WAT.ai

CLASSICAL MACHINE LEARNING

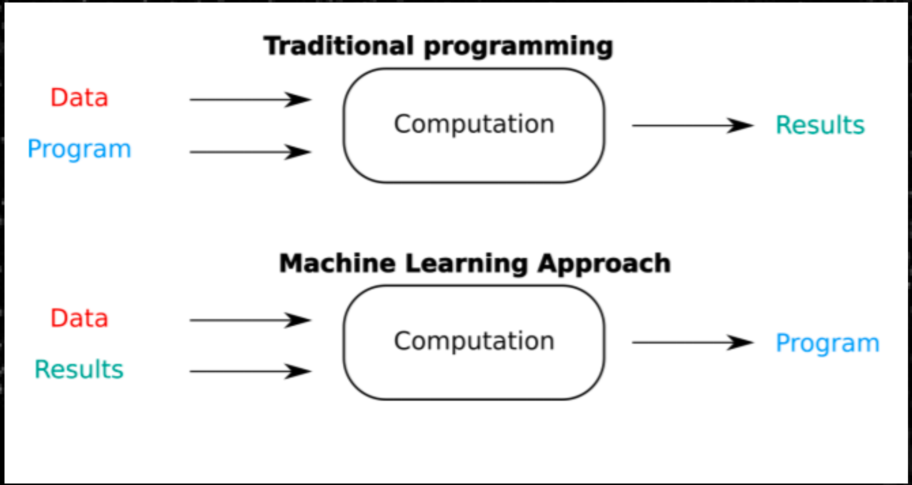
TREVOR YU & CARTER DEMARS

TODAY'S AGENDA

- Introduce the scikit-learn API and common practices in the field of machine learning.
- Provide intuition for various classical machine learning techniques regarding their complexity, performance, and effectiveness in the context of different applications.
- Explore concepts such as model selection, hyperparameter tuning, performance metrics, and the bias/variance trade-off.
- Apply this knowledge to a real-world dataset in a competition-style activity.

WHY LEARN ABOUT CLASSICAL ML?

- Statistical learning allows you to solve a different class of problems than traditional computing.

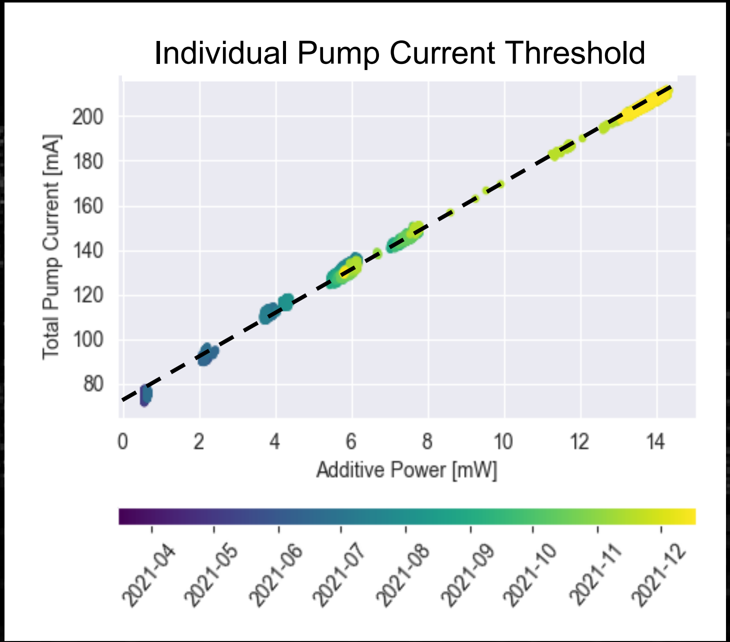


- Traditional computing is concerned with obtaining results from a set of inputs and instructions.
- Broadly speaking, classical machine learning seeks to estimate a function that maps predictors (X) to response variables (y).

OK BUT WHY NOT DEEP LEARNING?

- Sometimes it's not that deep (pun intended)
- Deep learning requires lots of data
- Deep learning models are expensive to train and challenging to run in production

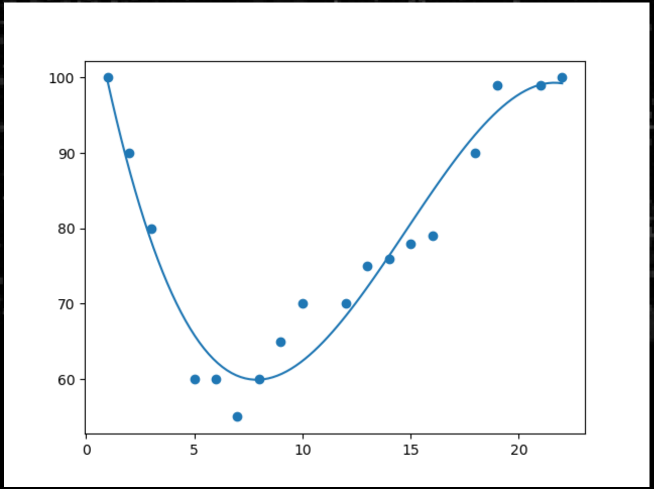
- Classical ML can still outperform Deep Learning for many tasks
    - Tree-based models like XGBoost routinely outperform deep learning models in machine learning competitions.
    - Why use deep learning when a simpler model will suffice?



COMMON MACHINE LEARNING TASKS

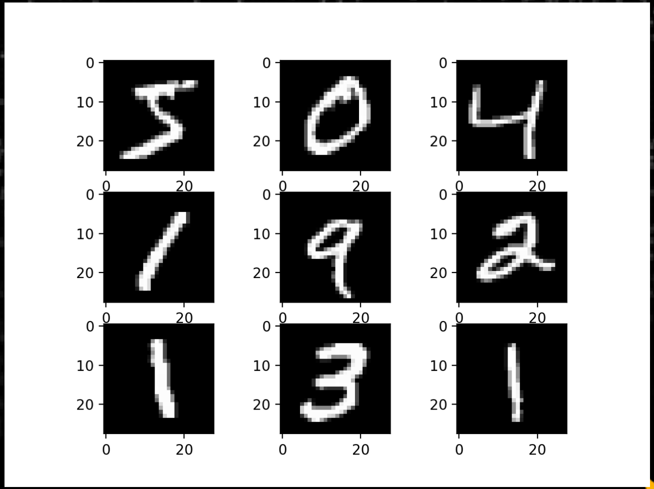
REGRESSION

- Prediction/inference for a quantitative (often continuous) response variable



CLASSIFICATION

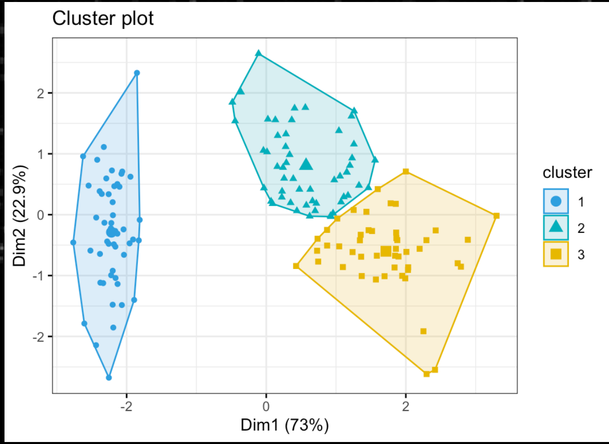
- Prediction/inference for qualitative (often discrete) response variable



COMMON MACHINE LEARNING TASKS

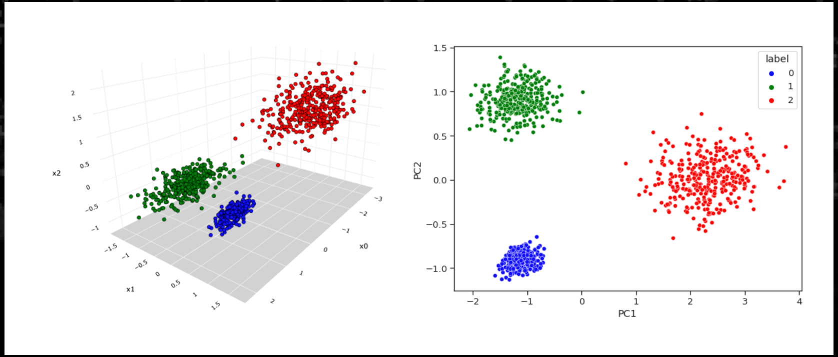
CLUSTERING

- Grouping unlabeled data such that similar inputs fall into the same “cluster”



DIMENSIONALITY REDUCTION

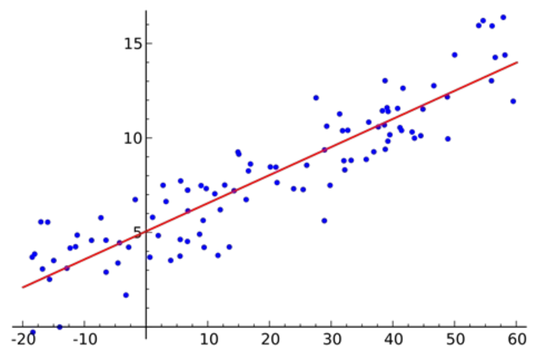
- Transforming data from a high-dimensional space to a low-dimensional space while maintaining as much information (variation) as possible



TYPES OF MODELS

PARAMETRIC MODELS

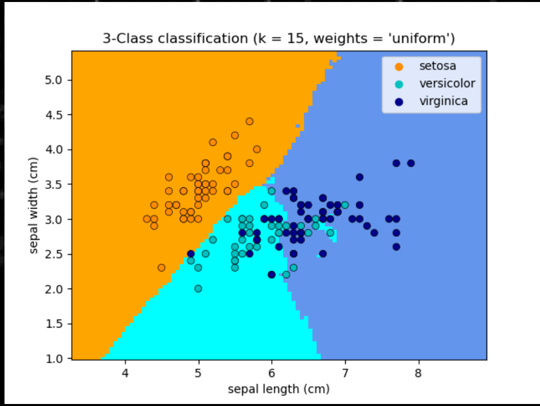
1. Assume the shape of *f(x)*
2. Fit a model with parameters that best estimate the true function



Ex: linear regression, kernel regression

NON-PARAMETRIC MODELS

1. Make no explicit assumptions about the shape of *f(x)*
2. Estimate *f* by fitting a function as close to the data points as possible without overfitting



Ex: k-nearest neighbours, support vector machines, certain tree-based models

MODEL SELECTION: HOW DO I CHOOSE?

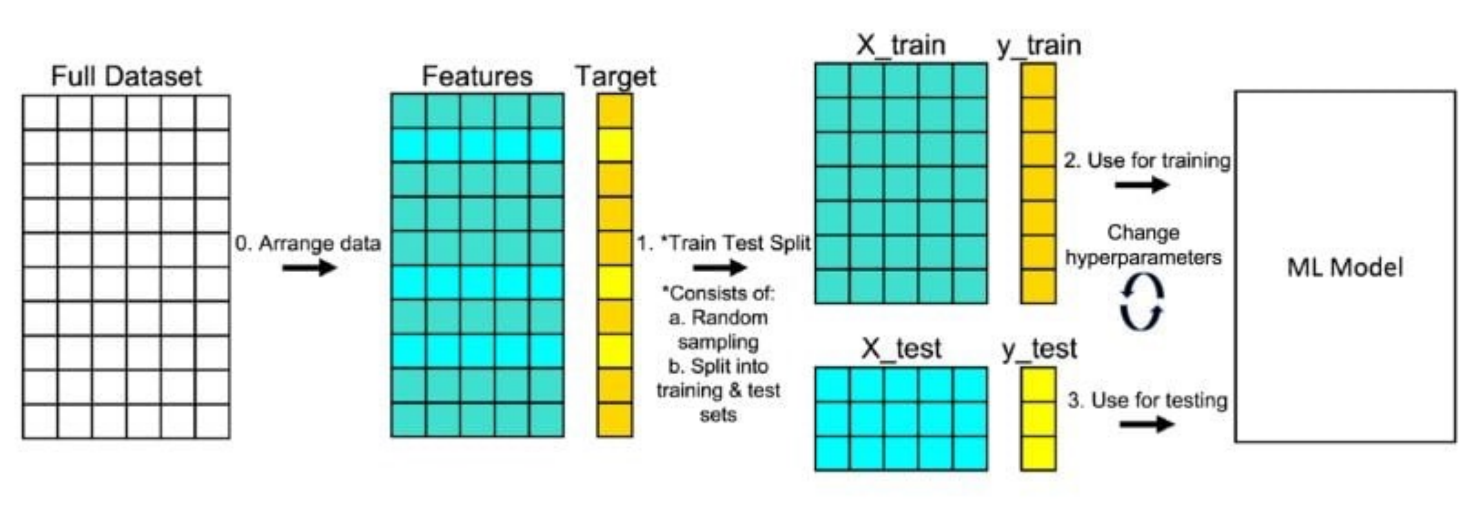
- Highly dependent on your task and your goals

Some things to consider when narrowing down your options:

1. What machine learning task am I trying to solve?
2. How much data do I have?
3. How is the data structured? How can the data be encoded?
4. Do I have intuition for the shape of the function I am trying to estimate? (parametric vs. nonparametric)
5. Are there other constraints on my model? Does it need to run inference in real-time?

MODEL SELECTION: TRAIN-TEST SPLIT

- To adequately compare models, you'll need to quantify each model's performance on unseen data
- Prior to training, the full dataset is split into a training set and a test set (and a validation set if used in production) through a process called train-test split



MODEL SELECTION: HYPERPARAMETERS

- Models can't always be compared directly, because there are many variable hyperparameters
- Hyperparameters are parameters whose value is generally set before model training
- Two models from the same model family could have wildly different accuracy based on the hyperparameters they are trained with
- Different classes of models will have unique hyperparameters

ASSESSING MODEL ACCURACY

- In regression, a commonly-used measure of accuracy is mean squared error (MSE)
- In a classification-setting, accuracy or error-rate is used

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$$

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i).$$

where n is the number of data points, y_i are the true values, and $\hat{f}(x_i)$ are the estimated values.

Task	Metrics	Alternatives
Regression	Mean squared error	MAE, RMSE,
Binary classification	Accuracy	F1, precision, recall
Multiclass classification	Accuracy	Micro/macro F1, per-class precision and recall

MACHINE LEARNING THEORY – MATHEMATICS INCOMING!

1. OVERFITTING VS. UNDERFITTING
2. BIAS-VARIANCE TRADE-OFF

OVERFITTING

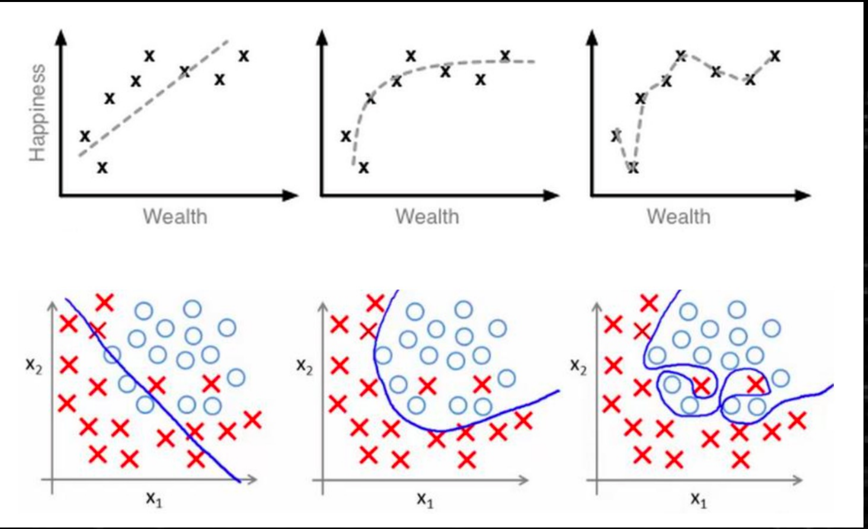
- When a given model yields a small error on the training data, but a large test error, we say that the model is overfit to the training data

- This happens when our model fits to patterns that are caused by noise/randomness in the data, rather than the properties of the true function

“Underfit” “Overfit”

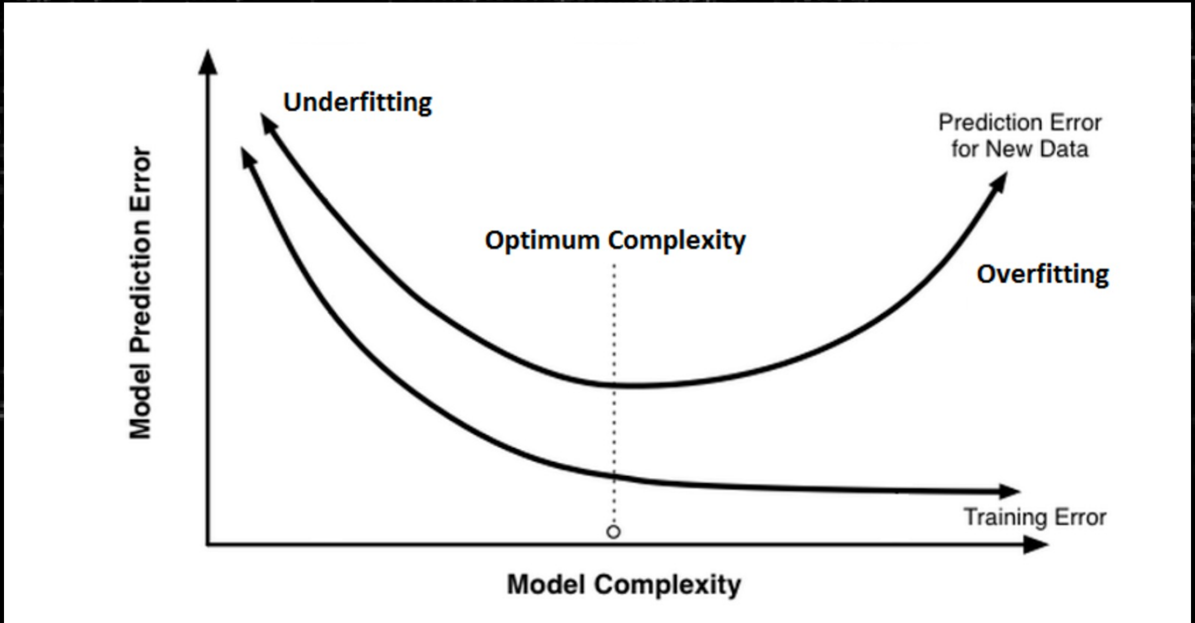
Regression

Classification



OVERFITTING

- Sometimes we can fix overfitting by adjusting model hyperparameters during training.
- Other times, it is due to poor model selection, and a different class of models might work better.



BIAS-VARIANCE TRADE-OFF

BIAS

- Bias is the error introduced by approximating a real-life problem with a simpler model
- The error due to bias is the difference between the average prediction of our model and the correct value which we are trying to predict

$$Err_{bias} = \frac{1}{m}\sum_{i=1}^{m}\left(E[\hat{f}(x_i)] - f(x_i)\right) = E_X[E[\hat{f}(x)] - f(x)]$$

VARIANCE

- Variance is the amount by which $\hat{f}$ would change if it were estimated using a different training set. If a model has high variance, small changes to the training data result in large changes to the function $\hat{f}$.

$$Err_{var} = Var\left(\hat{f}(x)\right) = E\left[\hat{f}(x) - E[\hat{f}(x)]\right]^2$$

BIAS-VARIANCE TRADE-OFF – LET'S PROVE IT!

Our function y $y = \hat{f}(x) + \varepsilon$

Mean Squared Error (MSE) $Err(x) = E\left[\left(y - \hat{f}(x)\right)^2\right]$

Formula for variance $Var\left(\hat{f}(x)\right) = E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] = E[\hat{f}(x)^2] - E[\hat{f}(x)]^2$

$$E[\hat{f}(x)^2] = E[\hat{f}(x)]^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right]$$

Isolate for $E[\hat{f}(x)^2]$ in variance formula

$$Err(x) = y^2 - 2yE[\hat{f}(x)] + E[\hat{f}(x)^2]$$

Expand MSE formula

$$Err(x) = y^2 - 2yE[\hat{f}(x)] + E[\hat{f}(x)]^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right]$$

Substitute into MSE formula

$$Err(x) = \left(E[\hat{f}(x)] - y\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right]$$

Complete the square

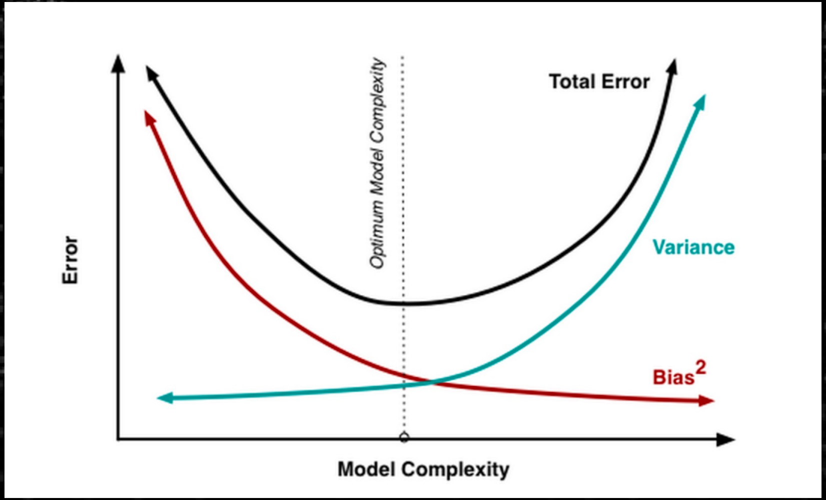
$$Err(x) = Bias^2 + Variance$$

Q.E.D.

BIAS-VARIANCE TRADE-OFF

The U-shaped total error curve is the result of two competing properties of machine learning models: bias and variance

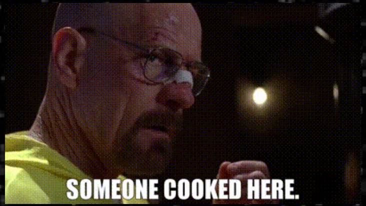
To minimize the expected test error, we want select a machine learning technique and suitable hyperparameters that simultaneously achieve low bias and low variance

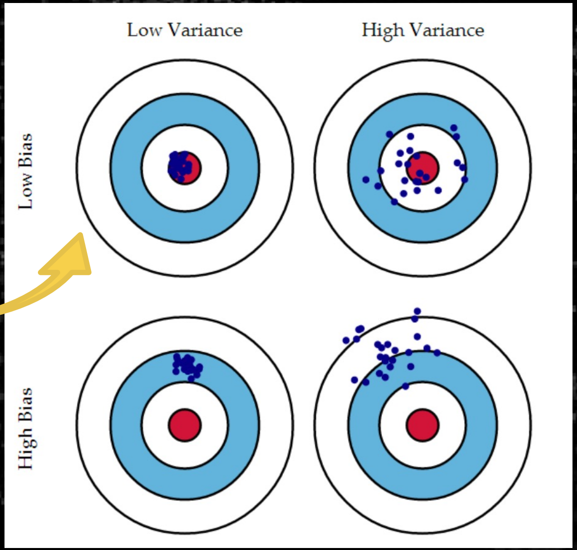


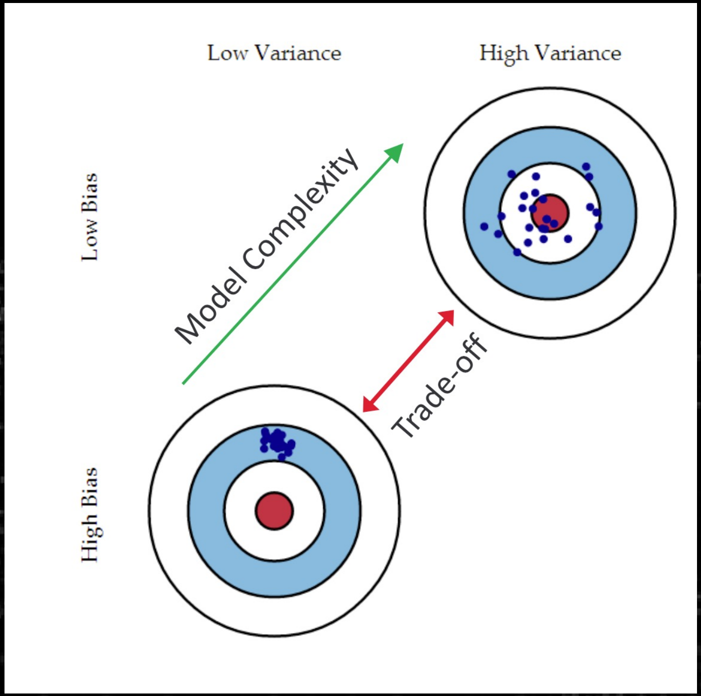
BIAS-VARIANCE TRADE-OFF

Ideal case, rarely occurs

Potentially overfit



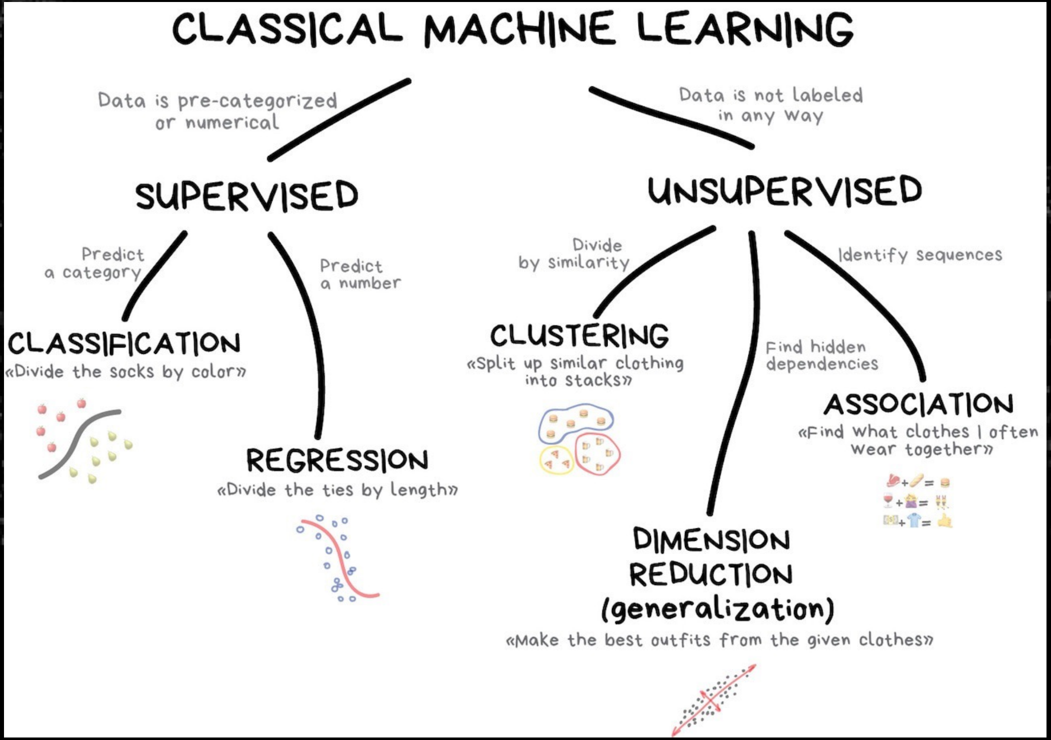




Potentially underfit

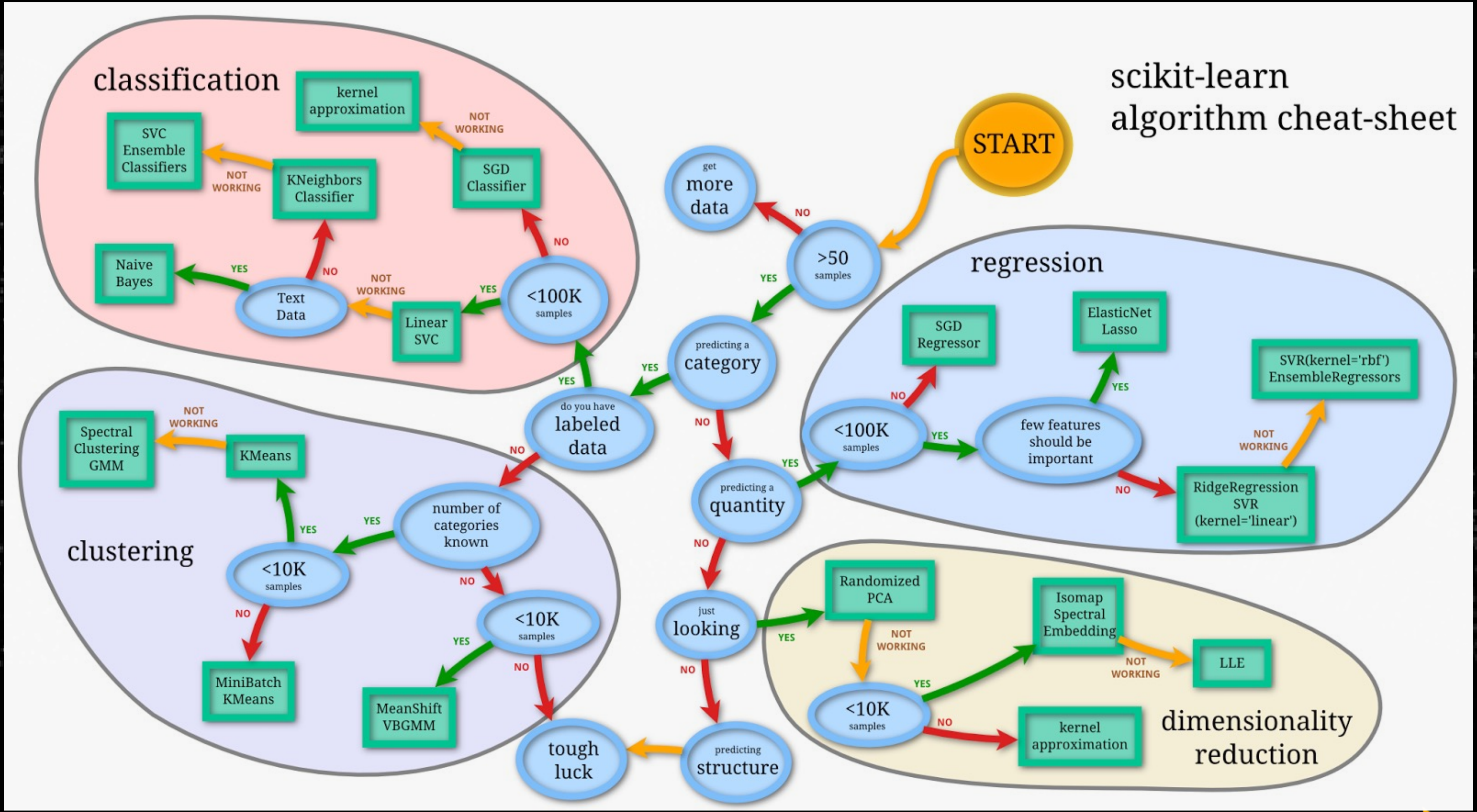
Complex model, awful accuracy, throw it away and try something else

Q&A SESSION – ASK US ANYTHING



SCIKIT LEARN DEMO

- Train-test split
- Training a logistic regression model
- Evaluating model performance



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

COMPETITION TIME

UPCOMING EDUCATION SESSIONS

- Neural Networks for Novices
- Dive into Deep Learning
- Discord:
- https://discord.gg/46KUMNGE8J
- Will post recordings, slides, etc.

EXIT SURVEY – ATTENDANCE!