Intro to Machine Learning

December BUG

Agenda

- Personal Background
- Defining machine learning
- Real World Applications
- Machine Learning / Data Science Tools
- Machine Learning Workflow
- Demo
- Resources

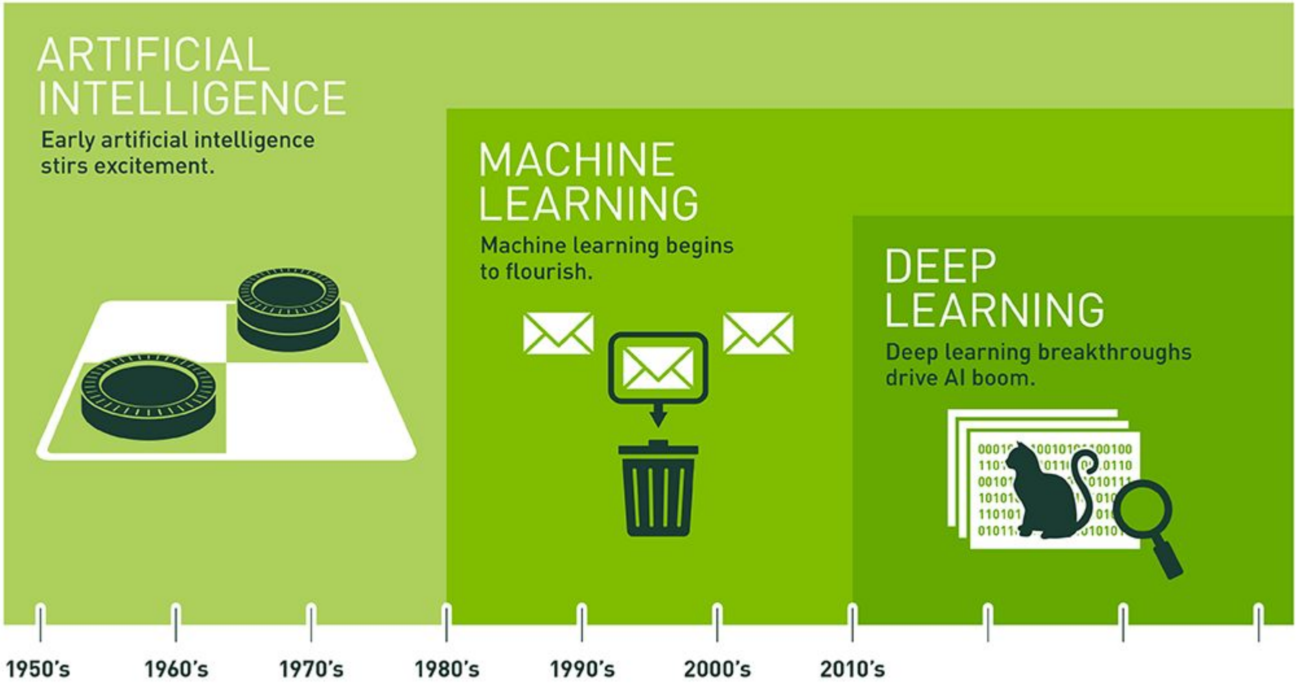
Personal Background

What is machine learning?

”Machine learning is the science of getting computers to act without being explicitly programmed.”
-Andrew Ng

- Machine learning today involves applying statistical models to datasets to predict data, discover patterns, etc.

https://www.coursera.org/learn/machine-learning/lecture/Ujm7v/what-is-machine-learning

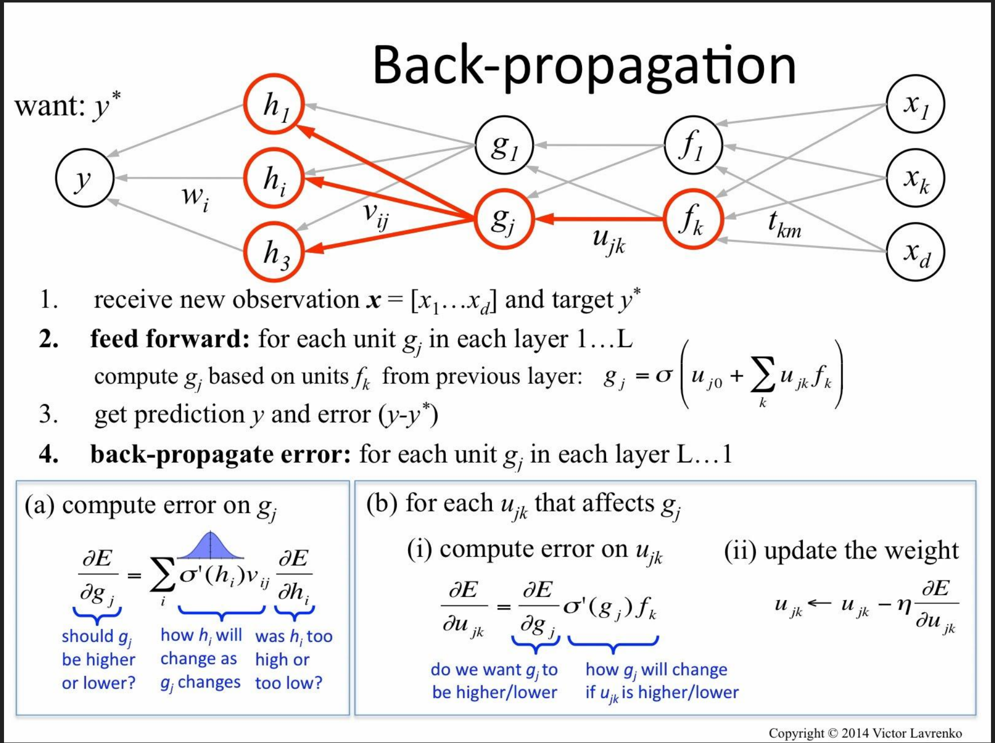


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

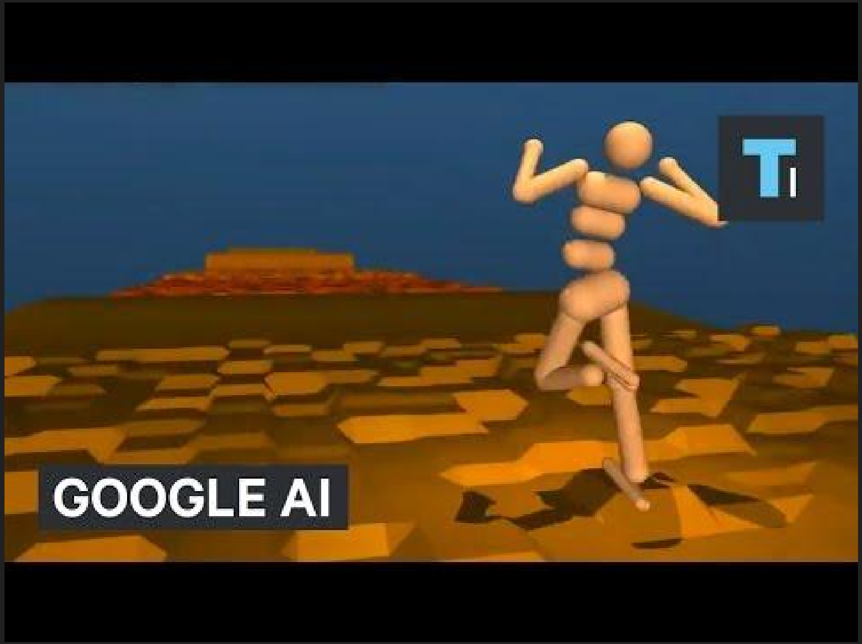
https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Deep Learning - Neural Nets and Backpropagation

Backpropagation - Backward propagation of errors

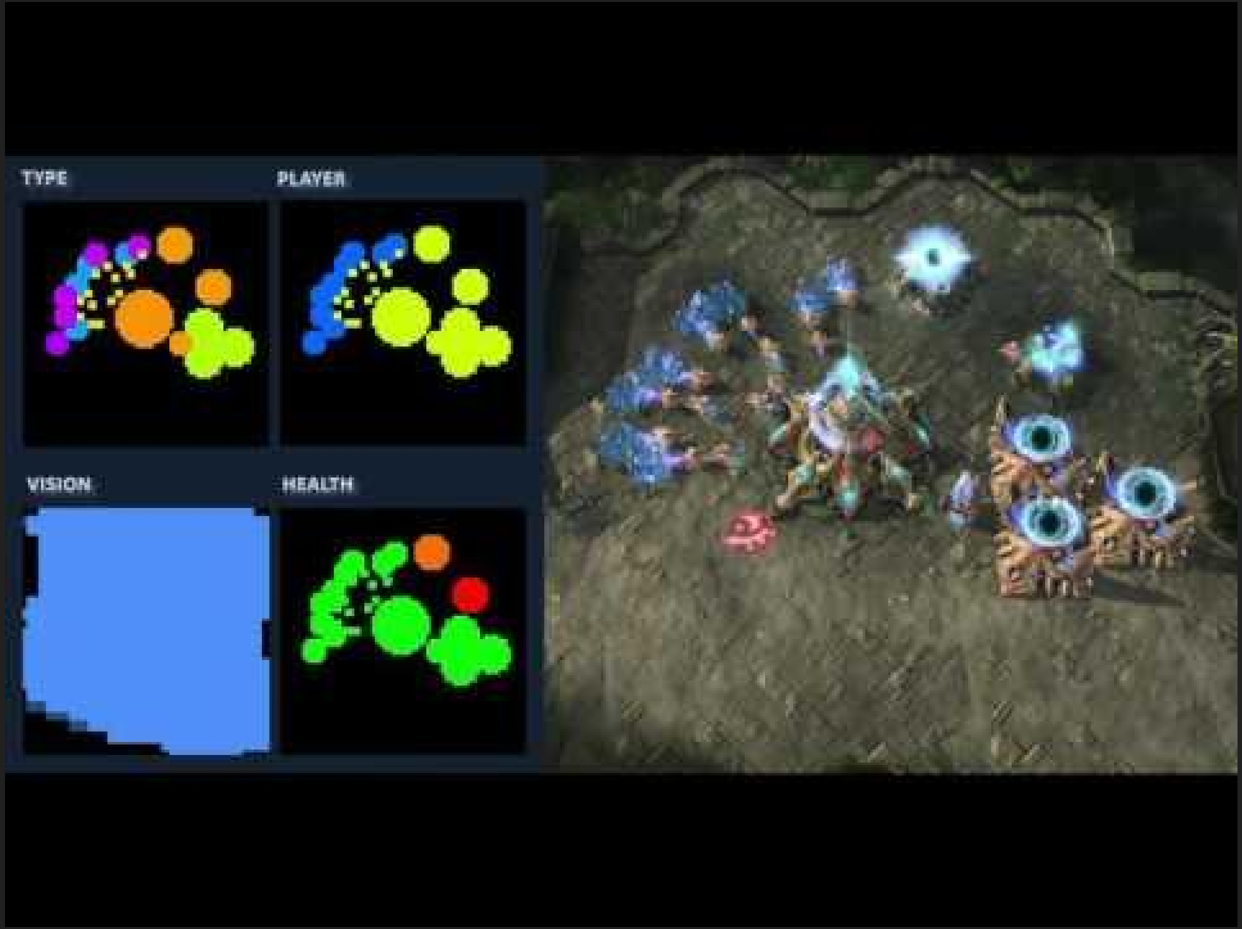


Real World Applications



Real World Applications

Real World Applications


TYPE
PLAYER
VISION
HEALTH

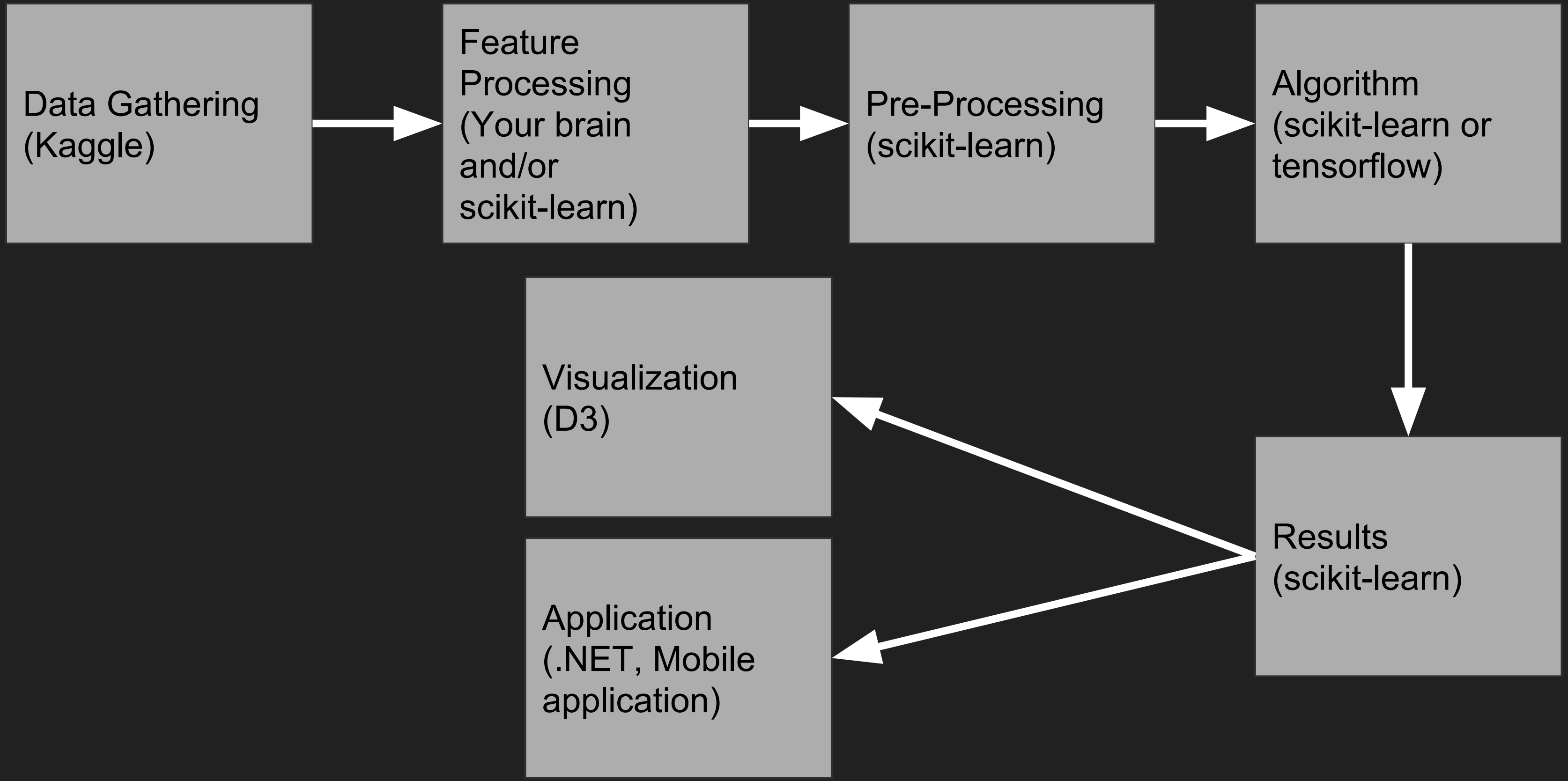
Machine Learning & Data Science Tools

Data Science / Machine Learning Tools

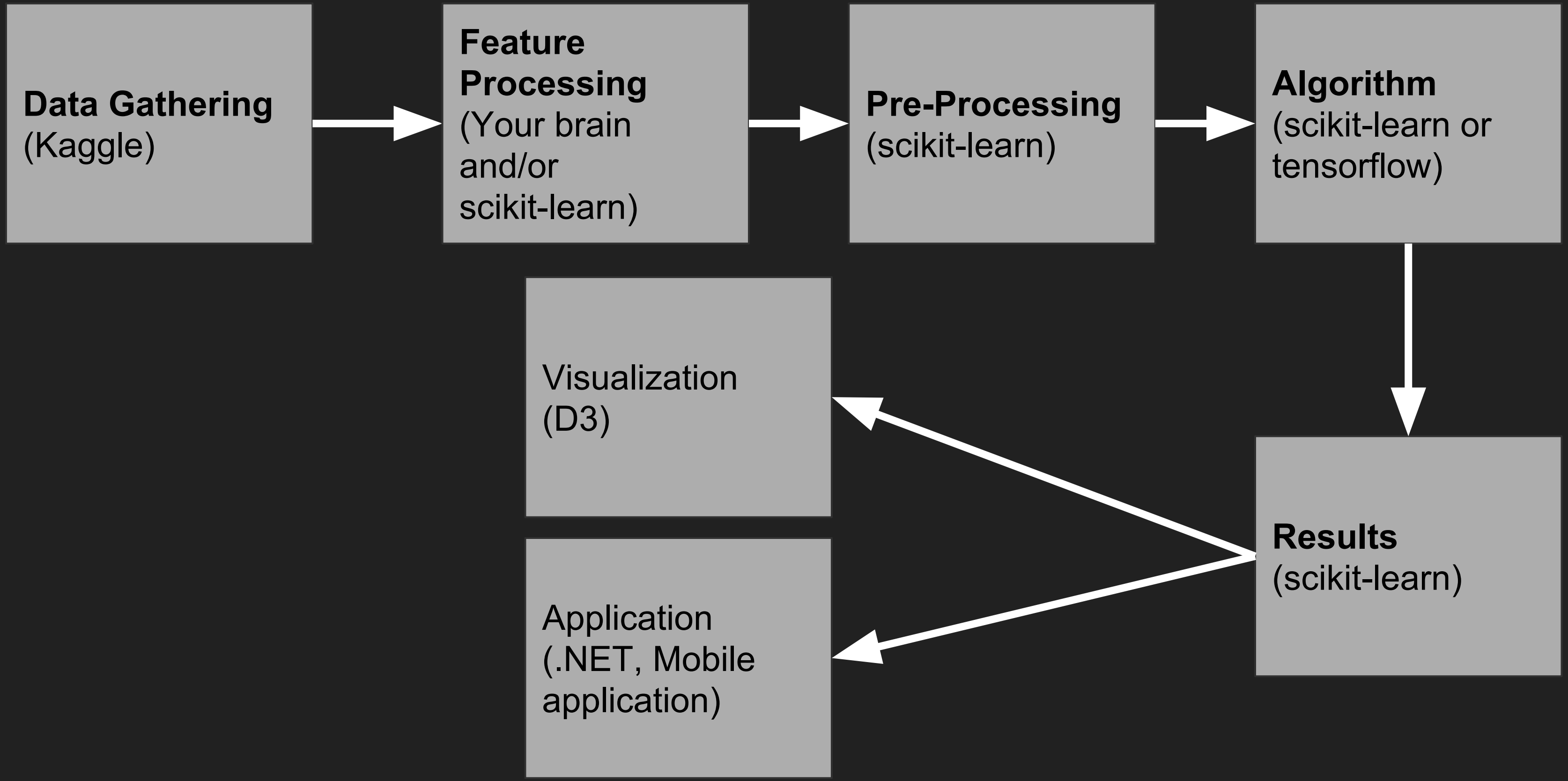
Data Science / Machine Learning Tools

Machine Learning Workflow

Machine Learning Workflow



Machine Learning Workflow



Data Gathering

kaggle

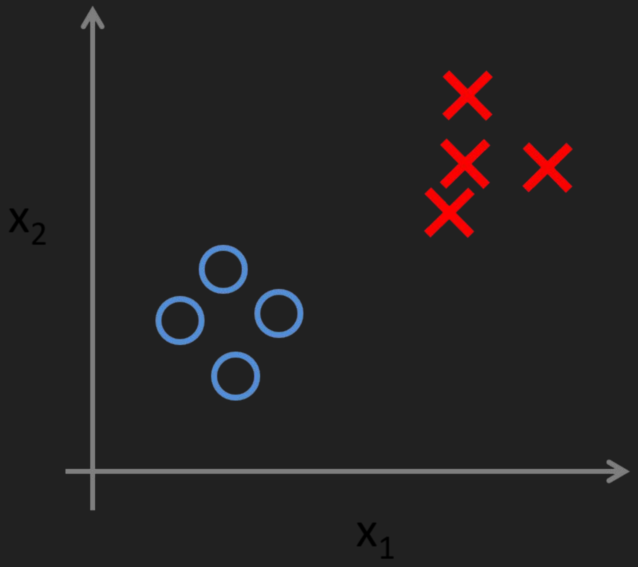
Probably the most important step in machine learning.

Labeled Data - Includes a meaningful “tag” to data. It’s typically what you’re trying to predict.

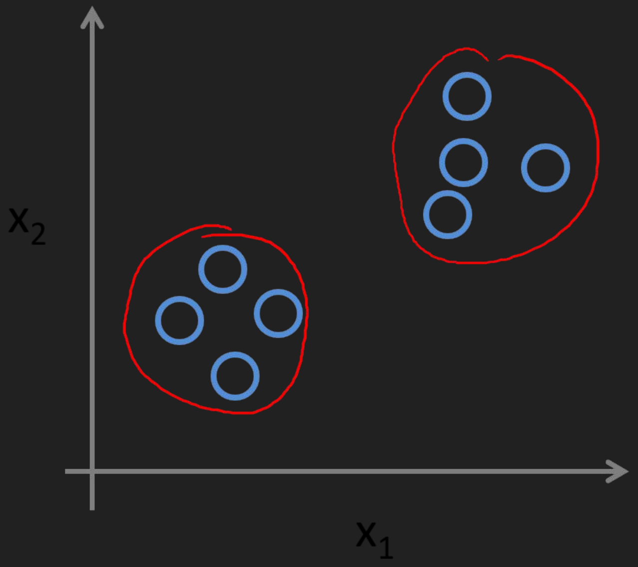
Unlabeled data - Just the data, no classification. (photos, videos)

Unsupervised vs. Supervised Learning

Supervised Learning



Unsupervised Learning

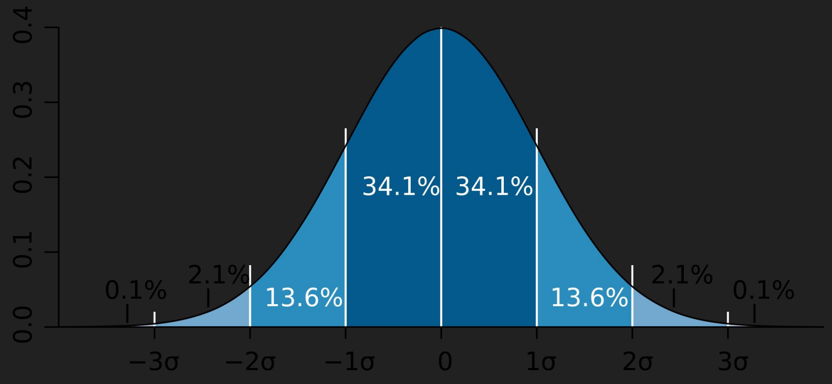


Feature Processing

- Data-Domain Research
- Filling in missing data
- Cartesian product
  - Population Density: urban, suburban, rural
  - State: Washington, Oregon, California
    - Product: urban_Washington, suburban_Washington, rural_Washington, urban Oregon, etc.
- Non-linear transformations
  - Binning
- Domain-specific features
  - Ex. length * width * height = volume
- Variable-specific features
  - Text features -

Pre-Processing

- Standardization
  - Some algorithms make assumptions about the data
    - SVM - assume that all features are centered around zero and have the same variance.
  - Standard normally distributed data (the bell curve)
- Scaling Features
- Normalization
- Binarization



Training vs Test Data

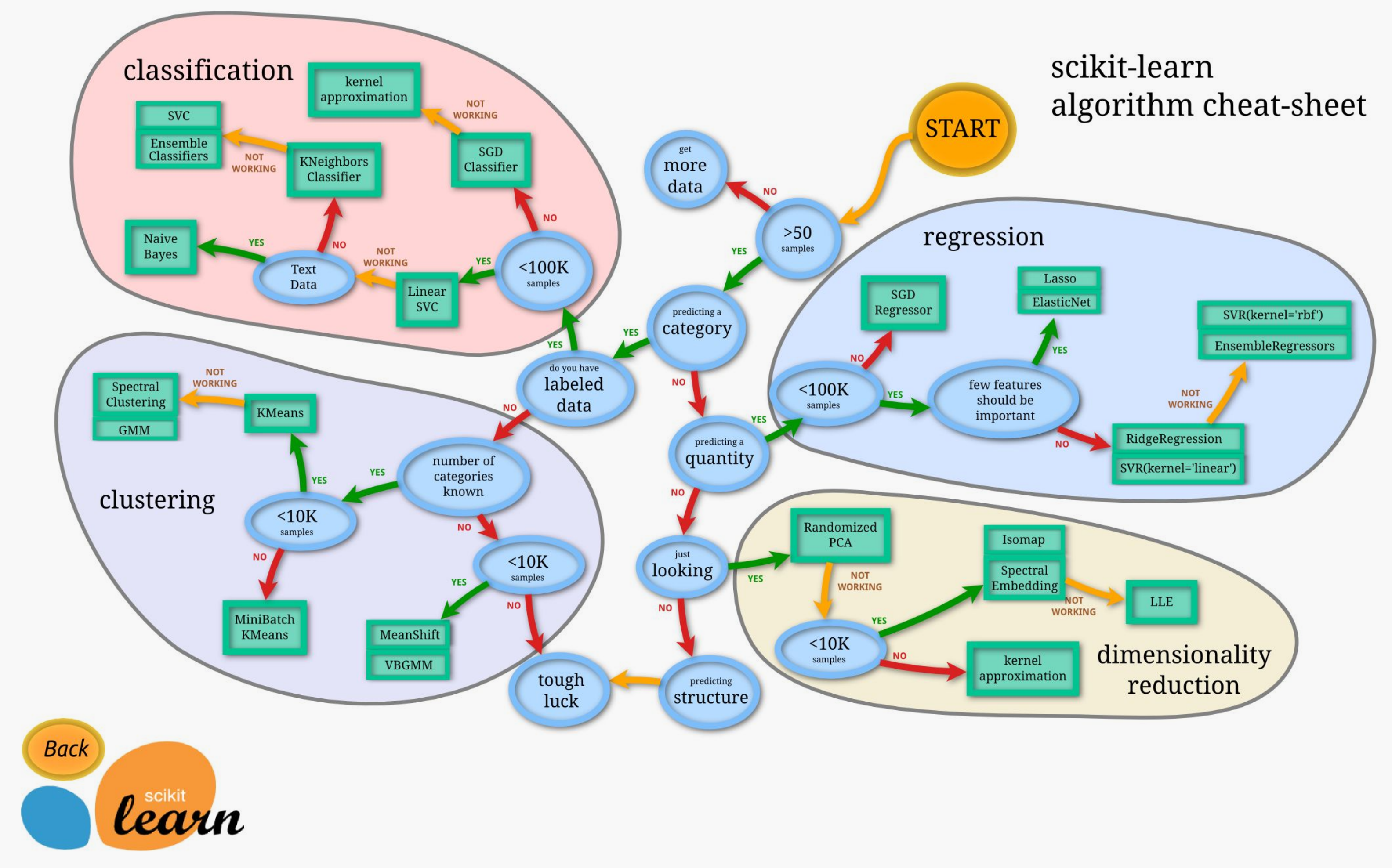
“The fundamental goal of ML is to generalize beyond the data instances used to train models.”

Training Data (70% - 80%) - Use this data to train the model.

Test Data (20% - 30%) - Use this data to evaluate the model.

- When splitting data, keep in mind that you want to accurately express your dataset in both your training and test dataset.
- Cross-Validation

http://docs.aws.amazon.com/machine-learning/latest/dg/splitting-the-data-into-training-and-evaluation-data.html


scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
NO
YES
predicting a category
do you have labeled data
YES
NO
<100K samples
SGD Classifier
NOT WORKING
kernel approximation
Linear SVC
NOT WORKING
Text Data
YES
Naive Bayes
NO
KNeighbors Classifier
NOT WORKING
SVC
Ensemble Classifiers
classification
number of categories known
<10K samples
KMeans
NOT WORKING
Spectral Clustering
GMM
MiniBatch KMeans
MeanShift
VBGMM
clustering
tough luck
predicting a quantity
<100K samples
SGD Regressor
few features should be important
Lasso
ElasticNet
RidgeRegression
SVR(kernel='linear')
SVR(kernel='rbf')
EnsembleRegressors
regression
just looking
predicting structure
Randomized PCA
Isomap
Spectral Embedding
LLE
kernel approximation
dimensionality reduction
Back
scikit learn

Algorithm - Train model

- Several different algorithms to choose from to model your data.
  - Few different options:
    - Implement the algorithm yourself.
    - Use an existing implementation.

```
>>> from sklearn import svm
>>> from sklearn import datasets
>>> clf = svm.SVC()
>>> iris = datasets.load_iris()
>>> X, y = iris.data, iris.target
>>> clf.fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

Algorithm - Evaluate model

```
>>> import pickle
>>> s = pickle.dumps(clf)
>>> clf2 = pickle.loads(s)
>>> clf2.predict(X[0:1])
array([0])
>>> y[0]
0
```

Results

The sklearn.metrics module includes score functions, performance metrics and pairwise metrics and distance computations.

Model Selection Interface

See the The scoring parameter: defining model evaluation rules section of the user guide for further details.

metrics.get_scorer (scoring)	Get a scorer from string
metrics.make_scorer (score_func[, …])	Make a scorer from a performance metric or loss function.

Classification metrics

See the Classification metrics section of the user guide for further details.

metrics.accuracy_score (y_true, y_pred[, …])	Accuracy classification score.
metrics.auc (x, y[, reorder])	Compute Area Under the Curve (AUC) using the trapezoidal rule
metrics.average_precision_score (y_true, y_score)	Compute average precision (AP) from prediction scores
metrics.brier_score_loss (y_true, y_prob[, …])	Compute the Brier score.
metrics.classification_report (y_true, y_pred)	Build a text report showing the main classification metrics
metrics.cohen_kappa_score (y1, y2[, labels, …])	Cohen's kappa: a statistic that measures inter-annotator agreement.
metrics.confusion_matrix (y_true, y_pred[, …])	Compute confusion matrix to evaluate the accuracy of a classification
metrics.f1_score (y_true, y_pred[, labels, …])	Compute the F1 score, also known as balanced F-score or F-measure
metrics.fbeta_score (y_true, y_pred, beta[, …])	Compute the F-beta score
metrics.hamming_loss (y_true, y_pred[, …])	Compute the average Hamming loss.
metrics.hinge_loss (y_true, pred_decision[, …])	Average hinge loss (non-regularized)
metrics.jaccard_similarity_score (y_true, y_pred)	Jaccard similarity coefficient score
metrics.log_loss (y_true, y_pred[, eps, …])	Log loss, aka logistic loss or cross-entropy loss.
metrics.matthews_corrcoef (y_true, y_pred[, …])	Compute the Matthews correlation coefficient (MCC)
metrics.precision_recall_curve (y_true, …)	Compute precision-recall pairs for different probability thresholds
metrics.precision_recall_fscore_support (…)	Compute precision, recall, F-measure and support for each class
metrics.precision_score (y_true, y_pred[, …])	Compute the precision
metrics.recall_score (y_true, y_pred[, …])	Compute the recall
metrics.roc_auc_score (y_true, y_score[, …])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
metrics.roc_curve (y_true, y_score[, …])	Compute Receiver operating characteristic (ROC)
metrics.zero_one_loss (y_true, y_pred[, …])	Zero-one classification loss.

Demo

Tutorials

http://scikit-learn.org/stable/tutorial/basic/tutorial.html

https://www.tensorflow.org/get_started/get_started

http://docs.aws.amazon.com/machine-learning/latest/dg/building-machine-learning.html

https://www.kaggle.com/c/titanic

Want to learn more?

https://openai.com/

https://www.tensorflow.org/

https://ipython.org/notebook.html

https://www.kaggle.com/

Jason Mayes - Machine Learning 101

Brush up on your math skills -> Statistics / Linear Algebra / Calculus

Questions????????

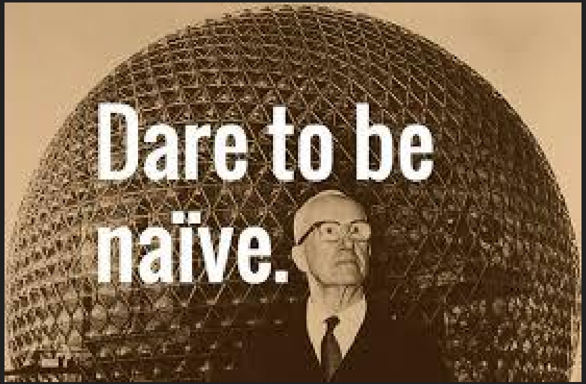
Extras

Machine Learning Tool Basics

IPython Notebook

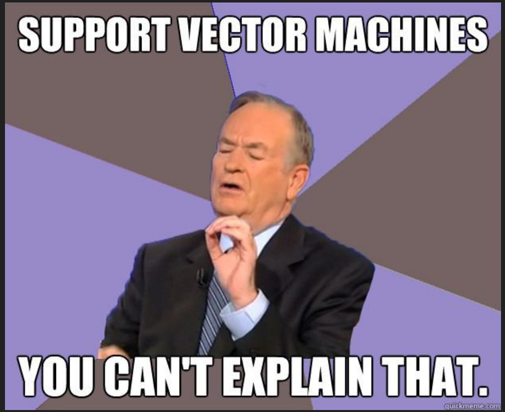
Python / Pandas

Naive Bayes



Random Forest

Support Vector Machine



K-Nearest Neighbors

ALPHAZERO
VS
STOCKFISH
Chess.com

Real World Applications


WORLD
1-1
TIME
389
x00

Applications

AlphaGo

BLIZZARD
ENTERTAINMENT

DeepMind