April 12th, 2021

Almost To The End

Agenda Until We Part Ways

Today - BIG DAY

- Scaling & Normalization
- Data Wrangling for ML
- Feature Generation
- Dealing with time, images and text
- Model Results

Today - Introduction to Neural Networks... FUN!!!

Next Week - Steve's Presentation, Advanced Topics, Recap of End-to-End Data Science Project

Exam

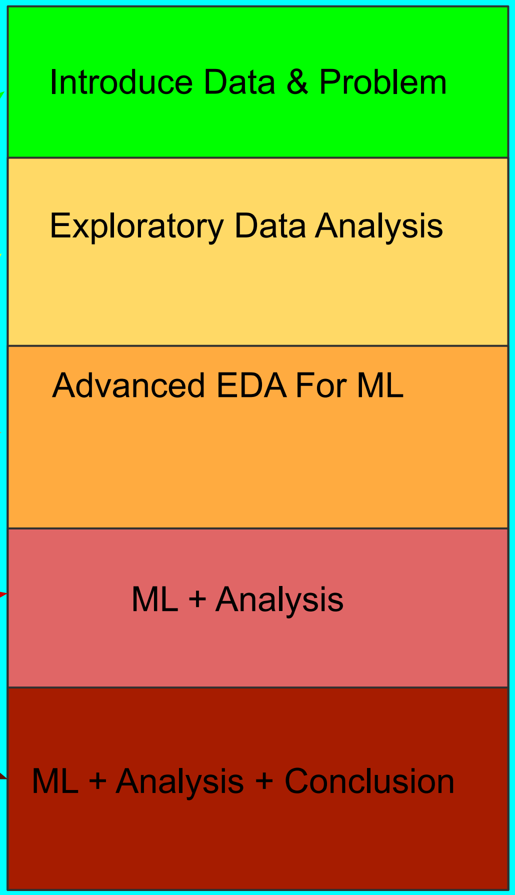
Let’s go over this - Jupyter

Final Project - Ok for 2.5 Hours??

8 minute presentations - how do we feel about time??

This is a suggested breakdown

- ➔ What is your data, what is your problem? 1min
- ➔ EDA - Midterm Presentations (what's in the data?) 1-2min
- ➔ EDA with a purpose for ML 1-2min
  - ◆ Distributions
  - ◆ Bootstrapping
  - ◆ Balancing
  - ◆ Train / Test / K Folds Splits?
- ➔ ML Baseline - what comes out of the box 1-2min
  - ◆ Describe the algorithm to us - teach me!
- ➔ ML Improvement - how did you improve a model? 1-2min
  - ◆ Comparisons from 'EDA with a purpose'
  - ◆ Explanations + teach me!
  - ◆ Wrap up - lessons learned



5 Minutes At The End

Prepping All Your Stuffs

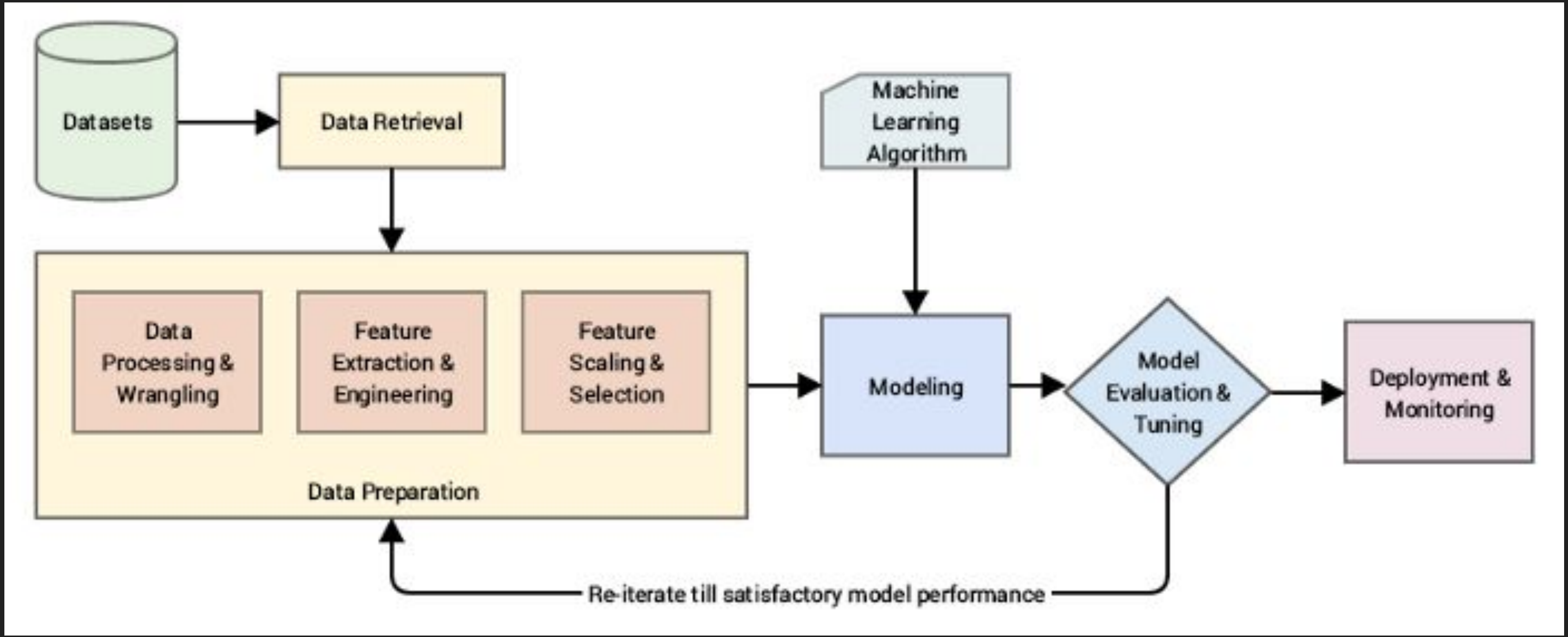
Github, Kaggle, LinkedIn, Resume, Website - AGAIN

Side Projects

Get your code base - database - organized somewhere!

Recall - this is what we are doing

You should be able to have (this week?) built a qualitative roadmap for the yellow boxes



Data Preprocessing

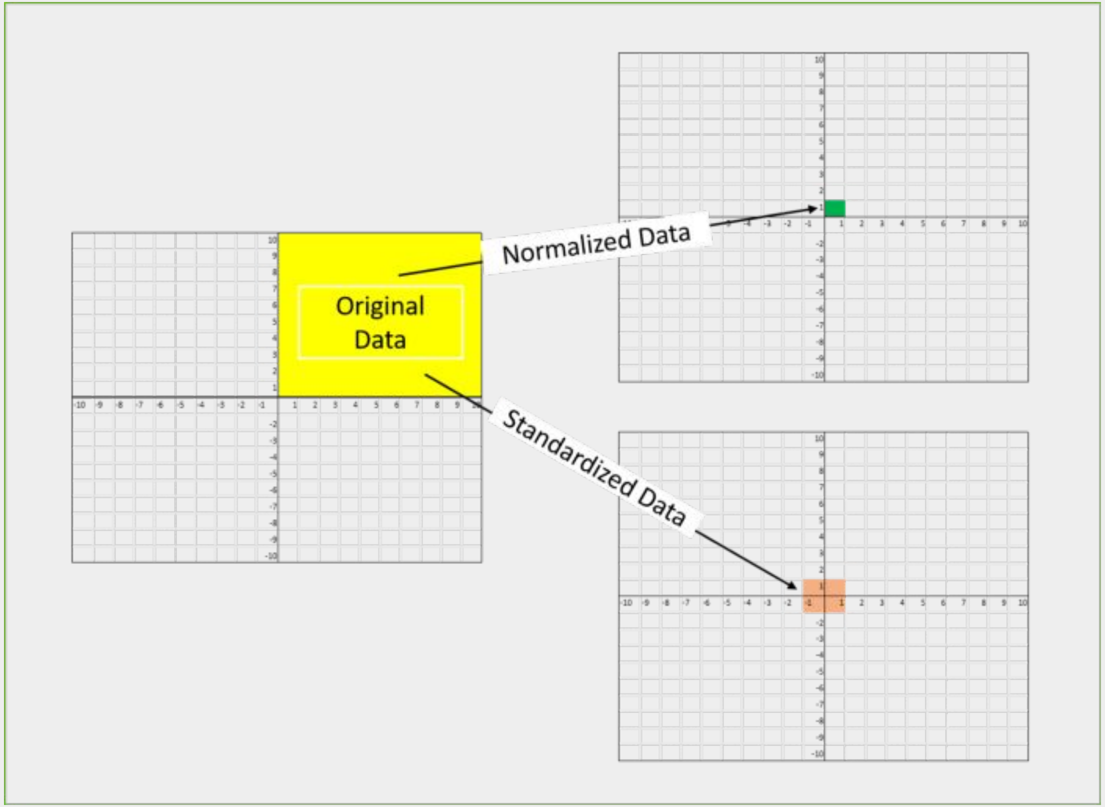
It’s an art...but here are some methods based on what you kinda already know…

Feature Generation

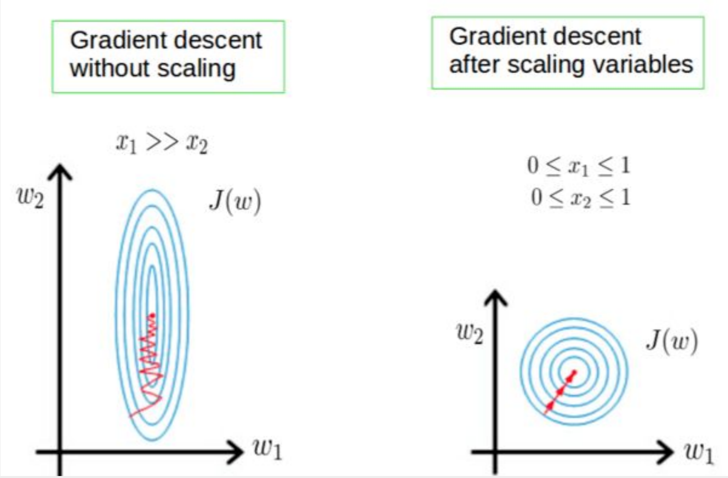
Creating columns (# per 100K), binning and bucketing, get_dummies()

Combinations of columns - can we afford to do this entirely? NO. Strategically YES.

Let’s start *thinking* about the maths



Machine learning algorithms only see numbers. Vectors. Matrices.



Let's start *thinking* about the maths

ML algorithms are sensitive to the "**relative scales of features,**" which usually happens when it uses the numeric values of the features rather than say their rank.

Scaling *Should Be Done*
K-Means
KNN
Gradient Based Algorithms

Scaling *Is Not Required*
Decision Trees
Random Forests
Naive Bayes

1) Min Max Scaler
2) Standard Scaler
3) Max Abs Scaler
4) Robust Scaler
5) **Quantile Transformer Scaler**
6) Power Transformer Scaler
7) Unit Vector Scalar

Like most other machine learning steps, feature scaling is a trial and error process, not a single silver bullet.

Data Preprocessing

Normalization:

The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

For example, consider a data set containing two features, age, and income(x2). Where age ranges from 0–100, while income ranges from 0–100,000 and higher. Income is about 1,000 times larger than age. So, these two features are in very different ranges. When we do further analysis, like multivariate linear regression, for example, the attributed income will intrinsically influence the result more due to its larger value. But this doesn’t necessarily mean it is more important as a predictor. So we normalize the data to bring all the variables to the same range.

Normalization is a good technique to use when you do not know the distribution of your data or when you know the distribution is not Gaussian (a bell curve). Normalization is useful when your data has varying scales and the algorithm you are using does not make assumptions about the distribution of your data, such as k-nearest neighbors and artificial neural networks.

Data Preprocessing

Standardization:

Standardizing the features around the center and 0 with a standard deviation of 1 is important when we compare measurements that have different units. Variables that are measured at different scales do not contribute equally to the analysis and might end up creating a bais.

For example, A variable that ranges between 0 and 1000 will outweigh a variable that ranges between 0 and 1. Using these variables without standardization will give the variable with the larger range weight of 1000 in the analysis. Transforming the data to comparable scales can prevent this problem. Typical data standardization procedures equalize the range and/or data variability.

Standardization assumes that your data has a Gaussian (bell curve) distribution. This does not strictly have to be true, but the technique is more effective if your attribute distribution is Gaussian. Standardization is useful when your data has varying scales and the algorithm you are using does make assumptions about your data having a Gaussian distribution, such as linear regression, logistic regression, and linear discriminant analysis.

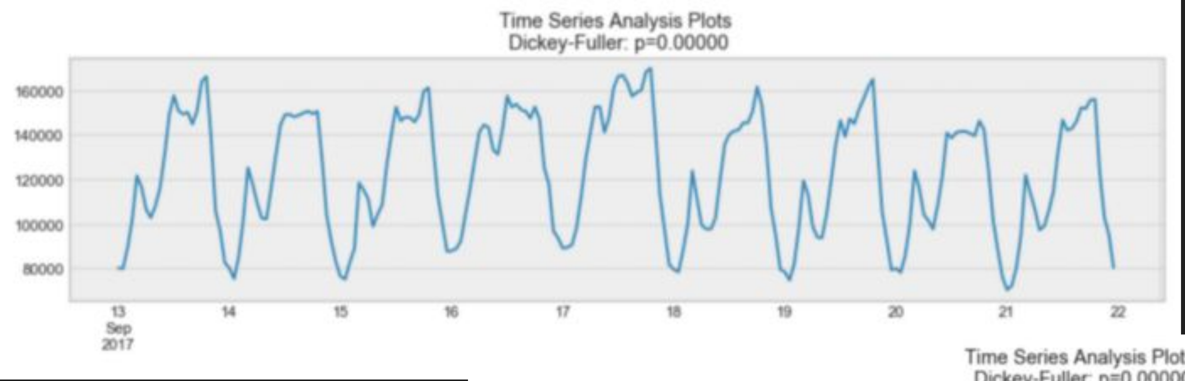
Data Preprocessing

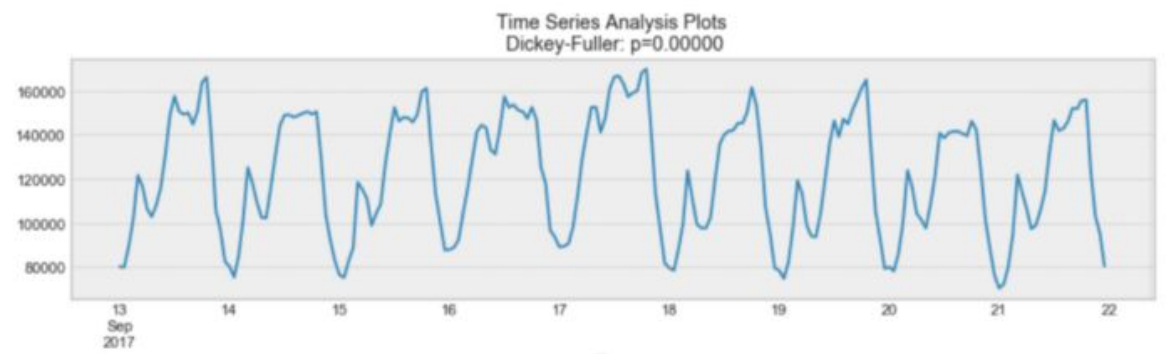
Binning:

Histograms are examples of binning. As is any bucketing of continuous variables to discrete categories. Examples would be floating point values between 1.0 and 100.0 where duplication is scarce. A good method for using this as classification labels or categorical features is to bin them into appropriate bins based off domain knowledge or exploratory data analysis. For example three bins of low, medium and high with ranges 1.0-33.0, 33.0-67.0, 67.0-100.0 where rounding is taken care of.

Time Series Techniques

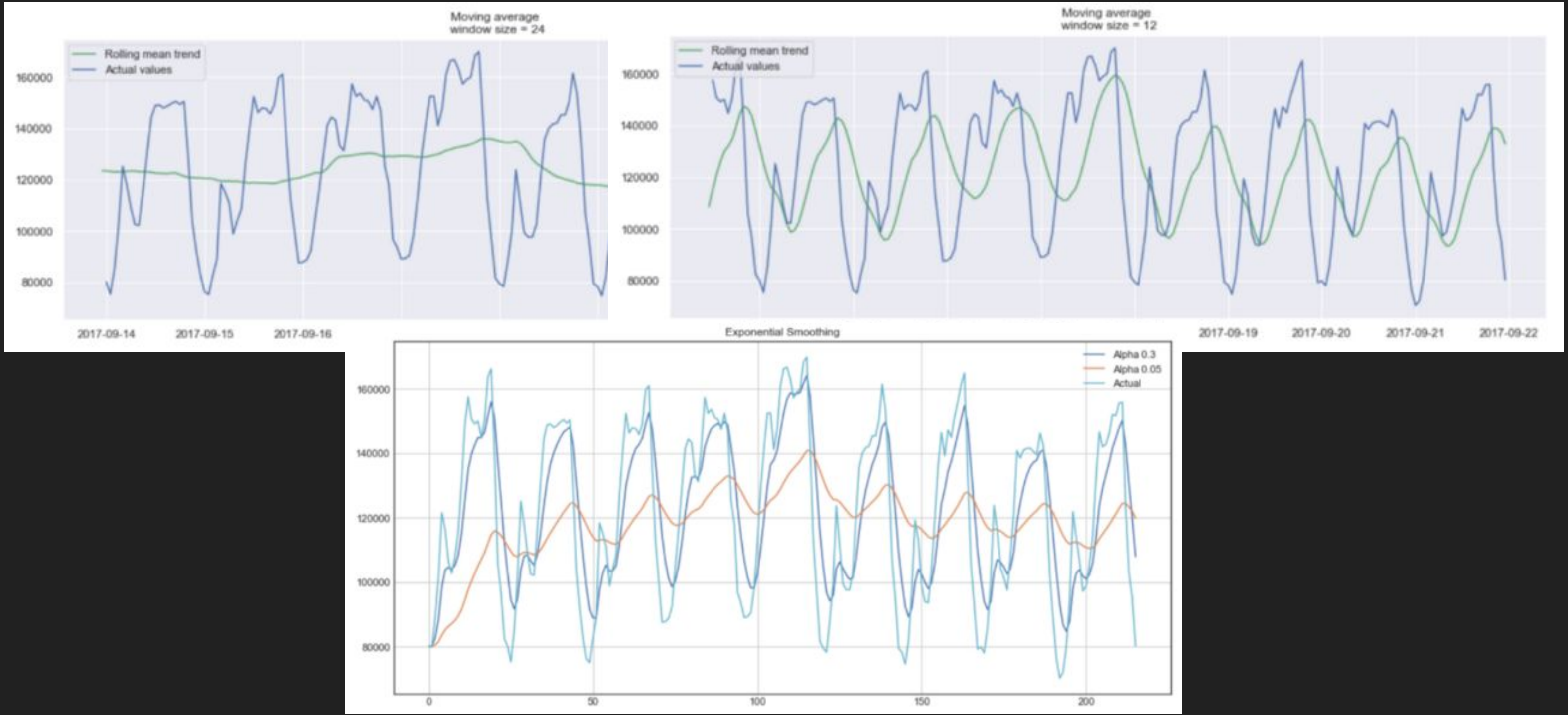
- Stationary? Seasonality?





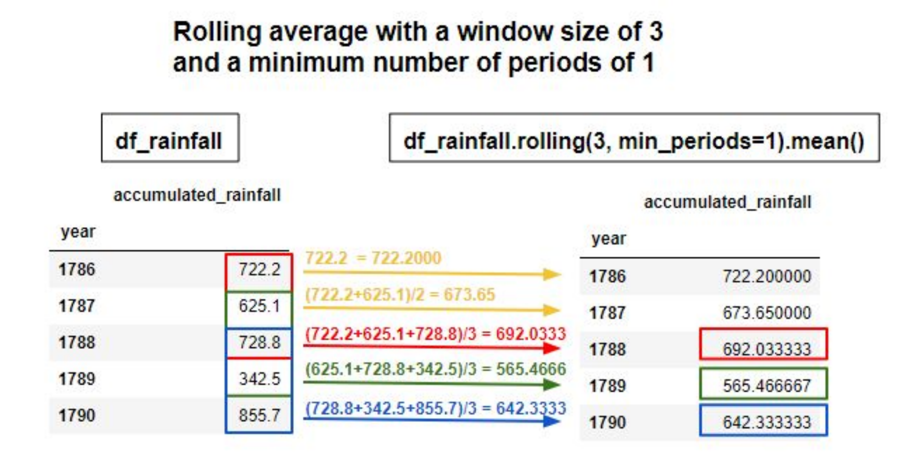
Time Series Techniques

Moving Averages



Time Series Techniques

Moving Averages



year	accumulated_rainfall
1786	722.2
1787	625.1
1788	728.8
1789	342.5
1790	855.7

year	accumulated_rainfall
1786	722.200000
1787	673.650000
1788	692.033333
1789	565.466667
1790	642.333333

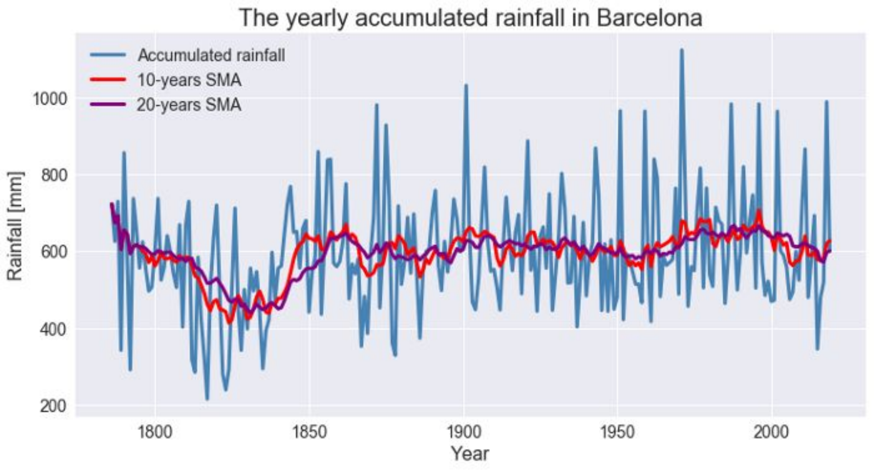


Image Techniques

Resizing Images

Denoising

Segmentation

Morphing

Text Techniques

Converting case

Numbers to words (or removing)

Removing punctuation, accents, etc.

Expand or otherwise handle abbreviations

Remove sparse terms

Discussion of NLTK (My favorite, but many others exist)

Train Test Validation (or the other way people say it)

Split your dataset into 3 components:

1) Training data
   a) Use this for all experimentation with data flow and getting a model working
2) Validation data
   a) Use this for exploring parameters, hyperparameters and feature generation / preprocessing
3) Test data
   a) Never look at this data - hold it out
   b) Use this dataset for your final results (accuracy, precision, recall)

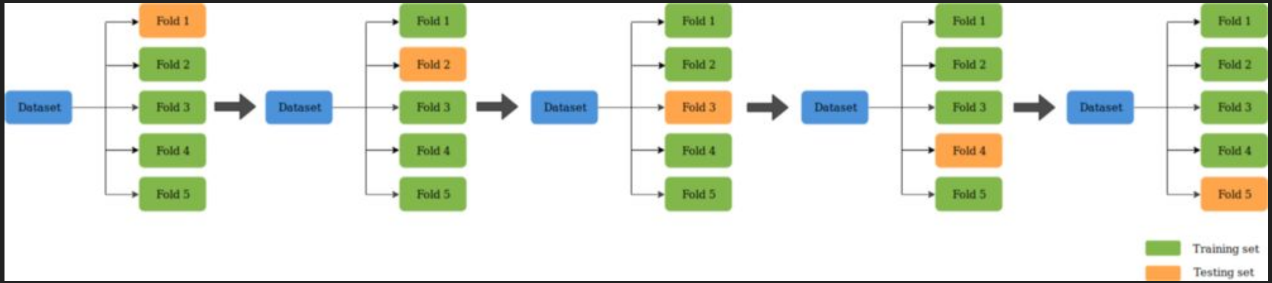
K Folds Cross Validation

```
cv = KFold(n_splits=10, random_state=42, shuffle=False)
```
cv is a list of (start, stop) indices

Cross validation is a good method to ensure your training set is, well, valid.

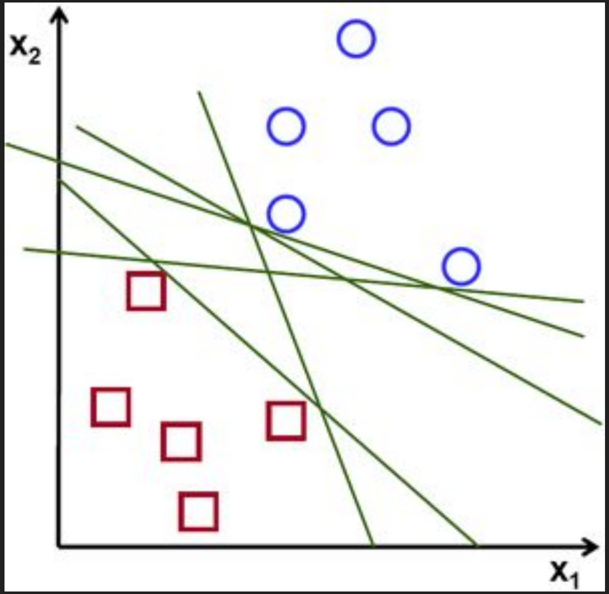
This means your results are purely due to luck (unluck) of the draw.

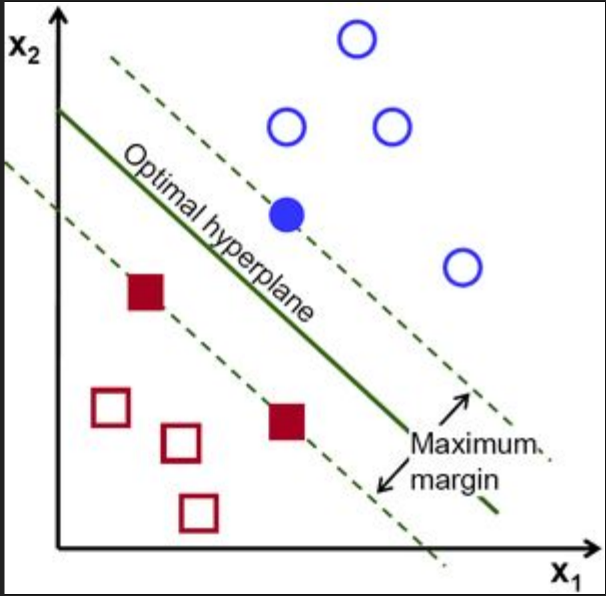
This is a best practice if you want to be taken seriously, especially with smaller datas.



A quick slide on Support Vector Machines

An algorithm we don't have time to really discuss, want to highlight the mechanics as a baseline as they once were powerful stallions. There are many algos!!





Introduction to Neural Networks

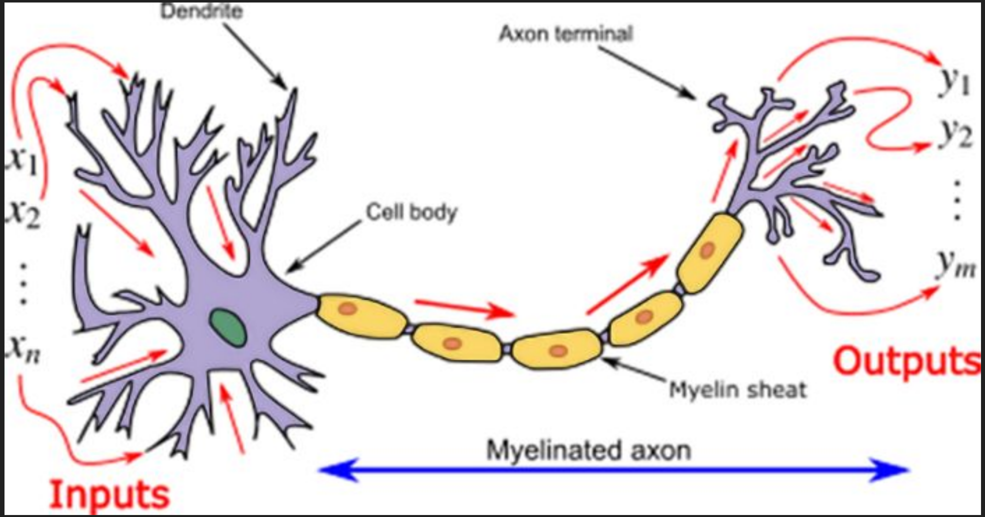
PyTorch & Keras

What is a NN?

How to think about it - a brain?

Simple Models

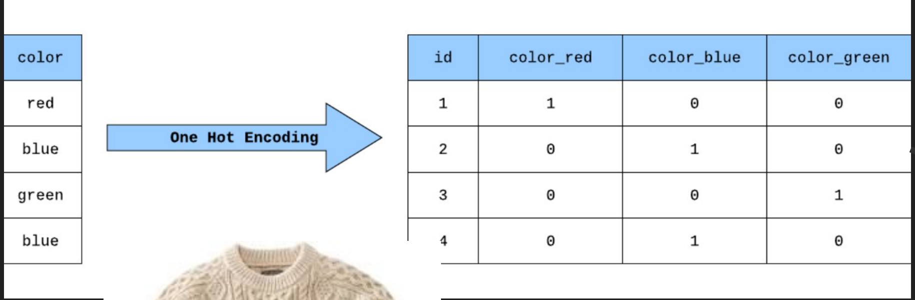
What they can do



Neural Networks

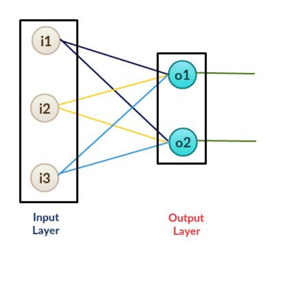
Feature generation? Where we’re going we don’t need feature generation.

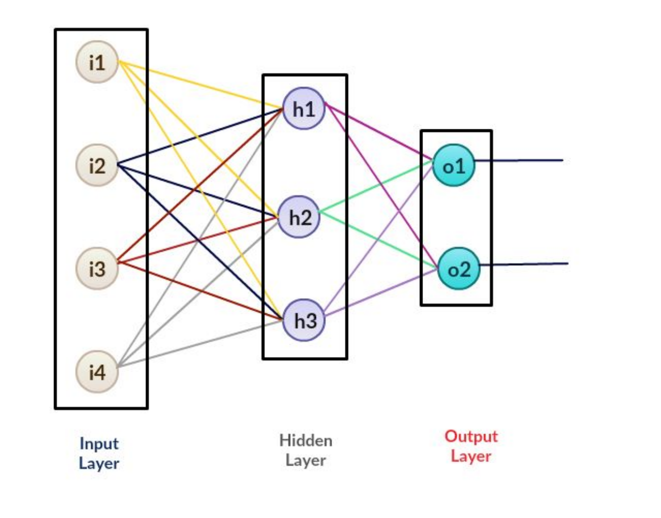
Large input space (think images, text, features you converted, to say, binary)



color
red
blue
green
blue

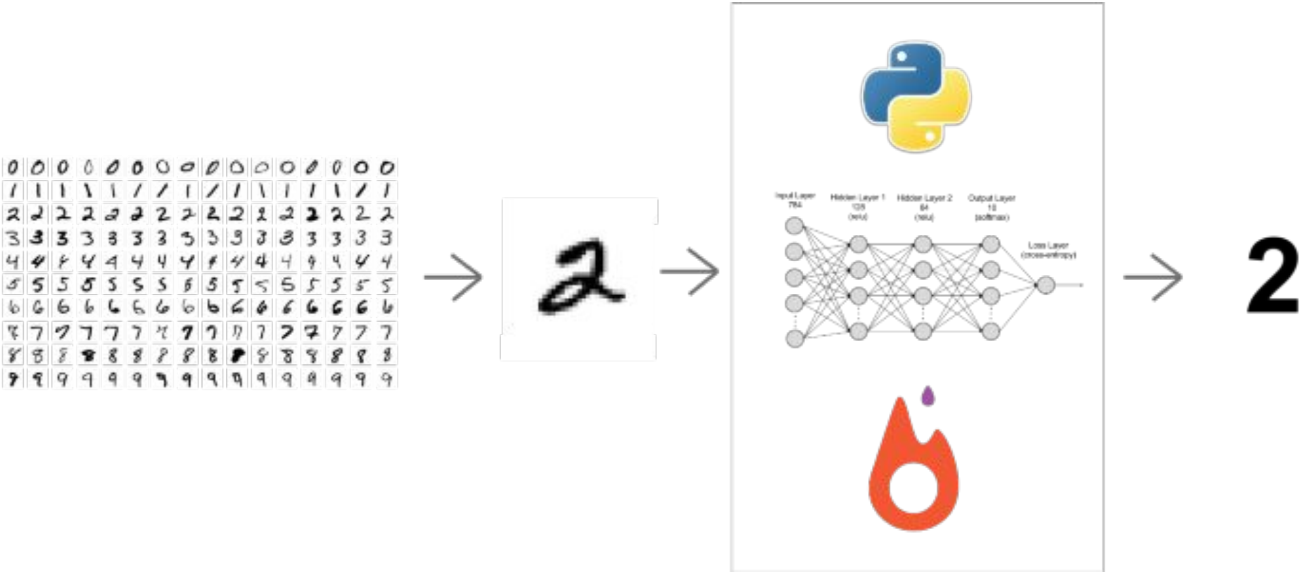
id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0



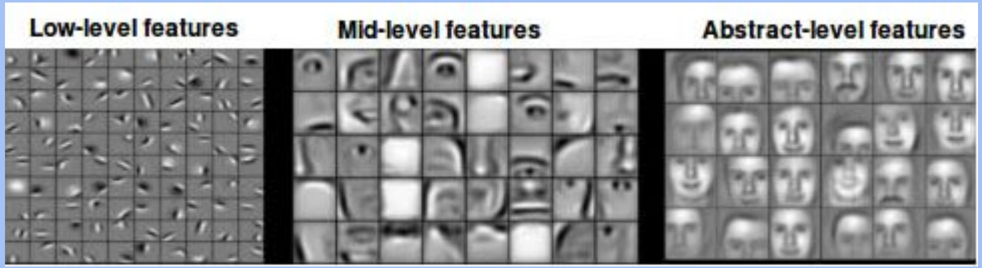


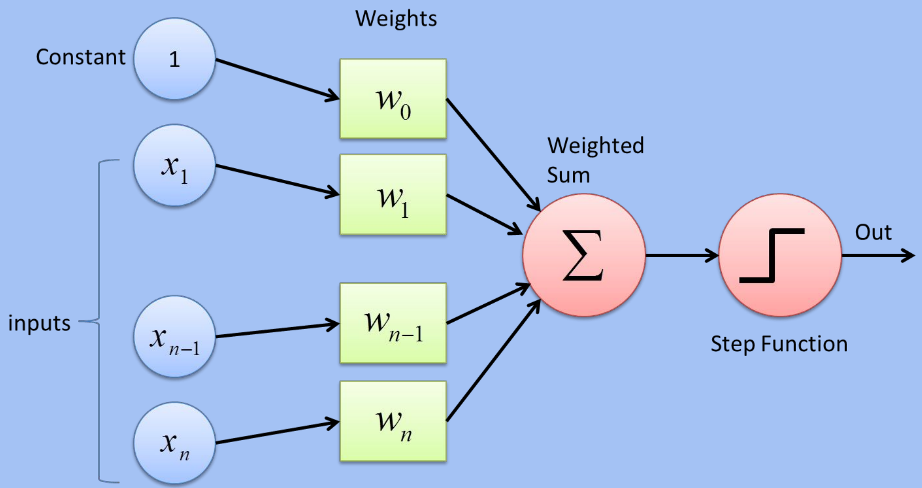
Deep Neural Networks

DNN's are used to learn 'feelz' as we say to those who don't like them...



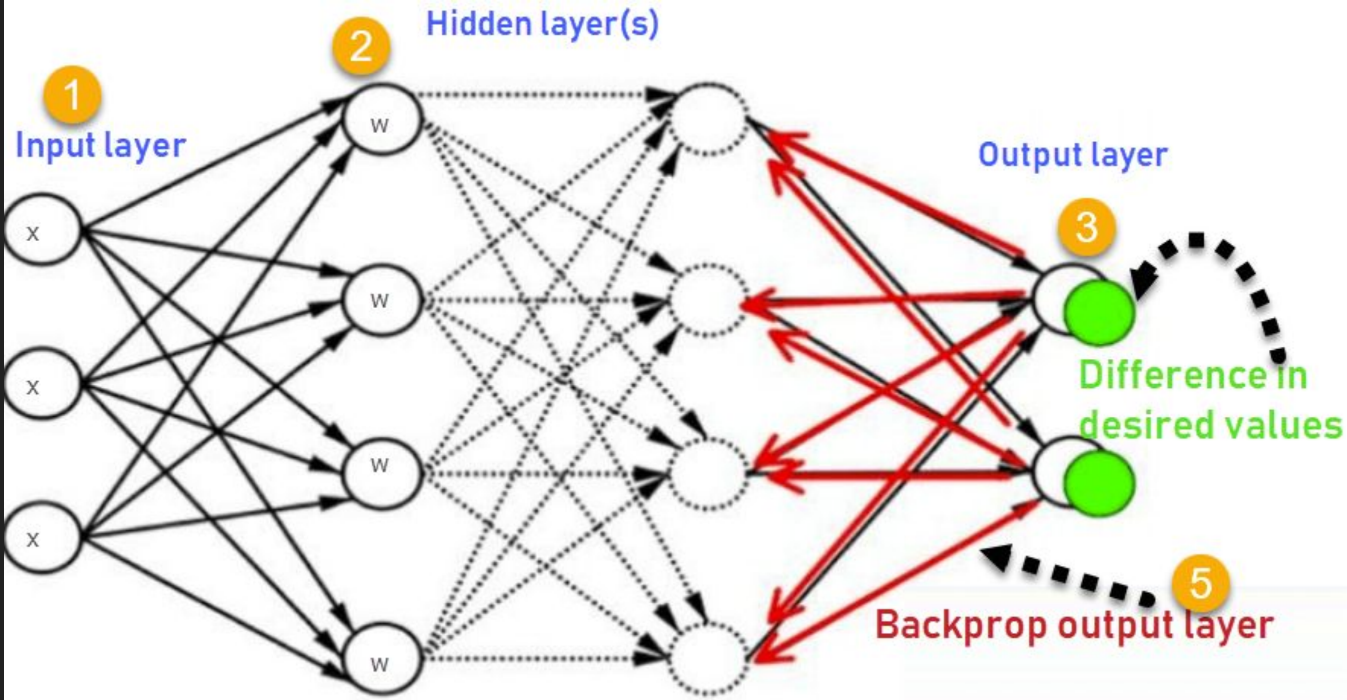
Little to no feature engineering





Deep Neural Networks

Gradient descent in the form of backpropagation



Data Flow In Deep Neural Networks?

It’s the same as for any other classification task!

1D or 2D (or ND) array to the input layer…

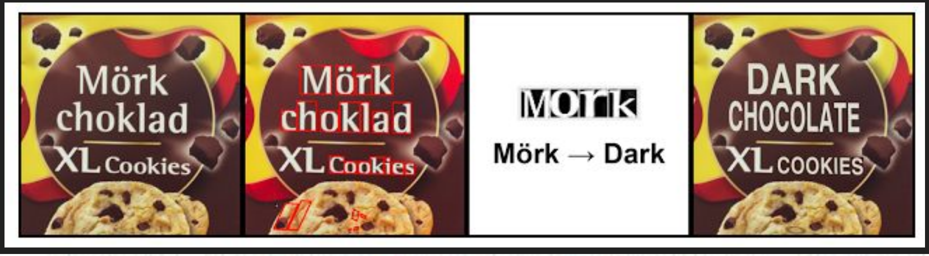
Binary or Multiclass outputs in the output layer…

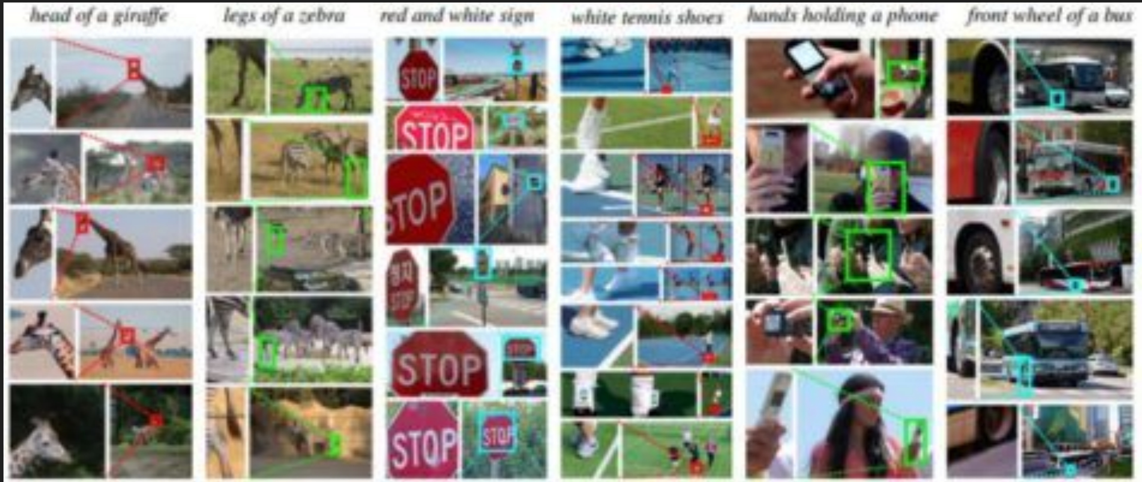
Let frameworks do the rest!

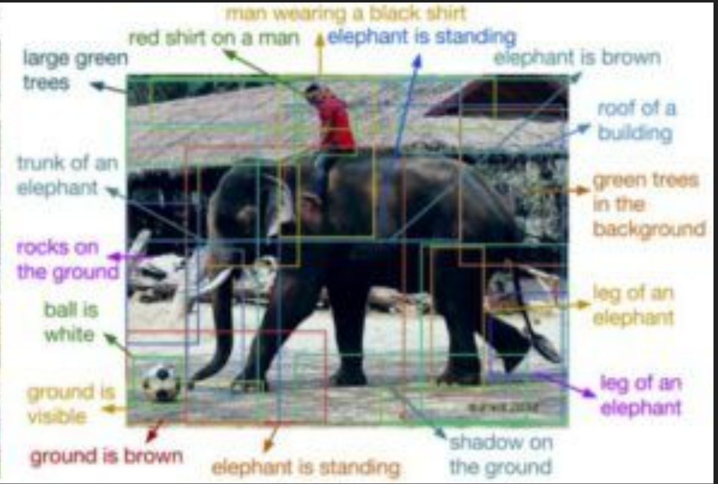
Well aside from the Data Science part of what works, what doesn’t and how to fix!

Deep Neural Networks

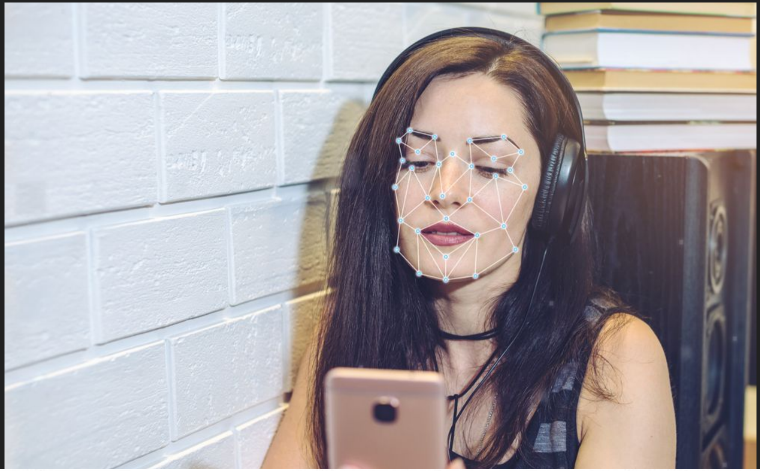
The coolest stuff…







Deep Neural Networks



Next Class - The Last Learning Class

End-to-End Recap

- Request > Data > EDA > Insights > Report > Preprocessing > ML > Results

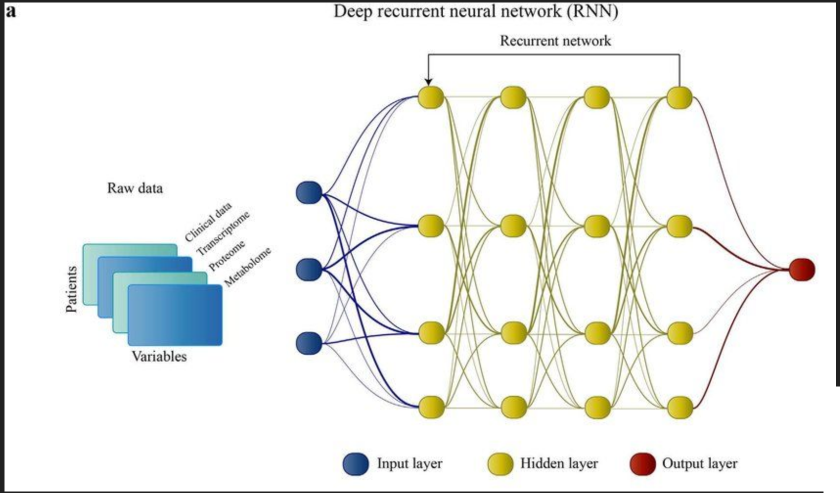
Steve's Presentation

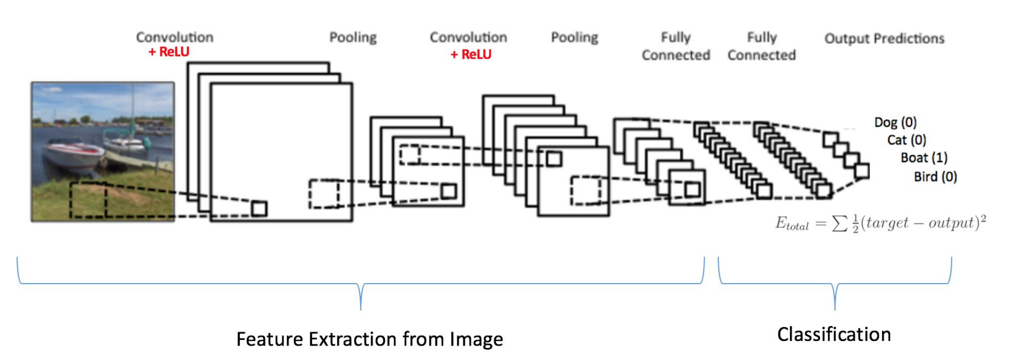
GAN Presentation

?? Anything ??

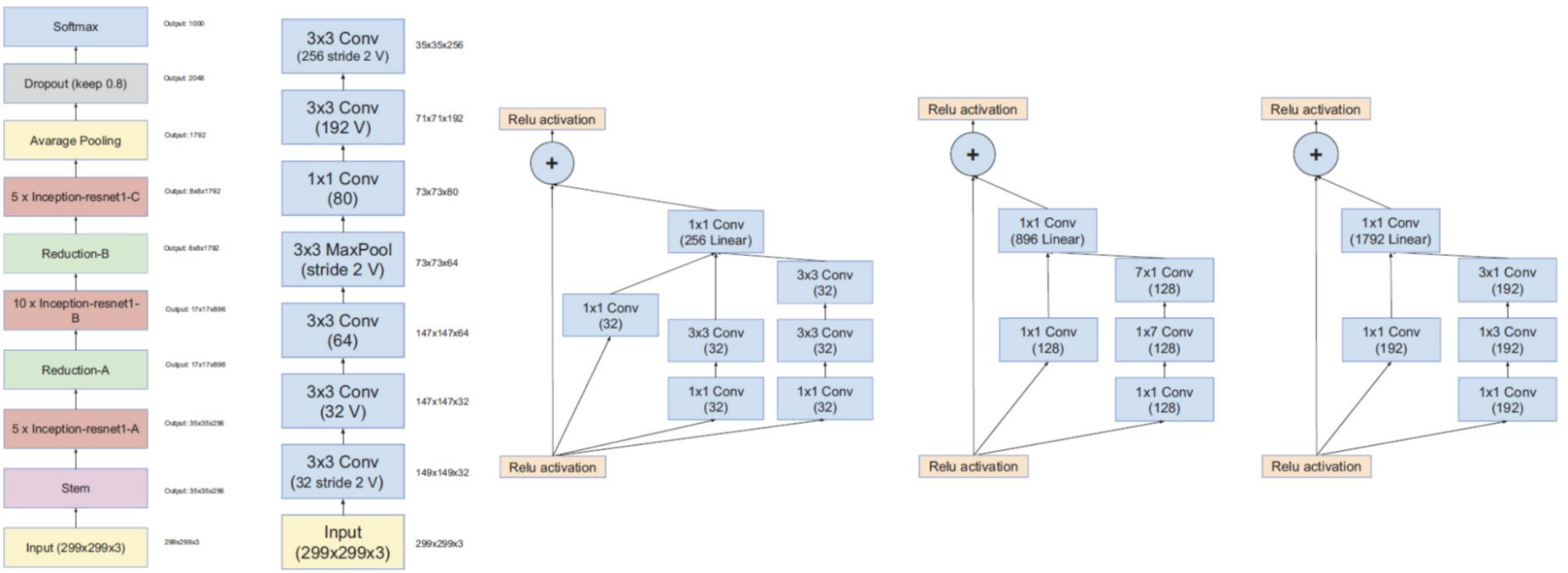
Deep Neural Networks - Architectures

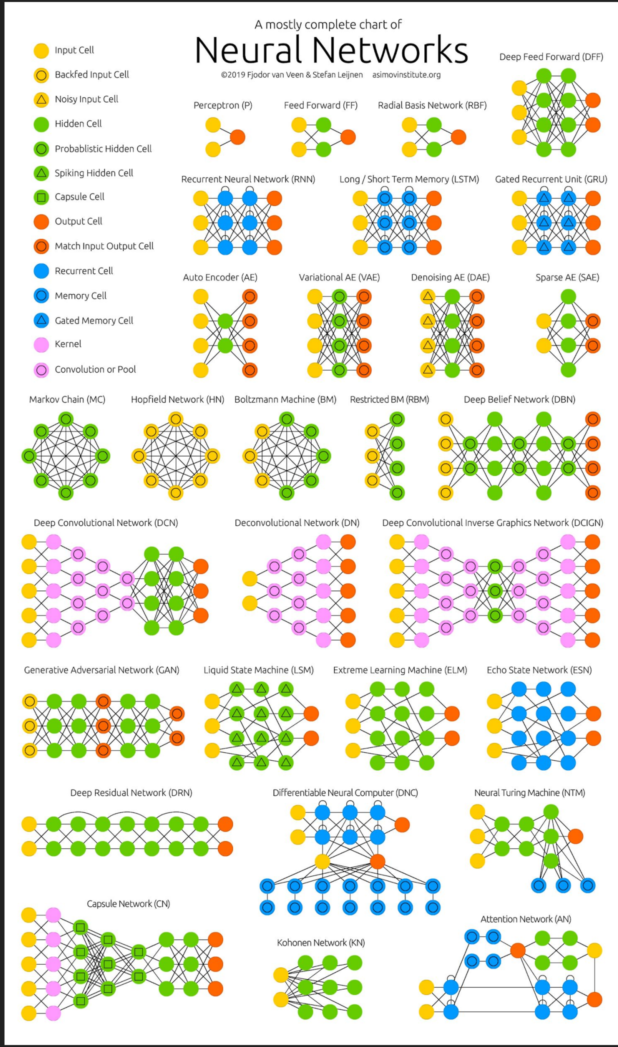
Convolutional Neural Networks, Deep Neural Networks, Recurrent Neural Networks





Example: Inception V4 - but do this MANY times...




A mostly complete chart of
Neural Networks
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org
Input Cell
Backfed Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Capsule Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Gated Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Differentiable Neural Computer (DNC)
Neural Turing Machine (NTM)
Capsule Network (CN)
Kohonen Network (KN)
Attention Network (AN)

Aren’t They The Greatest???

When and where they fail

Explain please?

https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

In order to train AlphaStar, we built a highly scalable distributed training setup using Google's v3 TPUs that supports a population of agents learning from many thousands of parallel instances of StarCraft II. The AlphaStar league was run for 14 days, using 16 TPUs for each agent. During training, each agent experienced up to 200 years of real-time StarCraft play. The final AlphaStar agent consists of the components of the Nash distribution of the league - in other words, the most effective mixture of strategies that have been discovered - that run on a single desktop GPU.

5 Minutes At The End

Prepping All Your Stuffs

Github, Kaggle, LinkedIn, Resume, Website - AGAIN

Side Projects

Get your code base - database - organized somewhere!