Getting Started with Data Science

Overview and Advice

Background

What is Data Science

What is Data Engineering

What is Machine Learning

What is Feature Engineering

What is Deep Learning

Data science is an inter-disciplinary field that uses **scientific** methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured **data**.

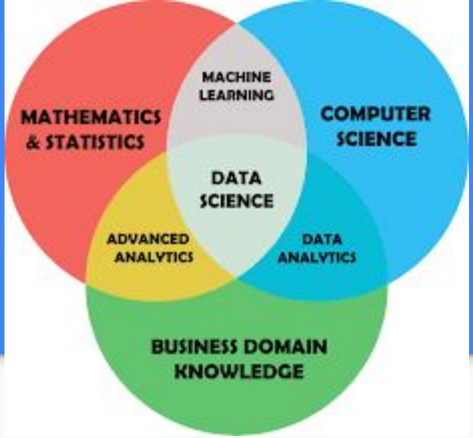
Data engineering is the aspect of **data** science that focuses on practical applications of **data** collection and analysis.

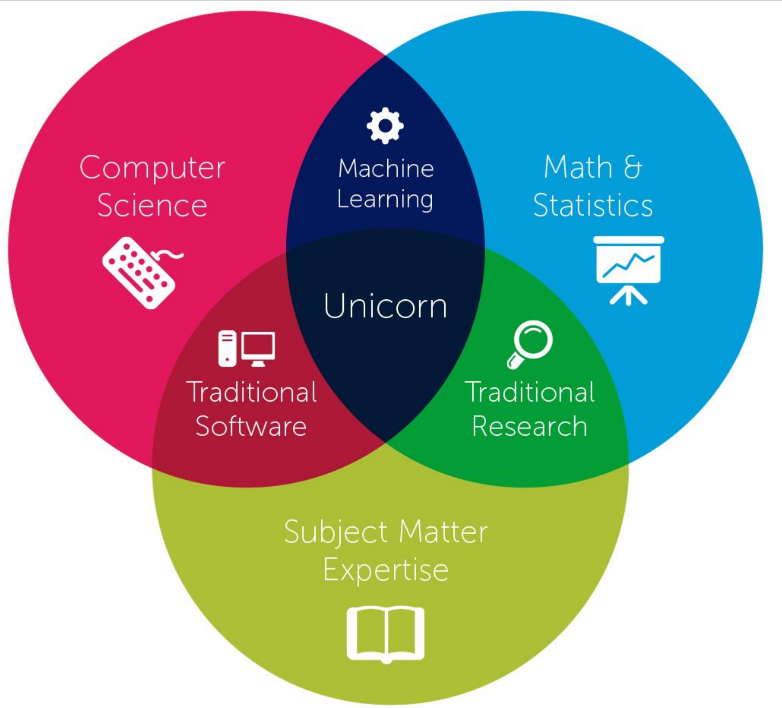
Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead.

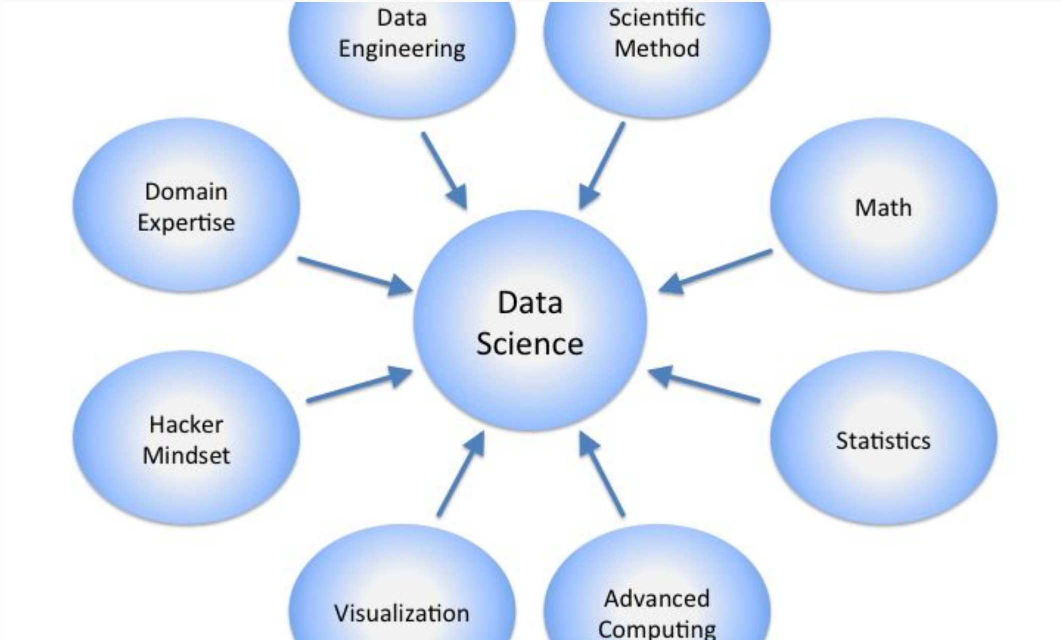
Feature engineering is the process of using domain knowledge to extract **features** from raw data via data mining techniques.

Deep learning is a subset of machine learning that processes data and creates patterns for use in decision making.

Data Science is







Data Engineering

Data engineers build pipelines that prepare and transform data for data scientists.

Data engineers prepare the data for use by data scientists.

Data engineers are aware of how data is used and have the goal of facilitating the data scientist and others.

Machine Learning

… perform a specific task without using explicit instructions, relying on patterns and inference instead.

Types of Learning:

SUPERVISED
Given features and known class or known value, build a model that predicts what the class or the value should be.

UNSUPERVISED
Given data, how does it relate or does it differ?

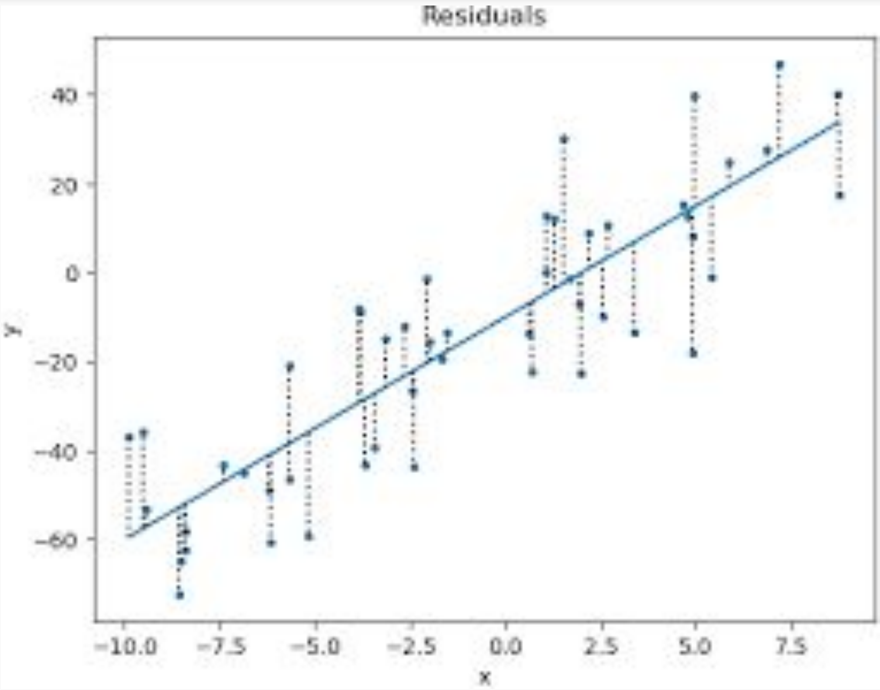
Two common tasks for Machine Learning:

CLASSIFICATION
What kind of thing is this? (e.g., Junk Mail vs Good Mail)

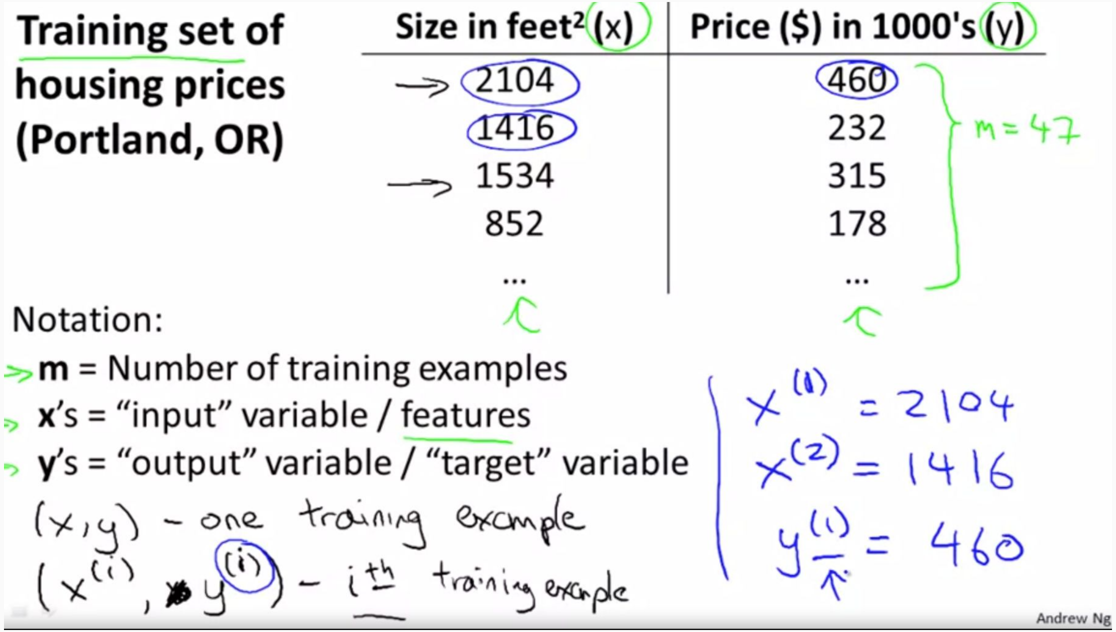
REGRESSION
Based on features, what will this value be?
(e.g., rooms in a house, sq footage, how much is my house going to be worth?)

Linear Regression

- Adjust Coefficients of every feature to alter the slope of the line towards minimizing the collective error across the samples.
- Use Gradient Descent to accomplish this optimization
- Articles that explain in more details:
  - https://towardsdatascience.com/gradient-descent-from-scratch-e8b75fa986cc
  - https://towardsdatascience.com/gradient-descent-explanation-implementation-c74005ff7dd1



Linear Regression (cont)



Training set of housing prices (Portland, OR)

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

m = 47

Notation:

- m = Number of training examples
- x's = "input" variable / features
- y's = "output" variable / "target" variable

(x, y) - one training example

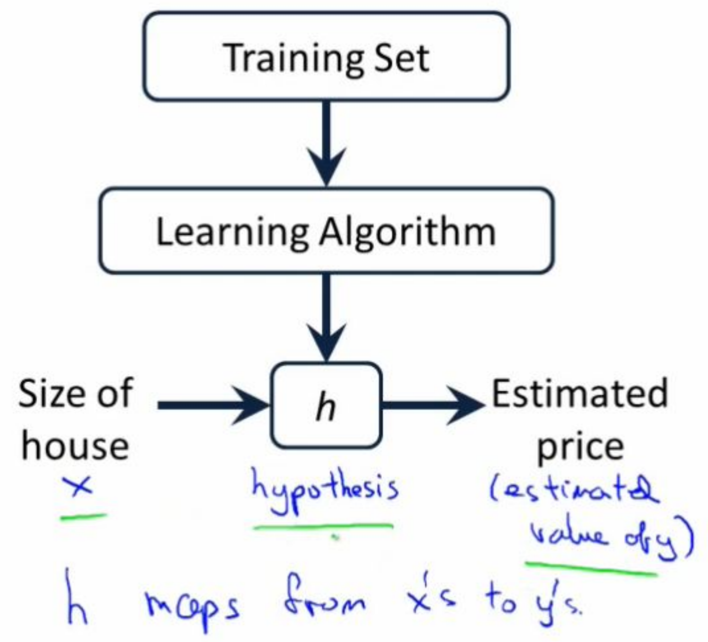
$(x^{(i)}, y^{(i)})$ - i^{th} training example

$x^{(1)} = 2104$

$x^{(2)} = 1416$

$y^{(1)} = 460$

Andrew Ng



Linear Regression (cont)

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Goal:

$$\underset{\theta_0, \theta_1}{minimize} J(\theta_0, \theta_1)$$

m	The number of training examples
$x^{(i)}$	The input vector for the i^{th} training example
$y^{(i)}$	The class label for the i^{th} training example
Θ	The chosen parameter values or "weights" (Θ_0, Θ_1, Θ_2)
$h_\theta(x^{(i)})$	The algorithm's prediction for the i^{th} training example using the parameters Θ.

Multiple Features

Note: [7:25 - θ^T is a 1 by (n+1) matrix and not an (n+1) by 1 matrix]

Linear regression with multiple variables is also known as "multivariate linear regression".

We now introduce notation for equations where we can have any number of input variables.

$$\begin{aligned} x_j^{(i)} &= \text{value of feature } j \text{ in the } i^{th} \text{ training example} \\ x^{(i)} &= \text{the input (features) of the } i^{th} \text{ training example} \\ m &= \text{the number of training examples} \\ n &= \text{the number of features} \end{aligned}$$

The multivariable form of the hypothesis function accommodating these multiple features is as follows:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

In order to develop intuition about this function, we can think about θ_0 as the basic price of a house, θ_1 as the price per square meter, θ_2 as the price per floor, etc. x_1 will be the number of square meters in the house, x_2 the number of floors, etc.

Using the definition of matrix multiplication, our multivariable hypothesis function can be concisely represented as:

$$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$

This is a vectorization of our hypothesis function for one training example; see the lessons on vectorization to learn more.

Remark: Note that for convenience reasons in this course we assume $x_0^{(i)} = 1 \text{ for } (i \in 1, \dots, m)$. This allows us to do matrix operations with theta and x. Hence making the two vectors 'θ' and $x^{(i)}$ match each other element-wise (that is, have the same number of elements: n+1).]

Gradient Descent For Multiple Variables

Gradient Descent for Multiple Variables

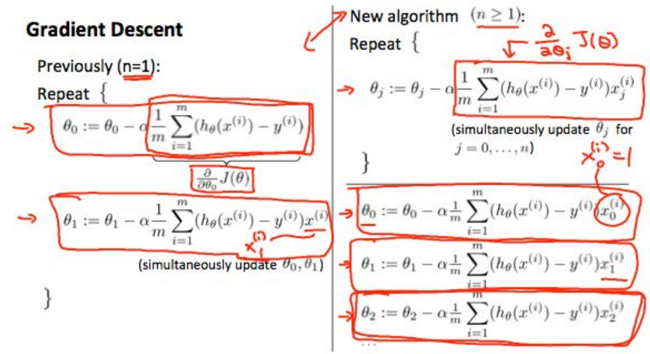
The gradient descent equation itself is generally the same form; we just have to repeat it for our 'n' features:

$$\begin{aligned} & \text{repeat until convergence: } \{ \\ & \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\ & \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \\ & \theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)} \\ & \cdots \\ & \} \end{aligned}$$

In other words:

$$\begin{aligned} & \text{repeat until convergence: } \{ \\ & \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \qquad \text{for j := 0...n} \\ & \} \end{aligned}$$

The following image compares gradient descent with one variable to gradient descent with multiple variables:



Gradient Descent

Previously (n=1):

Repeat {

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

$x_1^{(i)}$

(simultaneously update θ_0, θ_1)

}

New algorithm $(n \geq 1)$:

Repeat {

$\frac{\partial}{\partial \theta_j} J(\theta)$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update θ_j for $j = 0, \ldots, n$)

}

$x_0^{(i)} = 1$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

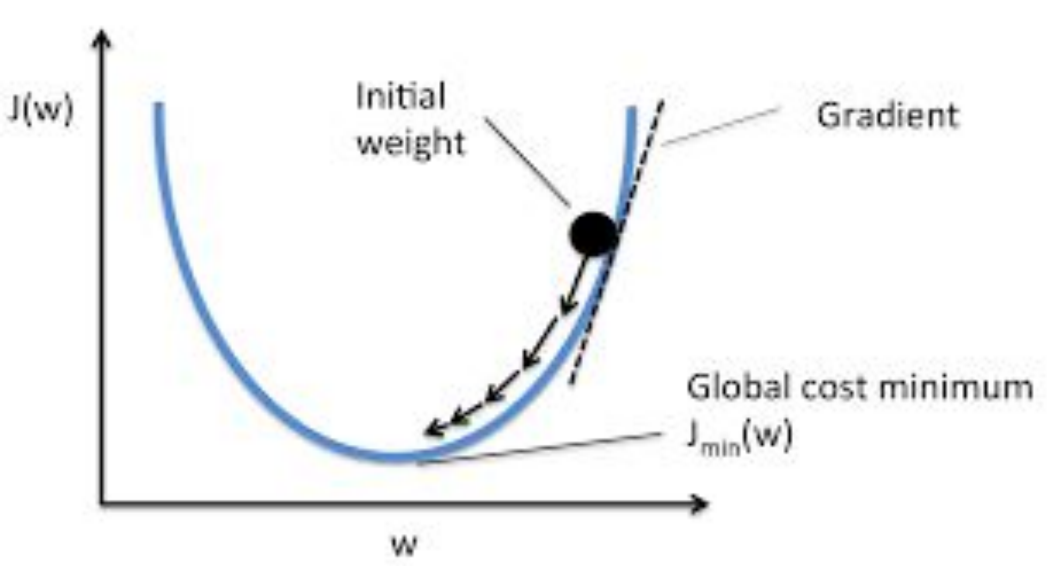
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

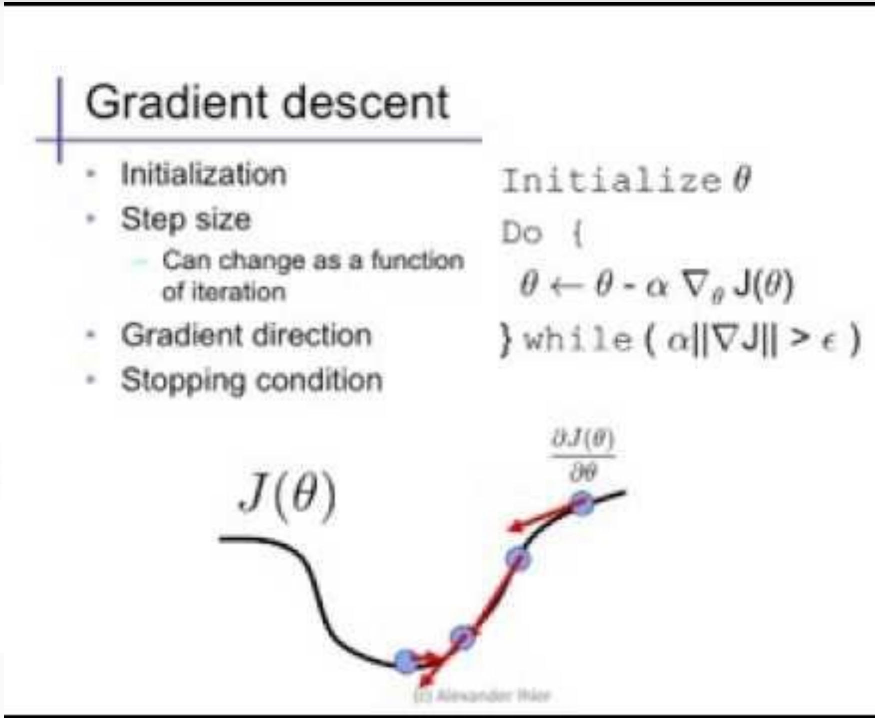
$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

...

Andrew Ng

Minimizing Cost Function





Gradient Descent - regularization (lambda)

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ $\quad x_0 = 1$

Parameters: $\cancel{\theta_0, \theta_1, \ldots, \theta_n}$ θ n+1 - dimensional vector

Cost function:

$$\cancel{J(\theta_0, \theta_1, \ldots, \theta_n)} \; J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

Repeat {

$$\theta_j := \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} \cancel{J(\theta_0, \ldots, \theta_n)} \; J(\theta)}$$

}   (simultaneously update for every $j = 0, \ldots, n$)

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \boxed{\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}} \quad \frac{\partial}{\partial \theta_0} J(\theta)$$

$$\theta_j := \theta_j - \boxed{\alpha} \boxed{\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j}$$

$(j = \cancel{0}, 1, 2, 3, \ldots, n)$

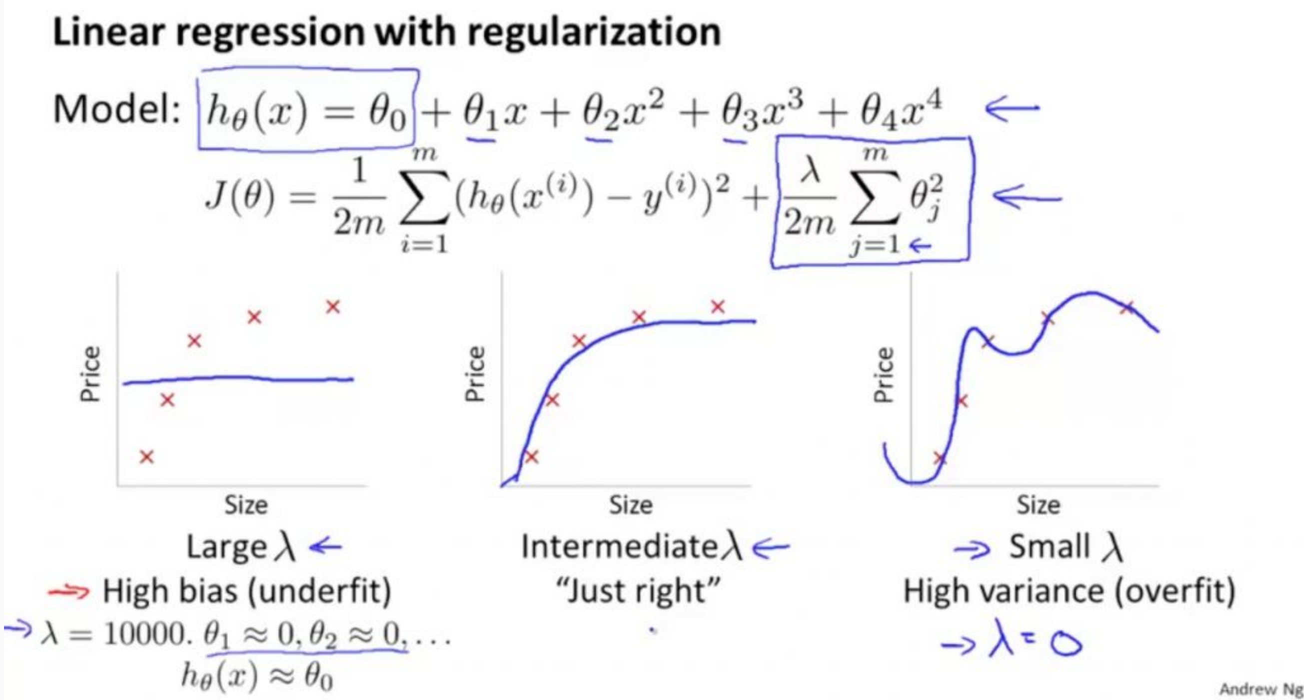
}

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$1 - \alpha \frac{\lambda}{m} < 1$$

Linear Regression Regularization

Large, medium, small Lambda



Linear Regression Regularization Optimization

```
function [jVal, gradient] = costFunction(theta)
    jVal = [ code to compute J(θ)];
```

$$J(\theta) = \left[-\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)}) \log 1 - h_\theta(x^{(i)}) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

```
    gradient(1) = [code to compute ∂/∂θ0 J(θ) ];
```

$$\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

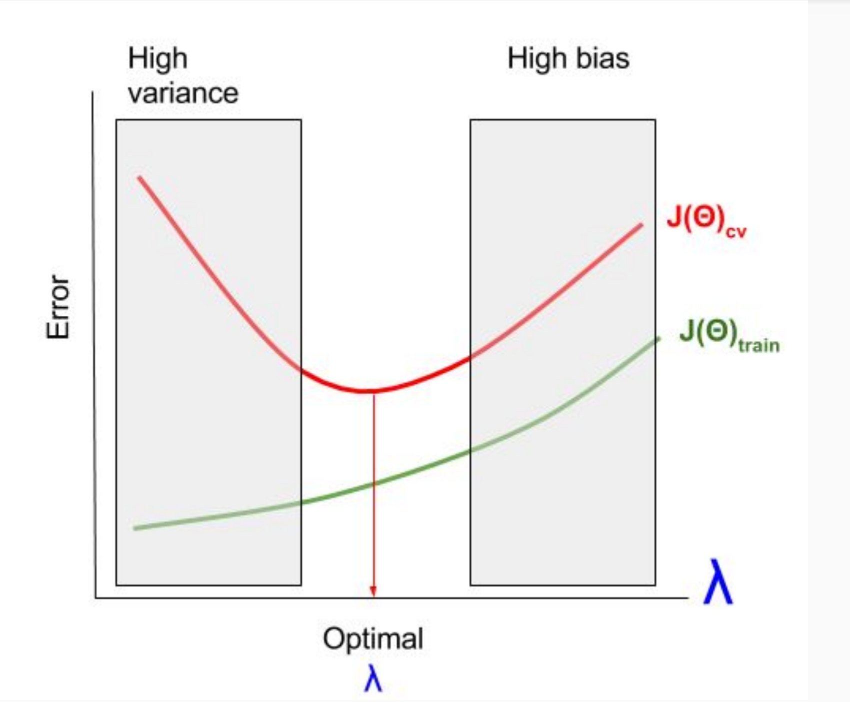
```
    gradient(2) = [code to compute ∂/∂θ1 J(θ) ];
```

$$\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} + \frac{\lambda}{m} \theta_1$$

```
    gradient(3) = [code to compute ∂/∂θ2 J(θ) ];
```

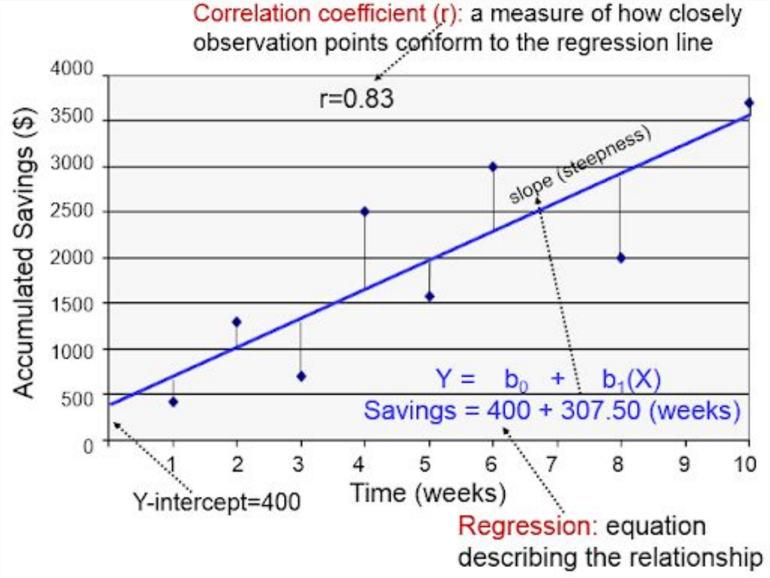
$$\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} + \frac{\lambda}{m} \theta_2$$

```
    :
    gradient(n+1) = [code to compute ∂/∂θn J(θ) ];
```

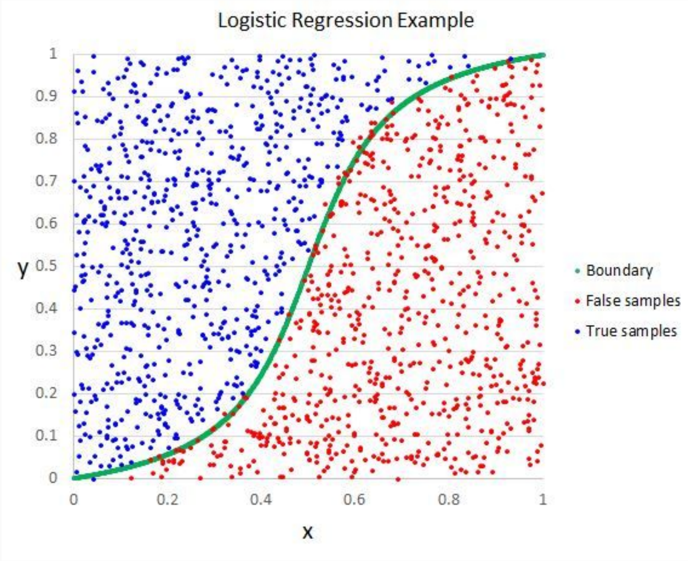


Classic Linear and Logistic Regression Examples

Linear Regression Examples:



Logistic Regression Examples:



Logistic Regression compared to Linear

	Linear Regression	Logistic Regression
Simple Hypothesis Function	**Hypothesis:** $h_\theta(x) = \theta_0 + \theta_1 x$	$h_\theta(x) = g(\theta^T x)$ $g(z) = \frac{1}{1+e^{-z}}$
Simple Cost Function	**Cost Function:** $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$ **Goal:** $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$	$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$ $= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$

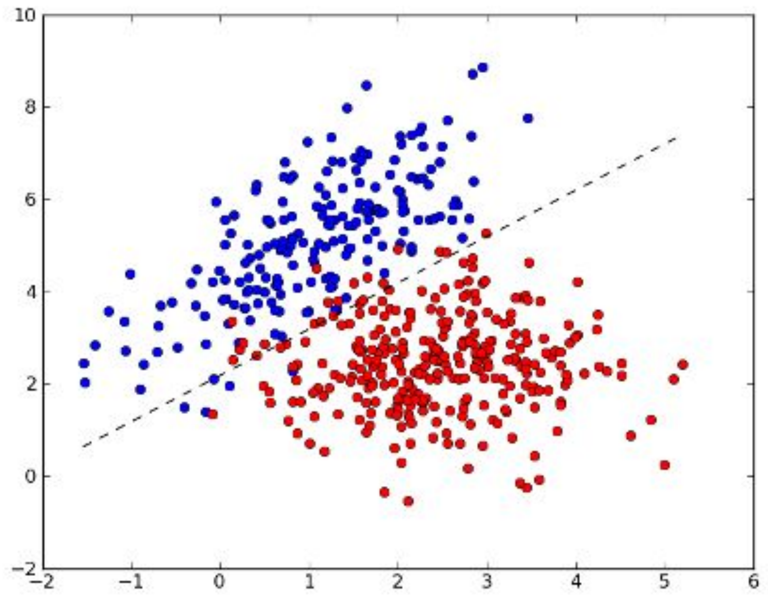
Linear Classification

Based on features, what is is the boundary between classes?

What separates Good Mail from Junk Mail?

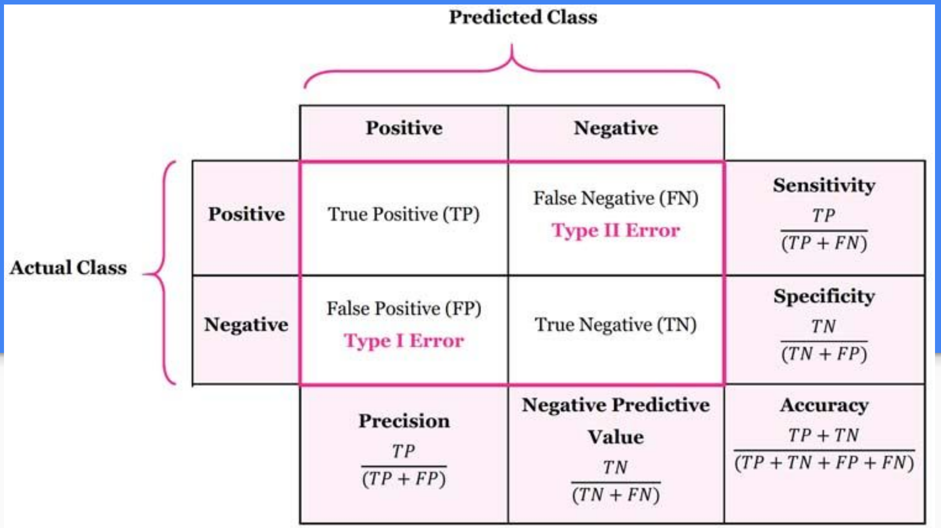
What separates fruits into classes such as Orange, Apple, Banana

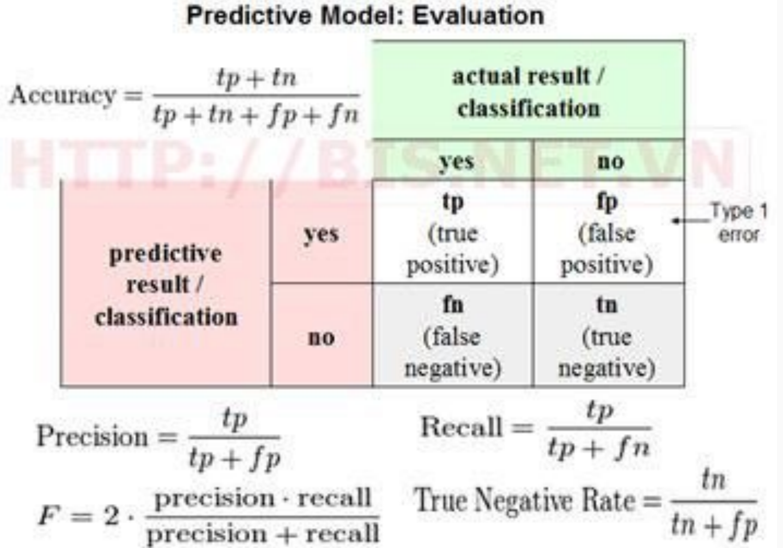
What separates orchids into their different species?



Classification Considerations
Confusion Matrix

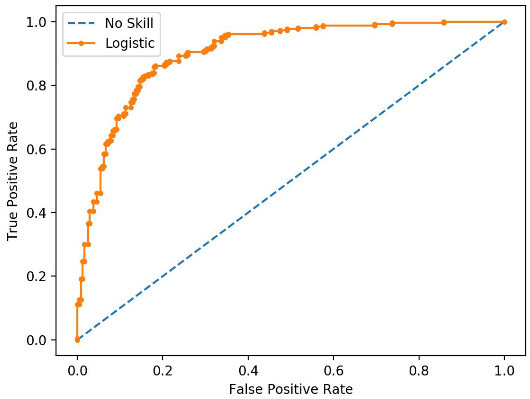
- Classification Metrics
  - Error Types
    - Type 1 - **False Positives** (Said yes but don’t have)
    - Type 2 - **False Negatives** (Said no but do have)
  - Sensitivity
    proportion of actual negatives that are correctly identified as such
  - Specificity
    actual positives that are correctly identified as such
  - Accuracy
  - F1 score

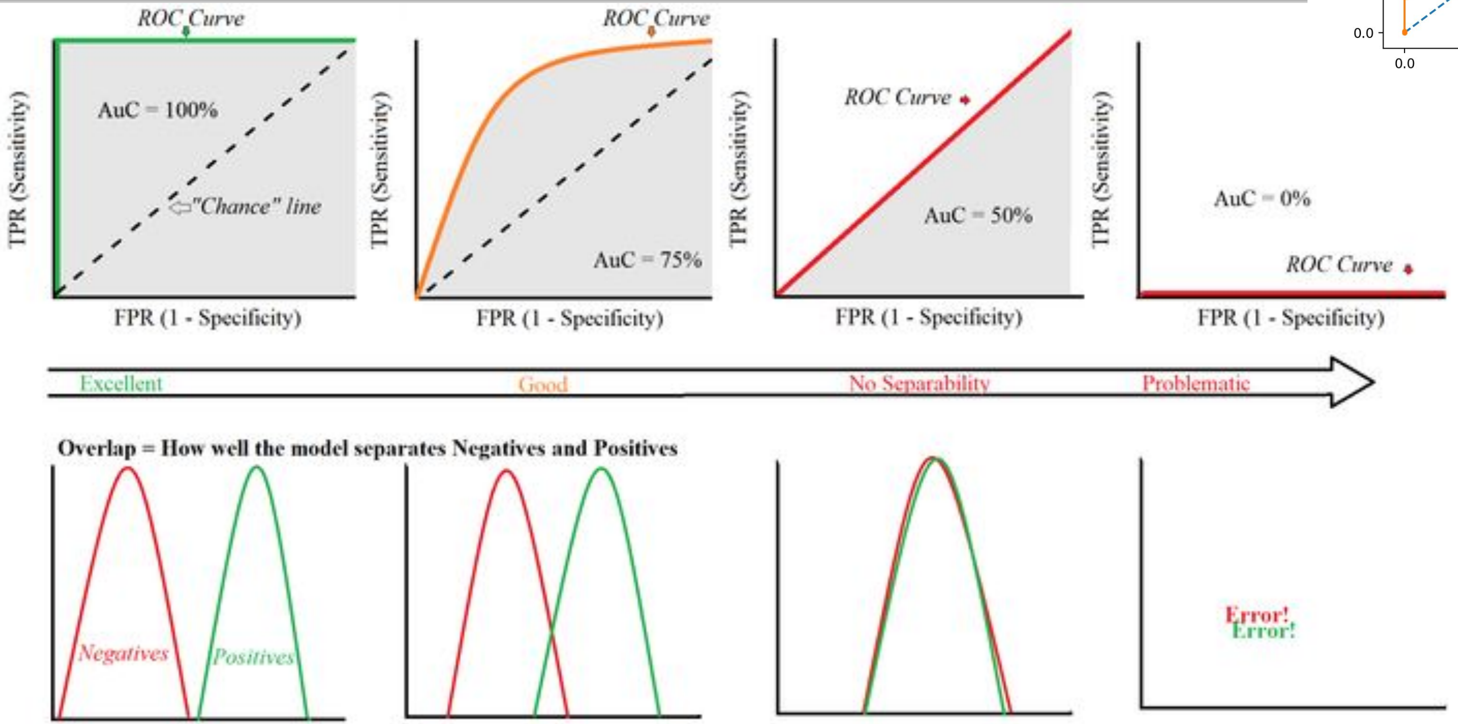




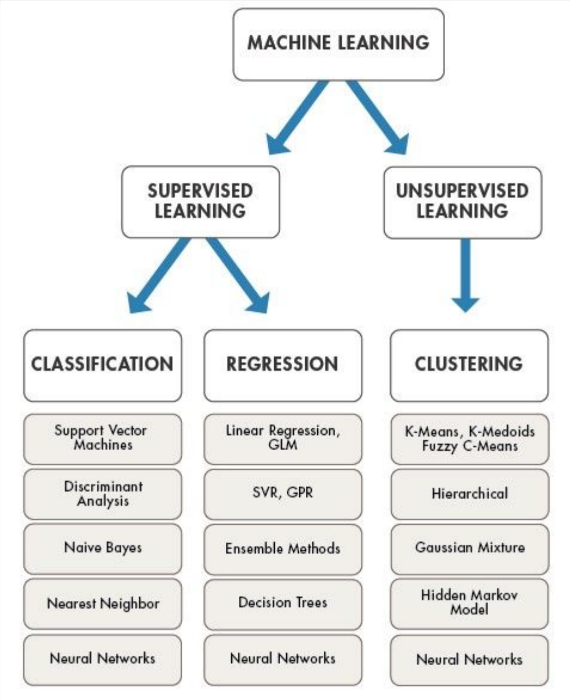
Classification Considerations

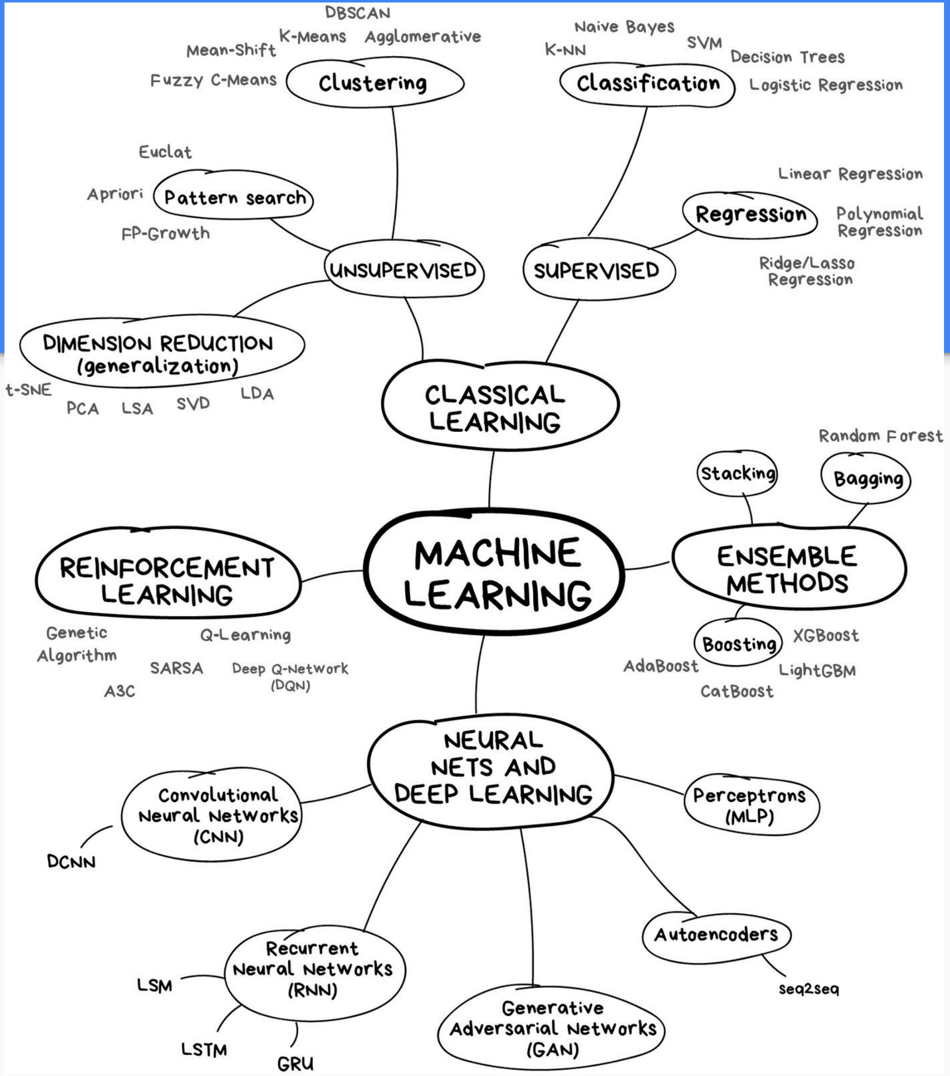
AUC/ROC





Many other types of models & algorithms





Common Data Science Tasks

- Data Acquisition
  - APIs
  - Databases
  - Data files
  - NLP / Data Extraction from semi-structured, unstructured data
- Dataset Combination
- Data Exploration
  - Summary Statistics
  - Correlation
  - Visualization
- Data Pruning
  - Instance Filtering
  - Column Filtering
- Data Imputation
- Feature Discovery
- Feature Reduction
  - PCA
- Split Dataset
  - 70% train - build a model,
  - 20% validation - validate model to tune parameters (e.g, alpha, lambda),
  - 10% test - final evaluation
- Build Model
- Evaluate Model

Basic Principles

- Keep it Simple
- Start Small
- Deep Narrow Slices that Work
- Limit Unknowns
- Get it Working Quickly then Build On little by little
- Start with what you understand and experiment a little from there

Starting out? Avoid:

- Large datasets
- Complex Algorithms
- Using things you don’t understand

Focus on:

- Learn a little
- Do a lot
- Understand more
- Repeat

Where to Start

- Data Sources
  - Python
    https://python-data-science.readthedocs.io/en/latest/datasets.html
  - R
    data()
  - https://www.kaggle.com/datasets
- Development IDEs
  - R
    - RStudio
    - MRan (MS)
  - Python
    - VS Code
    - Jupyter Notebooks

- Libraries
  - Python
    - NumPy
    - Pandas
    - Scikit-learn
    - Tensorflow (really need GPU)
    - Keras (really need GPU)
    - PyTorch (really need GPU)
  - R
    - Caret and many others

Where to go from here

- Explore
  - Sample datasets
  - follow a short tutorial
  - do some visualization
  - look at our other presentations
    https://github.com/dsindy/presentations
  - Regression
  - Classification
- Take a course
  - Data Analysis
  - Visualization
  - Beginning Machine Learning
- Background Courses
  - Python Programming
  - Linear Algebra
  - Databases / SQL

Some Reference Links

- Kaggle
- Anaconda
- VS Code
- RStudio
- Scikit-Learn and Caret

- Deep Learning when you are ready
  - https://www.fast.ai/
  - https://www.deeplearning.ai/
  - https://playground.tensorflow.org

- Examples from others
  - https://towardsdatascience.com/the-ultimate-guide-to-getting-started-in-data-science-234149684ef7
  - https://towardsdatascience.com/how-to-go-into-data-science-c1f6ef258438
- Courses
  - https://www.edx.org/course/subject/data-science
  - https://www.coursera.org/browse/data-science
  - https://www.udacity.com/
- Other
  - https://www.freecodecamp.org/

A couple simple tutorials to get you started

Beginner’s Guide to Linear Regression Python with Scikit-Learn

Linear Regression in R

Questions?

DSIndy Slack Channel

indydata.slack.com

DSIndy GitHub

https://github.com/dsindy/presentations

DSIndy Meetup

https://www.meetup.com/dsindy/

Good luck!