

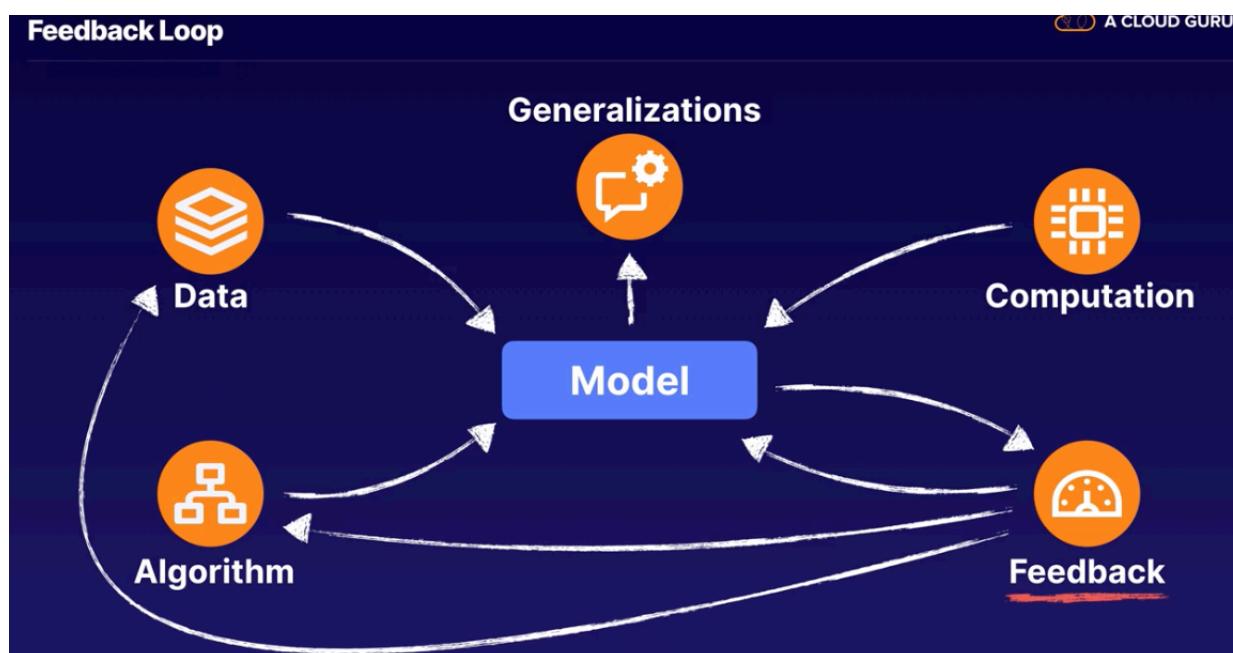
Cloud Guru - 7 - Evaluation and Optimization

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

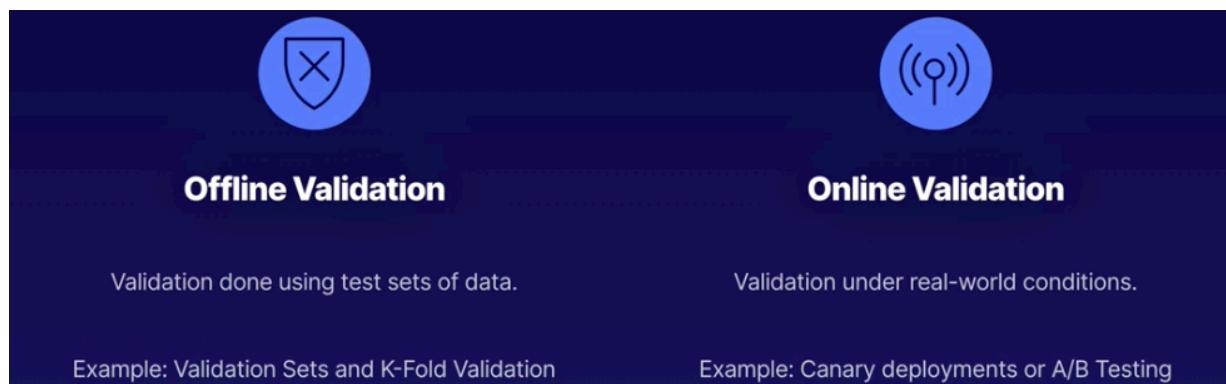
ML Objective



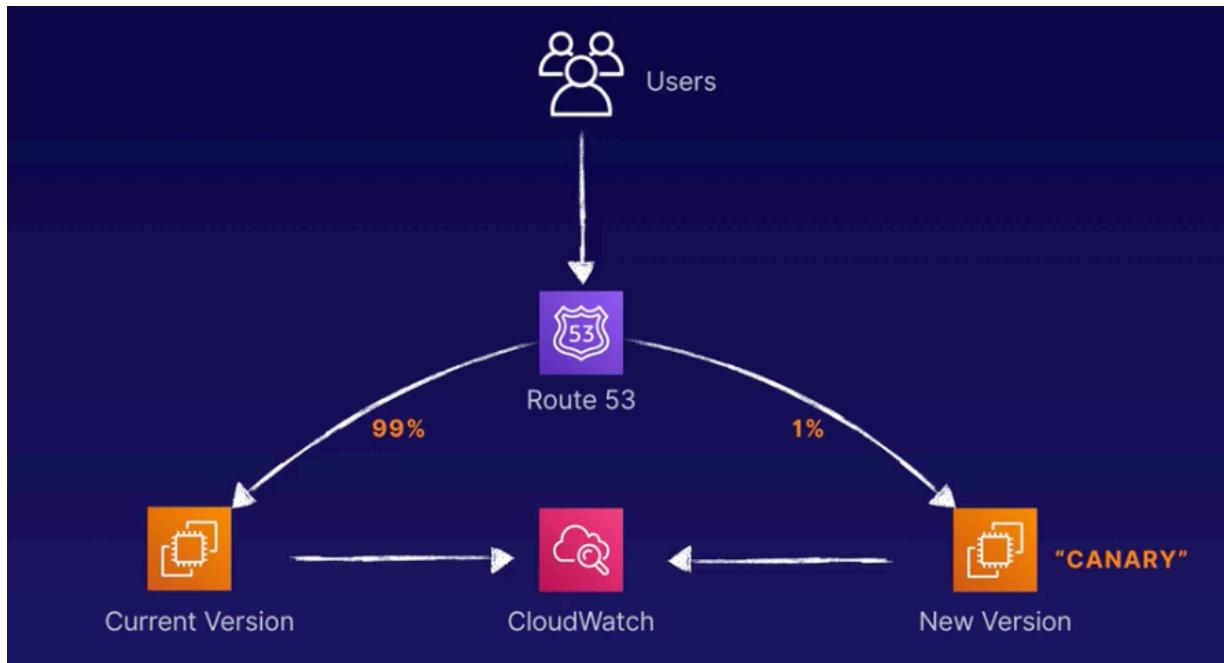
Need feedback loop to detect when we are trending in the path of memorization, or overfitting.



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>

Algorithm Metrics

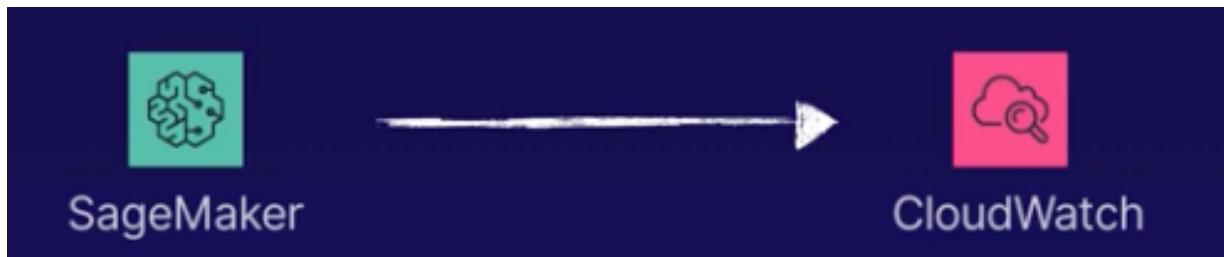
A CLOUD G

Metrics Computed by the Linear Learner Algorithm

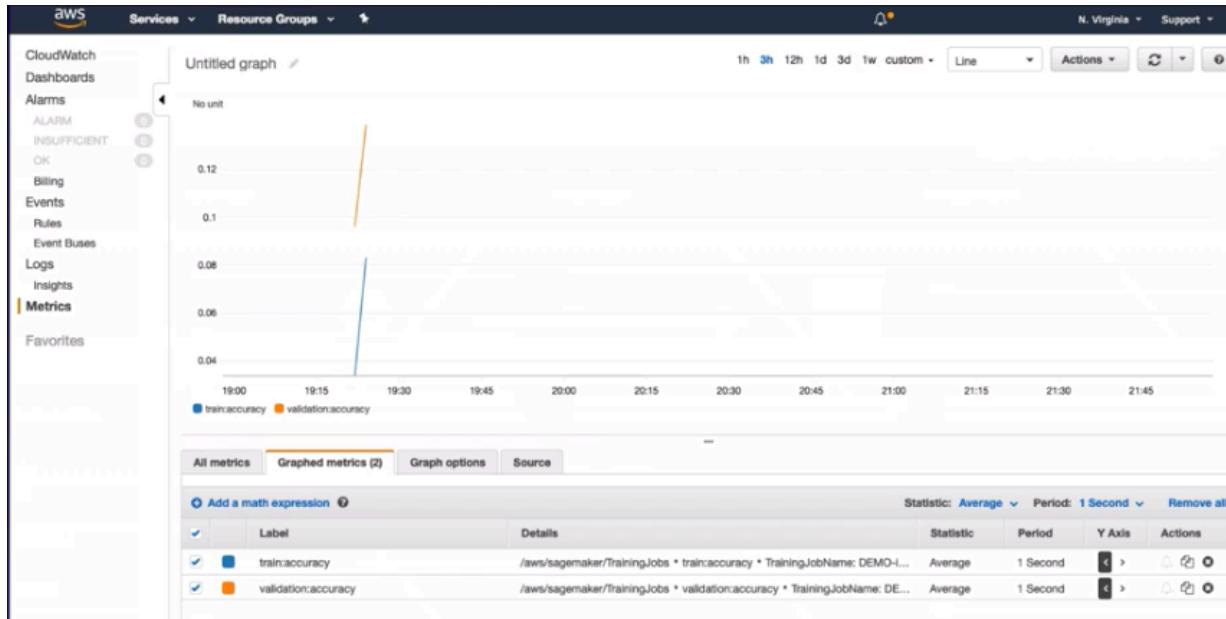
The linear learner algorithm reports five metrics, which are computed during training. Choose one of them as the objective metric. To avoid overfitting, we recommend tuning the model against a validation metric instead of a training metric.

Metric Name	Description	Optimization Direction
test:objective_loss	The mean value of the objective loss function on the test dataset after the model is trained. By default, the loss is logistic loss for binary classification and squared loss for regression. To set the loss to other types, use the loss hyperparameter.	Minimize
test:binary_classification_accuracy	The accuracy of the final model on the test dataset.	Maximize
test:binary_f_beta	The F _{beta} score of the final model on the test dataset. By default, it is the F1 score, which is the harmonic mean of precision and recall.	Maximize
test:precision	The precision of the final model on the test dataset. If you choose this metric as the objective, we recommend setting a target recall by setting the binary_classifier_model_selection hyperparameter to precision_at_target_recall and setting the value for the target_recall hyperparameter.	Maximize
test:recall	The recall of the final model on the test dataset. If you choose this metric as the objective, we recommend setting a target precision by setting the binary_classifier_model_selection hyperparameter to recall_at_target_precision and setting the value for the target_precision hyperparameter.	Maximize
validation:objective_loss	The mean value of the objective loss function on the validation dataset every epoch. By default, the loss is logistic loss for binary classification and squared loss for regression. To set loss to other types, use the loss hyperparameter.	Minimize
validation:binary_classification_accuracy	The accuracy of the final model on the validation dataset.	Maximize
validation:binary_f_beta	The F _{beta} score of the final model on the validation dataset. By default, the F _{beta} score is the F1 score, which is the harmonic mean of the validation:precision and validation:recall metrics.	Maximize
validation:precision	The precision of the final model on the test dataset. If you choose this metric as the objective, we recommend setting a target recall by setting the binary_classifier_model_selection hyperparameter to precision_at_target_recall and setting the value for the target_recall hyperparameter.	Maximize
validation:recall	The recall of the final model on the test dataset. If you choose this metric as the objective, we recommend setting a target precision by setting the binary_classifier_model_selection hyperparameter to recall_at_target_precision and setting the value for the target_precision hyperparameter.	Maximize

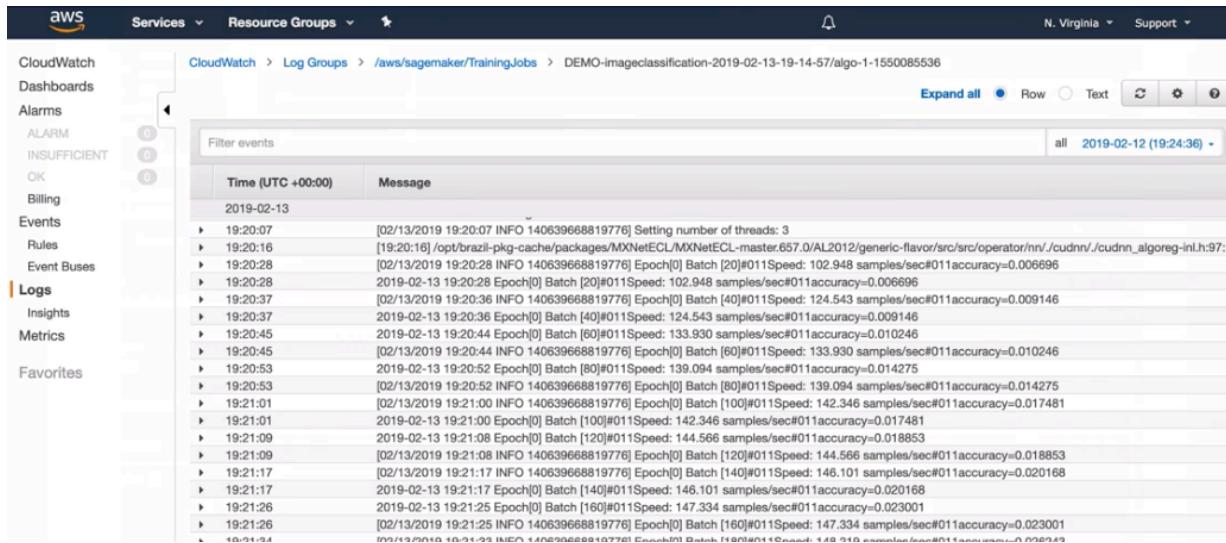
AWS scraps the logs of the containers and send them to CloudWatch
It's one of the big features of SageMaker with integration to CloudWatch "coming for free"



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.

Algorithm Metrics for Custom Algorithms

A CLOUD GURU

```
FancyAlgorithm: INFO: Train_error=0.138318; Valid_error = 0.324557;  
FancyAlgorithm: INFO: Train_error=0.148358; Valid_error = 0.546576;  
FancyAlgorithm: INFO: Train_error=0.168618; Valid_error = 0.785966;
```

SAGEMAKER PYTHON SDK

```
estimator =  
    Estimator(image_name=ImageName,  
              role='SageMakerRole', train_instance_count=1,  
              train_instance_type='ml.c4.xlarge',  
              train_instance_type='ml.c4.xlarge',  
              k=10,  
              sagemaker_session=sagemaker_session,  
              metric_definitions=[  
                  {'Name': 'train:error', 'Regex': 'Train_error=(.*?);'},  
                  {'Name': 'validation:error', 'Regex': 'Valid_error=(.*?);'}  
              ]  
    )
```

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:

1

More Data

Sometimes more data will provide enough additional entropy to steer your algorithm away from overfitting.

2

Train Longer

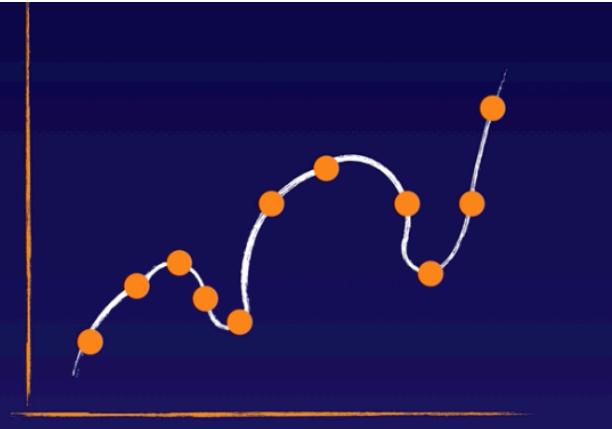
The algorithm needs more iterations with the data to minimize error.

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

Overfitting

Our model is too dependent on that specific data that we used to train. If it sees any new data, accuracy will likely be poor unless the data is identical to the training data.

We have trained our algorithm to **memorize** rather than **generalize**.



What we are after is a **Robust** model, that can **generalize** effectively, whenever it sees a new value it has not seen before.

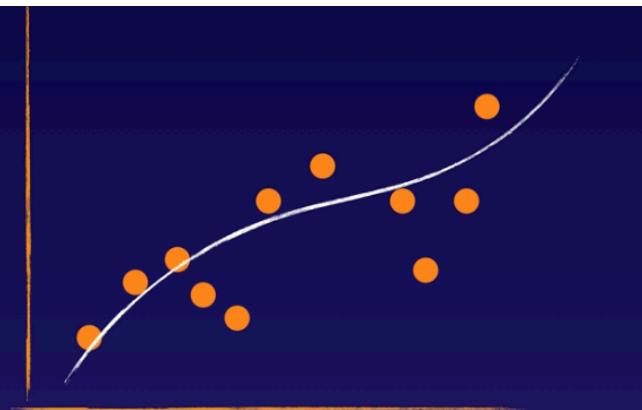
=> we want a model that follows the model as best as possible, but not too close, and can deal with noise.

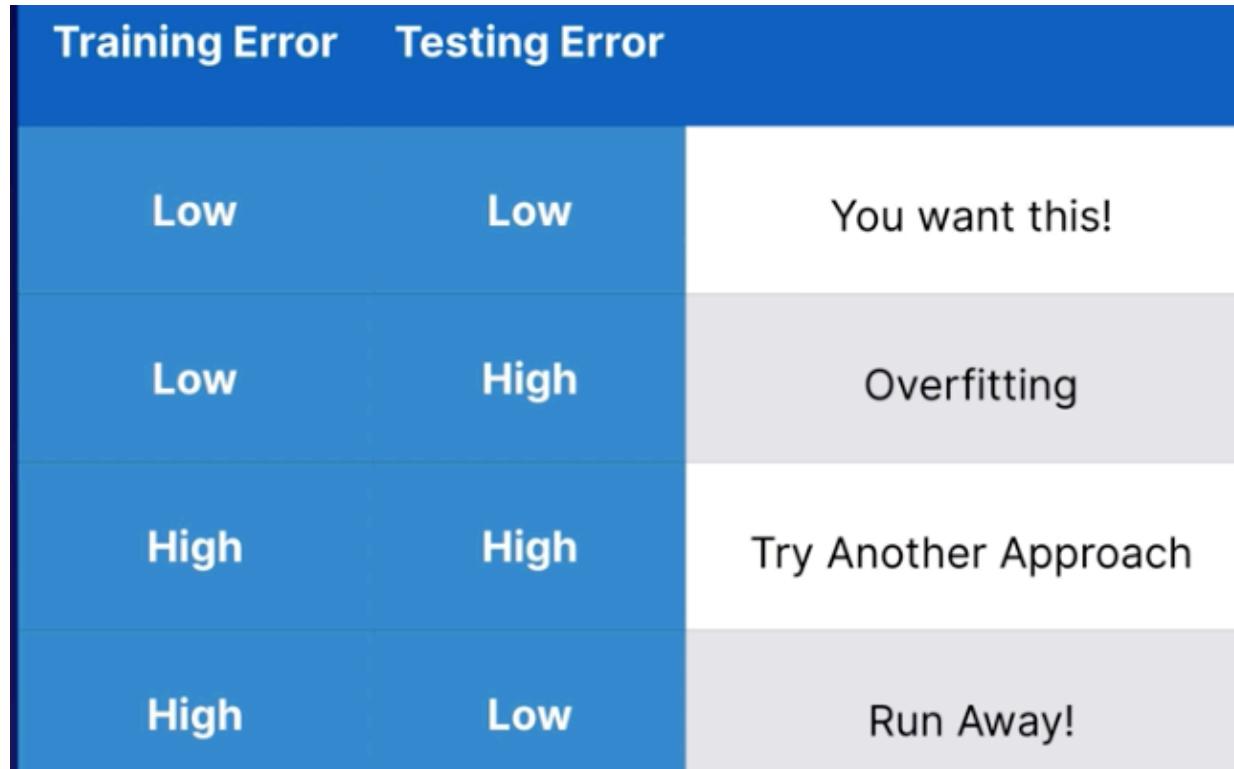
Robust Model

Our model fits the training data but also does reasonably well on new data which it has never seen.

It can "deal" with noise in the data.

It can **generalize** effectively for that new data.

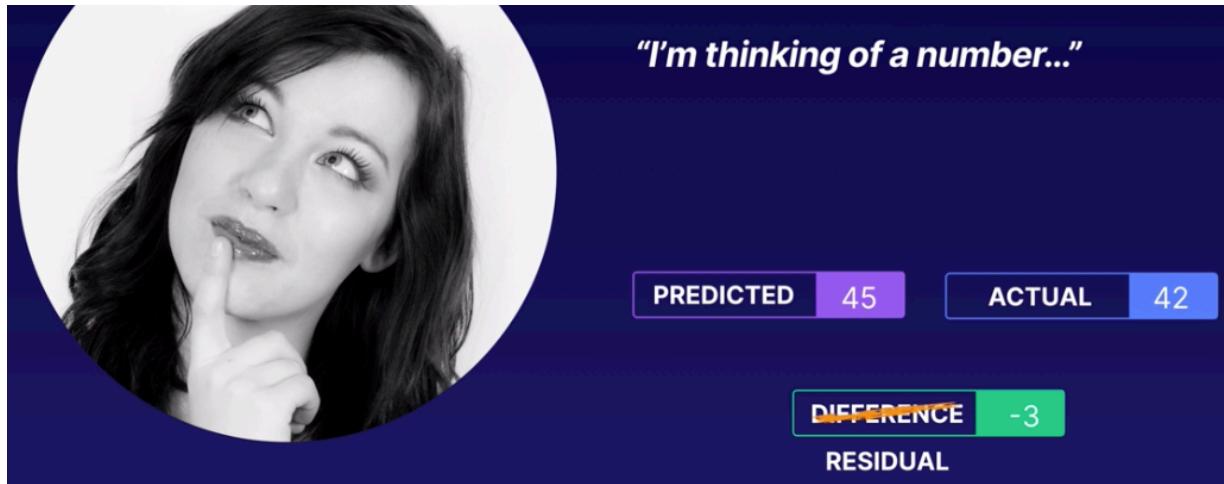




Ways to prevent overfitting:

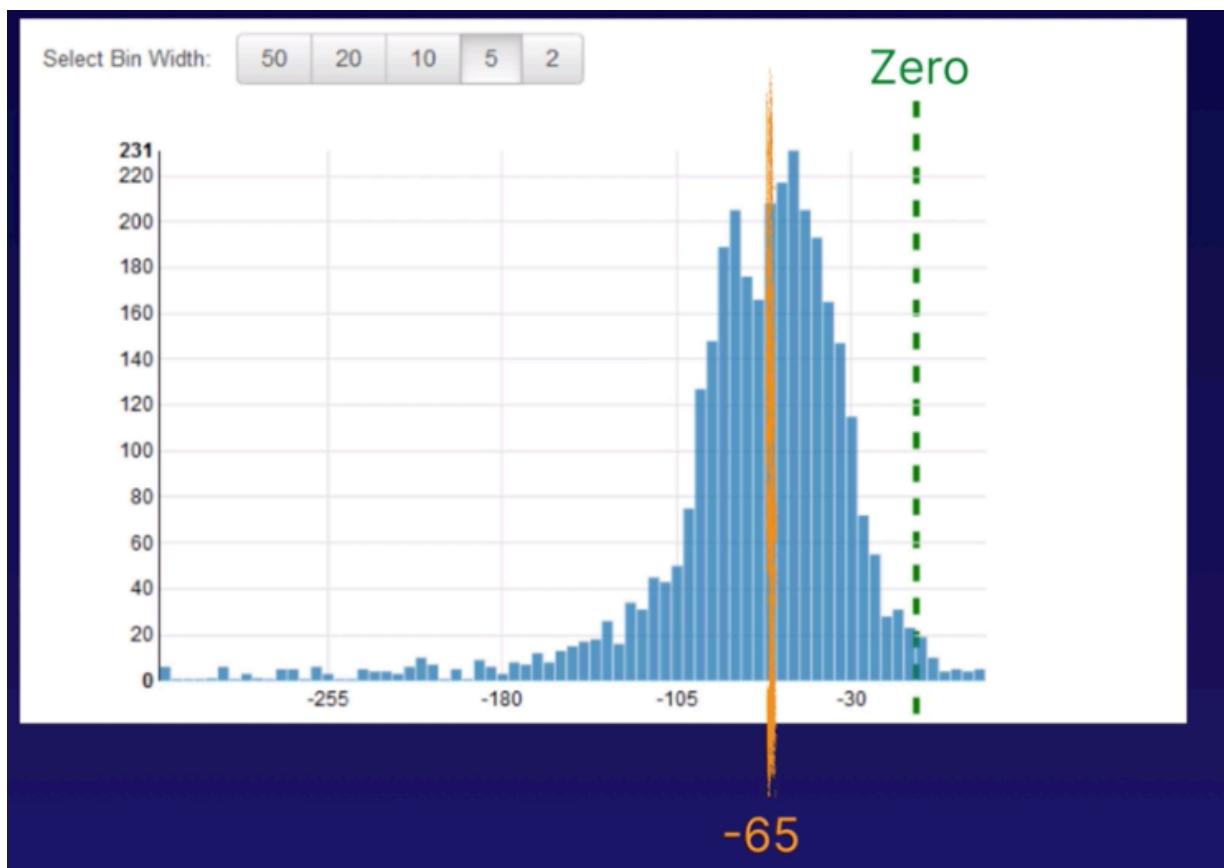
- 1 More Data**
Sometimes more data will provide enough additional entropy to steer your algorithm away from overfitting.
- 2 Early Stopping**
Terminate the training process before it has a chance to "overtrain". Many algorithms include this option as a hyperparameter.
- 3 Sprinkle In Some Noise**
Your training data could be TOO clean and you might need to introduce some noise to generalize the model.
- 4 Regulate!**
Regularization forces your model to be more general by creating constraints around weights or smoothing the input data.
- 5 Ensembles**
Combine different models together to either amplify individual weaker models (boosting) or smooth out strong models (bagging).
- 6 Ditch some Features**
(aka Feature Selection) Too many irrelevant features can influence the model in a negative way by drowning out the signal with noise.

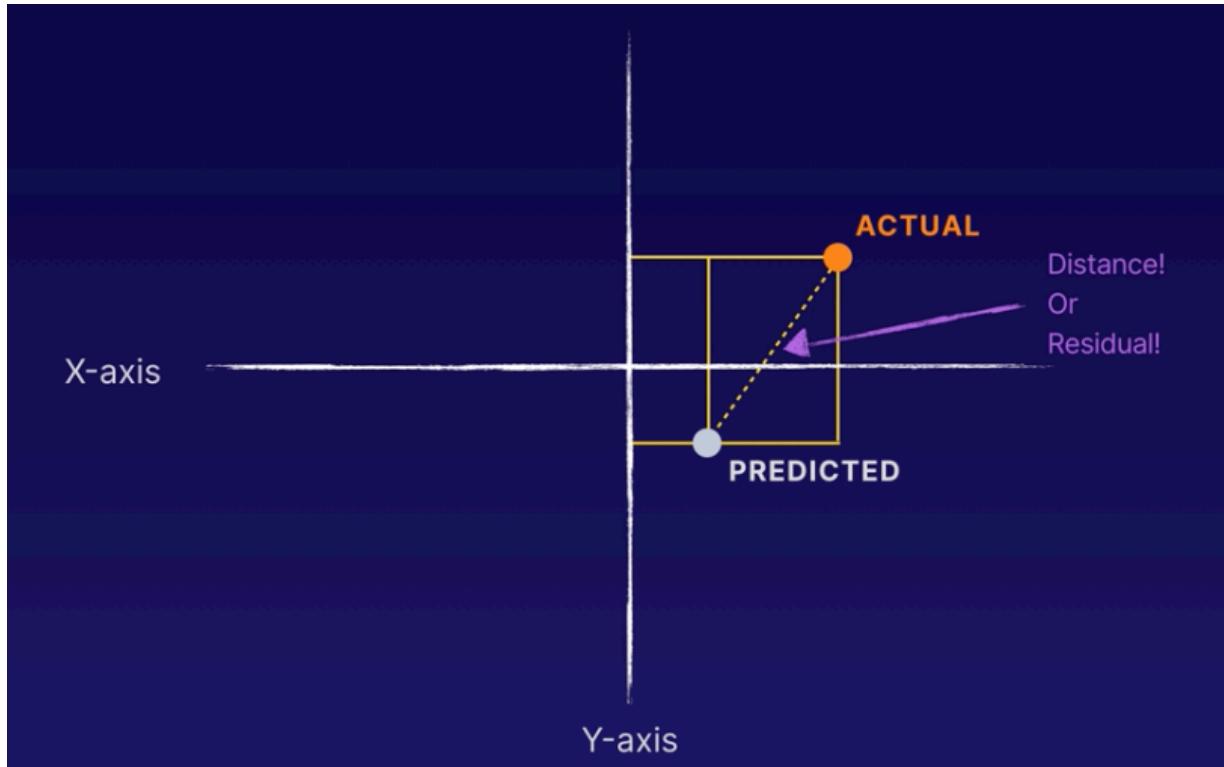
Regression Accuracy => Residual



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

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (actual_i - predicted_i)^2}$$



A measure of the distance between an actual observation and predicted value.

Lower RMSE is better.

Binary Classification Accuracy

Type 1 error => false positives

Type 2 error => false negatives

		Actual Outcome	
		TRUE	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

Binary Classification Accuracy

AWS Services Resource Groups N. Virginia Support

Amazon Machine Learning ML models ml-OzsRQmZDkVw

Evaluation Summary

ID: ev-DVmEEFzwTBy
 Name: Evaluation: ML model: Banking
 Datasource ID: ds-yvwYkRd0QBX
 Output location: Not available
 Creation time: Feb 14, 2019 11:00:37 AM
 Completion time: 5 mins.
 Compute Time (Approximate): 2 mins.
 Status: Completed
 Log: Download log

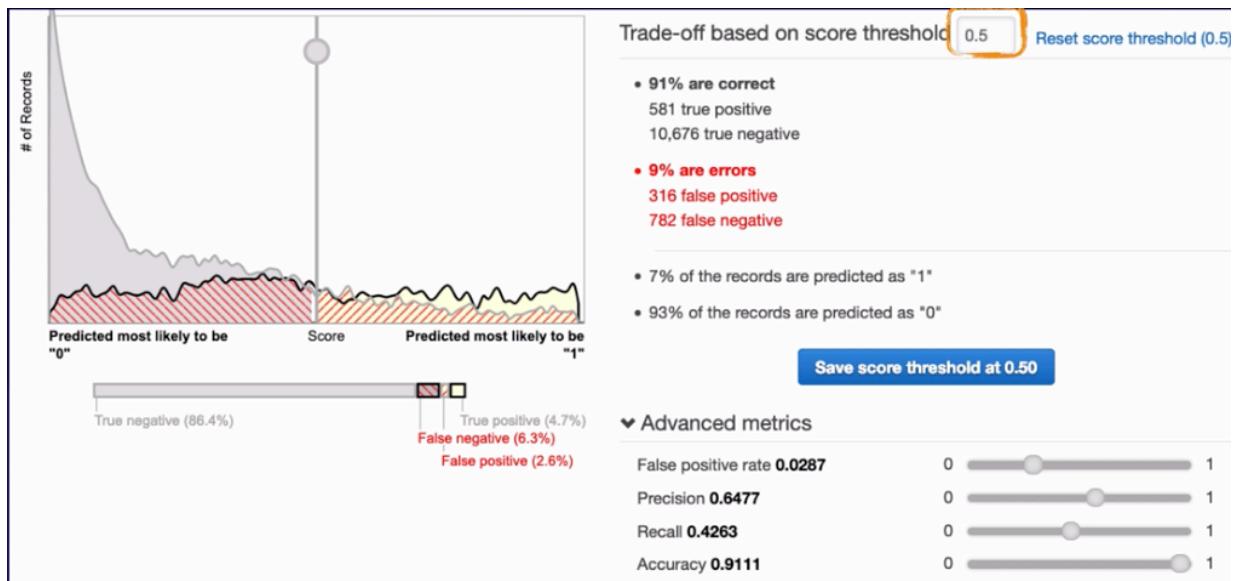
ML model performance metric

On your most recent evaluation, ev-DVmEEFzwTBy, the ML model's quality score is considered extremely good for most machine learning applications. AUC: 0.936
 Baseline AUC: 0.500
 Difference: 0.436

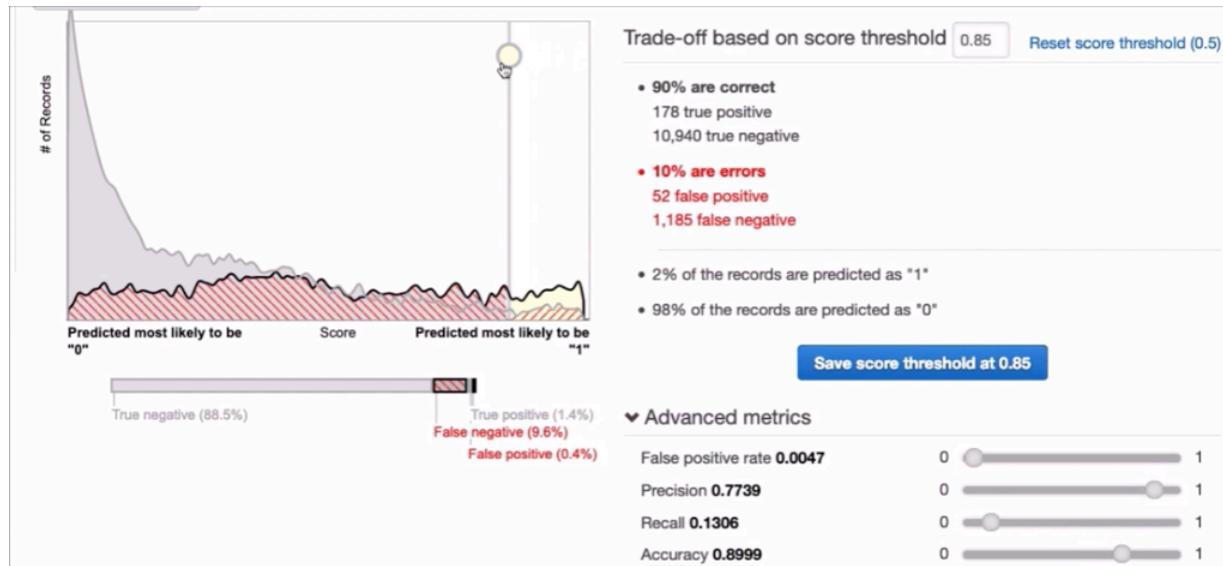
Next step: If you want to use this ML model to generate predictions, explore trade-offs to optimize the performance of your ML model first.

Score threshold: 0.5 | Adjust score threshold | Explore performance

Advanced metrics:



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



`binary_classifier_model_selection_criteria`

When predictor_type is set to binary_classifier, the model evaluation criteria for the validation dataset (or for the training dataset if you don't provide a validation dataset). Criteria include:

- accuracy—The model with the highest accuracy.
- f_beta—The model with the highest F1 score. The default is F1.
- precision_at_target_recall—The model with the highest precision at a given recall target.
- recall_at_target_precision—The model with the highest recall at a given precision target.
- loss_function—The model with the lowest value of the loss function used in training.

Optional

Valid values: accuracy, f_beta, precision_at_target_recall, recall_at_target_precision, or loss_function

Default value: accuracy

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.

F1 Score

A CLOUD GURU

$$\text{Recall} = \frac{\text{We Guessed Right}}{\text{We Guessed Right} + \text{We Should Have Flagged These Too}}$$

Spam Gets Through

$$\text{Precision} = \frac{\text{We Guessed Right}}{\text{We Guessed Right} + \text{We Flagged These but We Were Wrong}}$$

Legitimate Emails Get Blocked

F1Score

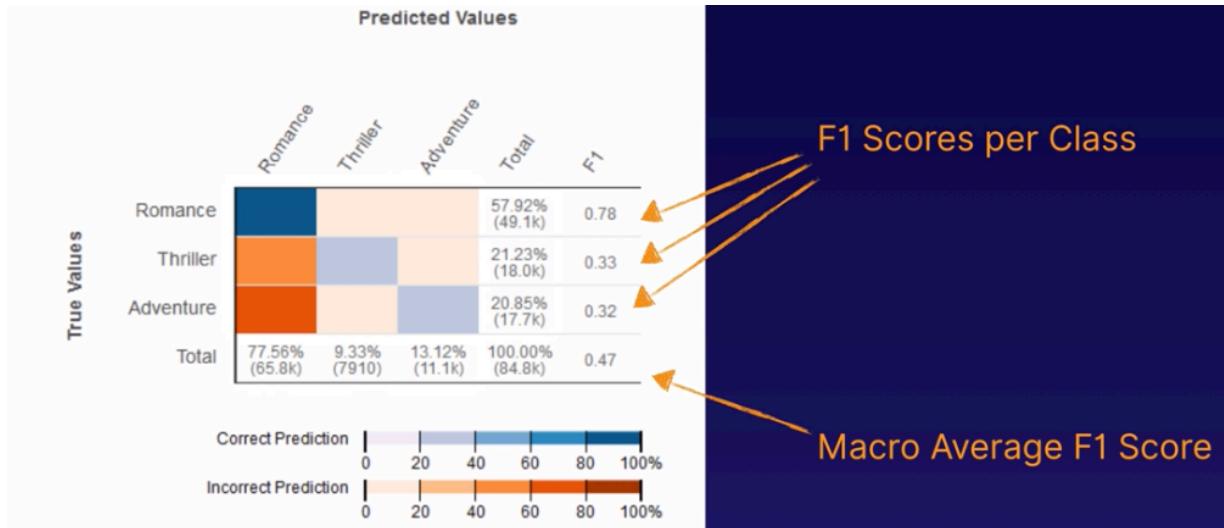
This is a balance between precision and recall

We generally look for a higher value that shows a better predictive accuracy. But that also depends on the use case

$$\text{F1 Score} = \frac{2 * (\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

The balance between Precision and Recall.
A larger value indicates better predictive accuracy.

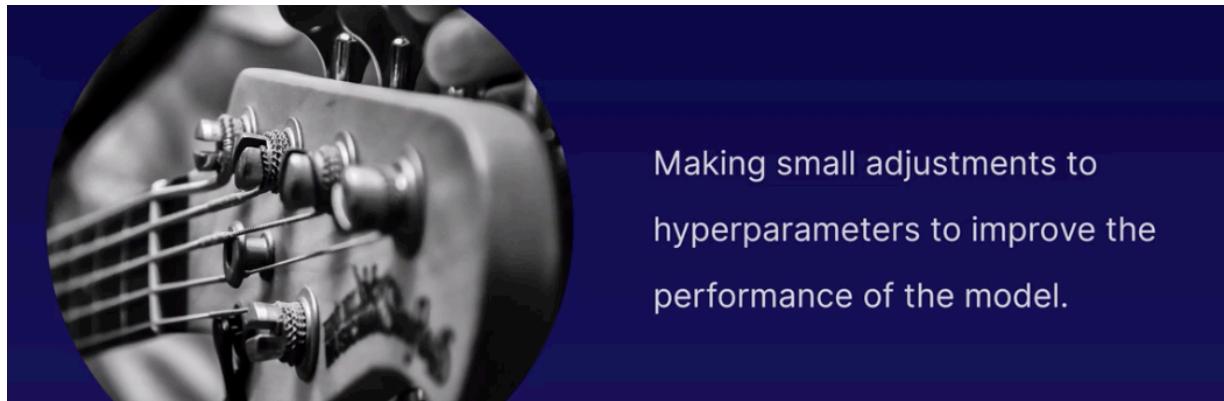
Multiclass classification



Improving Model accuracy



Model Tuning



Making small adjustments to hyperparameters to improve the performance of the model.

Example for Linear Learner hyper parameters

Linear Learner Hyperparameters

AWS Documentation > Amazon SageMaker > Developer Guide > Build a Model > Use Amazon SageMaker Built-in Algorithms > Linear Learner Algorithm > Linear Learner Hyperparameters

Linear Learner Hyperparameters

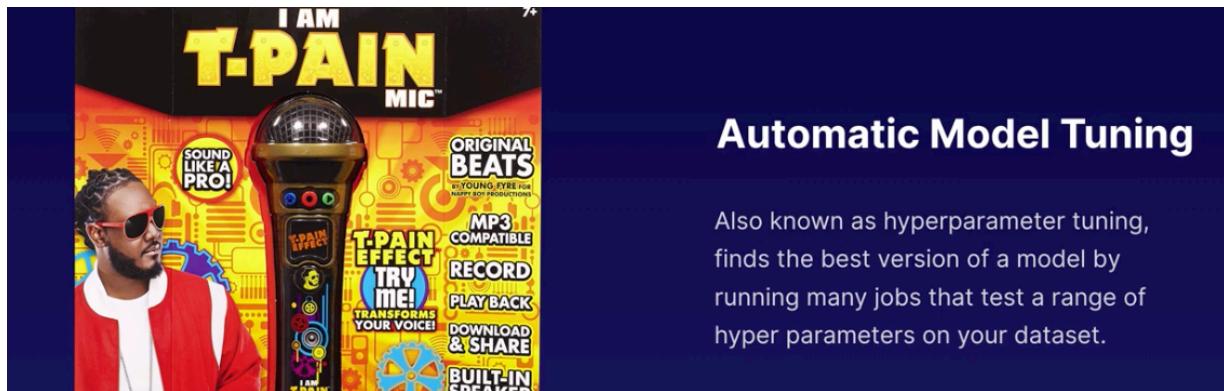
Parameter Name	Description
feature_dim	The number of features in the input data. Required Valid values: Positive integer
num_classes	The number of classes for the response variable. The algorithm assumes that classes are labeled 0, ..., num_classes - 1. Required when predictor_type is multiclass_classifier. 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
balance_multiclass_weights	Specifies whether to use class weights, which give each class equal importance in the loss function. Used only when the predictor_type is multiclass_classifier. Optional Valid values: true, false Default value: false
beta_1	The exponential decay rate for first-moment estimates. Applies only when the optimizer value is adam. Optional Valid values: auto or floating-point value between 0 and 1.0 Default value: auto

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
ll	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

Metrics Computed by the BlazingText Algorithm

The BlazingText Word2Vec algorithm (skipgram, cbow, and batch_skipgram modes) reports on a single metric during training: `train:mean_rho`. This metric is computed on [WS-353 word similarity datasets](#). When tuning the hyperparameter values for the Word2Vec algorithm, use this metric as the objective.

The BlazingText Text Classification algorithm (supervised mode), also reports on a single metric during training: the `validation:accuracy`. When tuning the hyperparameter values for the text classification algorithm, use these metrics as the objective.

Metric Name	Description	Optimization Direction
<code>train:mean_rho</code>	The mean rho (Spearman's rank correlation coefficient) on WS-353 word similarity datasets	Maximize
<code>validation:accuracy</code>	The classification accuracy on the user-specified validation dataset	Maximize

Use for Word2Vec

Use for Text Classification

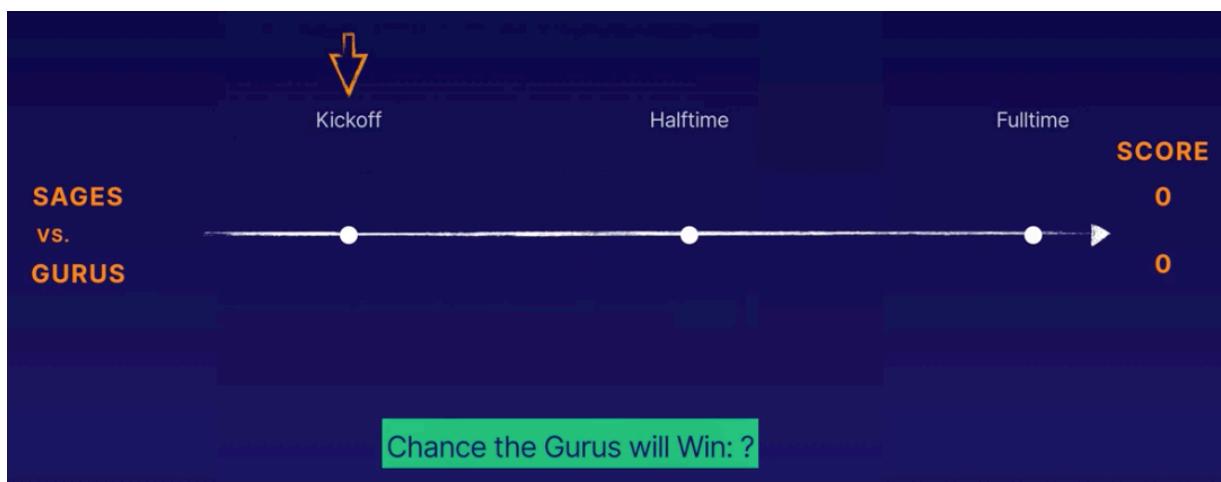
Tunable BlazingText Hyperparameters		
Tunable Hyperparameters for the Word2Vec Algorithm		
Tune an Amazon SageMaker BlazingText Word2Vec model with the following hyperparameters. The hyperparameters that have the greatest impact on Word2Vec objective metrics are: mode, learning_rate, window_size, vector_dim, 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

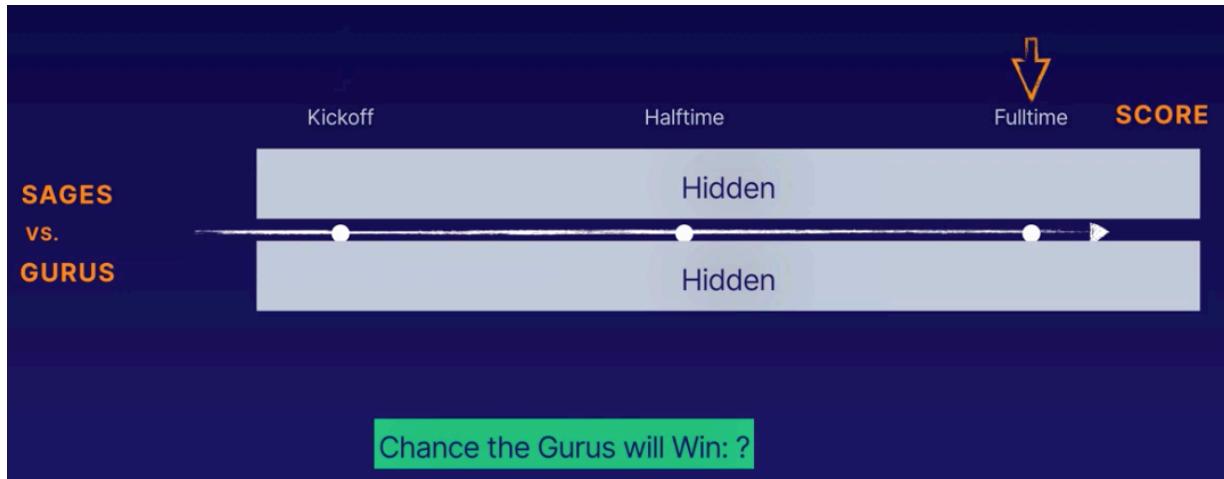
What is the chance Gurus will win?

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

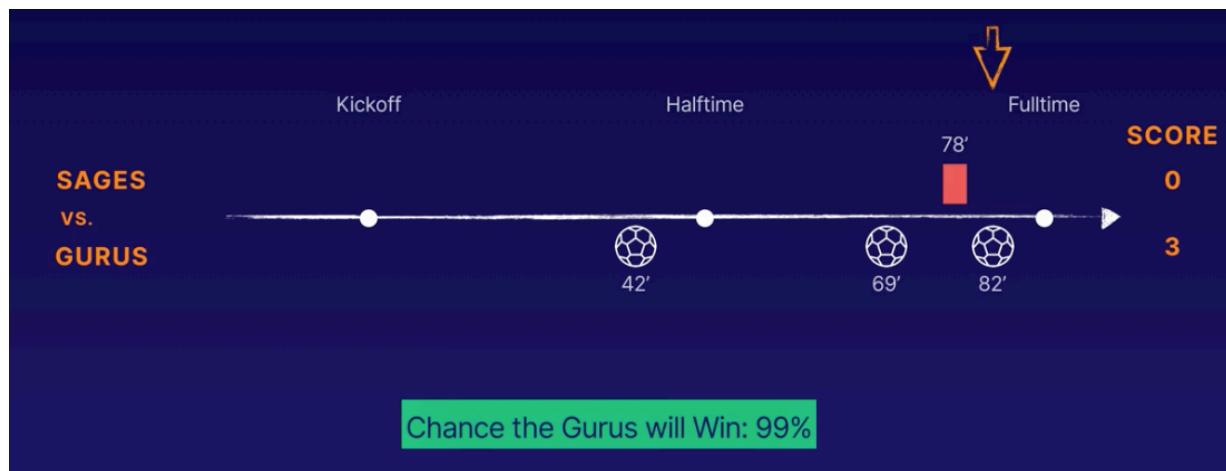
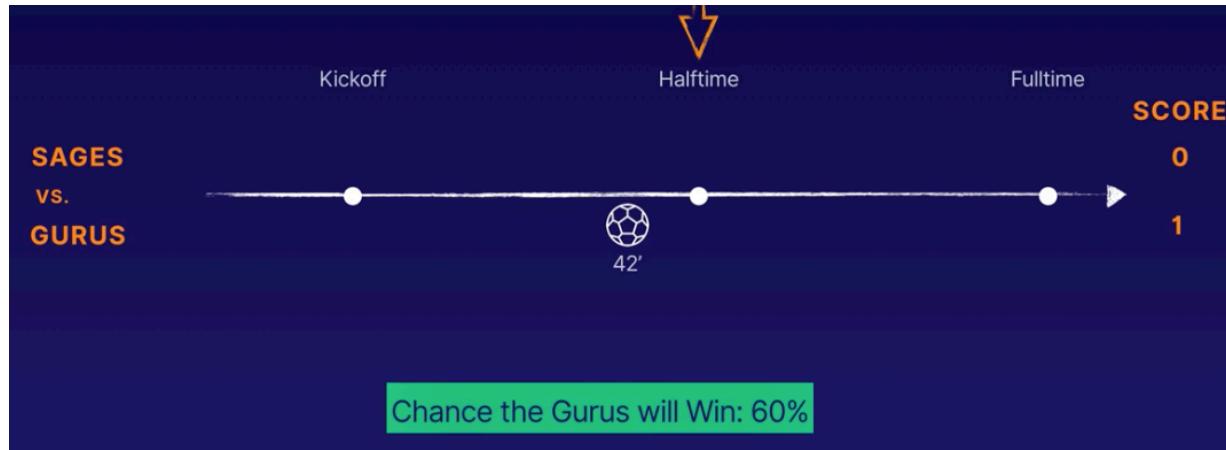


LET's say our target is 99% chances of winning for Gurus
 We can seek that values in a different ways.
 With a matrix trying out values till we find it

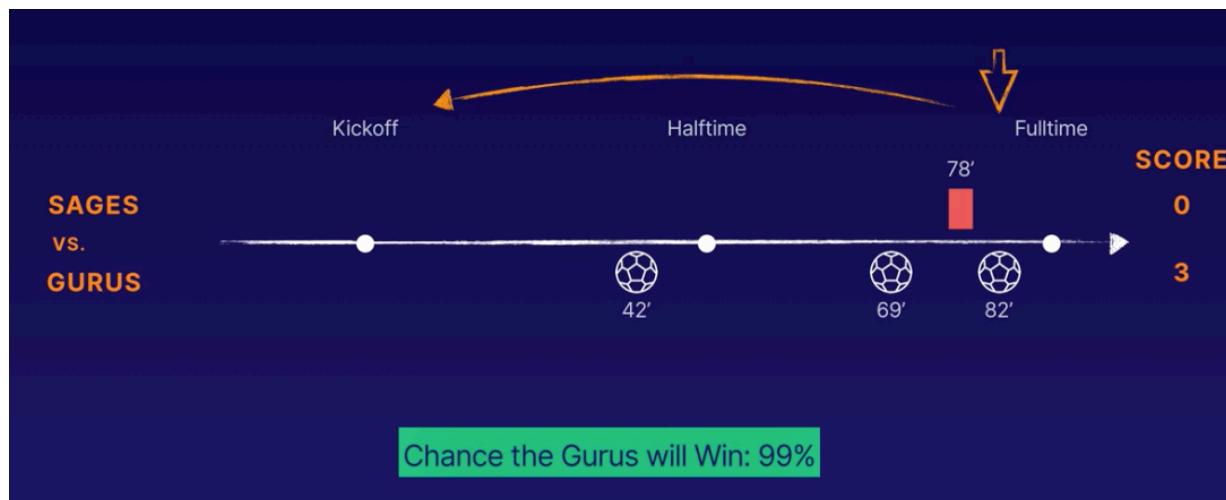
We could also pick values randomly.

MATRIX				RANDOM	
...	0.91	0.92	0.93		
0.94	0.95	0.96	0.97		
0.98	0.99			45	3.4
				3.14	99
				-4.3	24
				1.01	0.5

Bayesian reasoning comes to help there



We can venture 99% because we have the benefit of the **HISTORY** of the game.
=> Crux of **Bayesian**: 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)

Hyperparameter to Tune	Range	Optimization Metric
learning_rate	0.00001 to 1	validation:objective_loss (minimize)
	learning_rate = 0.00001	objective_loss: 1.0
	learning_rate = 0.00010	objective_loss: 0.4
	learning_rate = 0.00100	objective_loss: 0.1

My chances of less loss are better as the learning rate increases.

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

Exam tips:

Evaluation and Optimization	 A CLOUD GURU
<h3>Concepts</h3> <ul style="list-style-type: none">• Remember we want to Generalize...not Memorize.• Understand the difference between Offline Validation and Online Validation.• Know conceptually about a Canary deployment.	
<h3>Monitoring and Analyzing Training Jobs</h3> <ul style="list-style-type: none">• Know the difference between Training metrics and Validation metrics.• Know that CloudWatch integrates well with SageMaker for both logging and metric charting and dashboarding.• Understand how to define custom metrics for custom algorithms.	

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/](#)