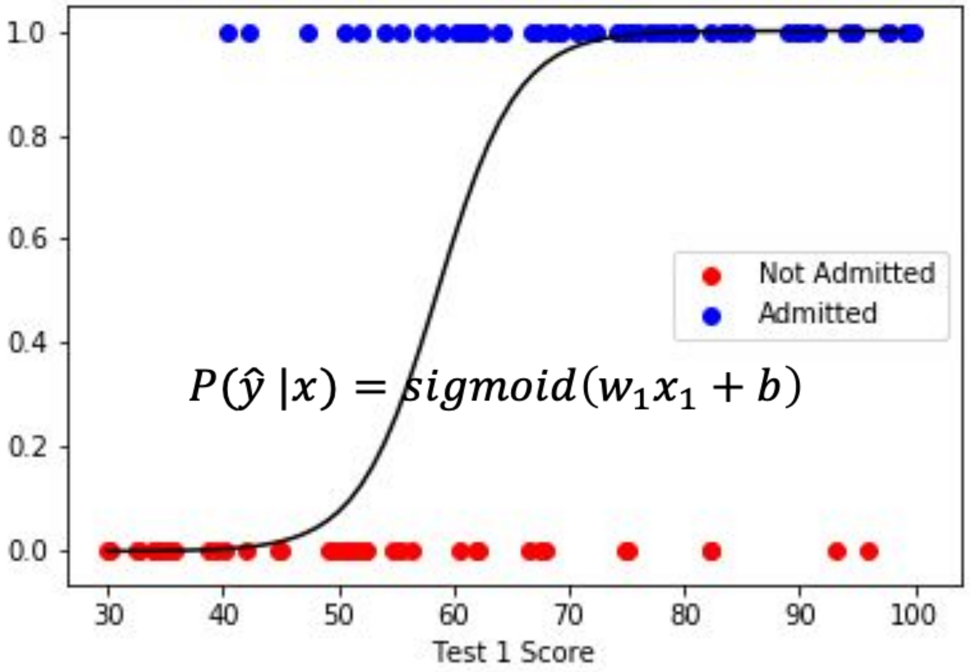


- The sigmoid function squishes numbers to (0,1)

Logistic Regression

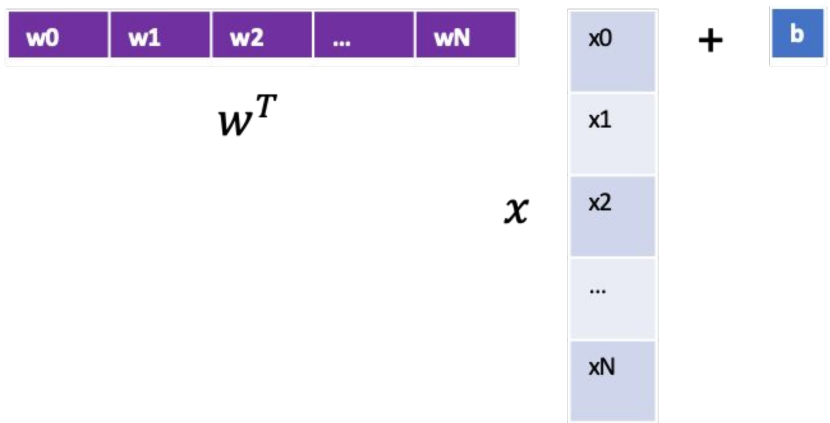
Unknowns

$$P(\hat{y} = 1|x) = sigmoid(w_1 x_1 + b) = \sigma(w_1 x_1 + b)$$
$$= \frac{1}{1 + e^{-(w_1 x_1 + b)}}$$



General Form:

$$P(\hat{y} = 1|x)$$
$$= \sigma(w_1 x_1 + w_2 x_2 + \ \ldots + w_N x_N + b)$$
$$= \sigma(w^T x + b)$$



[Notebook on Github](#)

[Scikit-learn implementation of Logistic Regression](#)

Q & A

Loss Minimization

- Most of machine learning involves some form of loss minimization
- Loss indicates how bad our predictions are.
- Let $\hat{y}$ be our predicted probability of admission of a student x, and y be the true label.
- Then, $L = f(\hat{y}, y)$ is the loss we incur, where $f(.,.)$ is called the loss function

Some Examples of Loss Functions

- **0-1 Loss:** $\hat{y} = 0, y = 1 \Rightarrow \; L = 1$
- **Squared Loss:** $\hat{y} = 0.2, y = 1 \Rightarrow L = (y - \hat{y})^2 = 0.8^2 = 0.64$
- **Log Loss / Binary Cross Entropy Loss:** This is the loss function used in Logistic Regression.

- **Important:** Loss is a function of the weights and not the data.

  - $$\begin{aligned} L &= f(\hat{y}, y) \\ &= f(h(x; w, b), y) \end{aligned}$$

$$h(x; w, b) = \sigma(\, w^T x + b)$$

Average Loss Function

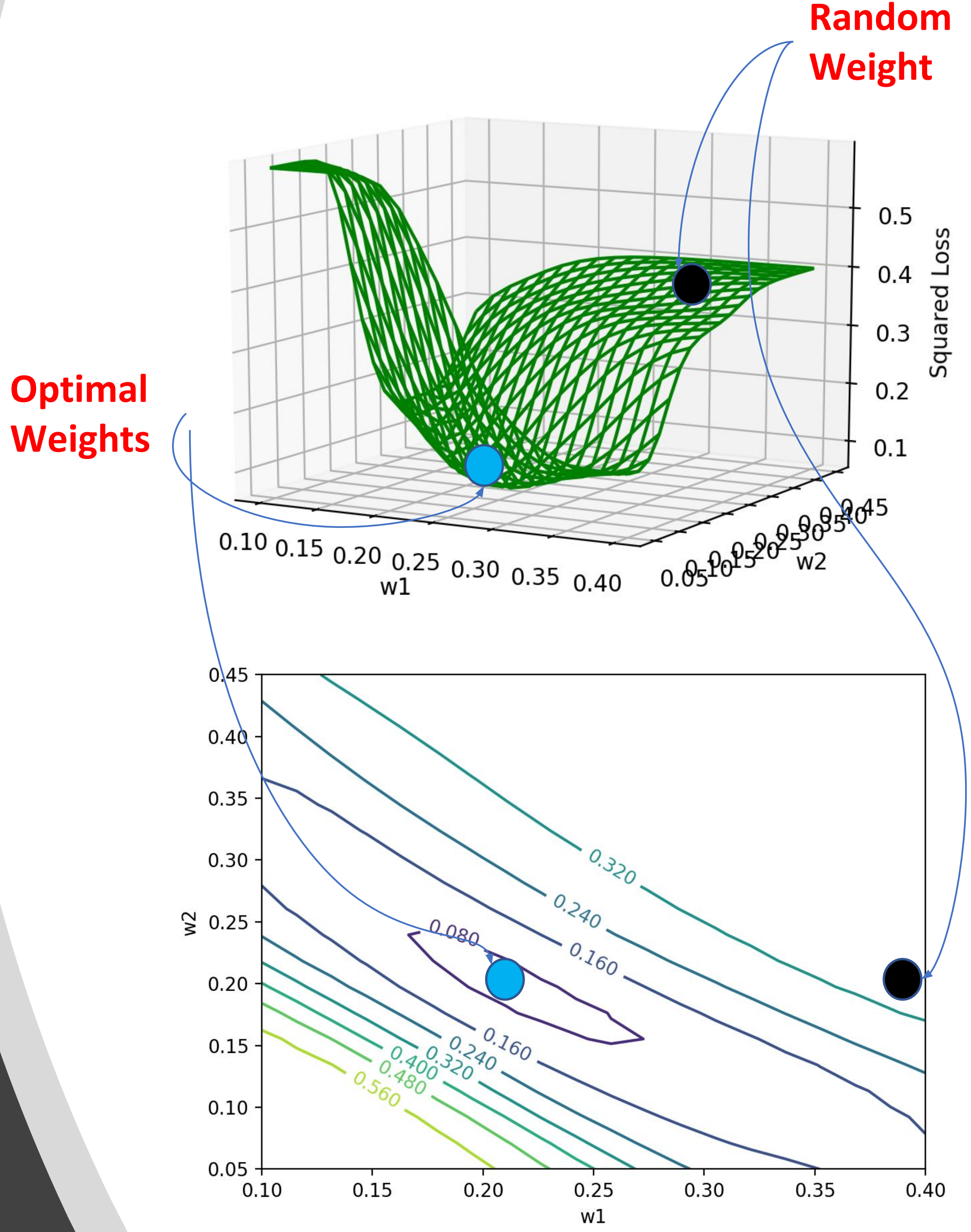
$$L = \frac{1}{M}\sum f(\hat{y_i}, y_i)$$

$$= \frac{1}{M}\sum f(h(x_i; w, b), y_i)$$

- $\hat{y_i} = h(x_i; w, b)$ is the prediction for the ith sample in dataset
- L is the average loss.

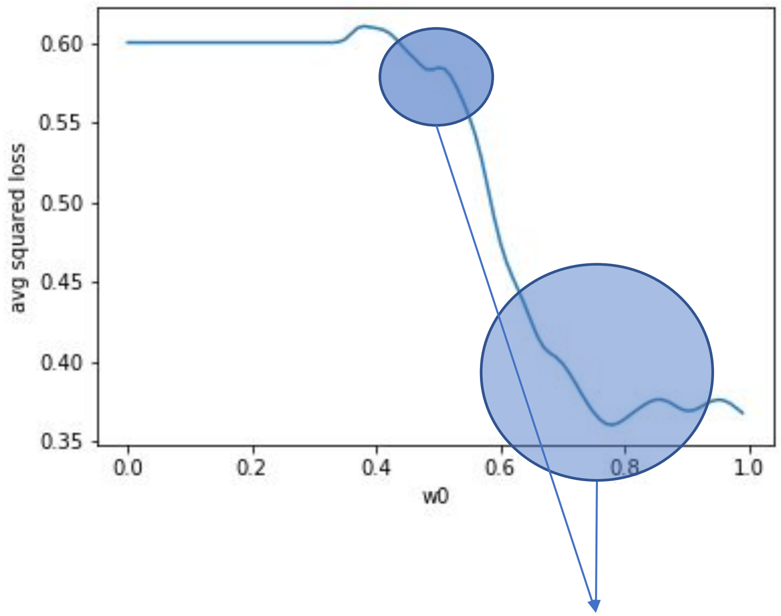
Plotting Loss Functions

- Plot average loss for every possible w = $[w_1, w_2]$
- Plotting this will give us a 3D plot. (top figure)
- Plotting equiloss surface will give us a contour plot (bottom figure)

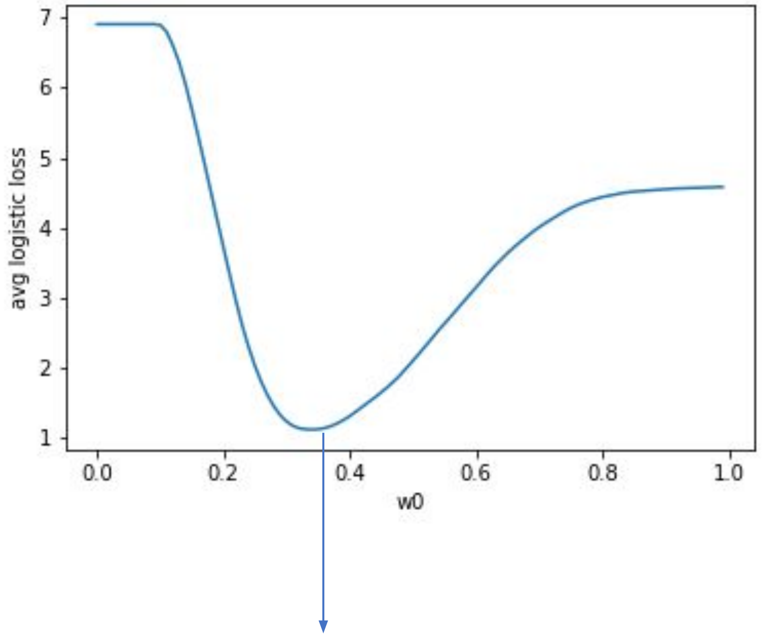


Appropriate Loss Function for Logistic Regression

Squared Loss



Log Loss



Word of Caution: Squared Loss isn't bumpy when used in Linear Regression.

Many Local Minima !

Fixed $w1 = -0.41$
And $b = -21$

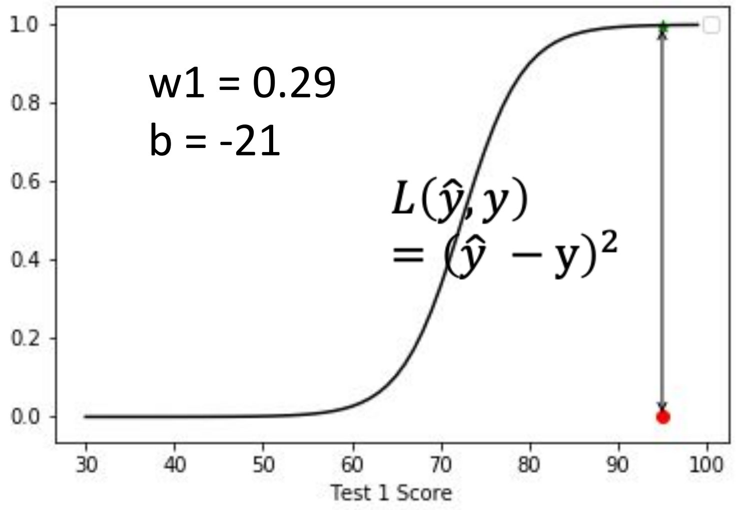
1 Global Minima

A Loss Function for Logistic Regression

Squared Loss

Observe:

The max loss can be only 1

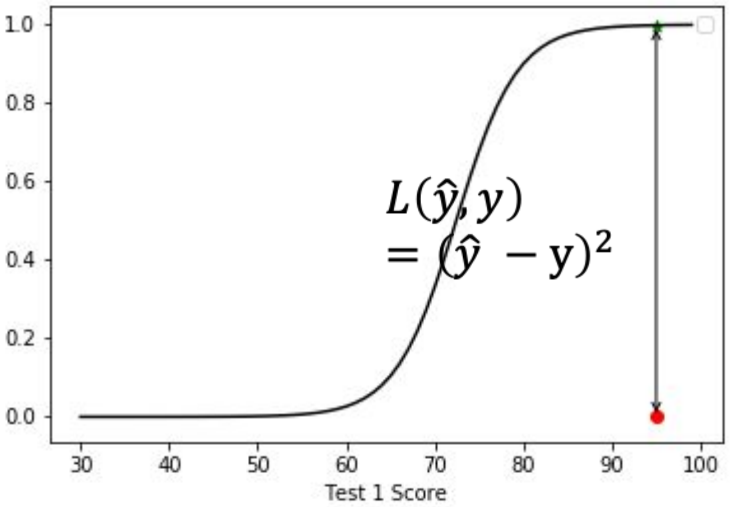


A Loss Function for Logistic Regression

Squared Loss

Observe:

The max loss can be only 1



What if ?

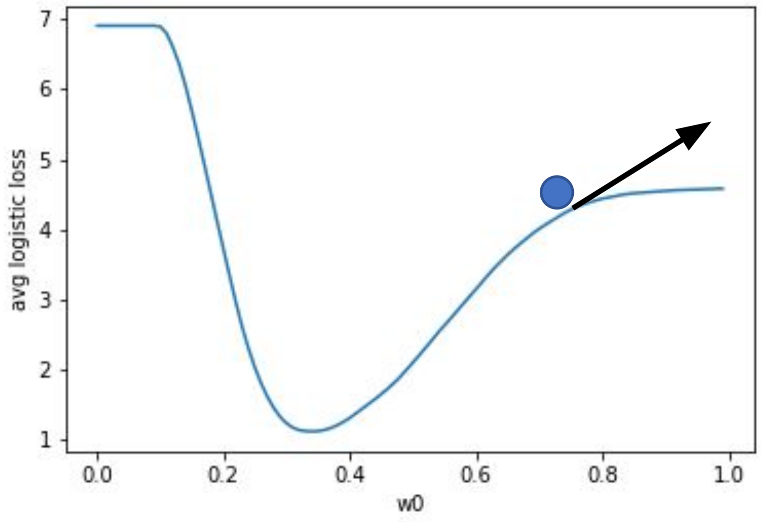
- When $y = 1$ and $\hat{y} = 0$, $f = \infty$
- When $y = 0$ and $\hat{y} = 1$, $f = \infty$

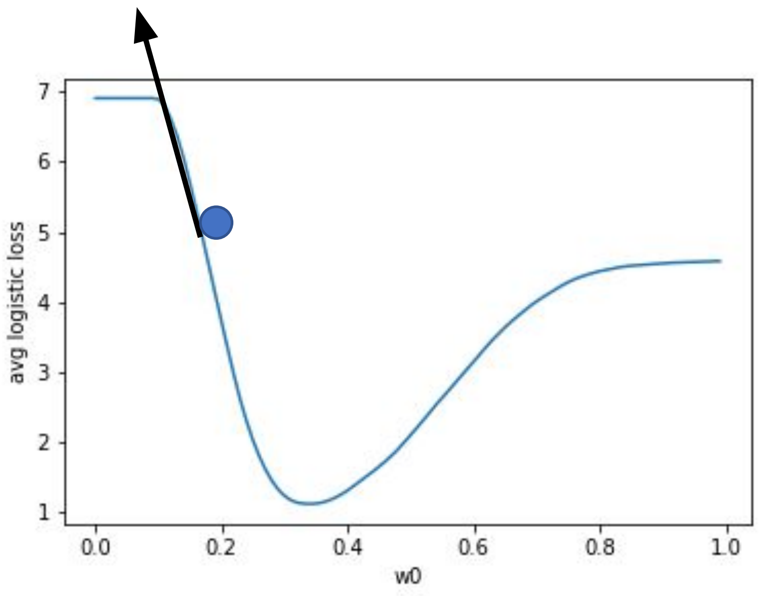
$$f(y, \hat{y}) = \begin{cases} -\ln \hat{y}\,, if\ y = 1 \\ -\ln(1 - \hat{y}), if\ y = 0 \end{cases}$$

$$f(y, \hat{y}) = -y \ln \hat{y} - (1 - y) \ln(1 - \hat{y})$$

Cross Entropy Loss

Gradient

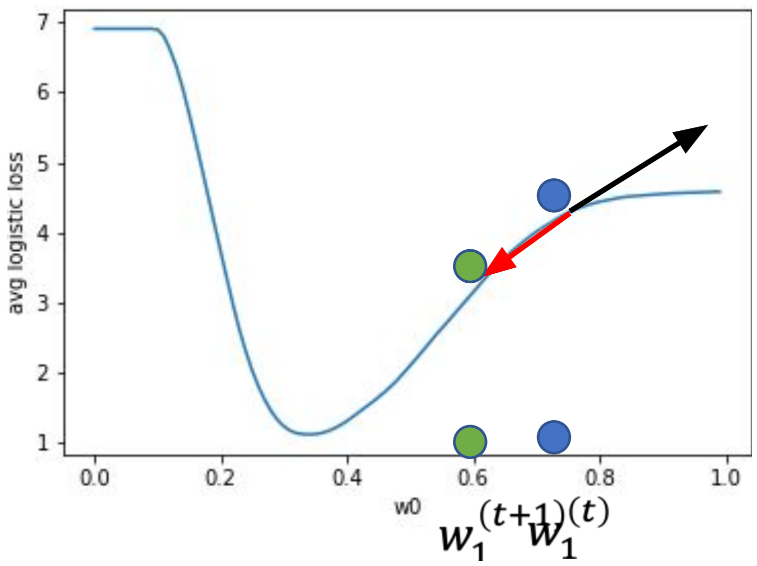


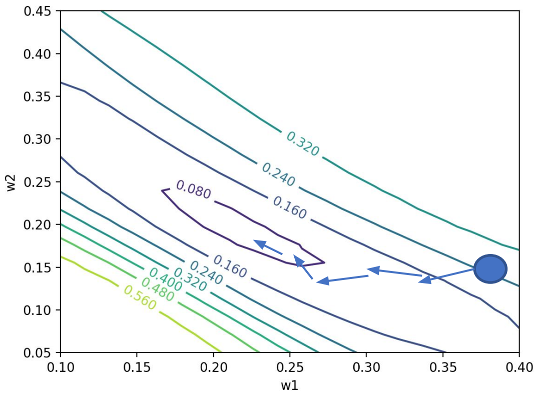


$$Tangent = \frac{\partial L}{\partial w_1}$$

Black arrow magnitude and direction of tangent (gradient)

Update Mechanism - Gradient Descent (Roll down the hill)





- Compute the gradient (tangent) at the current parameter values
- Step in the opposite direction

$$w_i^{(t+1)} := w_i^{(t)} - \eta_t \frac{\partial L}{\partial w_i^{(t)}}$$

Updated coeff

Learning Rate

Gradient

$$w_i^{(t+1)} = w_i^{(t)} - \eta \frac{\partial L}{\partial w_i^{(t)}}$$

$$= w_i^{(t)} - \eta \frac{1}{M} \sum (\hat{y}_i - y_i) x_i$$

Putting it all Together

1. **Given**: Dataset $D = \{(x_1, y_1), \ldots, (x_M, y_M)\}$
2. **Initialize:** coefficients w of model randomly
3. $L(w) = \frac{1}{M} \sum_i f(y_i, \hat{y}_i)$
4. For all coefficients w_j:
   1. $g_j = \frac{1}{M} \frac{\sum \partial f(y_i, \hat{y}_i)}{\partial w_j} = \frac{1}{M} \sum (y_i - \hat{y}_i) x_j$
5. For all coefficients:
   1. $w_j = w_j - \eta g_j$
6. Repeat 3-5 till change in loss is negligible

1 Pass / Epoch

Updates have to be simultaneous

Computing gradients over the full dataset might be expensive.

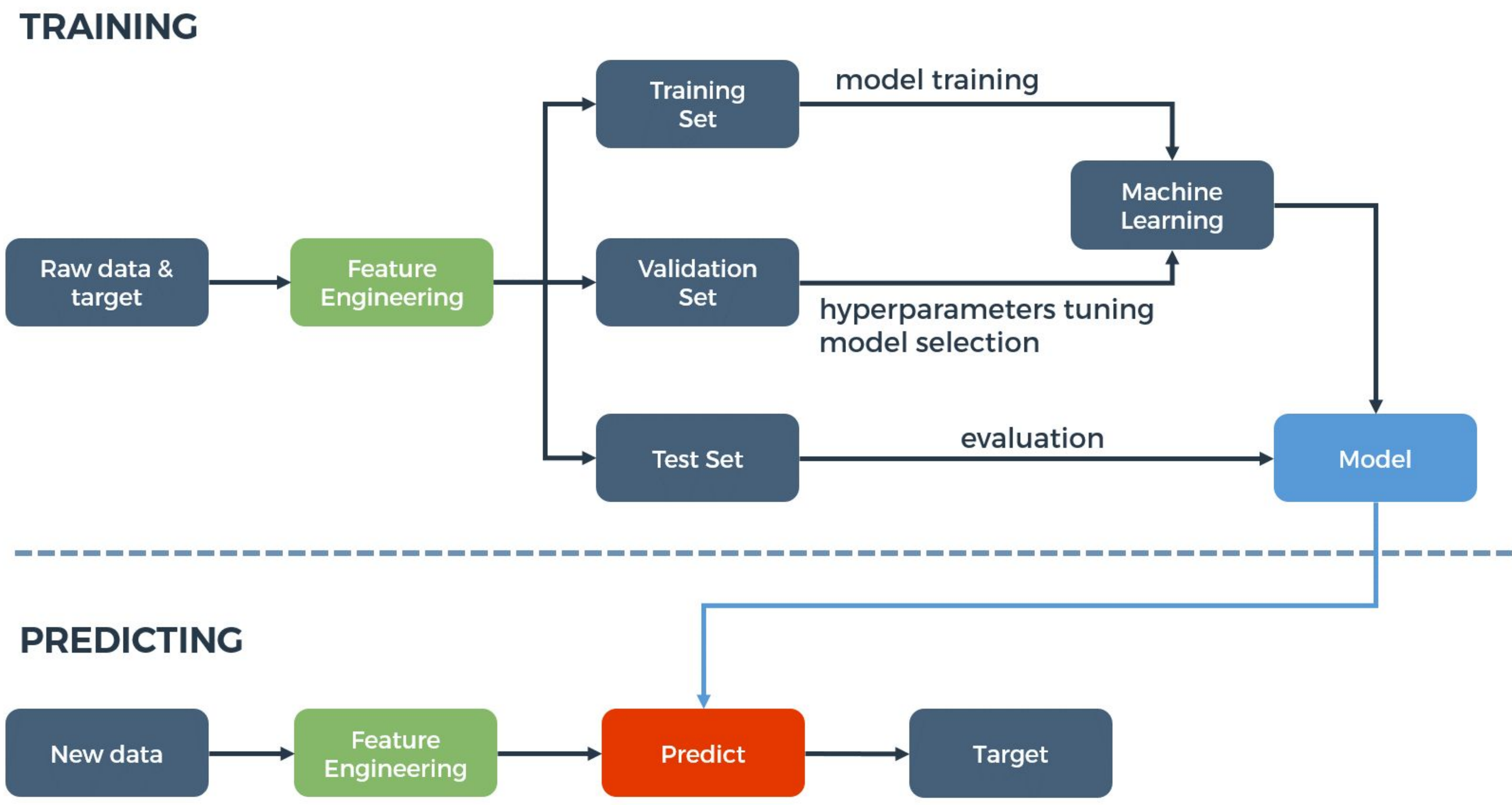
- Compute over mini batches of data instead (mini-batch gradient descent)

Scikit-learn has an efficient Logistic Regression implementation:

Choose a learning rate just high enough so that training doesn't diverge (i.e. losses don't increase with steps)

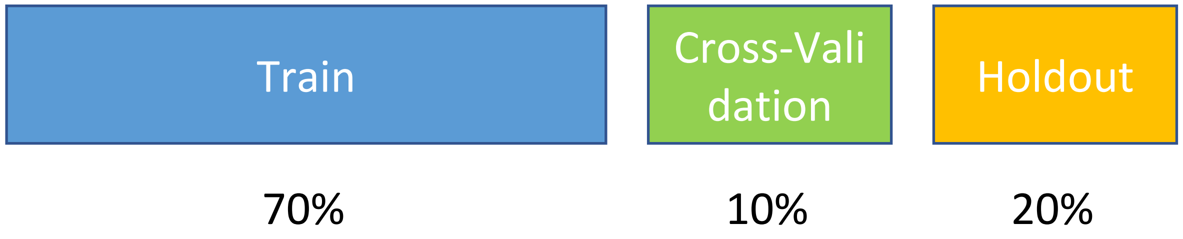
Q & A

Data Science Pipeline


TRAINING
Raw data & target
Feature Engineering
Training Set
Validation Set
Test Set
model training
Machine Learning
hyperparameters tuning
model selection
evaluation
Model
PREDICTING
New data
Feature Engineering
Predict
Target

Train – Test Splits

For Large Data:



Tune your model after observing results on this

Report results/evaluation metrics on this

For small data, look at K-Fold splits.

Split Strategies:

1) Random (70-10-20) split

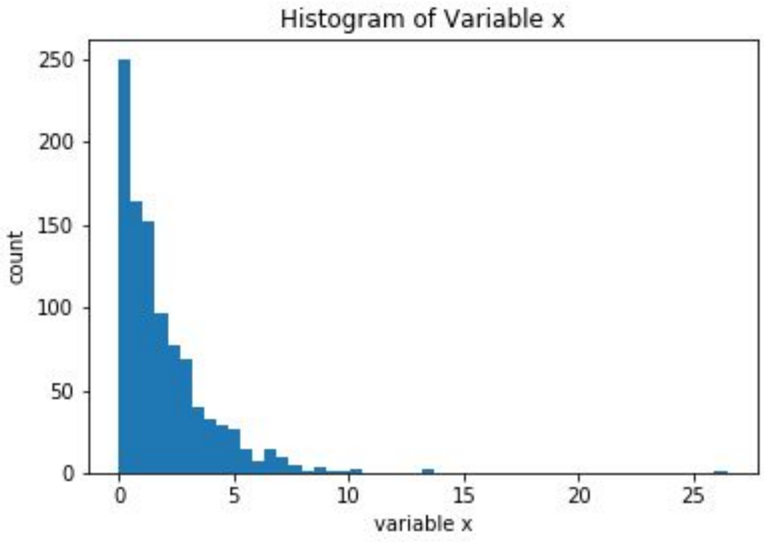
2) Out of time cross validation and holdout

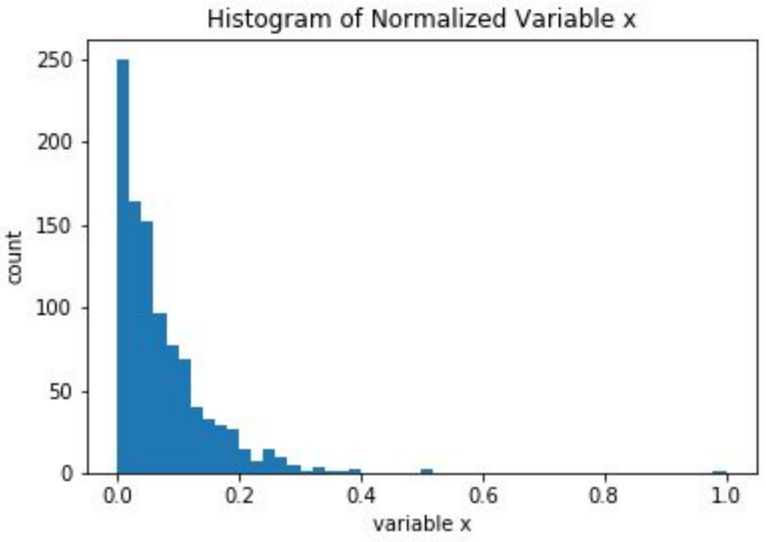
Scikit-learn has implementations of
- Train-Test Split
- K-Fold

Feature Pre-Processing for Continuous variables

- Scale your features to small values around 0
  - Min-Max Scaler $x := \frac{x - x_{mn}}{x_{mx} - x_{mn}}$ [Scikit Learn MinMaxScaler]

Scaling your features will help the gradient descent converge faster

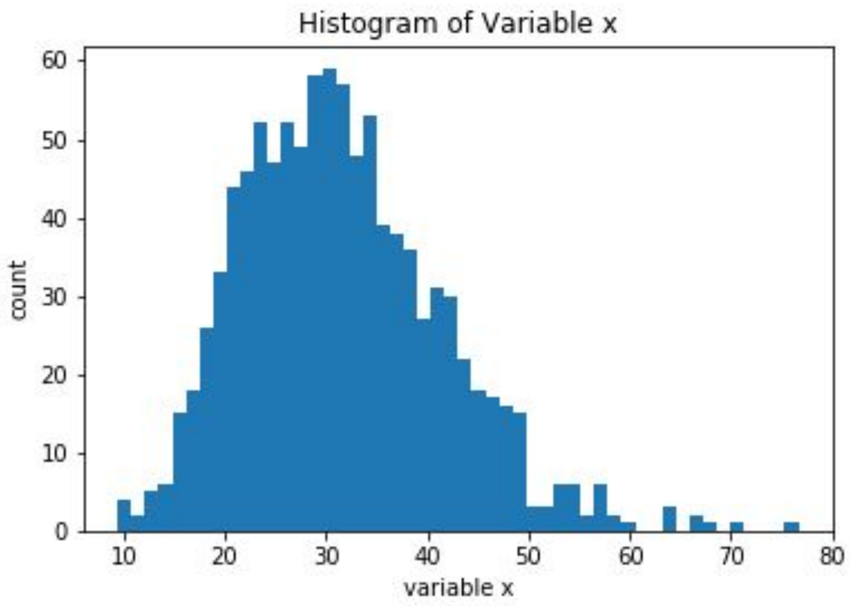


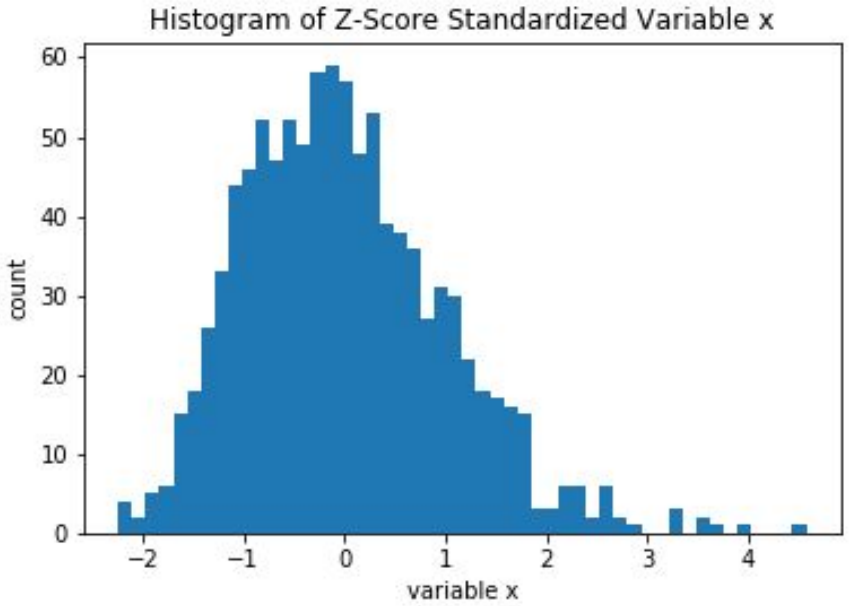


Feature Pre-Processing for Continuous variables

- Scale your features to small values around 0
  - Z-Score $x := \frac{x-\mu}{\sigma}$ [Scikit Learn Standard Scaler]

Scaling your features will help the gradient descent converge faster





Feature Pre-Processing for Discrete/Categorical Variables

A categorical variable can take K discrete values with no notion of ordering or rank between them.

Device Type:

Mobile
Laptop
Tablet

1-hot encode

1	0	0
0	1	0
0	0	1
Mobile	Laptop	Tablet

Scikit-learn

- One Hot Encoder

For a variable with large K, look at other techniques:

- Hashing Trick
- Target Statistics

Model Evaluation Measure - Accuracy

S.No	Predicted Label	Ground Truth Label
1	1	1
2	0	0
3	0	1
4	1	0
5	0	0
6	0	0

$$Accuracy = \frac{N_{correct}}{N} = \frac{4}{6} = 66.67\%$$

Imagine a scenario where 99% of the ground truth labels are 0s

A classifier which labels every example as a 0, will also have 99% accuracy !

Using Accuracy for imbalanced classes will be misleading !!

Model Evaluation Measures

	Actual Positive	**Actual Negative**
Predicted Positive	tp (True Positives)	fp (False Positives)
Predicted Negative	fn (False Negatives)	tn (True Negatives)

Confusion Matrix

$$\text{Accuracy} = \frac{tp + tn}{tp + fp + fn + tn}$$

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

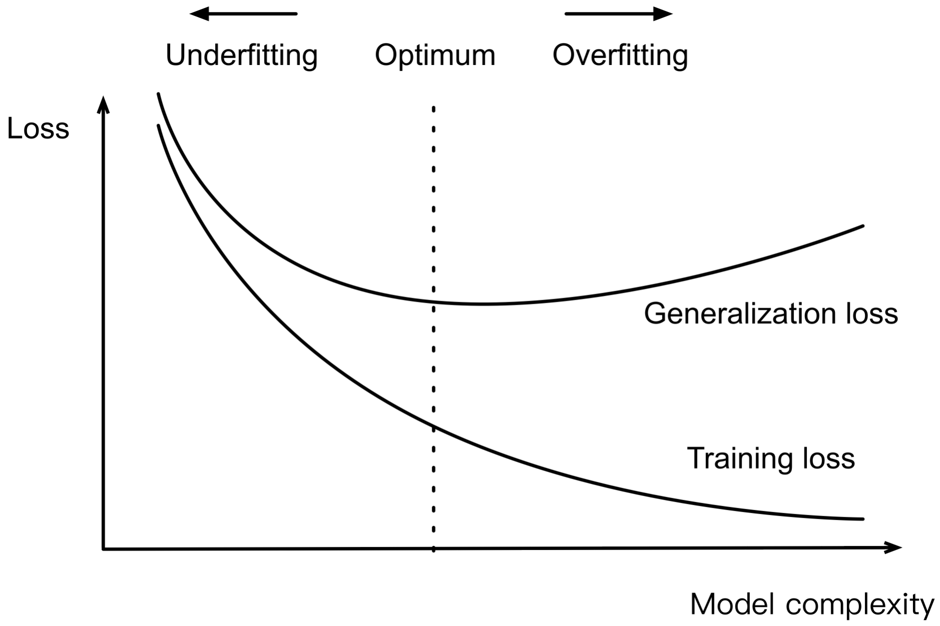
$$Precision = \frac{tp}{tp + fp}$$

$$F1 = 2\,\frac{Recall * Precision}{Recall + Precision}$$

In the case of Ad-Click Prediction:

- If optimizing for reach, then tune for recall
- If optimizing for ad dollars spent, then tune for precision

Overfitting and Underfitting



$$P(\hat{y} = 1|x) = \sigma(w_1 * x_1 + w_2 * x_2 + b)$$

A more Complex Model

$$P(\hat{y} = 1|x)$$
$$= \sigma(w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2 + b)$$

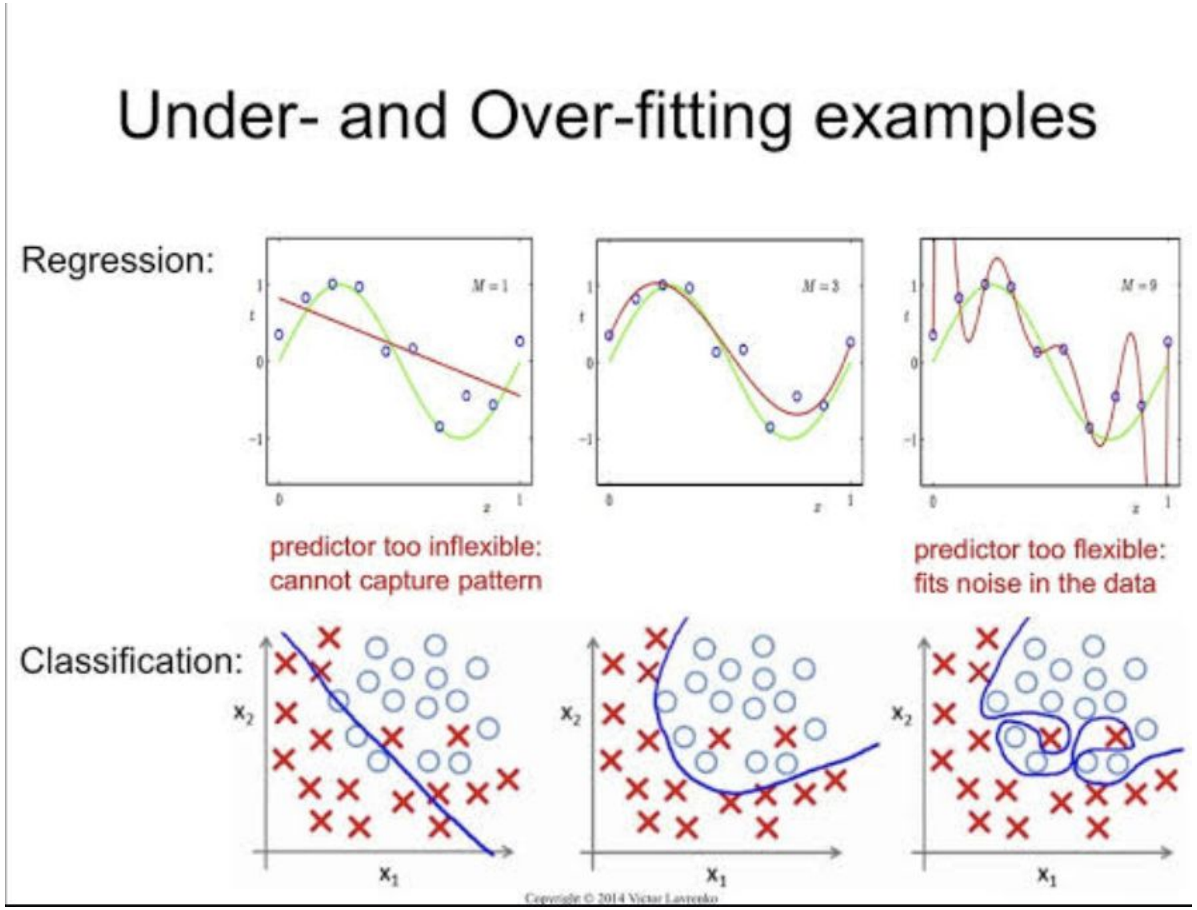
A simpler model

$$P(\hat{y} = 1|x) = \sigma(w_1 x_1 + b)$$

A highly simplistic model

$$P(\hat{y} = 1|x) = \sigma(b)$$

Model Complexity



Underfit Models:

- Model not complex enough to capture the underlying distribution of the data.

Overfit Models:

- Model too complex and tries to capture every bit of information in the dataset.
- Such models do not generalize well to unseen data.

Model Complexity in the case of Logistic Regression could be due to:

- Large number of parameters.

Fixing Underfitting

- Add more features to your Logistic Regression Model

- Try using a more complex model, such as :
  - Decision Trees
  - Neural Nets

Fixing Overfitting

- Collect more data
- Then reduce model complexity
  - Try regularization
  - Then try a simpler model

QnA & Code Walkthrough

Scikit Learn Cheat Sheets

- https://scikit-learn.org/stable/tutorial/machine_learning_map/
- https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Scikit_Learn_Cheat_Sheet_Python.pdf
- https://bit.ly/2Kwg36X
- https://towardsdatascience.com/resources-to-start-your-journey-in-data-science-bf960a8d928c

Thank you

Supervised Machine Learning