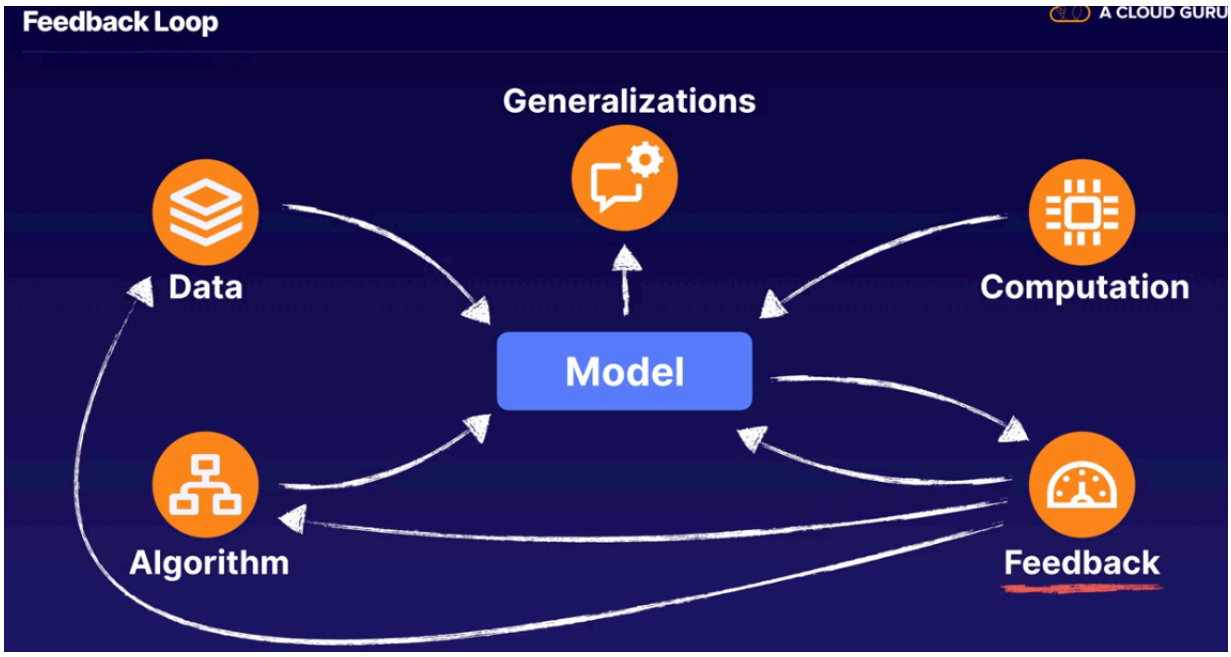
Cloud Guru - 7 - Evaluation and Optimization

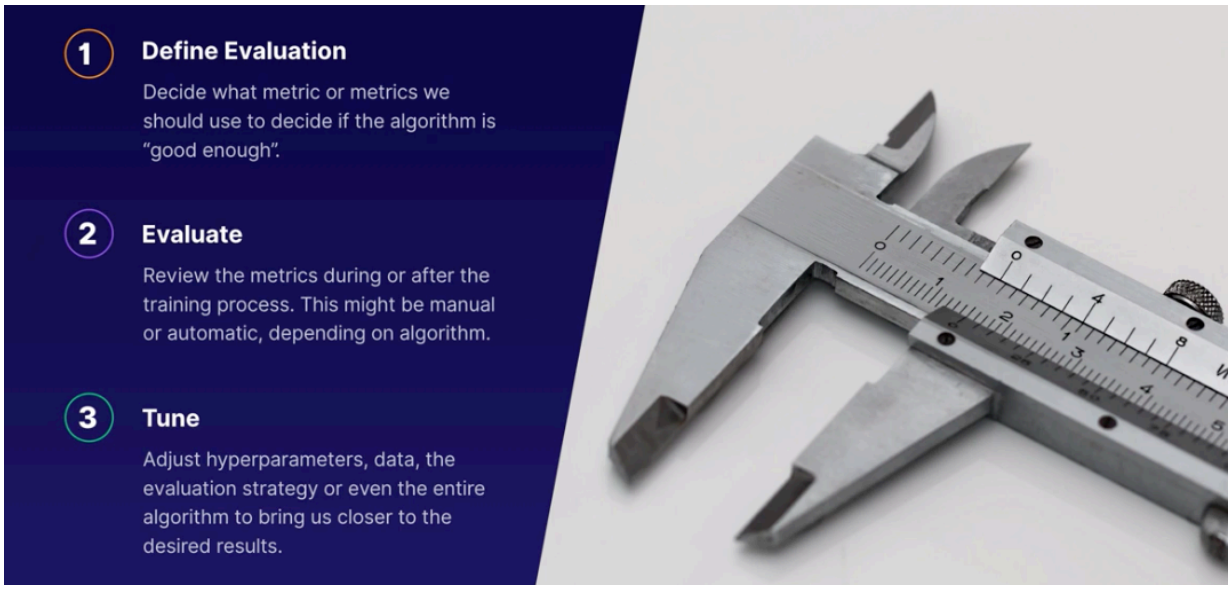
https://learn.acloud.guru/course/aws-certified-machine-learning-specialty/learn/c865bf80-7377-1ff0-4077-ff6cb94bf923/77f1dce7-6d2f-d2d4-3748-c16406b26606/watch

ML Objective

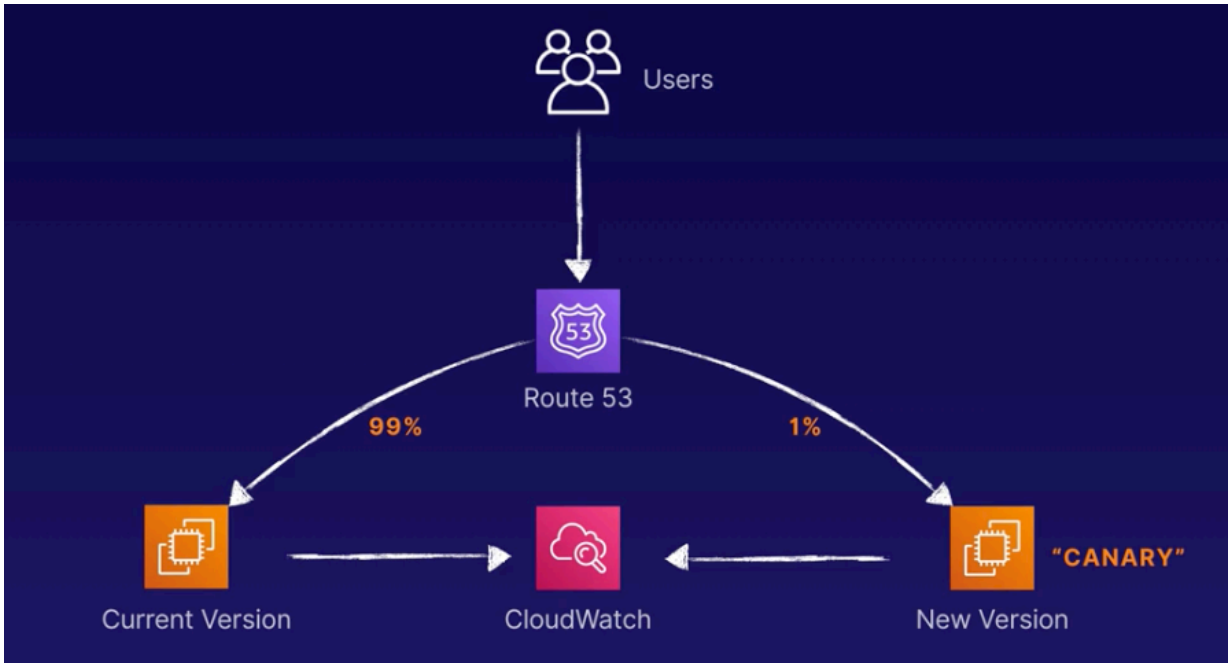
Need feedback loop to detect when we are trending in the path of memorization, or overfitting.



3 steps to Evaluation:



Canary Deployment:

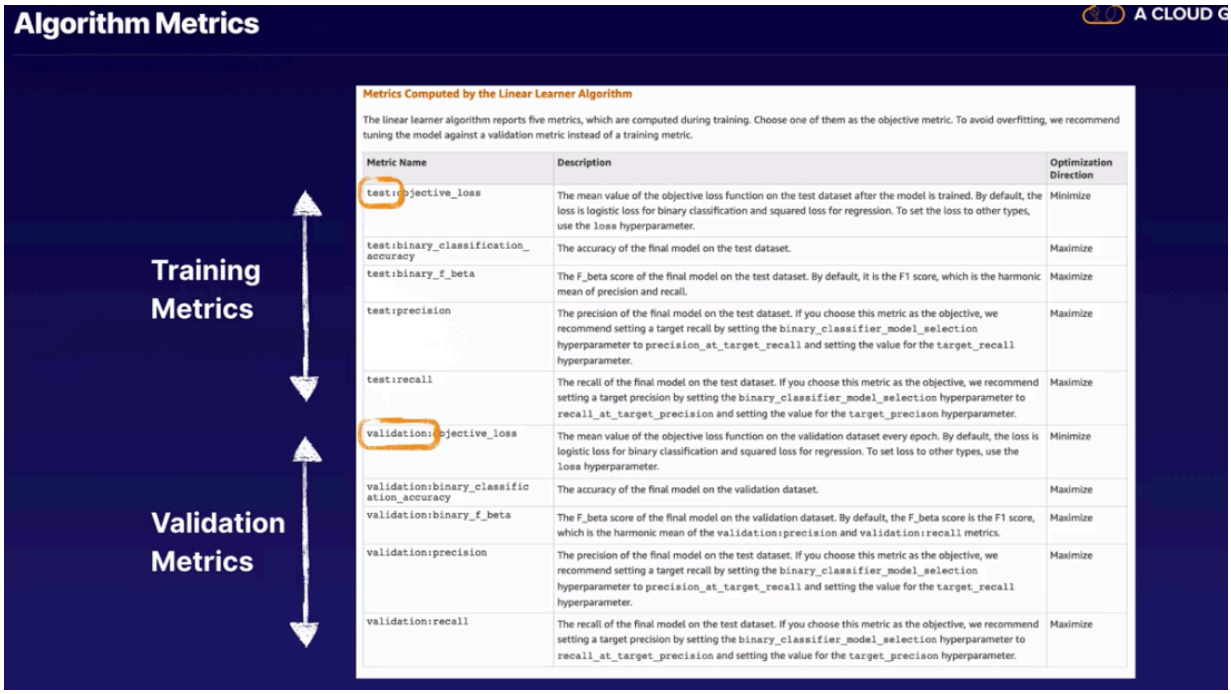


Canary Vs A/B deploymentsL
https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/canary-deployment.html

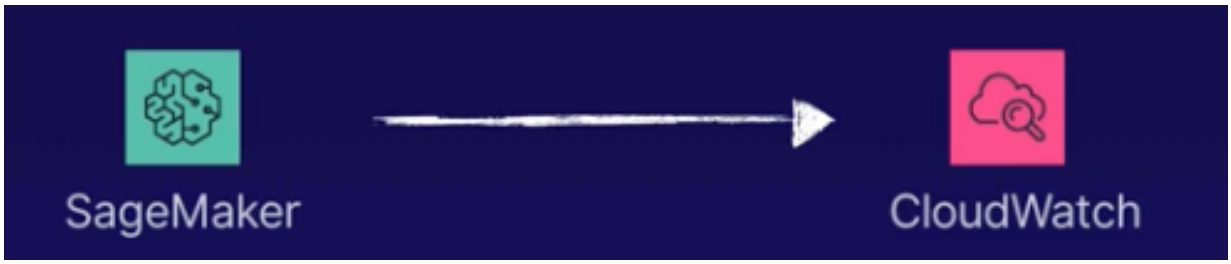
https://docs.aws.amazon.com/wellarchitected/latest/machine-learning-lens/bluegreen-deployments.html

Validation metrics are broken down into 2:
- Training metrics / used during training process
- Validation metrics / when testing a model

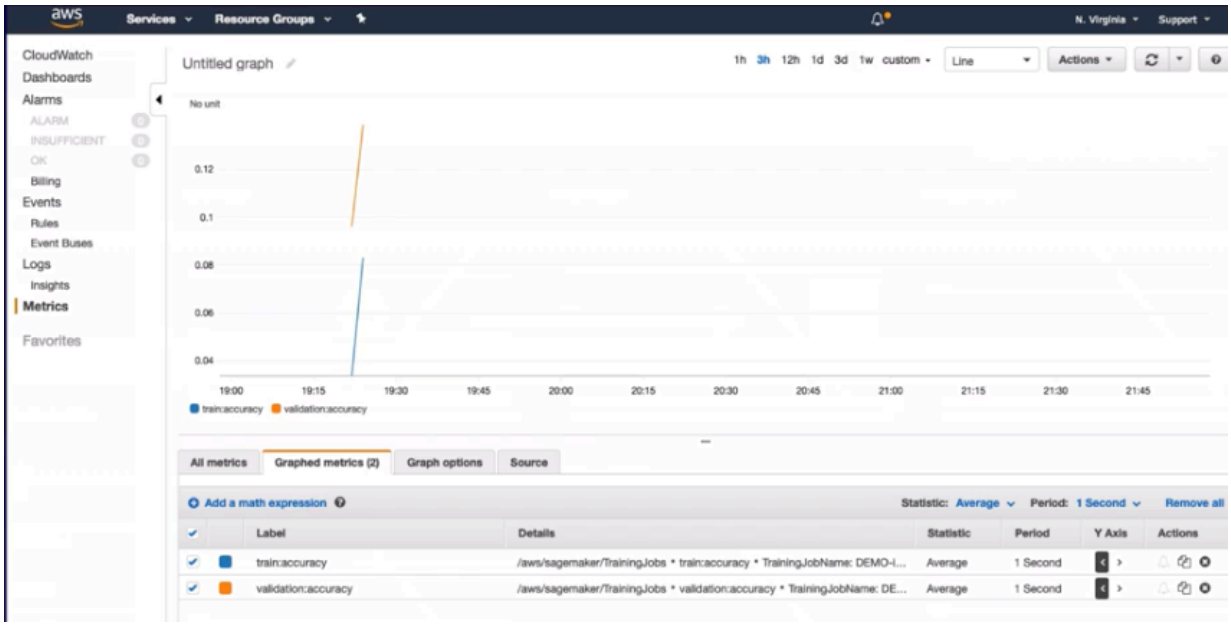
Example: https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner-tuning.html



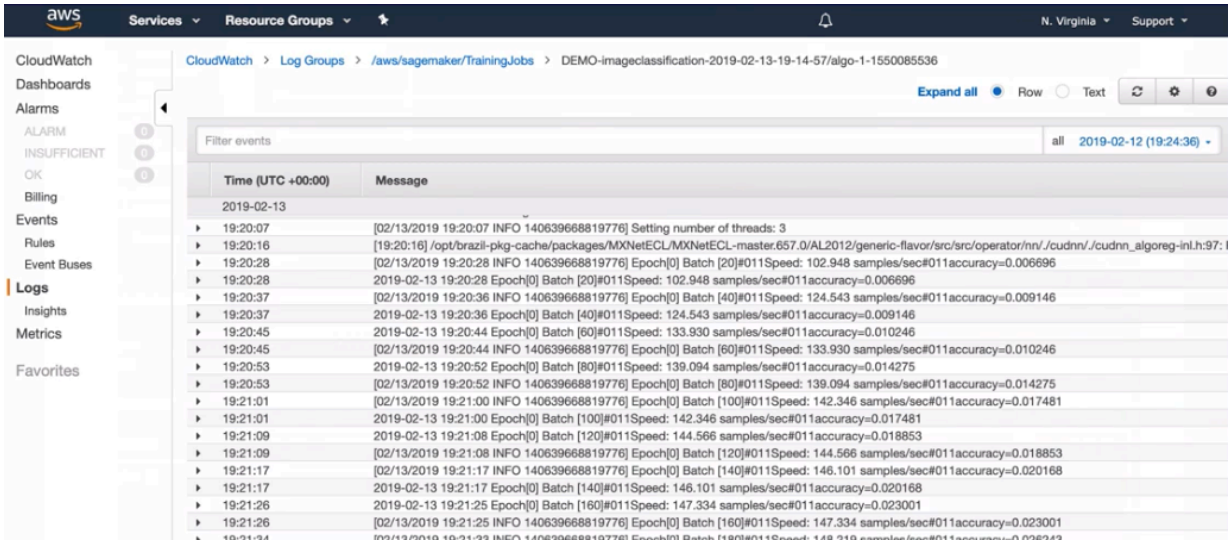
AWS scraps the logs of the containers and send them to CloudWatch
It’s one of the big features of SageMaker with integration to CloudWatch “coming for free”



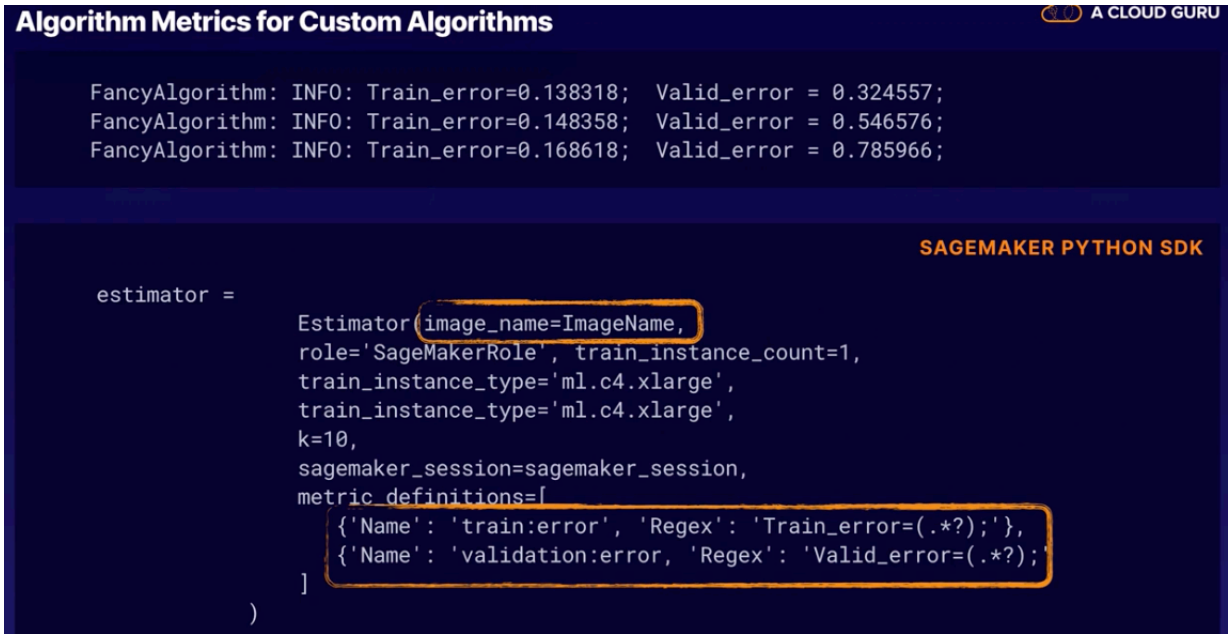
Can create charts from Cloudwatch console



And can drill through logs

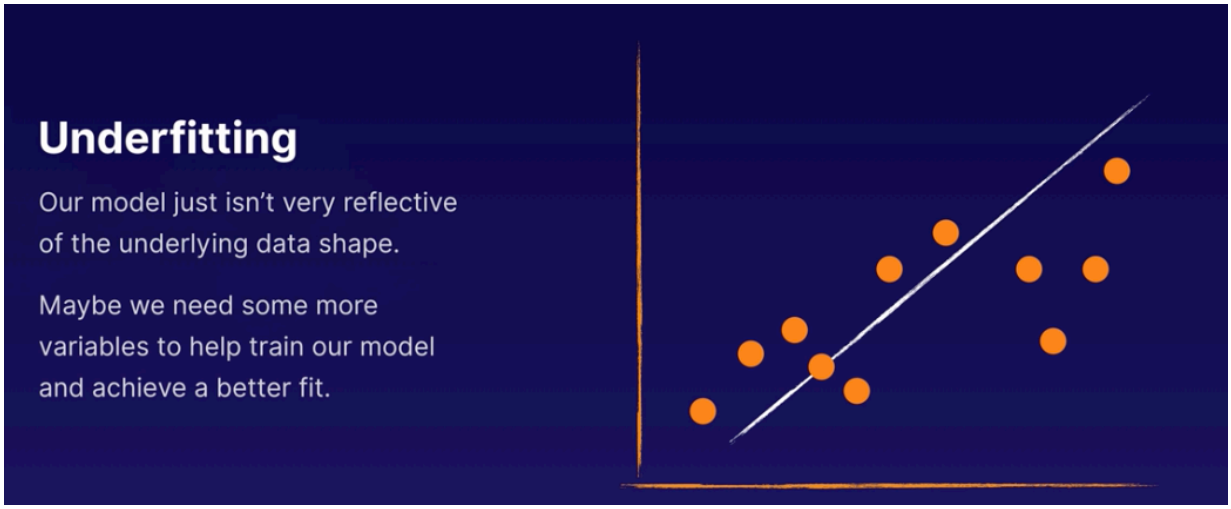


To note Pagemaker allows us to send logs from our custom algorithms to CloudWatch.
When we define our SM jobs, we can also define our metrics that we want SM to pull from that custom algorithm.
In below case, we use a regex expression.

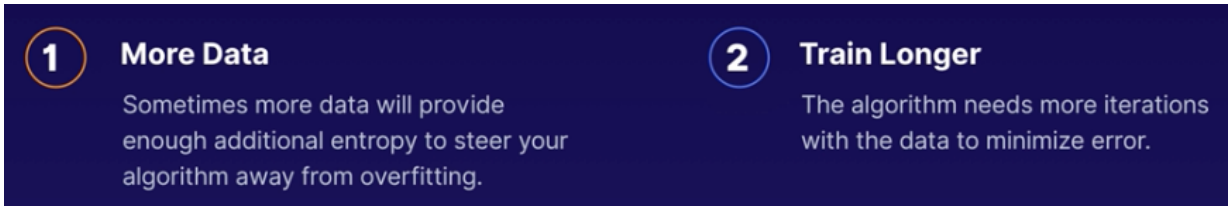


Evaluation Model Accuracy:

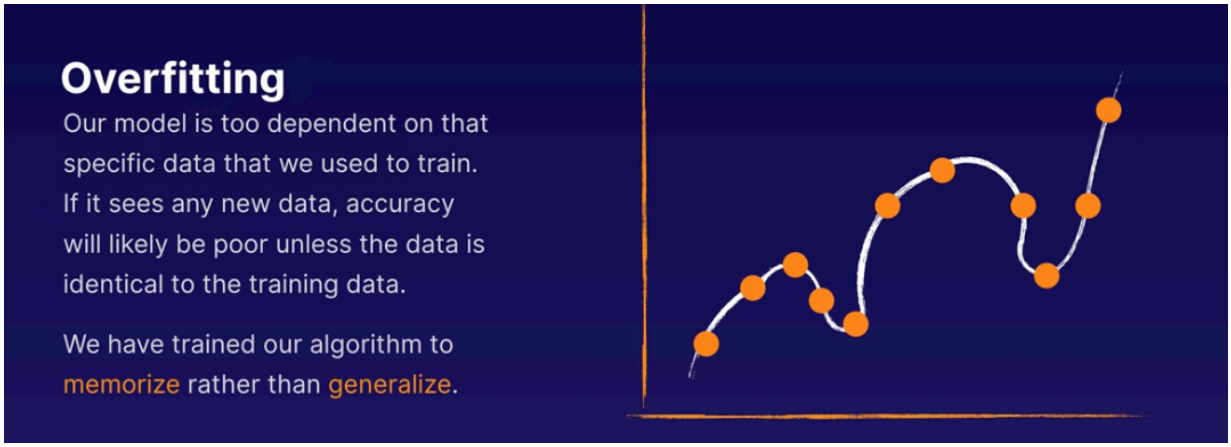
One of the risk we run is **underfitting** - model is not very reflective of the underlying data



Things we can do to prevent under fitting:



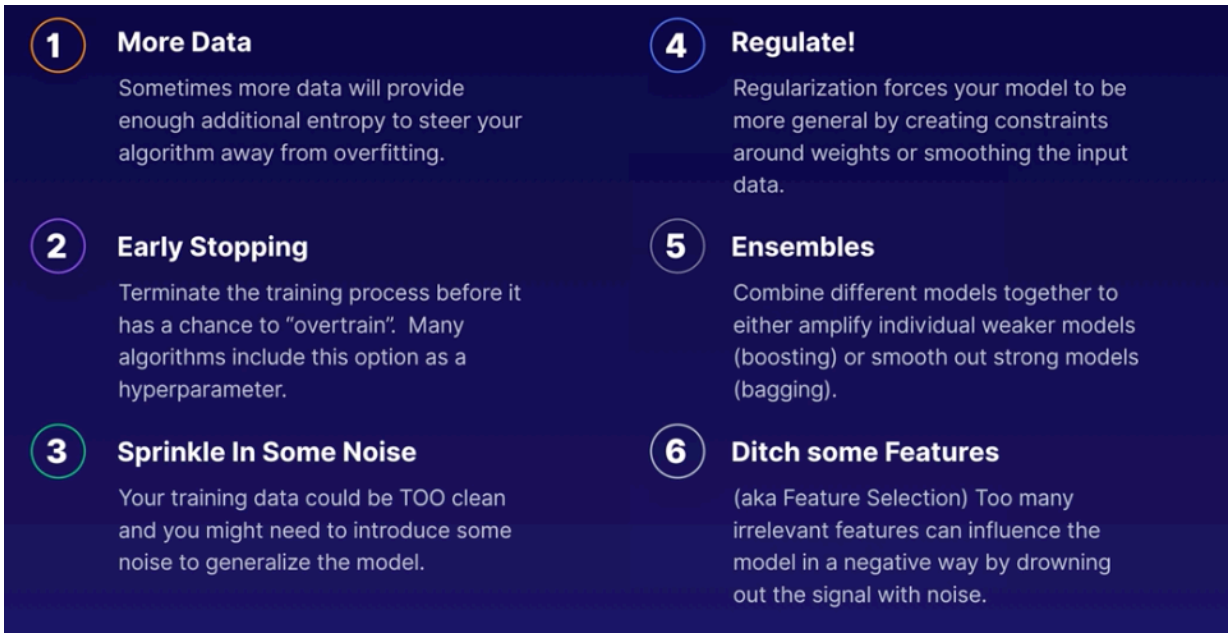
The opposite pb is **Overfitting**. We have basically memorized the data. If we see any new data outside of the training data, the model won't know what to do



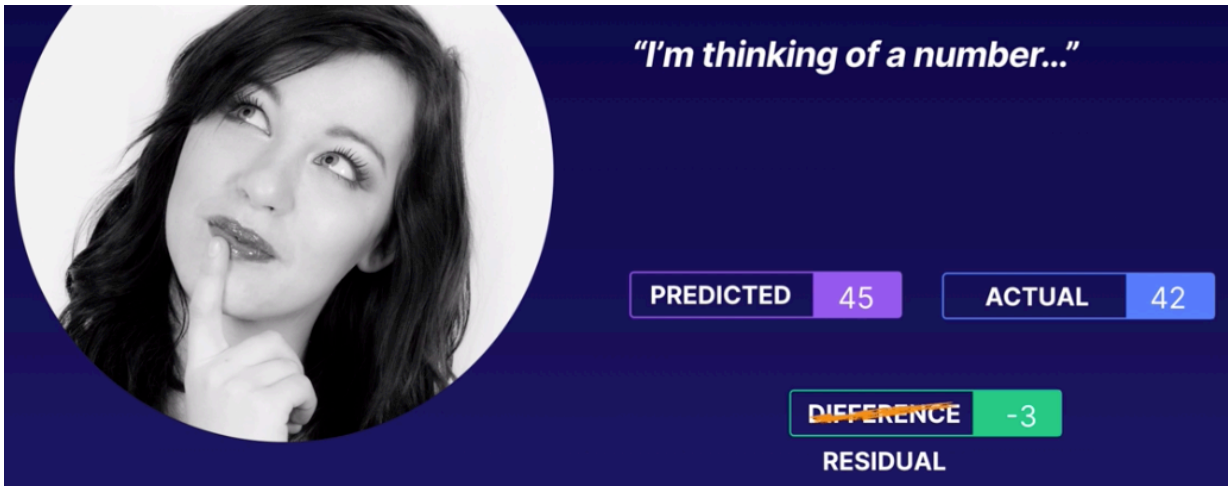
What we are after is a **Robust** model, that can **generalize** effectively, whenever it sees a new value it has not seen before.
=> we want a model that follows the model as best as possible, but not too close, and can deal with noise.

Training Error	Testing Error	
Low	Low	You want this!
Low	High	Overfitting
High	High	Try Another Approach
High	Low	Run Away!

Ways to **prevent overfitting**:

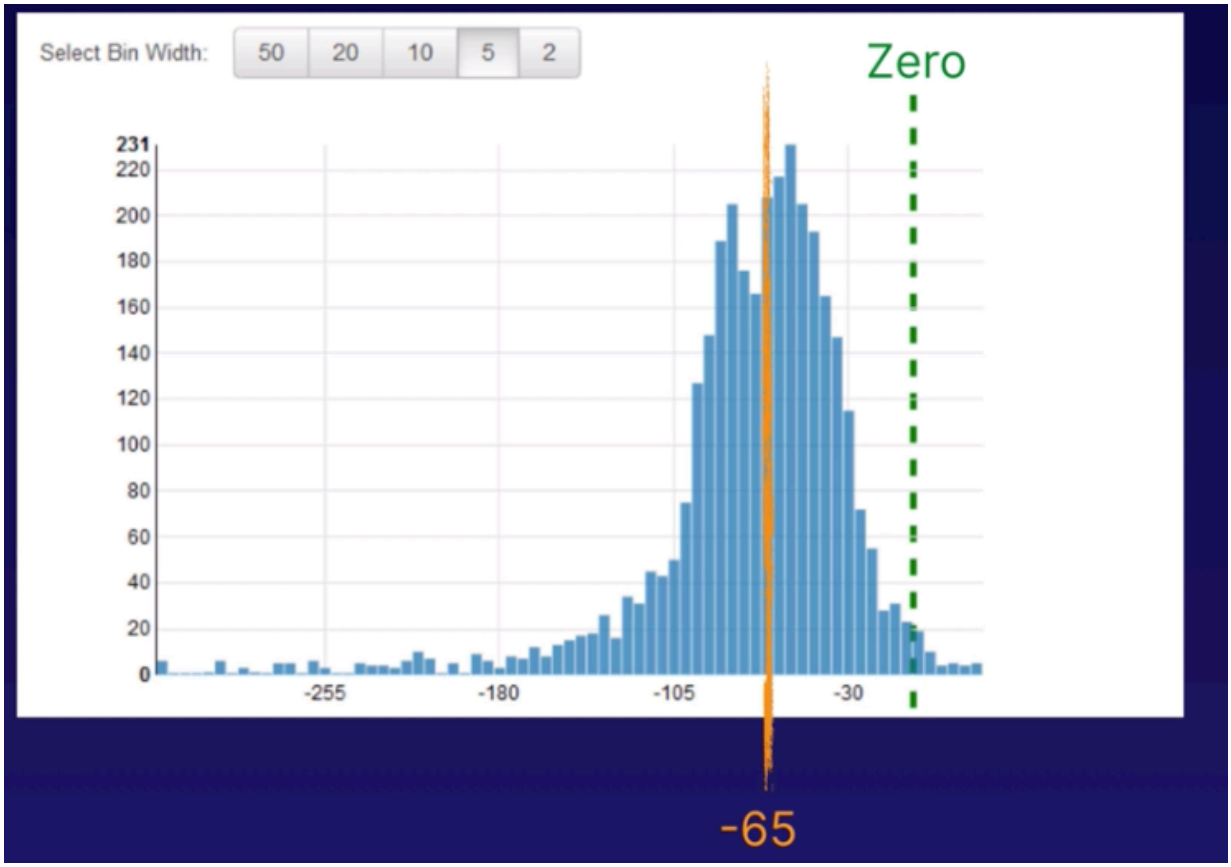


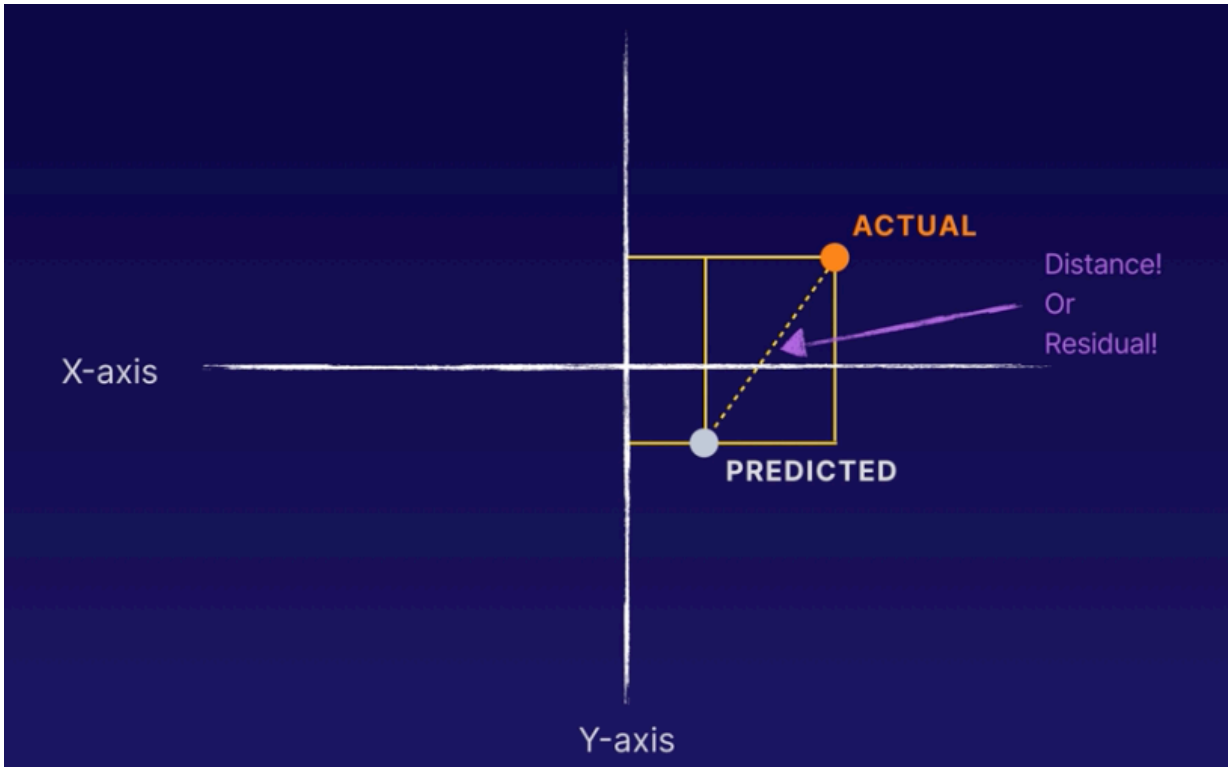
Regression Accuracy => Residual



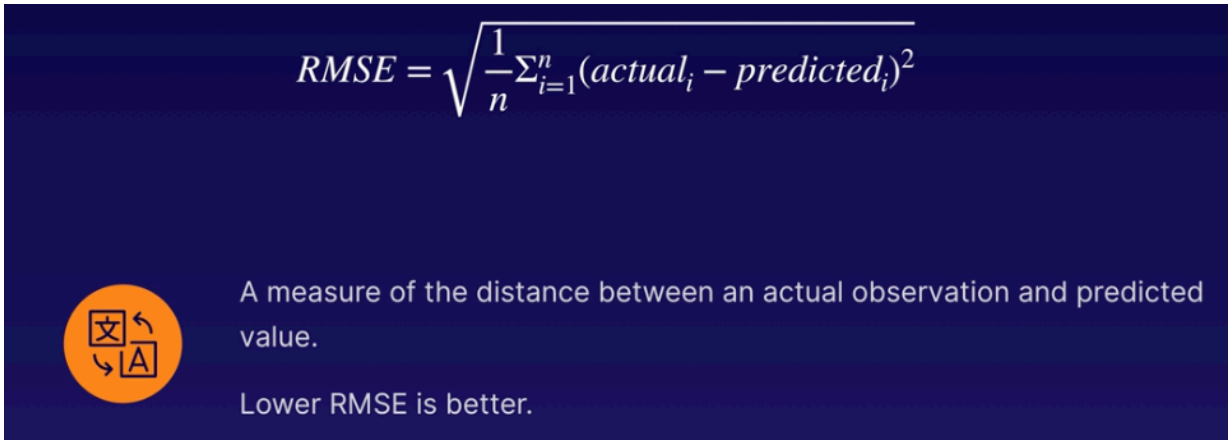
Residual distribution

We would like to see this bell curve centered around 0, so we predict too high as much as we predict too low
At -65, it means we are predicting higher more often.





Regression model error with RMSE, using the distance between actual and predicted
We want a low RMSE



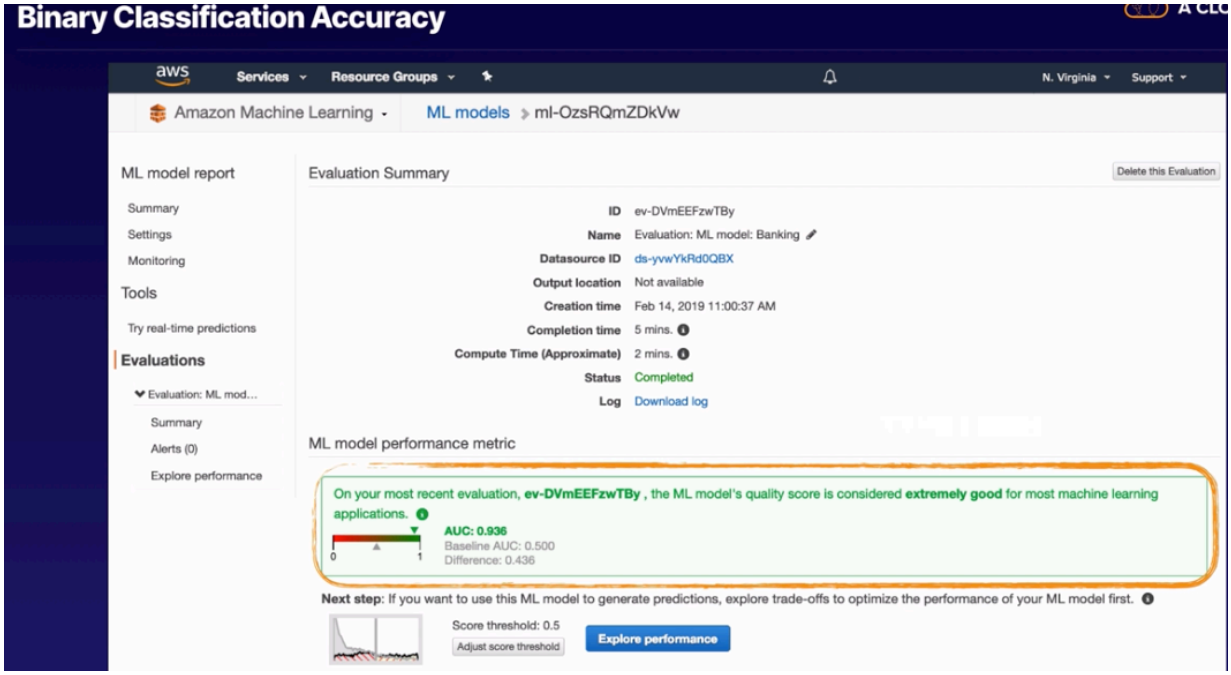
Binary Classification Accuracy
Type 1 error => false positives
Type error => false negatives

		Actual Outcome: TRUE	Actual Outcome: FALSE
Predicted Outcome	TRUE	I predicted correctly!	I was wrong. (False Positive) **TYPE I ERROR**
	FALSE	I was wrong. (False Negative) **TYPE II ERROR**	I predicted correctly!

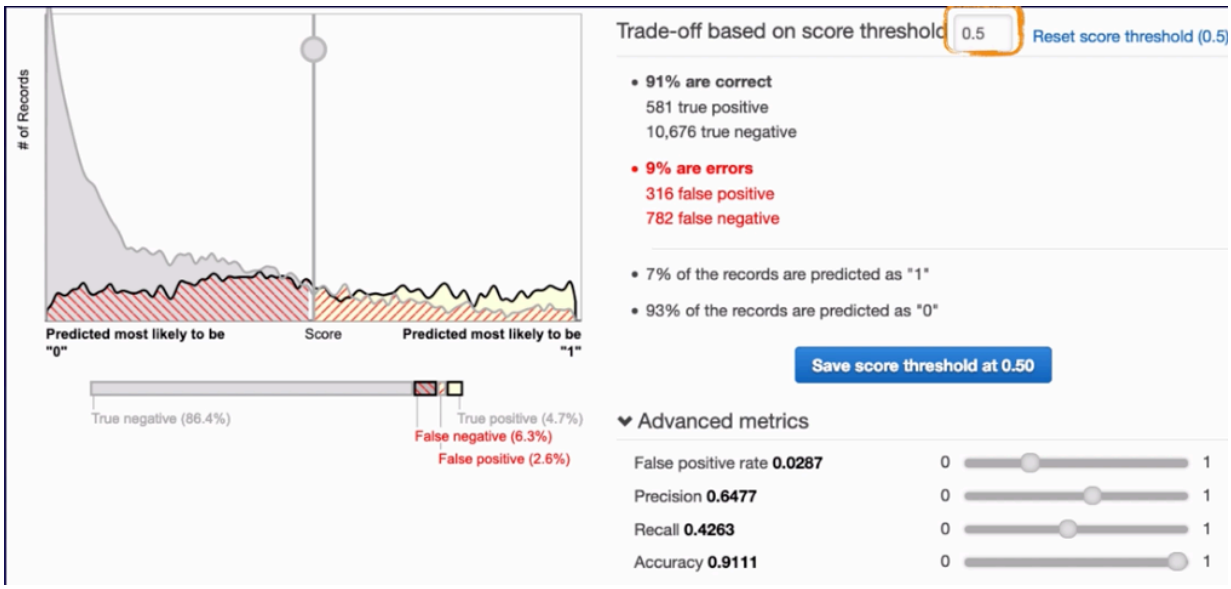
In the past, AWS had Amazon Machine Learning - but now AWS encourages everyone to use SM

Machine learning had some cool visualizations though

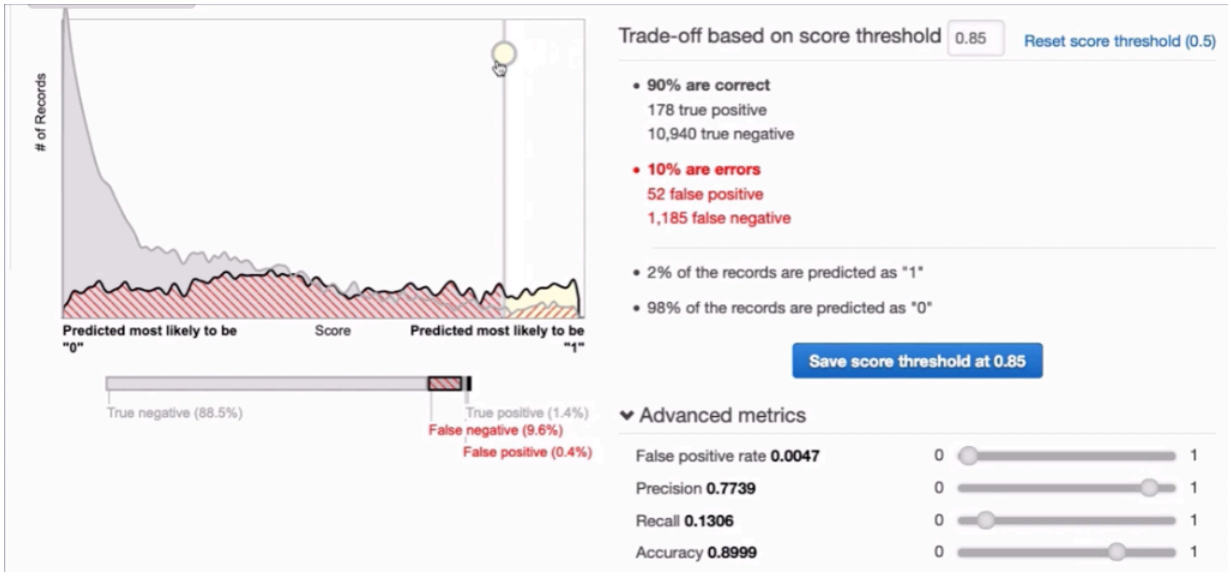
Example: area **under the curve: AUC**
Want score to be as close as possible to 1

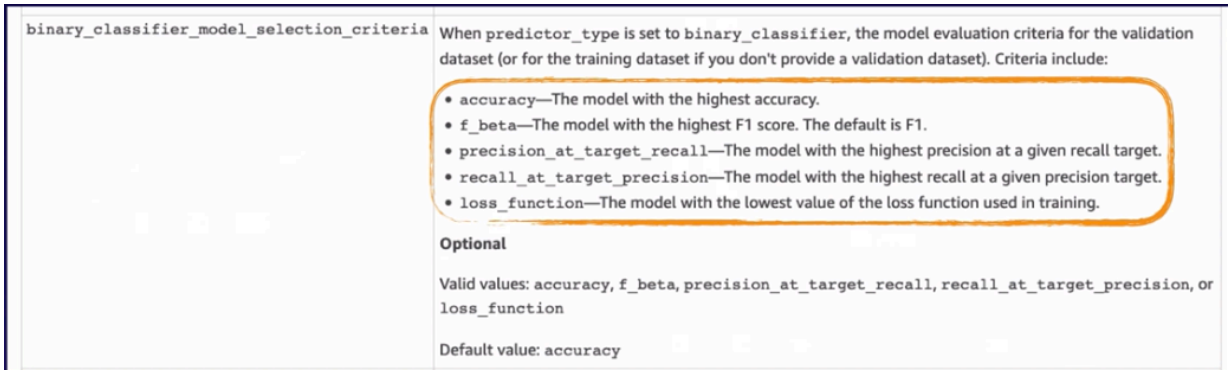


Advanced metrics:



Machine learning allows us to use that threshold and we can se metrics changing, with tradeoffs

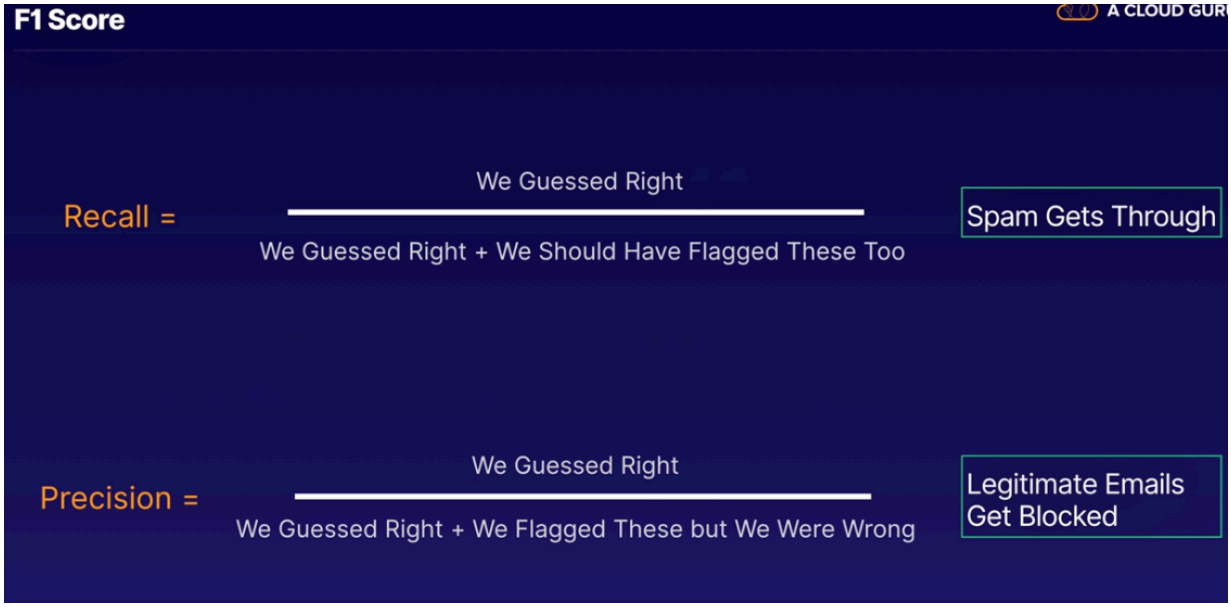




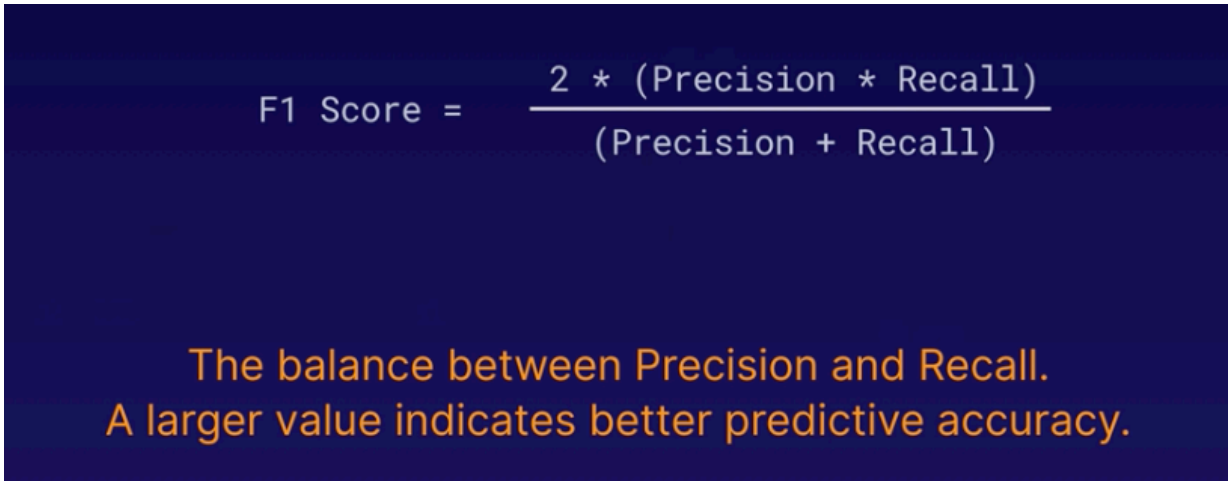
Recall Vs Precision

If someone asks me to "**recall**" something from 10 years ago I am probably not going to remember everything, we won't guess correctly all the time "**Precision**" -> it's very precise. It's important we get all the spams. In this case, we run the risk of flagging legitimate emails.

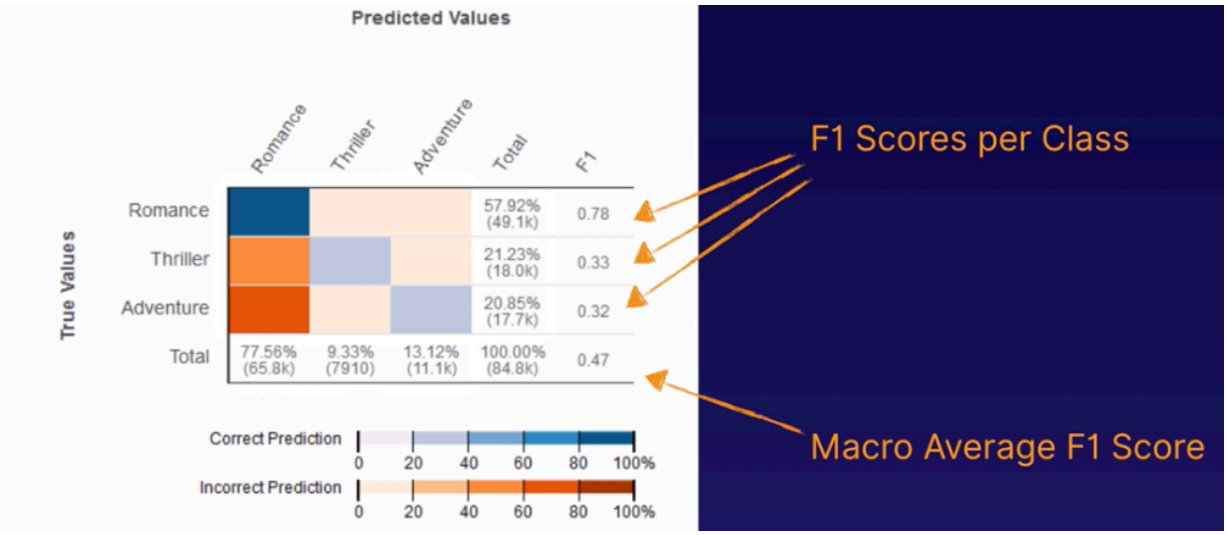
There are tradeoffs.
As recall goes up, precision decreases.



F1Score
This is a balance between precision and recall
We generally look for a higher value that shows a better predictive accuracy. But that also depends on the use case



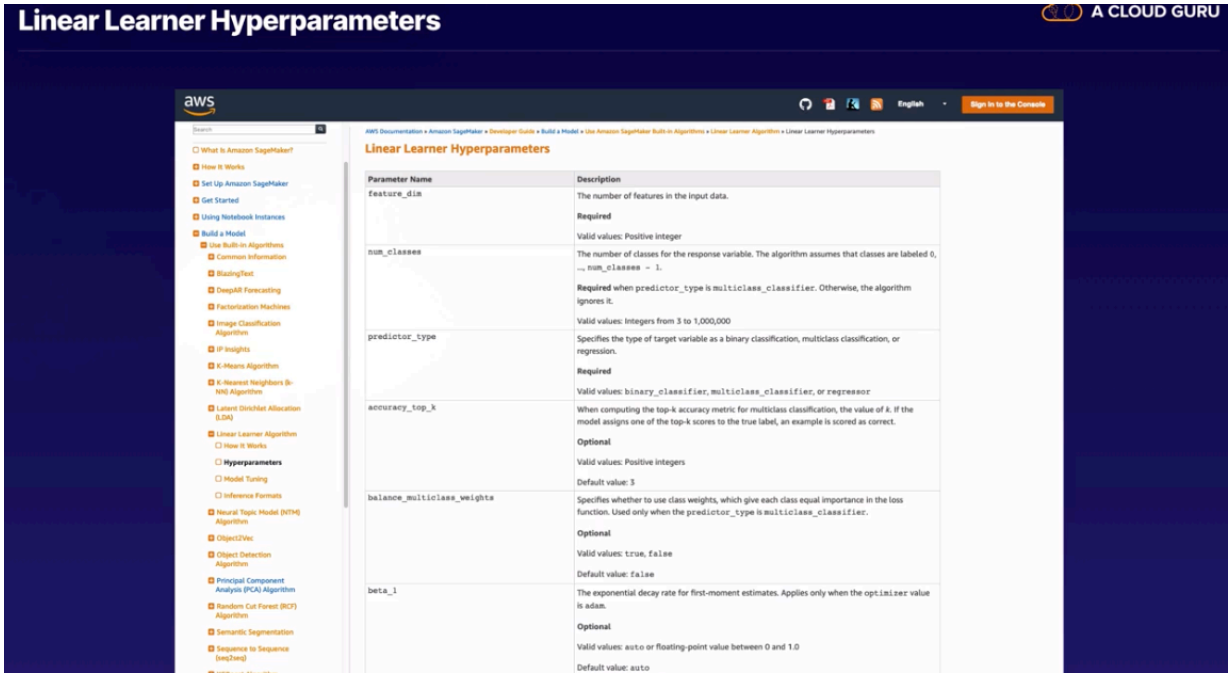
Multiclass classification



Improving Model accuracy

Model Tuning

Example for Linear Learner hyper parameters

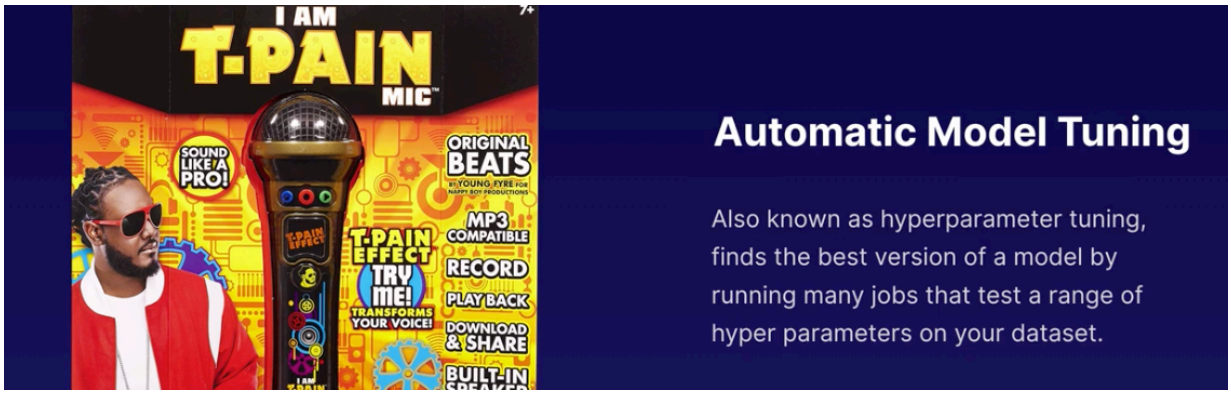


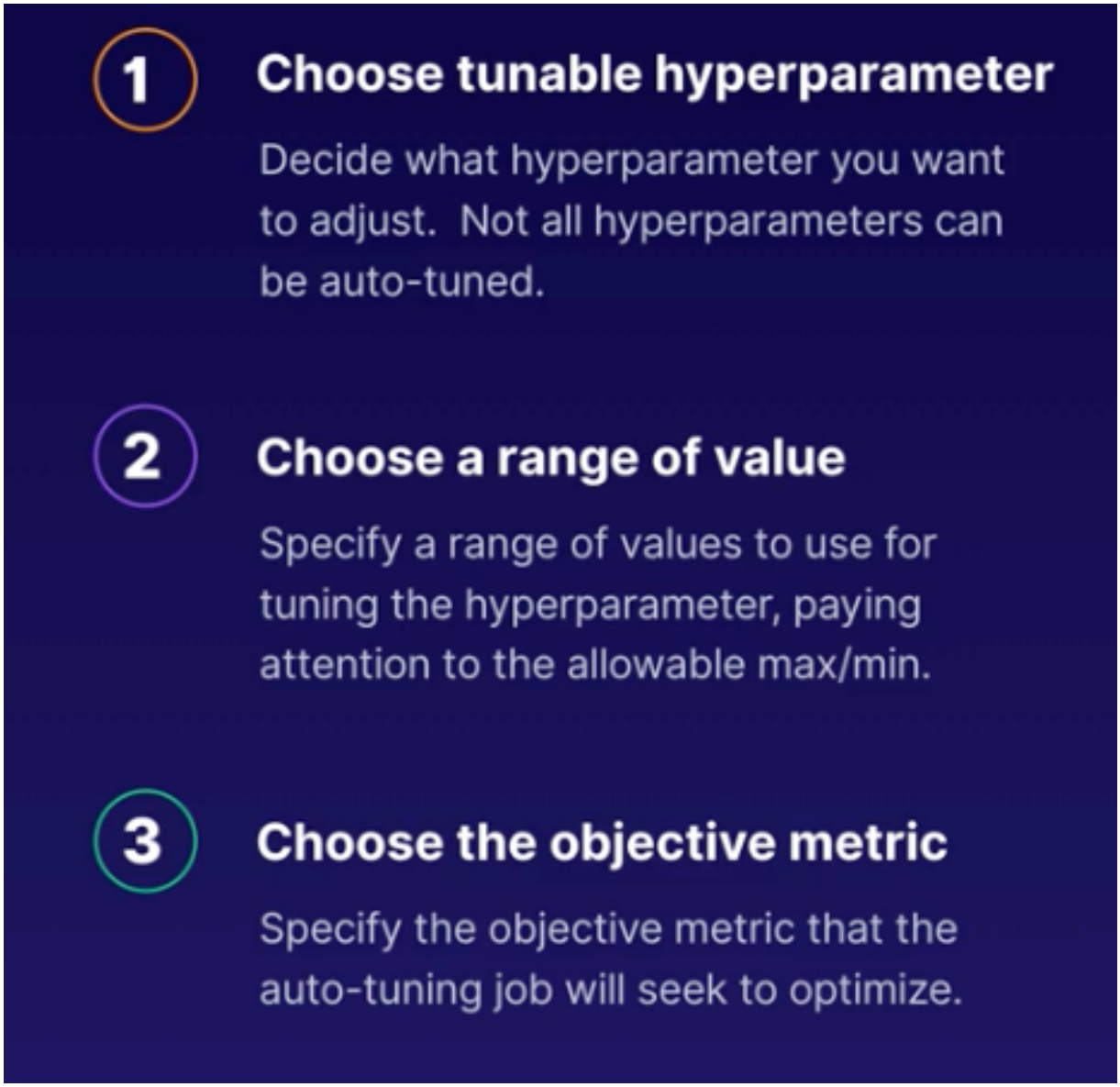
Some are required vs optional

Linear Learner Hyperparameters

Parameter Name	Description
`feature_dim`	The number of features in the input data. **Required** Valid values: Positive integer
`num_classes`	The number of classes for the response variable. The algorithm assumes that classes are labeled 0, ..., `num_classes - 1`. **Required** when `predictor_type` is `multiclass_classifier`. Otherwise, the algorithm ignores it. Valid values: Integers from 3 to 1,000,000
`predictor_type`	Specifies the type of target variable as a binary classification, multiclass classification, or regression. **Required** Valid values: `binary_classifier`, `multiclass_classifier`, or `regressor`
`accuracy_top_k`	When computing the top-k accuracy metric for multiclass classification, the value of *k*. If the model assigns one of the top-k scores to the true label, an example is scored as correct. **Optional** Valid values: Positive integers Default value: 3

Hyper parameter Tuning



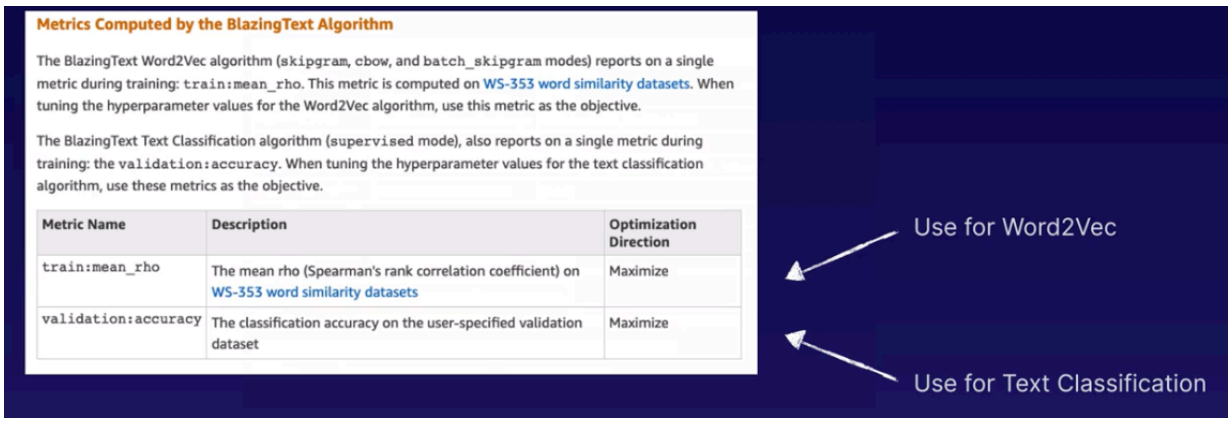

1
Choose tunable hyperparameter
Decide what hyperparameter you want to adjust. Not all hyperparameters can be auto-tuned.
2
Choose a range of value
Specify a range of values to use for tuning the hyperparameter, paying attention to the allowable max/min.
3
Choose the objective metric
Specify the objective metric that the auto-tuning job will seek to optimize.

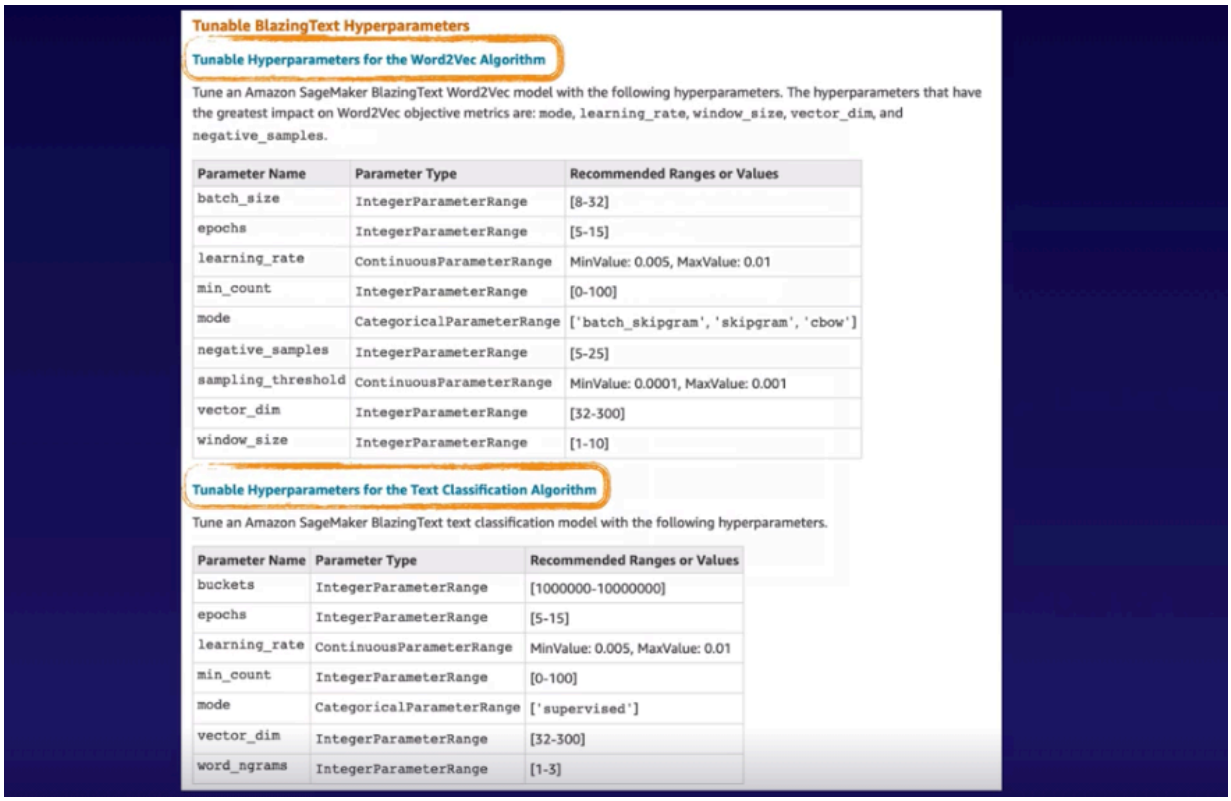
Tuning Linear Learner Hyperparameters

You can tune a linear learner model with the following hyperparameters.

Parameter Name	Parameter Type	Recommended Ranges
wd	ContinuousParameterRanges	MinValue: 1e-7, MaxValue: 1
l1	ContinuousParameterRanges	MinValue: 1e-7, MaxValue: 1
learning_rate	ContinuousParameterRanges	MinValue: 1e-5, MaxValue: 1
mini_batch_size	IntegerParameterRanges	MinValue: 100, MaxValue: 5000
use_bias	CategoricalParameterRanges	[True, False]
positive_example_weight_mult	ContinuousParameterRanges	MinValue: 1e-5, MaxValue: 1e5

Optimization metrics can vary depending on the algorithm





Tunable BlazingText Hyperparameters

Tunable Hyperparameters for the Word2Vec Algorithm

Tune an Amazon SageMaker BlazingText Word2Vec model with the following hyperparameters. The hyperparameters that have the greatest impact on Word2Vec objective metrics are: `mode`, `learning_rate`, `window_size`, `vector_dim`, and `negative_samples`.

Parameter Name	Parameter Type	Recommended Ranges or Values
batch_size	IntegerParameterRange	[8-32]
epochs	IntegerParameterRange	[5-15]
learning_rate	ContinuousParameterRange	MinValue: 0.005, MaxValue: 0.01
min_count	IntegerParameterRange	[0-100]
mode	CategoricalParameterRange	['batch_skipgram', 'skipgram', 'cbow']
negative_samples	IntegerParameterRange	[5-25]
sampling_threshold	ContinuousParameterRange	MinValue: 0.0001, MaxValue: 0.001
vector_dim	IntegerParameterRange	[32-300]
window_size	IntegerParameterRange	[1-10]

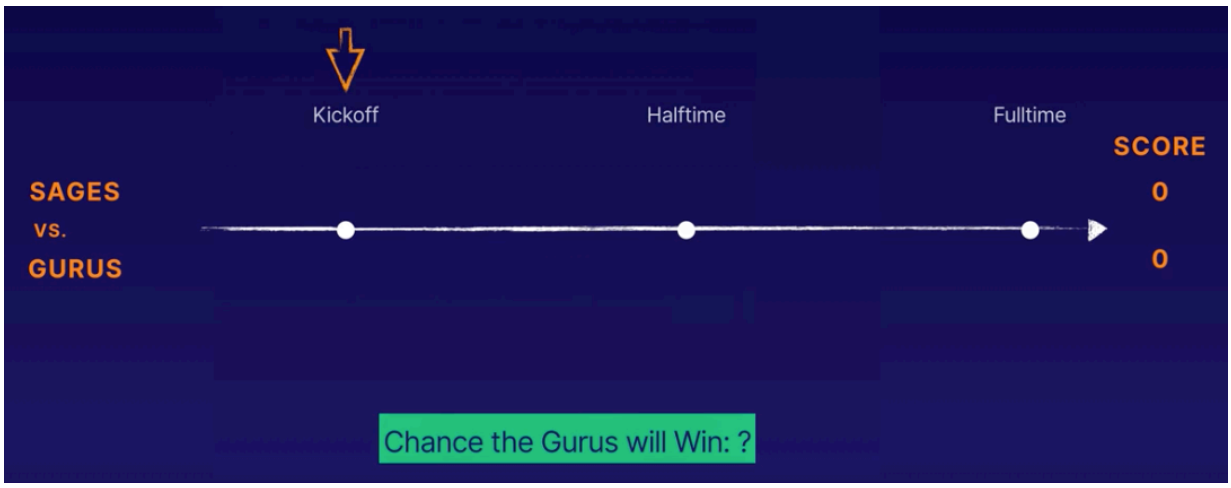
Tunable Hyperparameters for the Text Classification Algorithm

Tune an Amazon SageMaker BlazingText text classification model with the following hyperparameters.

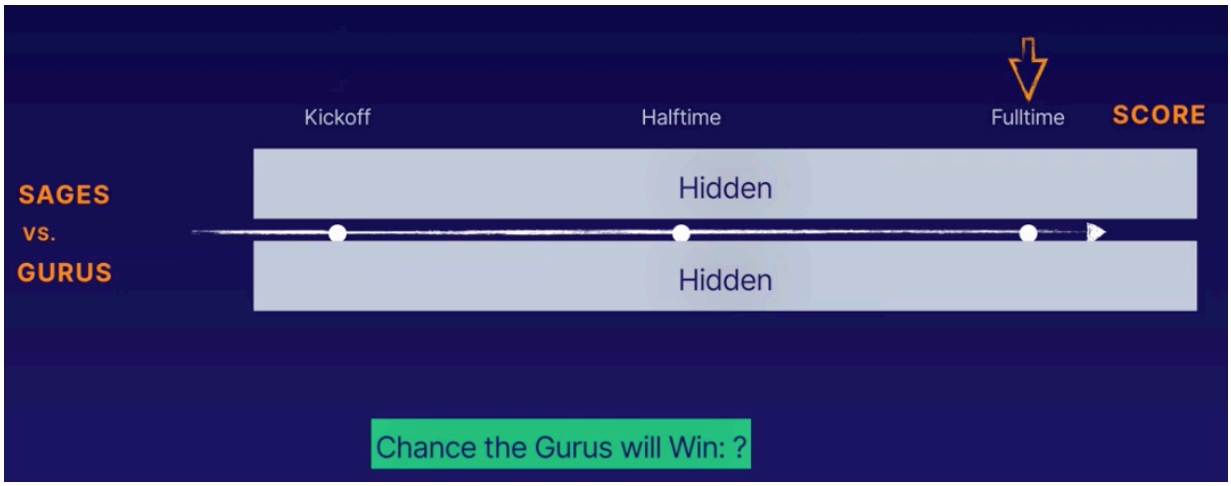
Parameter Name	Parameter Type	Recommended Ranges or Values
buckets	IntegerParameterRange	[1000000-10000000]
epochs	IntegerParameterRange	[5-15]
learning_rate	ContinuousParameterRange	MinValue: 0.005, MaxValue: 0.01
min_count	IntegerParameterRange	[0-100]
mode	CategoricalParameterRange	['supervised']
vector_dim	IntegerParameterRange	[32-300]
word_ngrams	IntegerParameterRange	[1-3]

How does AWS do hyper parameter tuning

Example - we are at the beginning of a soccer game: Sages vs Gurus
What is the chance Gurus will win?
At this stage we don't know



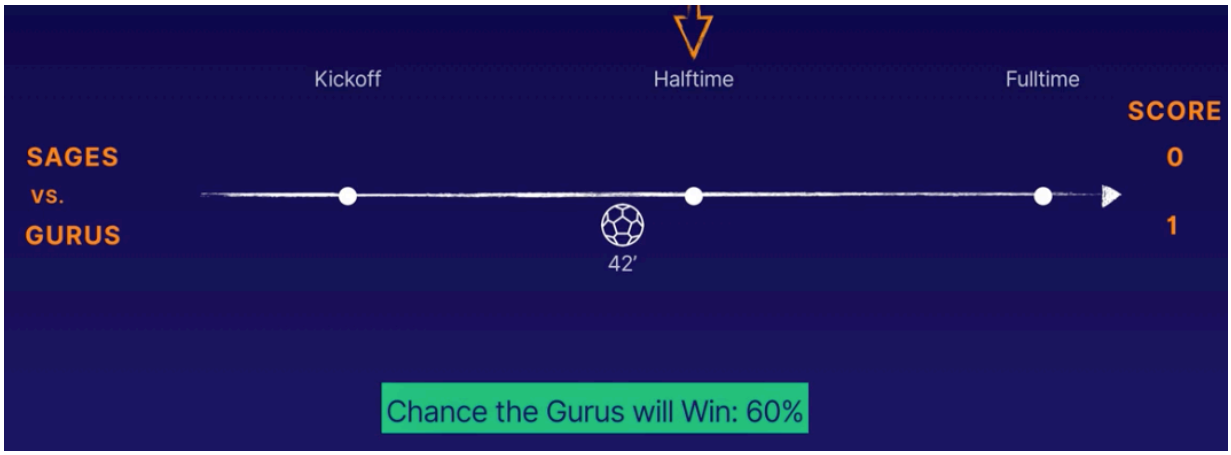
Now game is over, and same Q - no detail was provided.
We have same chances of guessing as before

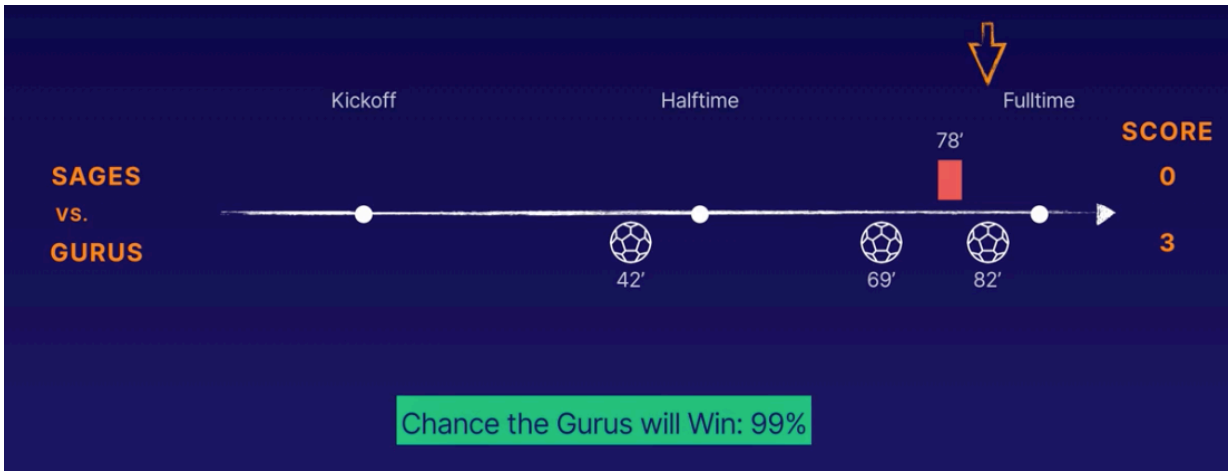


LEt’s say our target is 99% chances of winning for Gurus
We can seek that values in a different ways.
With a matrix trying out values till we find it

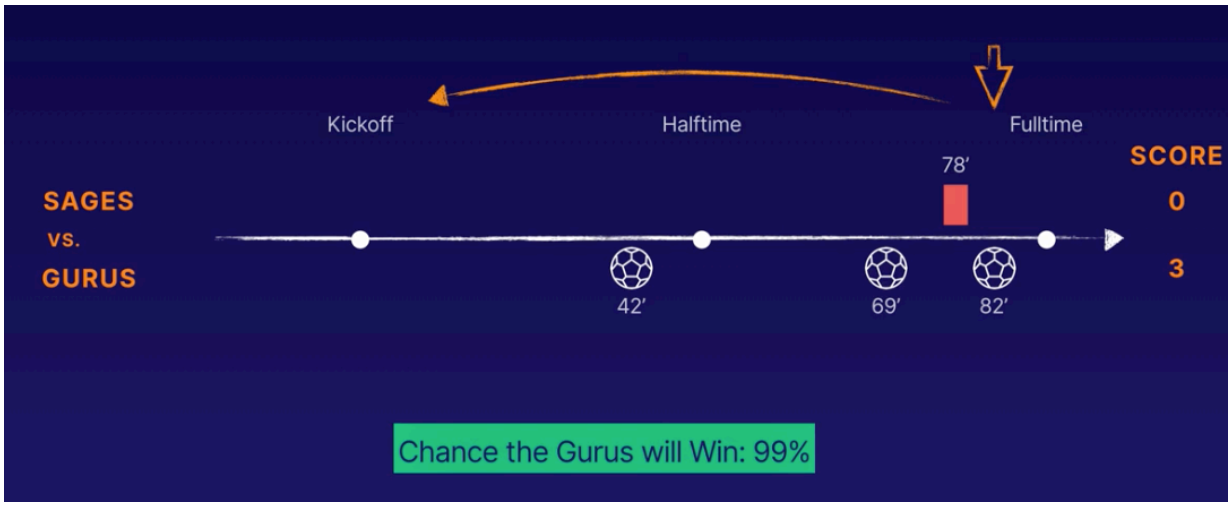
We could also pick values randomly.

Bayensian reasoning comes to help there

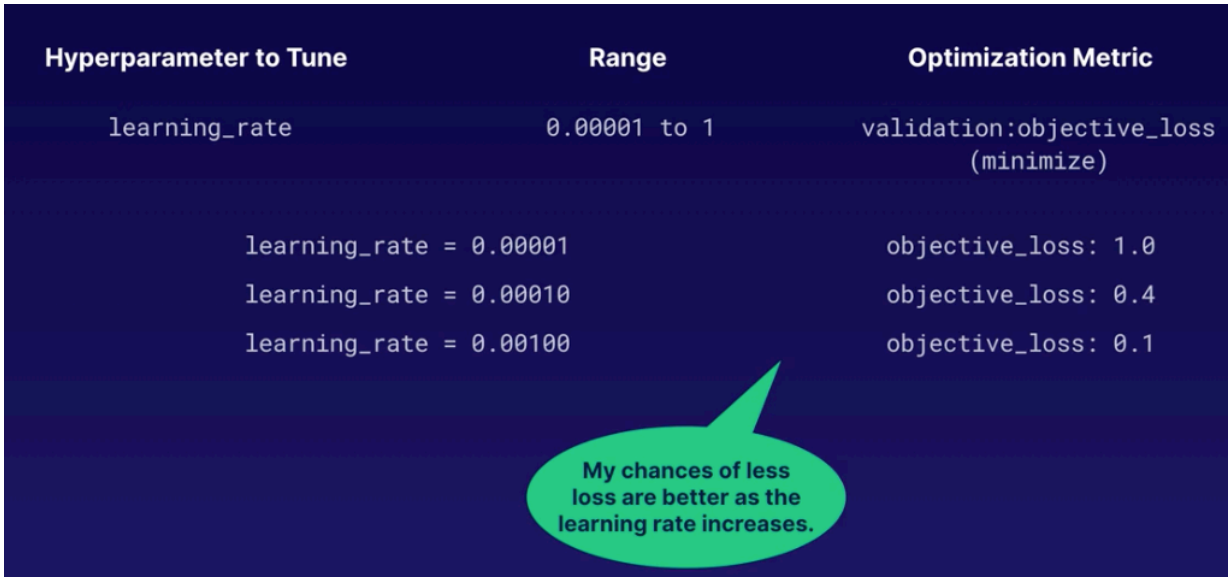




We can venture 99% because we have the benefit of the **HISTORY** of the game.
=> Crux of **Bayensian**: We pay attention to what happened in the past, and build a probability model for what we should choose as the next value



AWS will send better candidates for learning rate - try to avoid the costly iteration, sending more intelligently optimization values (as opposed to random or sequential attempts)



Try to avoid costly iterations by focusing on what hyperparameter value is likely to give more improvement.

Exam tips:

Evaluating Model Accuracy

- Understand underfitting and overfitting as well as potential causes and countermeasures.
- Know that regression accuracy is measured by RMSE.
- Be able to explain what it means if a histogram of residuals is skewed negatively or positively.
- For binary classification, know what the AUC metric indicates
- Understand trade-offs and how different scenarios might call for different optimizations.
- Understand how to read a confusion matrix and know what an F1 Score and Macro Average F1 Score metric can tell you.
- Know some ways to improve model accuracy.

Model Tuning

- Recall hyperparameters and how they can be adjusted to control your learning process.
- Know the three components you must define for automatic tuning.
- Understand that the best metrics to optimize varies by algorithm and sometimes is specific to how you use the algorithm (BlazingText for example).
- Conceptually understand Bayesian Optimization.
- Automatic Tuning is not the perfect solution and sometimes may result in weaker models.

Resources:
- Incremental training: https://docs.aws.amazon.com/sagemaker/latest/dg/incremental-training.html
- Automatic Model Tuning: https://aws.amazon.com/blogs/aws/sagemaker-automatic-model-tuning/
- Easily monitor and visualize metrics while training on SM: https://aws.amazon.com/blogs/machine-learning/easily-monitor-and-visualize-metrics-

while-training-models-on-amazon-sagemaker/