FLATIRON SCHOOL

Optimizing Outcomes with Model Evaluation & Tuning Techniques.

hosted by KASH

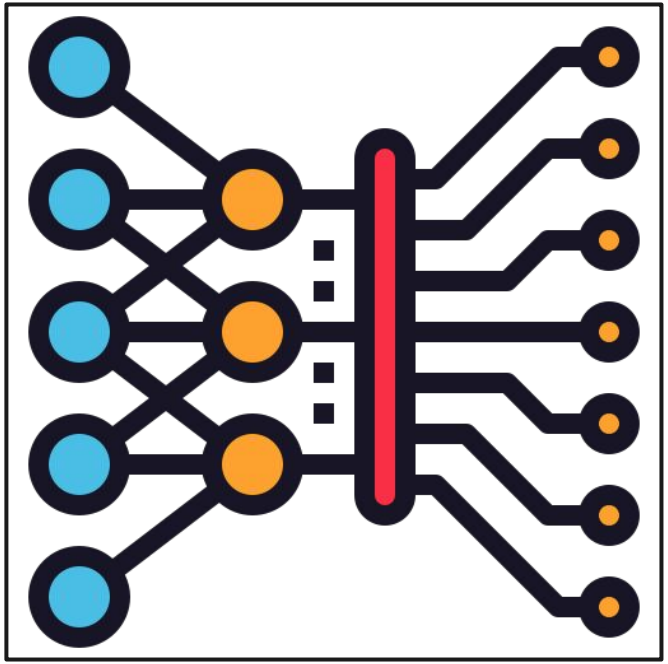
CRITICAL QUESTION

How can we competently evaluate and improve any model's performance while extending our development pipeline?

THE NATURE OF A PREDICTIVE MODEL

Many inexperienced data scientists tend to think of machine learning algorithms as **highly robust** and **immune to external changes** – the data science equivalent of "*it is what it is*."

However, algorithms are code, and **code can be altered**; as such, there exist many methods and techniques for tuning and evaluating algorithmic ability that experienced modeling experts use regularly and wisely as part of their development arsenal.



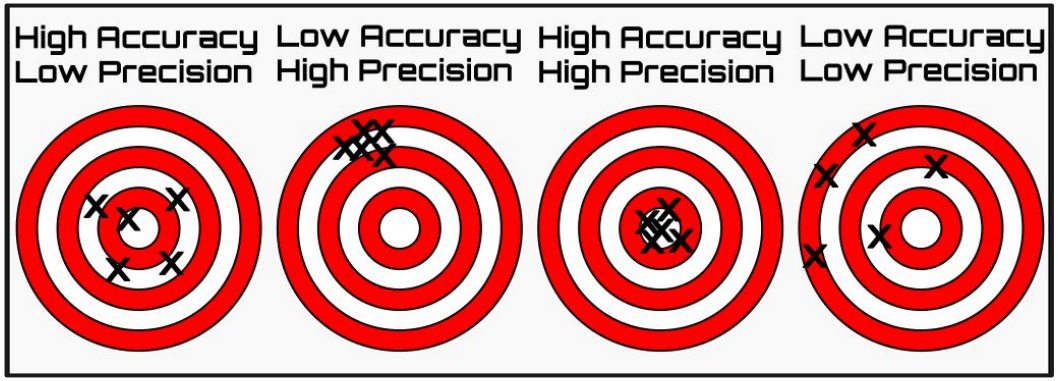
//

WRITTEN IDEATION

HOW CAN WE BE CERTAIN THAT OUR **ACCURACY SCORES** AND OVERALL **PERFORMANCE METRICS** ARE **RELIABLE**?

MODELING OUTCOMES: BALANCING ACCURACY AND ERROR

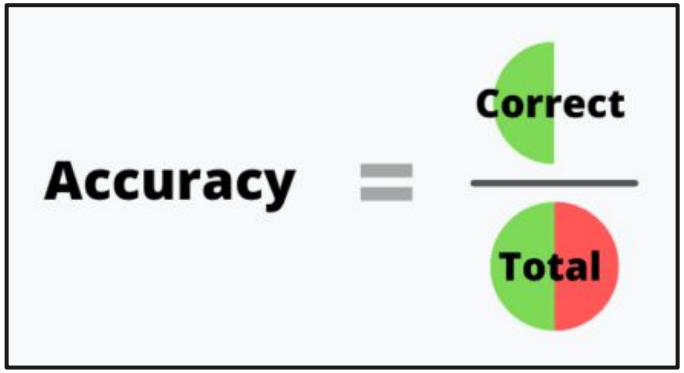
Understanding accuracy is tantamount to understanding **how effective a model is at capturing heuristics across a dataset** – while relying on standard accuracy is mostly definitive in depicting algorithmic fitness, knowing the intricacies of **how accuracy and error are related** can unveil new and more nuanced techniques for analyzing and assessing a model.



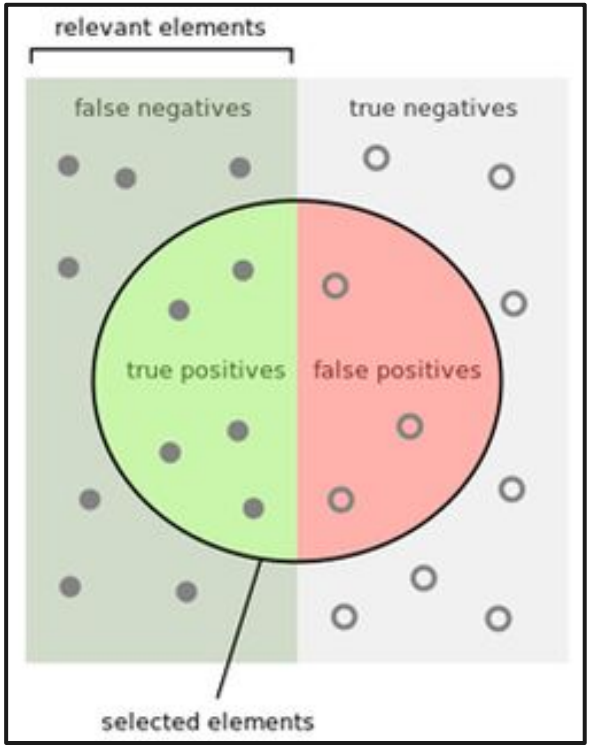
ACCURACY MEASURES: STANDARD ACCURACY

Standard accuracy – a.k.a. the accuracy score that is often utilized in standard modeling demonstrations – is calculated very simply by **analyzing the ratio of correctly predicted labels** (true positives and true negatives) **to all predicted labels**, including both correct and incorrect assessments.

While this works well for generally evaluating how "correct" an algorithm is, it actually doesn't show that much as to how "incorrect" it was.



ACCURACY MEASURES: THE PRECISION AND RECALL SCORES



To better understand an algorithm's overall effectiveness, we often make use of the **precision** and **recall** scores.

Precision calculates the **ratio of correctly predicted positives to all predicted positives**, while recall calculates the **ratio of correctly predicted positives to all actual positives**.

Both scores are highly similar but measure slightly distinct measures for inaccuracy in our model.

How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

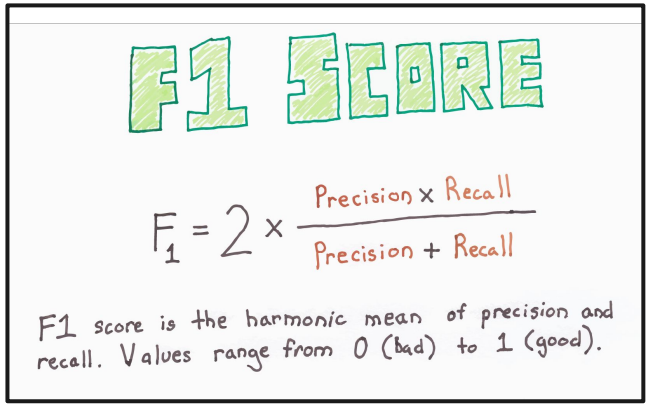
How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

ACCURACY MEASURES: THE F_1 SCORE

By **combining our precision and recall scores into one composite score**, we can induce a modified score that serves as a "weighted index" of both accuracy and inaccuracy across a model's produced predictions.

This weighted score – referred to as the **F1 score** – is often regarded as an improved accuracy score and generally trusted as a means of better understanding **how well our model performed for both positive and negative assessments**.

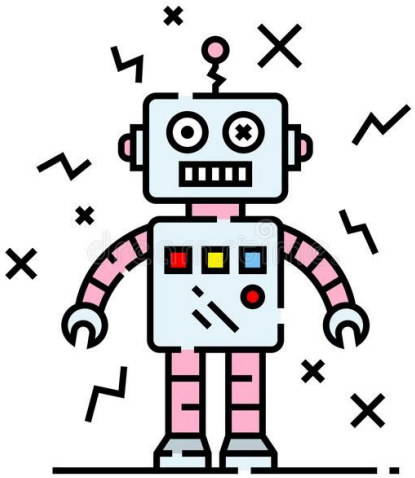


//

WRITTEN IDEATION

ARE SOME **ERRORS** BETTER THAN OTHERS? IS IT THAT SIMPLE?

MODELING OUTCOMES: ASSESSING CLASSIFICATION ERRORS



Even after understanding how to programmatically detect and compute errors, there's still much to be done to further study and improve upon them, and that starts with understanding **what types of errors can occur after the modeling pipeline has been constructed**.

Regression models bear their own suite of specific errors as well as accuracy measures, which we'll introduce during our relevant introduction to regression algorithms.

CLASSIFIER ERROR ASSESSMENT: TYPE-1 AND TYPE-2 ERRORS

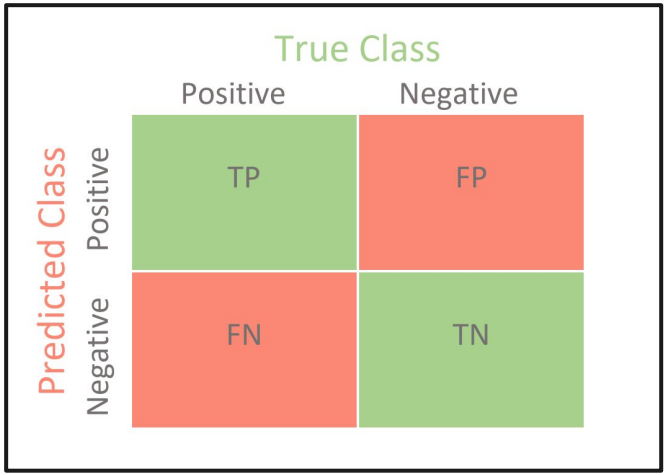
To start, the worth of **specific types of errors is highly dependent on our domain and data-centric task** – however, most tasks display some accumulation of Type-1 and Type-2 errors.

Type-1 errors indicate false positive rates (rejection of the null hypothesis when it is actually true) while **Type-2 errors indicate false negative rates** (failure to reject the null hypothesis when it is actually false).

CLASSIFIER ERROR ASSESSMENT: THE CONFUSION MATRIX

One of the most reliable techniques to visualize and assess the occurrence of label-based errors across a model is the **confusion matrix**, which charts class accuracies and errors in a grid-like manner.

This matrix generally follows the same patterns in design and development: the **upper-left-to-lower-right diagonal indicates the general accuracies across your model**, while all other squares display inappropriate class assignments as errors.



CLASSIFIER ERROR ASSESSMENT: THE CLASSIFICATION REPORT

Classification reports are also highly useful for further understanding and communicating in-depth accuracy information for a performant model.

These reports can be manually generated and are highly configurable based on what specific information you want to compose; in most examples, it's most important to curate **accuracy metrics**, **sample sizes**, and **class distributions**.

	precision	recall	f1-score	support
0	0.77	0.86	0.81	37584
1	0.84	0.75	0.79	37577
accuracy			0.80	75161
macro avg	0.81	0.80	0.80	75161
weighted avg	0.81	0.80	0.80	75161

//

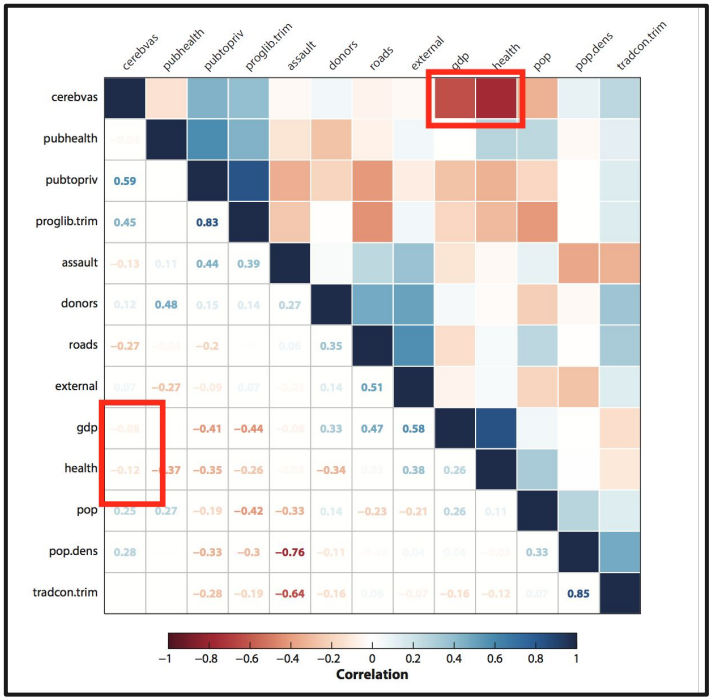
WRITTEN IDEATION

HOW CAN WE BETTER UNDERSTAND HOW THE **MODEL** ACTUALLY **LEARNS RELEVANT SIGNALS** ACROSS OUR DATA?

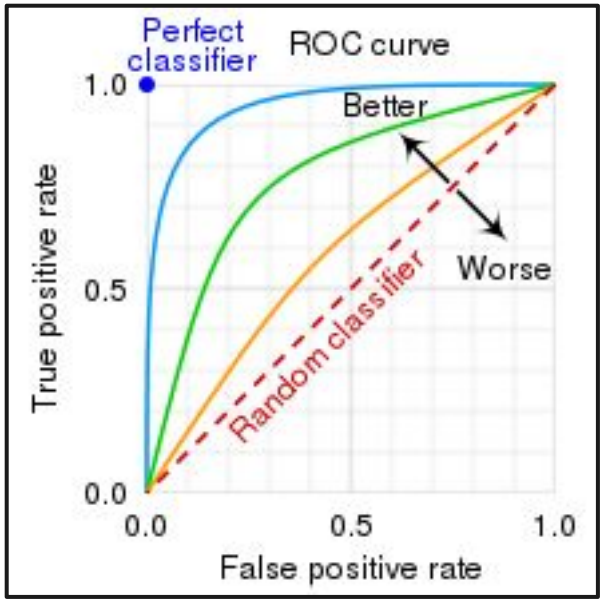
MODELING OUTCOMES: VISUALIZING ALGORITHMIC FITNESS

As we know from previous courses, data visualization is a powerful technique to summarize and adequately showcase patterns and behaviors of our data that can otherwise be obscured – the same holds for **visualizing accuracy, error, and overall performance**.

While methods like confusion matrices do count as visualization, other techniques exist to more clearly depict a **model's demonstrable "learning rate" and fitness**.



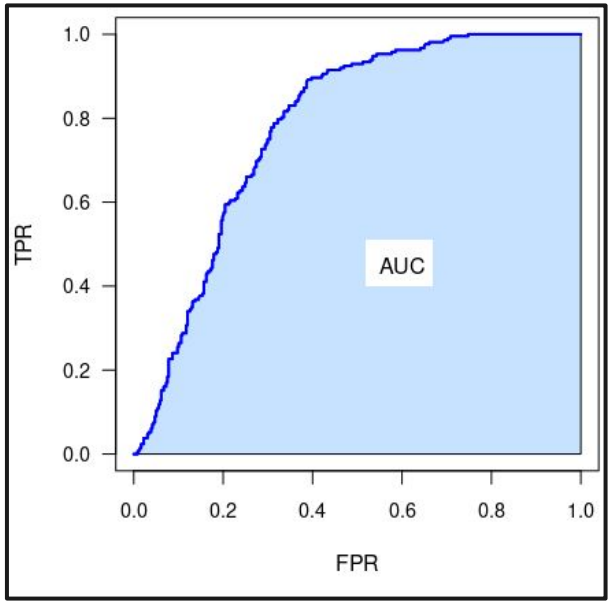
VISUALIZING MODEL FITNESS: THE R.O.C. CURVE



The **receiver-operating characteristic** is a reliably powerful method of depicting a model's learning rate and ability to extract useful heuristics from the data.

The chart displays **error rate ratio** (true positives against false positives) in a manner that visually showcases how fast and well a model is able **to optimize general accuracy during the training segment of model fitness**.

VISUALIZING MODEL FITNESS: THE A.U.C. CURVE



Interestingly, the **area under the ROC curve** depicts even more useful information – the ability of our algorithm **to partition and induce differences across all classes** from our data.

The higher our AUC score is, the better our model's **"separability"**, and the lower our relative frequency of Type-1 and Type-2 errors.

//

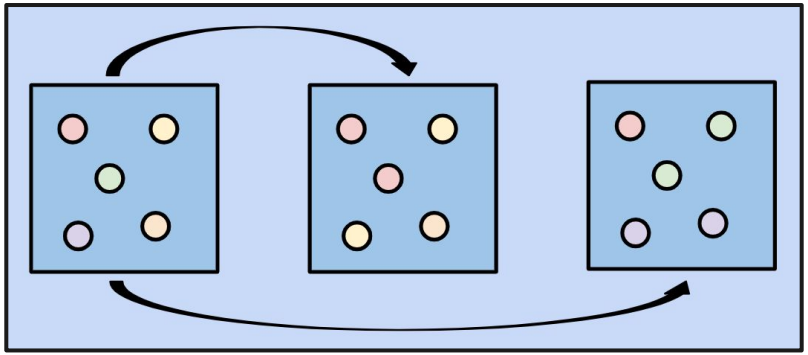
WRITTEN IDEATION

WHAT HAPPENS IF OUR MODEL IS **OVERFITTING TO SPECIFIC CLASSES** IN THE DATA? HOW CAN WE **FIX IT**?

MODELING OUTCOMES: DATA RESAMPLING

Many times when working with real-world and unideal data, we may find that **class imbalances** plague our dataset's overall distribution – unfortunately, this is a pertinent issue as **it could lead to overfitting or underfitting** of our model.

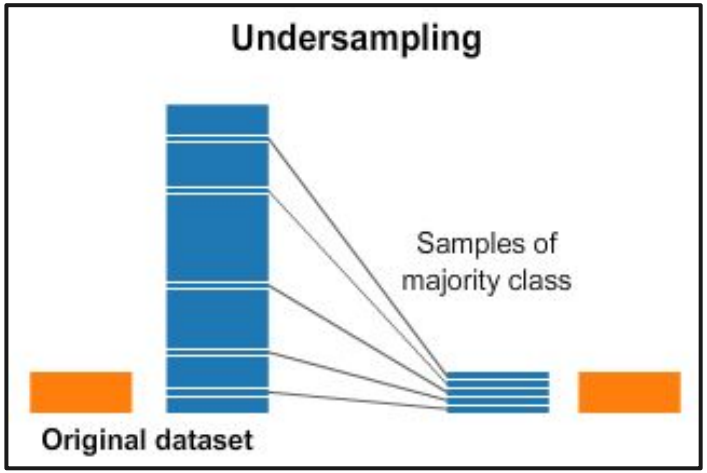
We can correct for class imbalances and improve performance by attempting **to resample and/or rebalance our classes** across our data.



DATA RESAMPLING: DATA UNDERSAMPLING

One straightforward technique is to **undersample our "majority class(es)"**; in other words, classes that are much more populous across our dataset are sampled from and reduced in quantity until all classes are equally represented.

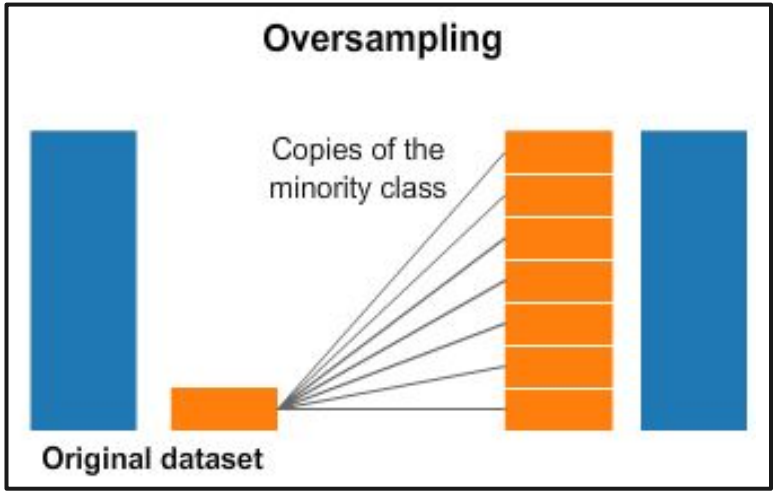
This does obviously lose some data and **run the risk of losing valuable signal**, which is why undersampling should be done carefully with programmed stochasticity and actively reduced bias.



DATA RESAMPLING: DATA OVERSAMPLING

On the other hand, we can also **oversample the "minority class(es)"**; in other words, classes that are much more sparse across our dataset are sampled from and inflated in quantity until all classes are equally represented.

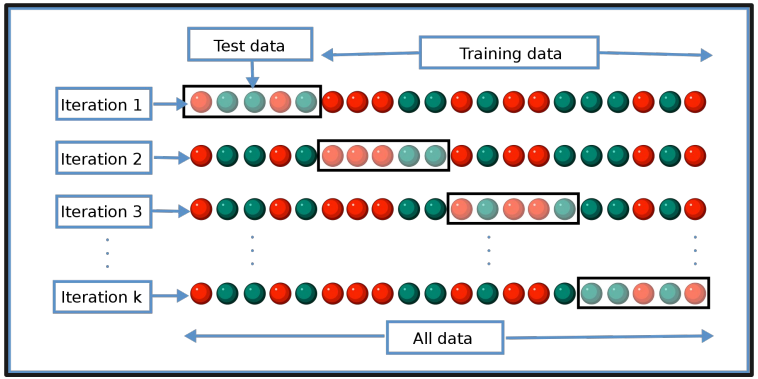
This can also lead to **signal reduction through the form of addition of noise** to the dataset – duplicates can overwhelm our model, which is why this must also be conducted with programmed randomness and care.



DATA RESAMPLING: CROSS VALIDATION

One highly reliable technique that avoids the bloat of resampling is **“cross validation”**, which performs several iterations of model fitting by **taking randomly sampled and independent segments of our overall data as training and testing splits**.

These **“folds”** of splits likely contribute to variances in our accuracy, which is why **cross-validated model performances are averaged** upon completion.

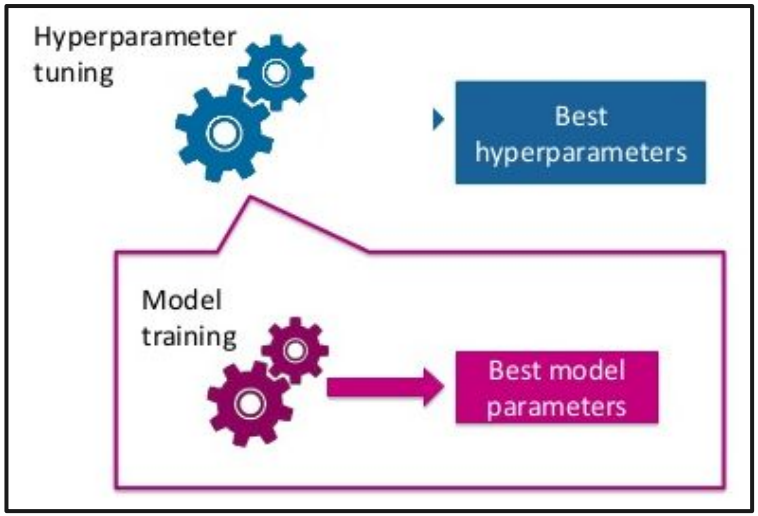


//

WRITTEN IDEATION

CAN WE **MODIFY THE MODELS THEMSELVES** TO MAKE AN IMPACT ON **PERFORMANCE**? IF SO, HOW?

MODELING OUTCOMES: HYPERPARAMETER TUNING



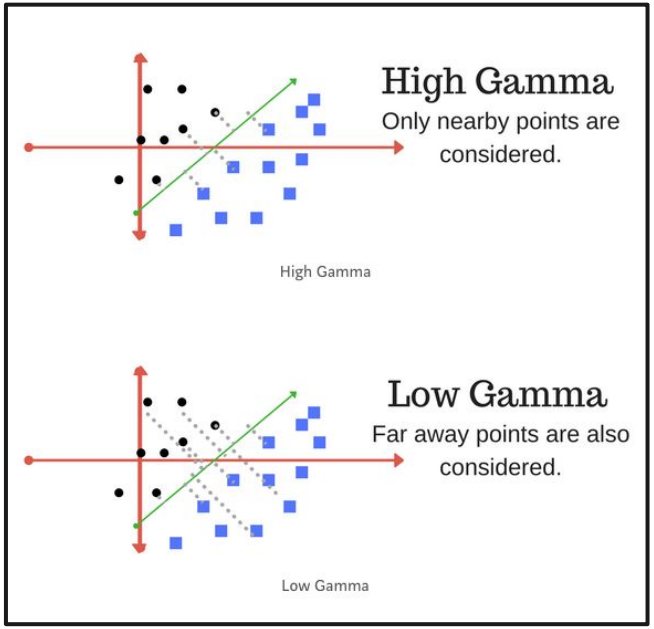
Finally, our models can simply be improved by **curating specific arguments and parameters that impact the way the model actually works** – this becomes incredibly model-specific as you can imagine, but is undoubtedly one of the most powerful ways to tune accuracy and performance.

Specifically impactful variables that heavily alter an algorithm's in-depth function are referred to as **"hyperparameters"**.

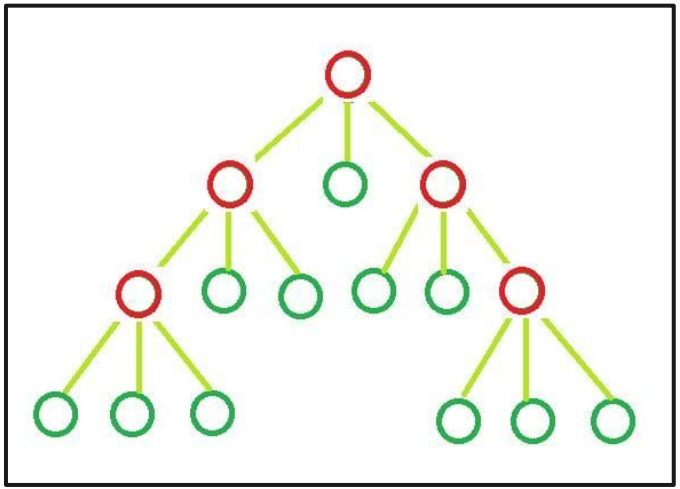
HYPERPARAMETER TUNING FOR INSTANCE-BASED MODELS

Understanding **how hyperparameters relate to specific types of algorithms** can both empower your data science arsenal while also unveiling the secrets of how specific models function as mathematical constructs.

For instance (get it?), **instance-based model hyperparameters** tend to affect the **distance calculations**, **number of neighbors/support vectors**, and **overall expected margin** where the decision boundary is drawn.



HYPERPARAMETER TUNING FOR DECISION TREE MODELS

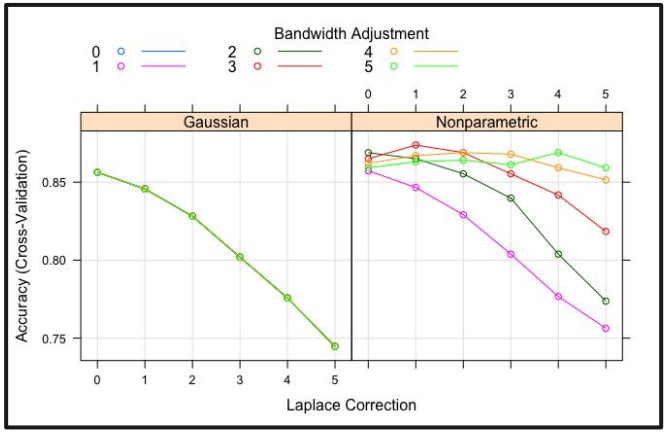


Decision tree hyperparameters tend to affect mathematical and programmatic properties such as the **number of splits per level**, **allowable number of nodes per split**, specific methods or constraints for **how to make a decision split**, and the **overall breadth and depth** of the tree as a whole.

HYPERPARAMETER TUNING FOR BAYESIAN MODELS

Bayesian model hyperparameters are much more nuanced as they tend to be statistical and probabilistic properties and can mandate a much more thorough understanding of the relevant mathematics.

However, some of the most easily interpretable and mutable hyperparameters include **smoothing of the distribution** and **added noise to avoid zero-based probabilities of classes**.



THANK YOU FOR YOUR TIME!