

Cloud Guru - 6 - Algorithms part 2

Text Analysis

SM algorithm: **LDA - Latent Dirichlet Allocation**

Latent Dirichlet Allocation (LDA)



Latent Dirichlet Allocation (LDA) algorithm is an unsupervised learning algorithm that attempts to describe a set of observations as a mixture of distinct categories. LDA is most commonly used to discover a user-specified number of topics shared by documents within a text corpus. Here each observation is a document, the features are the presence (or occurrence count) of each word, and the categories are the topics.



Used to figure out how similar documents are based on the frequency of similar words.

Use Cases

- **Article Recommendation**

Example: Recommend articles on similar topics which you might have read or rated in the past.

- **Musical Influence Modeling**

Example: Explore which musical artists over time were truly innovative and those who were influenced by those innovators.

Other SM Algorithm: **NTM (Neural Topic Model)**



Neural Topic Model

UNSUPERVISED

Unsupervised learning algorithm that is used to organize a corpus of documents into topics that contain word groupings based on their statistical distribution. Topic modeling can be used to classify or summarize documents based on the topics detected or to retrieve information or recommend content based on topic similarities.



Similar uses and function to LDA in that both NTM and LDA can perform topic modeling. However, NTM uses a different algorithm which might yield different results than LDA.

Other SM Also: **seq2seq (Sequence to Sequence)**



Sequence to Sequence

SUPERVISED

Supervised learning algorithm where the input is a sequence of tokens (for example, text, audio) and the output generated is another sequence of tokens.



Think a language translation engine that can take in some text and predict what that text might be in another language. We must supply training data and vocabulary.

1

Steps consist of embedding, encoding and decoding

Using a neural network model (RNN and CNN), the algorithm uses layers for embedding, encoding and decoding into the target.

2

Commonly initialized with pre-trained word libraries.

A standard practice is initializing the embedding layer with a pre-trained word vector like FastText or Glove or to initialize it randomly and learn the parameters during training.

3

Only GPU instances are supported.

Currently Amazon SageMaker seq2seq is only supported on GPU instance types and is only set up to train on a single machine. But it does also offer support for multiple GPUs.

Use cases of seq2seq:

- **Language Translations**

Example: Using a vocabulary, predict the translation of a sentence into another language.

- **Speech to Text**

Example: Given an audio “vocabulary”, predict the textual representation of spoken words.

Other Also: BlazingText (based on FastText from FB)



BlazingText

Highly optimized implementations of the Word2vec and text classification algorithms. The Word2vec algorithm is useful for many downstream natural language processing (NLP) tasks, such as sentiment analysis, named entity recognition, machine translation, etc.

SUPERVISED
UNSUPERVISED



Really, really optimized way to determine contextual semantic relationships between words in a body of text.

BlazingText Modes:

Modes	Word2Vec (Unsupervised)	Text Classification (Supervised)
Single CPU Instance	Continuous Bag of Words Skip-gram Batch Skip-gram	Supervised
Single GPU Instance (1 or more GPUs)	Continuous Bag of Words Skip-gram	Supervised with 1 GPU
Multiple CPU Instances	Batch Skip-gram	None

① Expects single pre-processed text file Each line in the file should contain a single sentence. If you need to train on multiple text files, concatenate them into one file and upload the file in the	② Highly Scalable Improves on traditional Word2Vec algorithm by supporting scale-out for multiple CPU instances. FastText text classifier can leverage GPU acceleration.	③ Around 20x faster than FastText Supports pre-trained FastText models but also can perform training about 20x faster than FastText.
--	---	---

Use Cases for BlazingText:

AWS encapsulates these services into other services:

<ul style="list-style-type: none"> Sentiment Analysis Example: Evaluate customer comments in social media posts to evaluate whether they have a positive or negative sentiment. 	 Amazon Comprehend
<ul style="list-style-type: none"> Document Classification Example: Review a large collection of documents and detect whether the document should be classified as containing sensitive data like personal information or trade secrets. 	 Amazon Macie

One other also: Object2Vec



Object2Vec

SUPERVISED

General-purpose neural embedding algorithm that can learn low-dimensional dense embeddings of high-dimensional objects while preserving the semantics of the relationship between the pairs in the original embedding space.



A way to map out things in a d-dimentional space to figure out how similar they might be to one another.

WORD2VEC

sad

upset

angry

lonely

scared

frightened

happy

nervous

appreciative

Object2Vec is similar to Word2Vec

OBJECT2VEC



Based on ratings between person and album, and ultimately relationship, we can group things together and recommend music albums based on people#4 previous choices



Object2Vec :

1	2	3
Expect Pairs of Things	Feature Engineering	Training Data Required
Looking for pairs of items and whether they are "positive" or "negative" from a relationship standpoint. Accepts categorial labels or rating/score-based labels.	Embedding can be used for downstream supervised tasks like classification or regression.	Officially, Object2Vec requires labeled data for training, but there are ways to generate the relationship labels from natural clustering.

Use Cases:

- **Movie Rating Prediction**

Example: Predict the rating a person is likely to give a movie based on similarity to other's movie ratings.

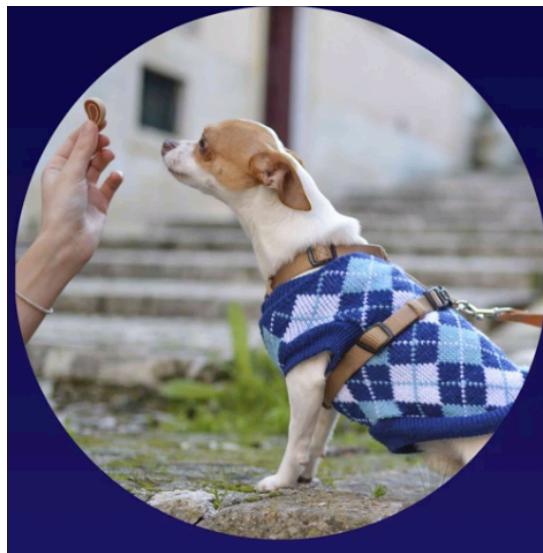
- **Document Classification**

Example: Determine which genre a book is based on its similarity to known genres (history, thriller, biography, etc.)

Reinforcement Learning

We can use 2 strategies:

- positive reinforcement
- negative reinforcement



Reinforcement Learning

"The carrot and the stick."

Positive

Provide a positive reward thereby motivating the subject to repeat the behavior, presumably for another positive reward.

Negative

Provide a displeasurable experience or response thereby motivating the subject to NOT repeat the undesired behavior.

Goal is to maximize reward

Reward is based on the behavior of a subject/agent



Reinforcement Learning

Reinforcement Learning (RL) is a machine learning technique that attempts to learn a strategy, called a policy, that optimizes for an agent acting in an environment. Well suited for solving problems where an agent can make autonomous decisions.



Find the path to the greatest reward.

Markov Decision Process: MDP

An **Agent** is placed into an **Environment**

Goal is to get to candy (to note a goal is not required in reinforcement learning) => Reward

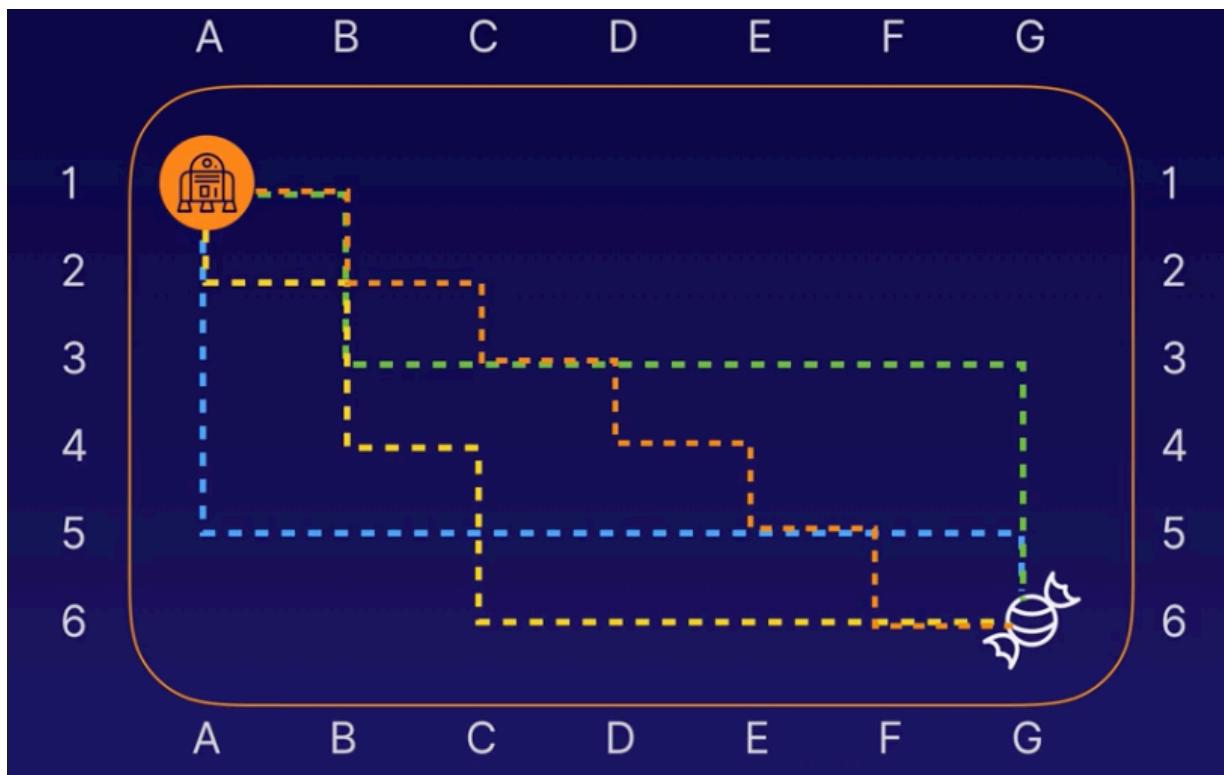
Current **State** (no need to know how we got there)

Then we have **actions** that an Agent can perform (go up, down, right,...)

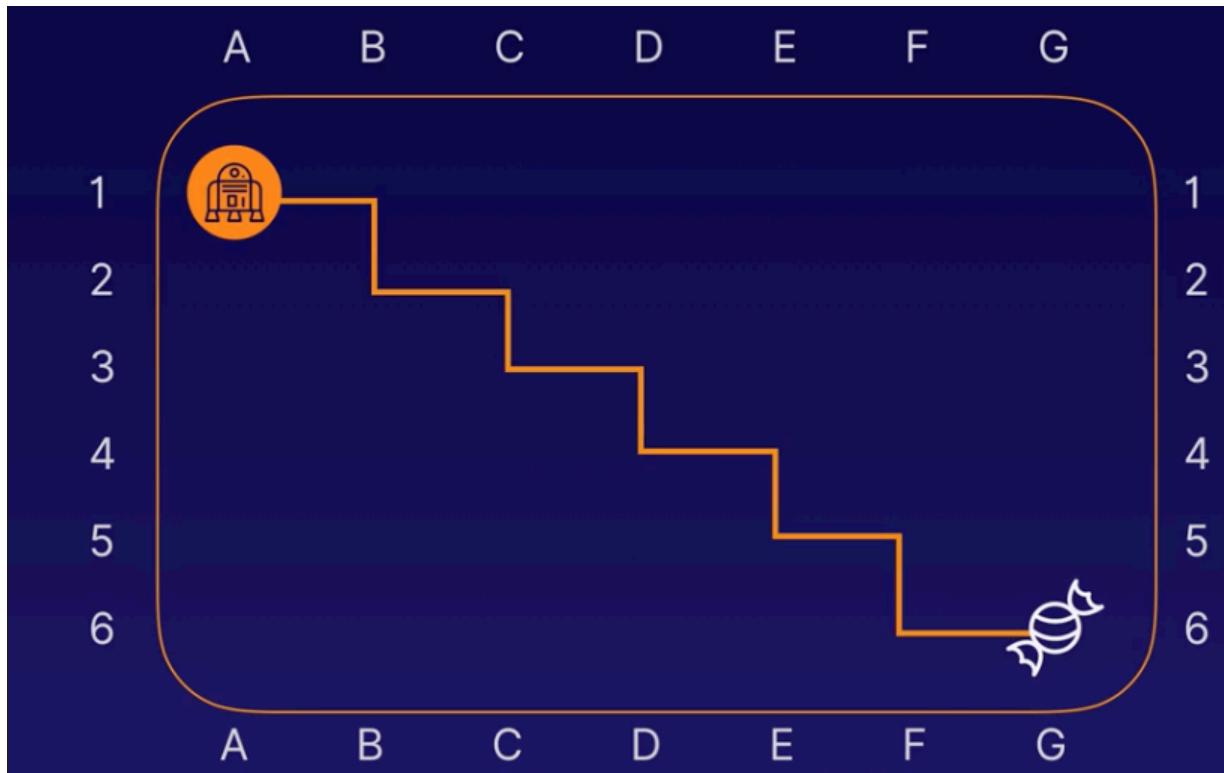
Agent access **Observation**.

Agent	A	B	C	D	E	F	G	
Environment								
Reward	1	6	6	6	6	6	6	1
State	1							
Action	2	6	7	7	7	7	7	2
Observation	3		8	8	8	8	8	3
	4	5	6	7	8	9	9	4
	5	5	6	7	8	9	10	5
	6	5	6	7	8	9	10	6

Then we have **episodes** (iterations) - there can be many path to get to goal.
 Model is trying to figure out the strategy to maximize reward in the least number of steps



Optimal path;



In developing that optimal path, it will learn lessons that will help him make those decision in the future, given a similar objective.

This is called the **Policy** - the decision making part of the reinforcement learning model.

A good example in AWS: Deepracer project

DeepRacer
A CLOUD GURU



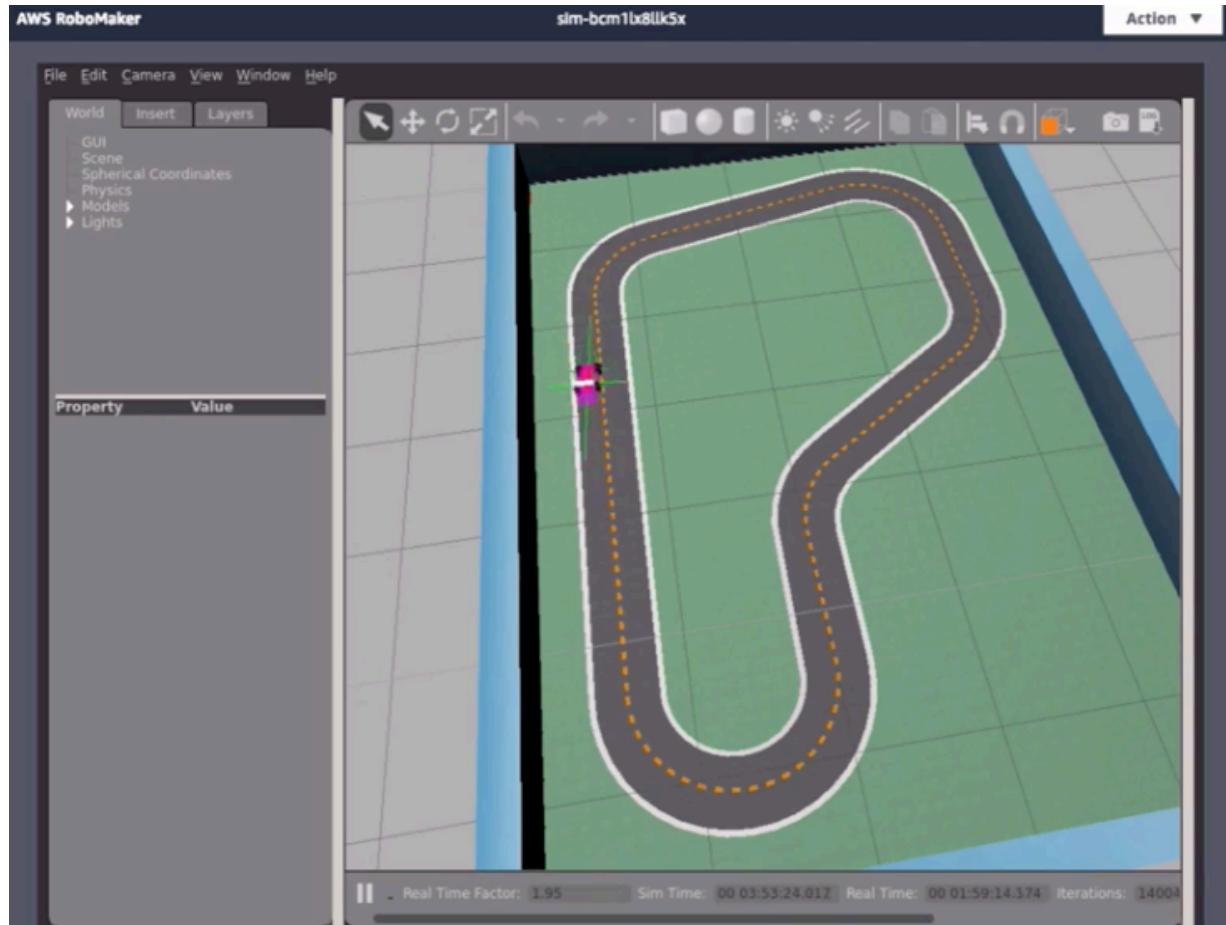

```
def reward_function(self, on_track, x, y, distance_from_center, car_orientation,
progress, steps, throttle, steering, track_width, waypoints, closest_waypoints):

    reward = 1e-3
    marker_1 = 0.1 * track_width
    marker_2 = 0.2 * track_width
    marker_3 = 0.4 * track_width

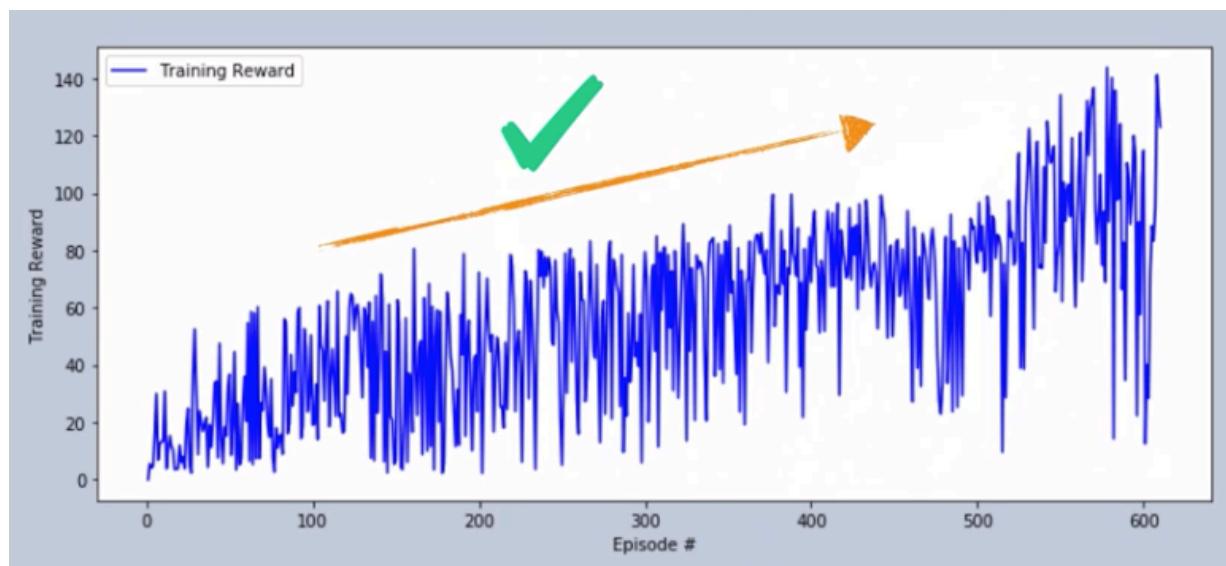
    if distance_from_center >= 0.0 and distance_from_center <= marker_1:
        reward = 1
    elif distance_from_center >= marker_1 and distance_from_center <= marker_2:
        reward = 0.5
    elif distance_from_center >= marker_2 and distance_from_center <= marker_3:
        reward = 0.1
    else:
        reward = 1e-3 # likely crashed/ close to off track

    return reward
```

Example of simulation



Car learns that if it stays closer to the center of track, it will get a higher reward
Graph of cumulative reward



It's cumulative which means the **policy** is continuing to evolve and develop and earn more reward

Use cases of Reinforcement learning:

- **Autonomous Vehicles**

Example: A self-driving car model can “learn” to stay on the road through iterations of trial and error in a simulation. Once the model is good enough, it can be tested in a real vehicle on a test track.

- **Intelligent HVAC Control**

Example: An RL model can learn patterns and routines of building occupants, impact of sunlight as it transitions across the sky and equipment efficiency to optimize the temperature control for lowest energy consumption.

Forecasting

Imense number of variable for what a market believe is a fair price for a company price

Forecasting

A CLOUD GURU



In this case, the stock peaked out and started a downstream trend



Illustration of: Past performance is not indicative of future results



SM forecasting algo: DeepAR

Can predict point and time values and estimated values over some time



DeepAR

SUPERVISED

Forecasting algorithm for scalar times series using recurrent neural networks (RNN). DeepAR outperforms standard autoregressive integrated moving average (ARIMA) and exponential smoothing (ETS) by training a single model over multiple time series as opposed to each individual time series.



Can predict both point-in-time values and estimated values over a timeframe by using multiple sets of historic data.

Use case: we are a shoe company and want to forecast how many we can sell.
PB: it's a brand new show => Cold Start pb



How many can we expect to sell?

Cold Start Problem

Little or no history to use for building a forecasting model.

We could take an existing show and assume it may be the same pattern.
Pb is - it's a brand new product.
Maybe it has the characteristic of multiple shoes to create an aggregate forecast



Forecast Type	Example
Point Forecast	"Number of sneakers sold in a week is X"
Probabilistic Forecast	"Number of sneakers sold in a week is between X and Y with Z% probability."

1	Support for various time series Time series can be numbers, counts, or values in an interval (such as temperature readings between 0 and 100.)	4	You must supply some hyperparameters Context Length, Epochs, Prediction Length and Time Frequency are required hyperparameters.
2	More time series is better Recommend training a model on as many time series as are available. DeepAR really shines with hundreds of related time series.	5	Automatic Evaluation of the Model Uses a backtest after training to evaluate the accuracy of the model automatically.
3	Must supply at least 300 observations DeepAR requires a minimum number of observations across all time series.		

- **Forecasting New Product Performance**

Example: Incorporate historical data from other products to create a model that can predict performance of a newly released product.

- **Predict Labor Needs for Special Events**

Example: Use labor utilization rates at other distribution centers to predict the required level of staffing for a brand new distribution center.

Ensemble Learning



Ensemble Learning

Using multiple learning algorithms and models collectively to hopefully improve the model accuracy.

SM Extreme Gradient Boosting - SGBoost

Swiss Army knife

XGBoost



Open-source implementation of the gradient boosted trees algorithm that attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models.

SUPERVISED



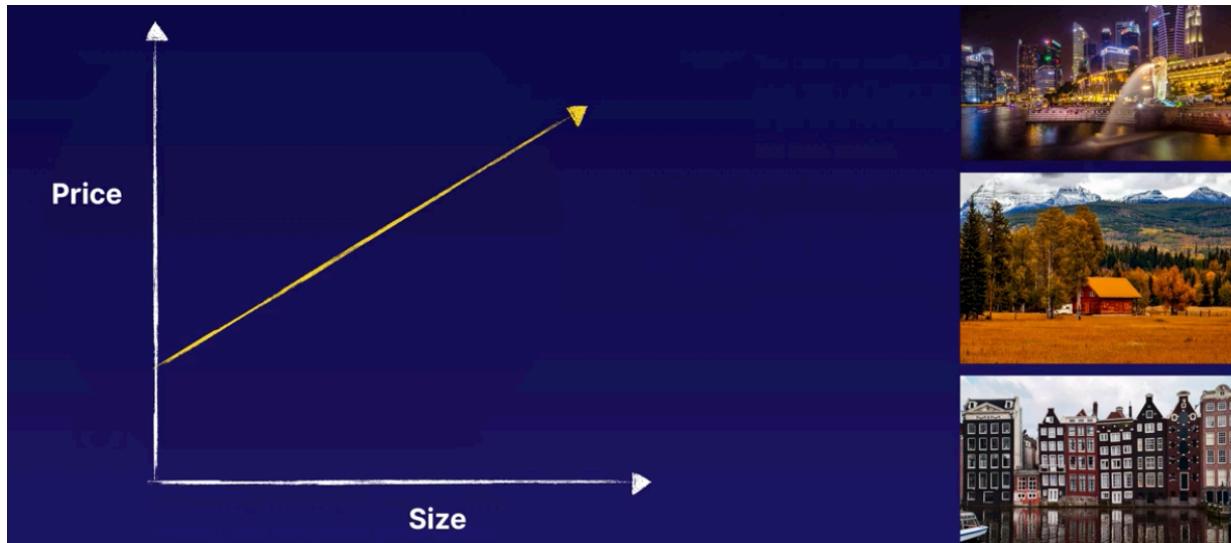
A virtual “Swiss army knife” for all sorts of regression, classification and ranking problems, with 2 required and 35 optional hyperparameters to tune.

<p>1 Accepts CSV and libsvm for training and inference</p> <p>Uses tabular data with rows representing observations, one column representing the target variable or label and the remaining columns representing features.</p>	<p>4 Spark Integration</p> <p>Using the SageMaker Spark SDK, you can call XGBoost direct from within the Spark environment.</p>
<p>2 Only trains on CPU and memory bound.</p> <p>Currently only trains on CPU instances and is memory-bound as opposed to compute-bound.</p>	
<p>3 Recommend lots of memory.</p> <p>AWS recommends using an instance with enough memory to hold the entire training data for optimal performance.</p>	

Example - price house when selling it

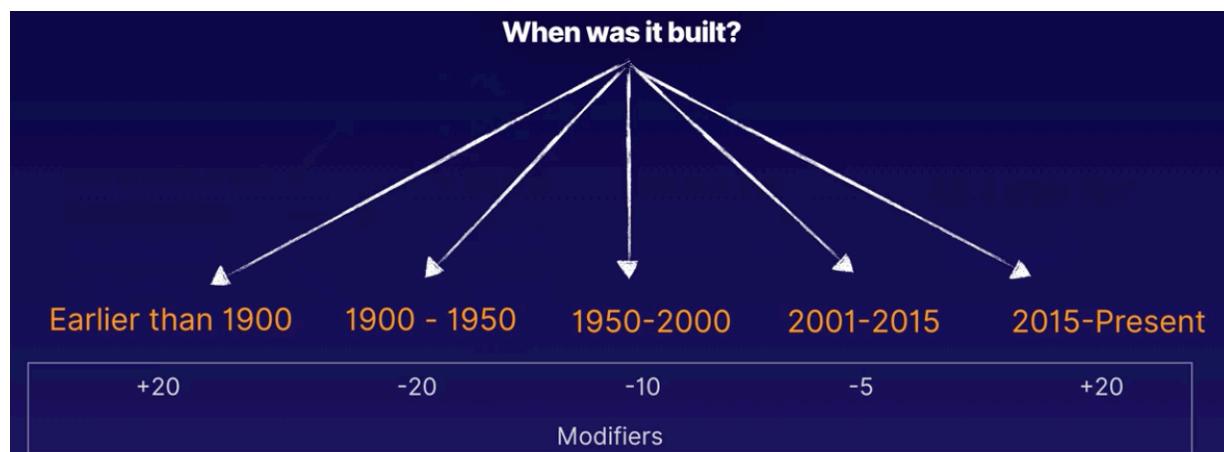
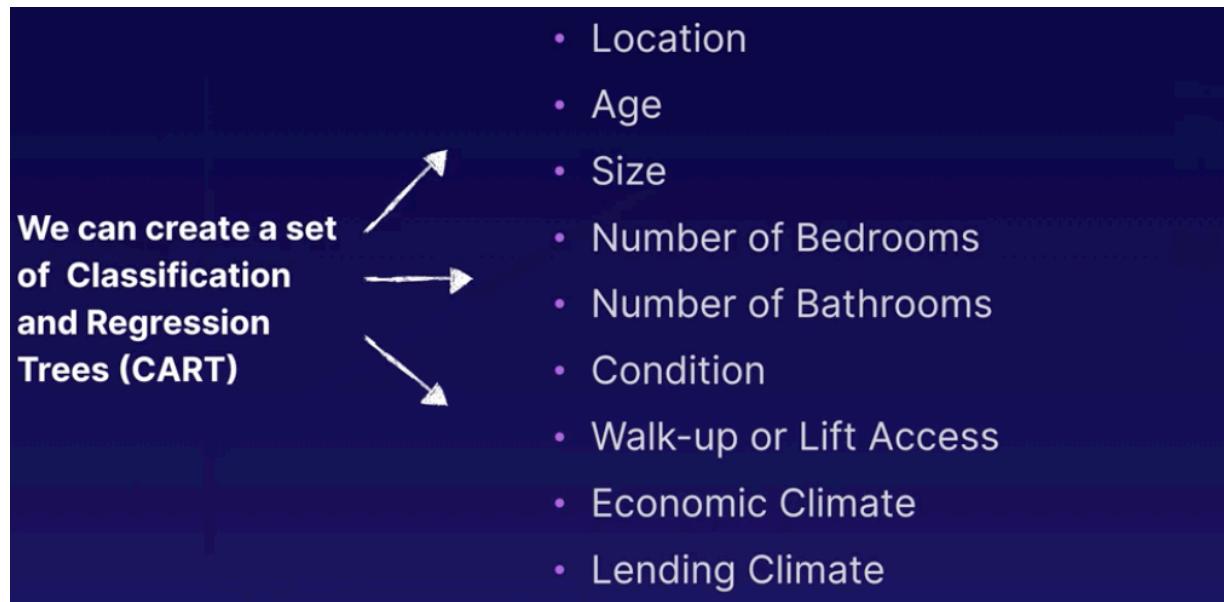


Price is not just related to size



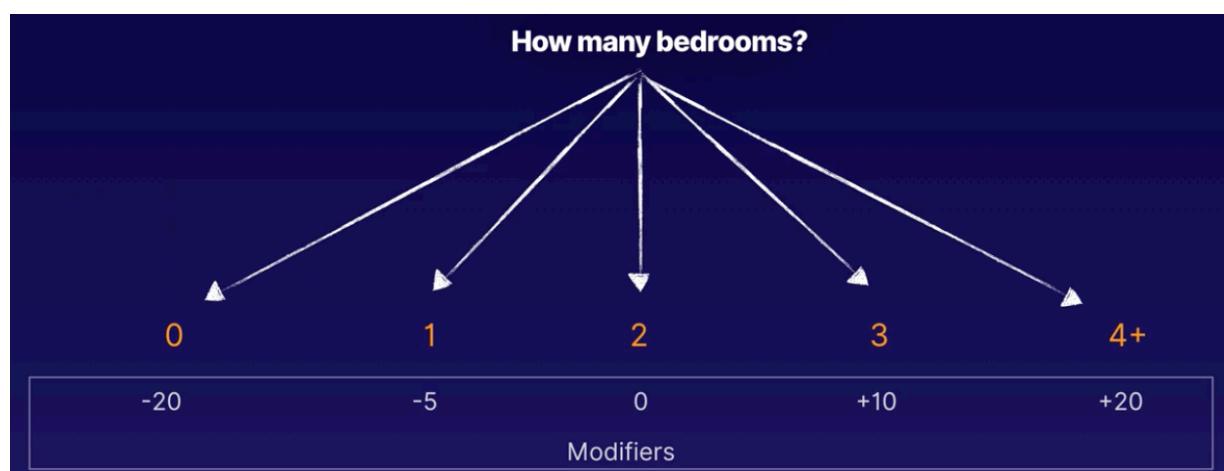
Need to account for many other things to have an accurate predictor.
In a decision Tree Ensemble, we can create a set of Classification and Regression

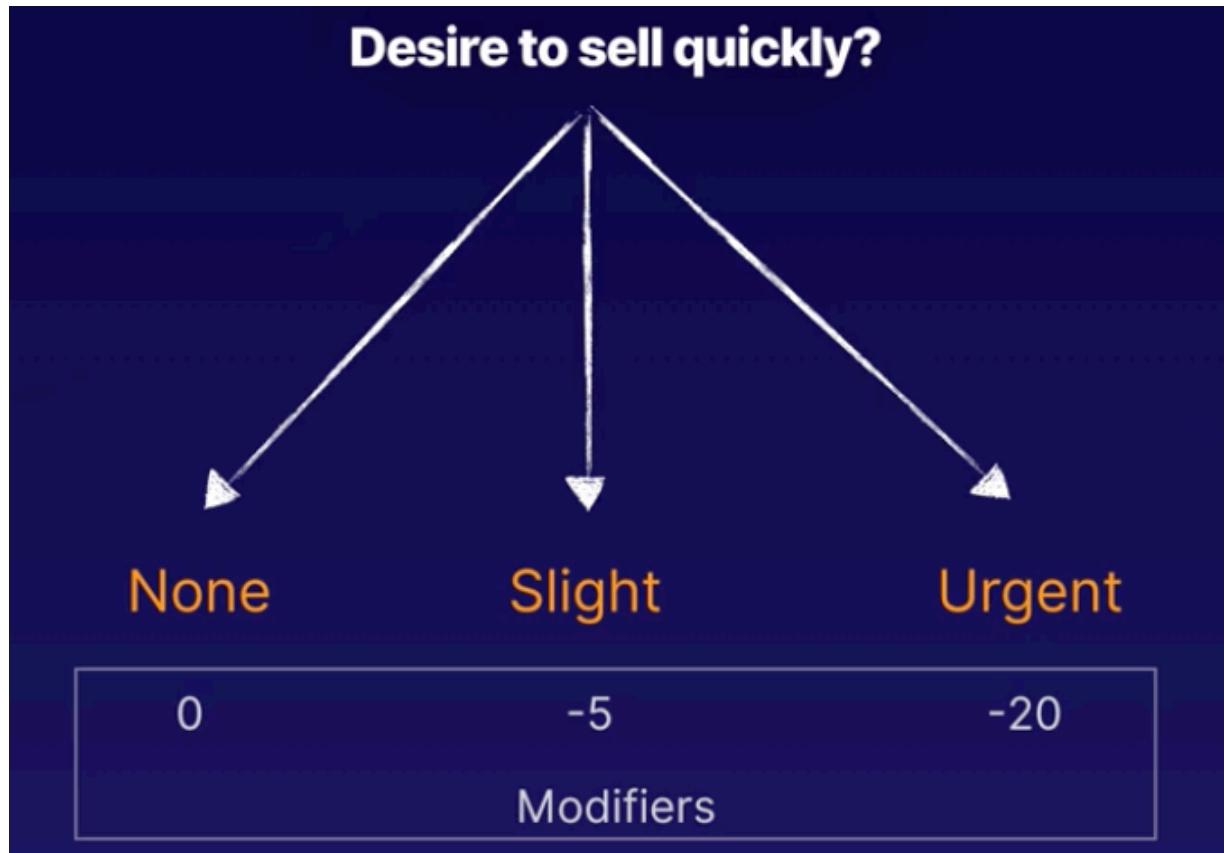
Trees, known as CART



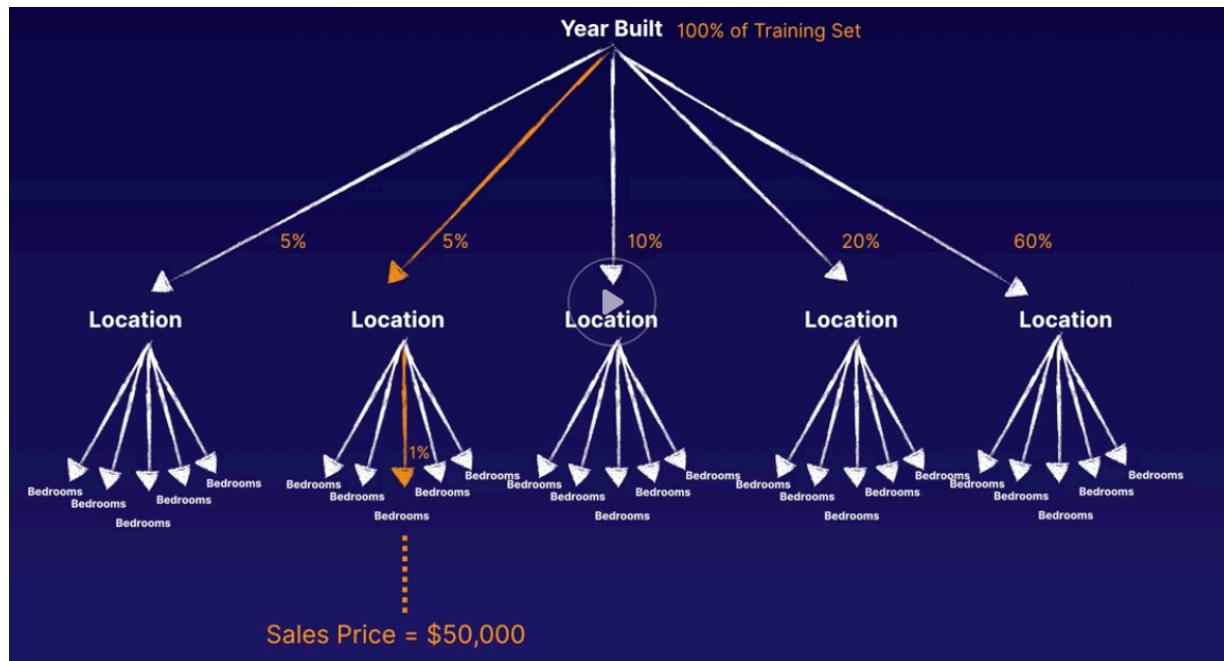
Add modifiers:

If house is very old, it could be desirable => +20





We then have all these trees tacked one on the other



Each tree has a modifier

Tree 1	Year Built	-10
Tree 2	Location	+20
Tree 3	Number of Bedrooms	-5
Tree 4	Number of Bathrooms	+1
Tree 5	Economic Condition	-5
Tree 6	Desire to Sell Quickly	-20

Using multiple trees to create a more realistic estimation model than any one tree could have provided.

Home Value Modifier -19

Asking Price = Median Home Price * (100 - 19)

When all you know is a hammer, everything looks like a nail.



=> pick the algorithm that gives us the best performance

Use cases of XGboost

- **Ranking**

Example: On an e-commerce website, you can leverage data about search results, clicks, and successful purchases, and then use XGBoost to train a model that can return relevance scores for searched products.

- **Fraud Detection**

Example: When XGBoost is given a dataset of past transactions and whether or not they were fraudulent, it can learn a function that maps input transaction data to the probability that transaction was fraudulent.