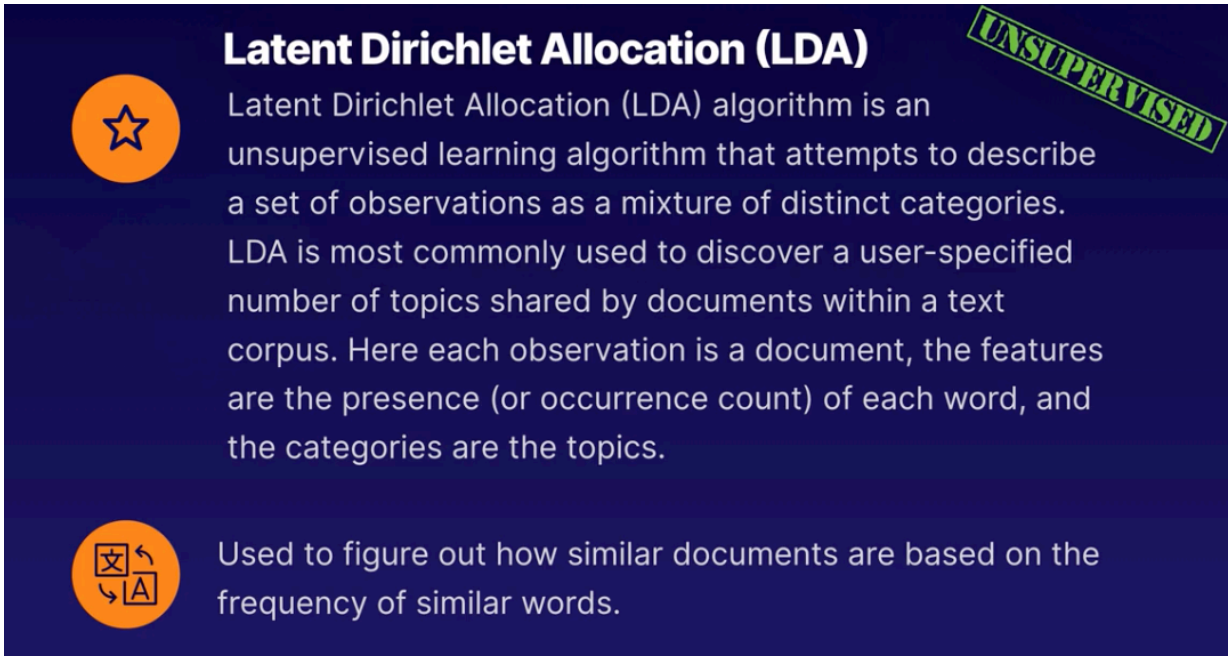
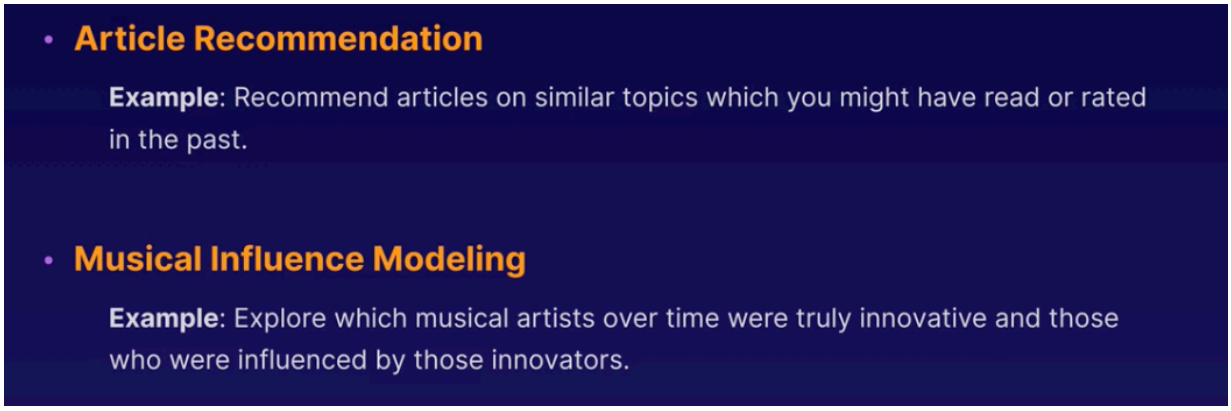
Cloud Guru - 6 - Algorithms part 2

Text Analysis

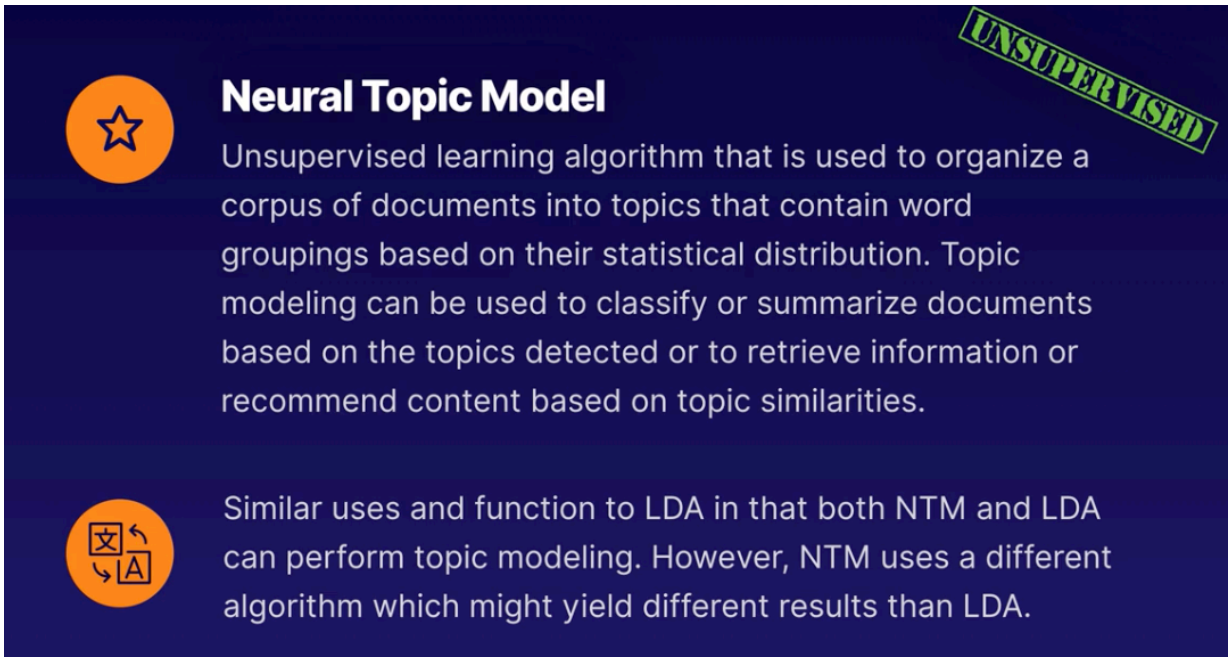
SM algorithm: **LDA - Latent Dirichlet Allocation**



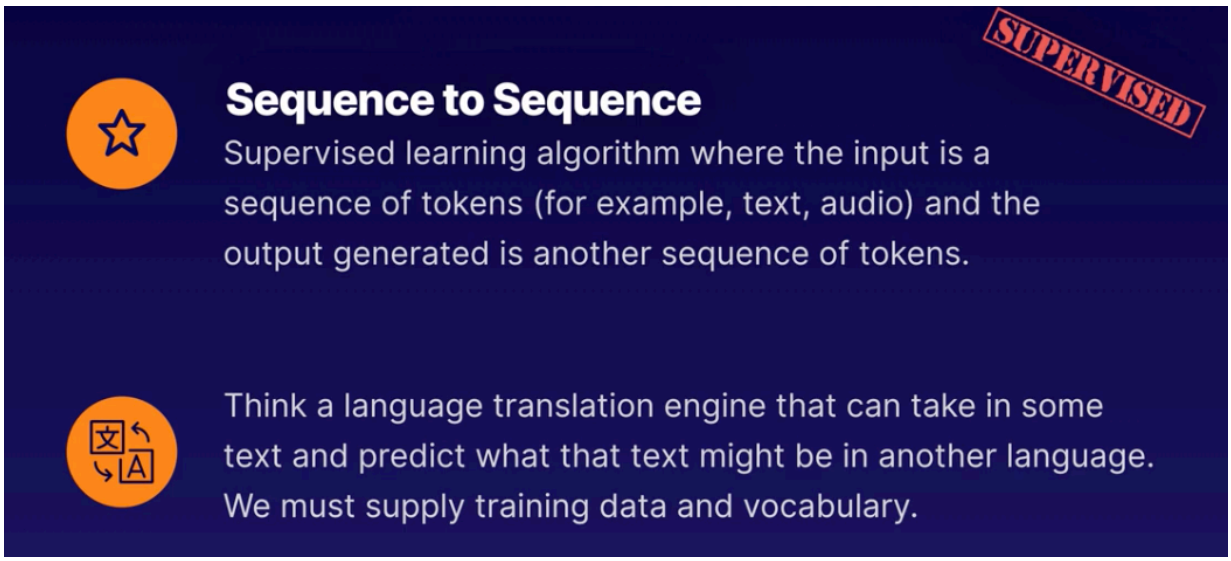
Use Cases

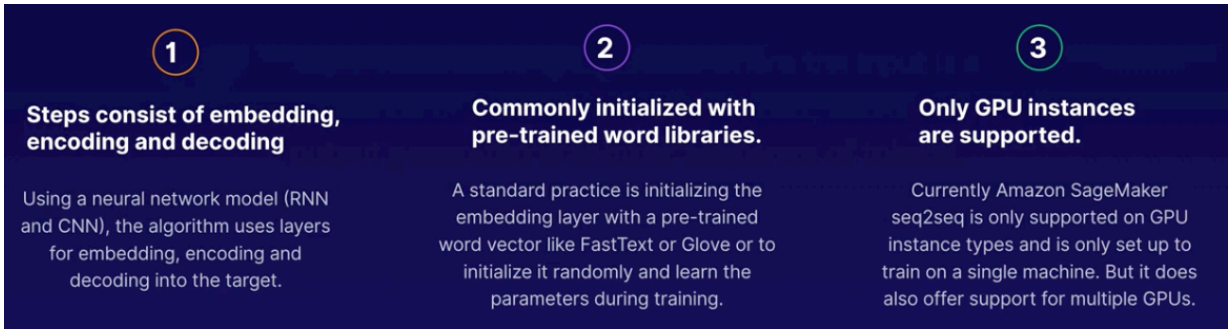


Other SM Algorithm: NTM (Neural Topic Model)

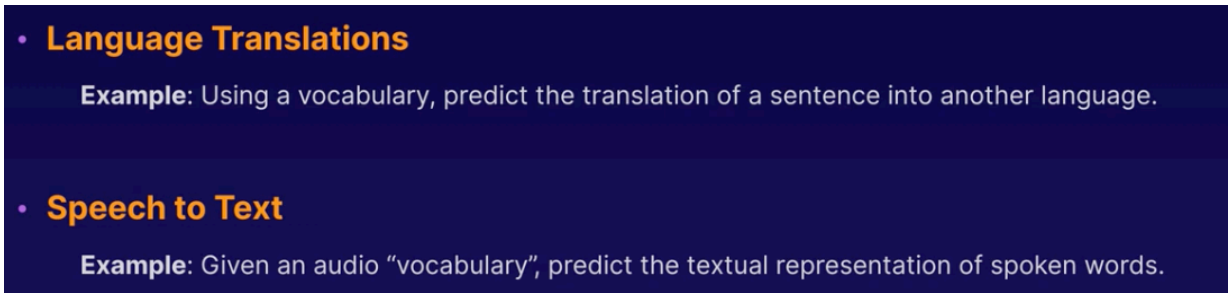


Other SM Also: **seq2seq (Sequence to Sequence)**

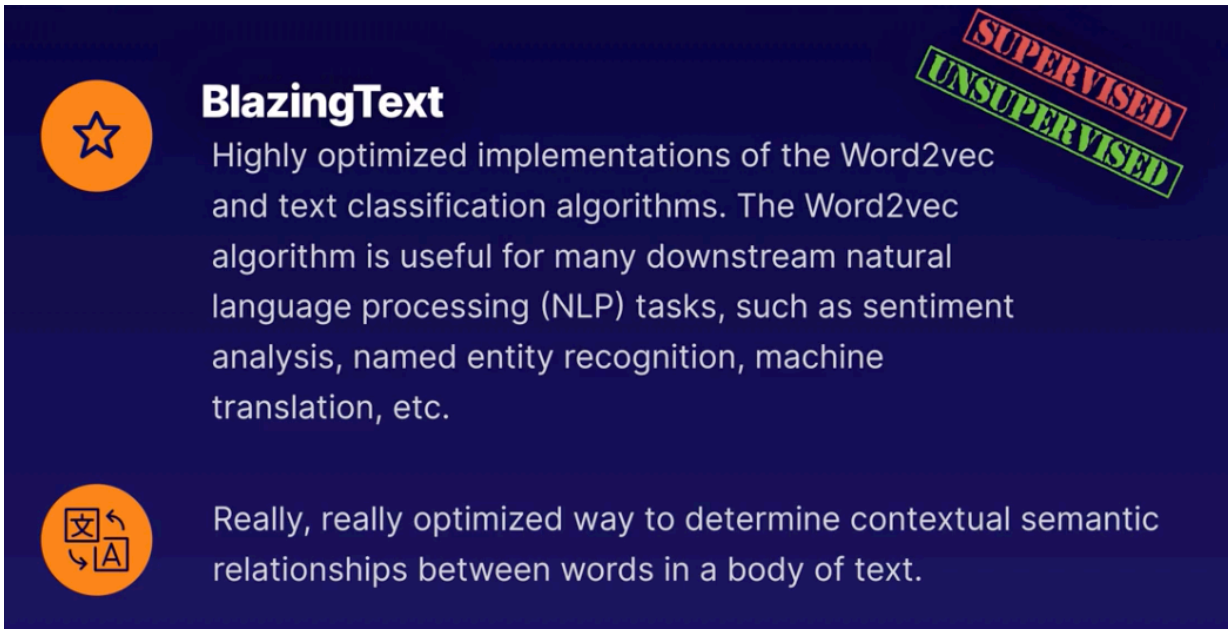




Use cases of seq2seq:

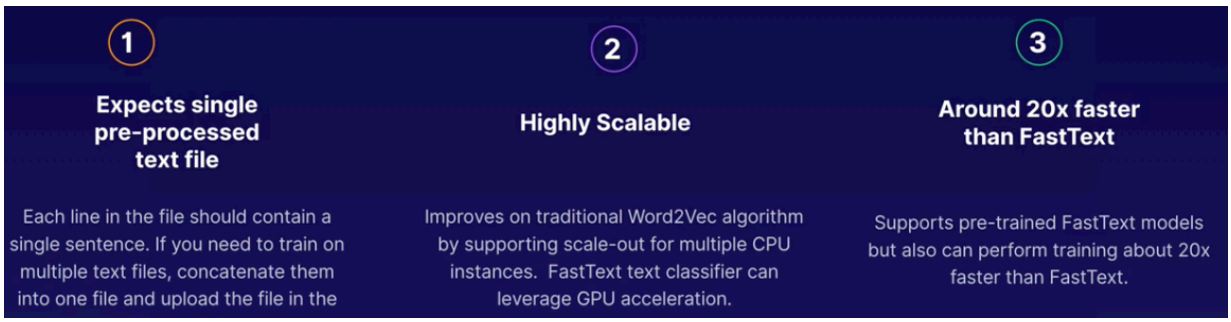


Other Also: BlazingText (based on FastText from FB)



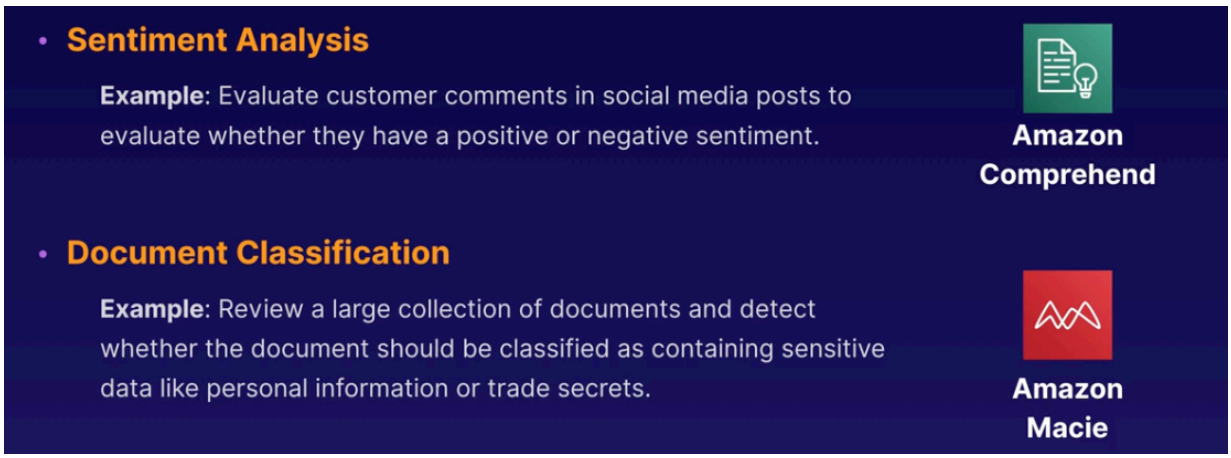
BlazingText Modes:

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

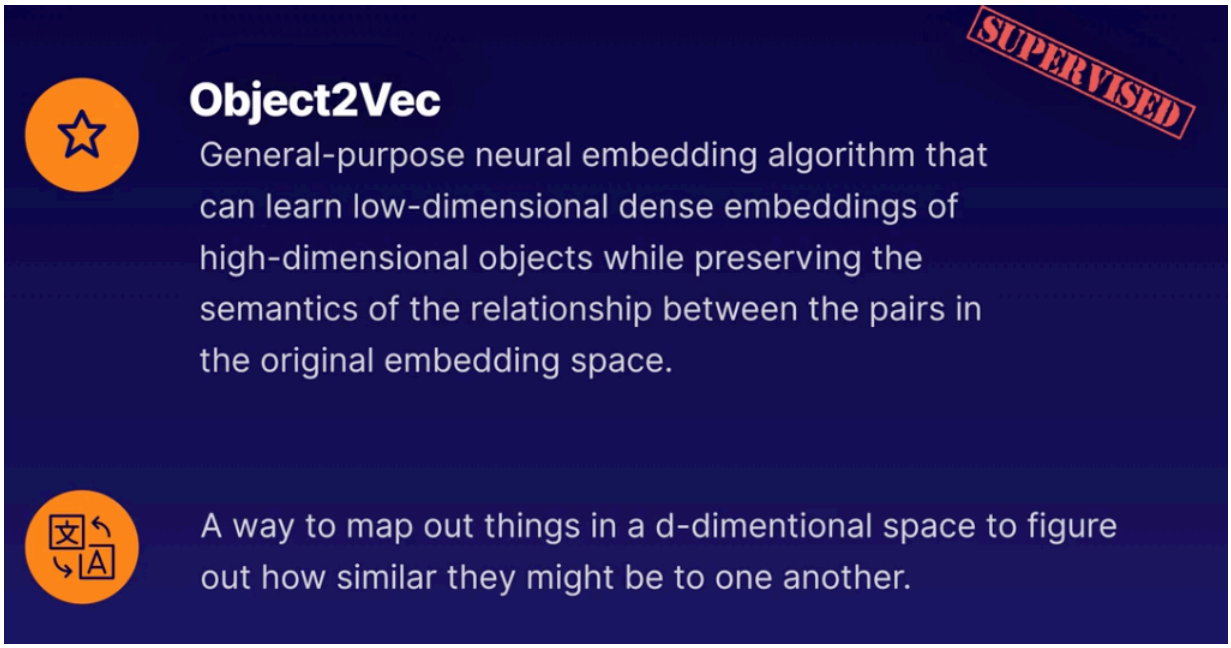


Use Cases for BlazingText:

AWS encapsulates these services into other services:



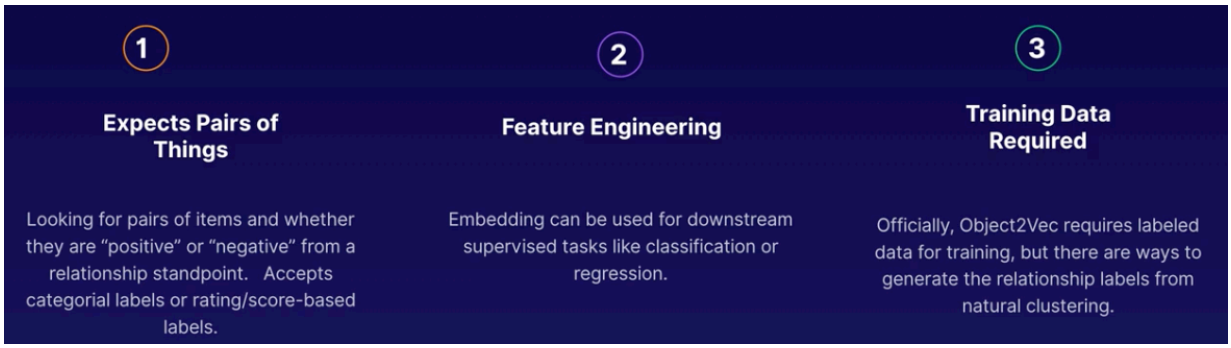
One other also: Object2Vec



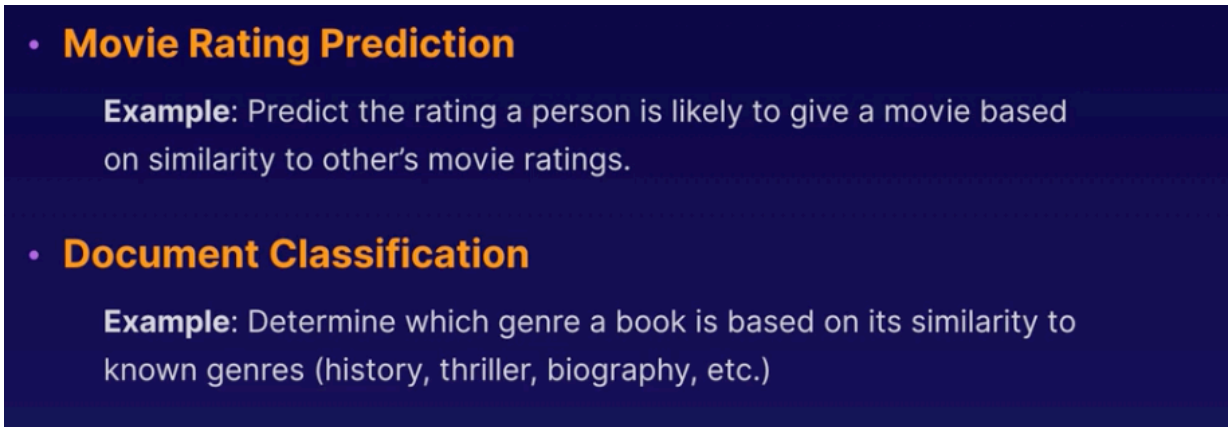
Object2Vec is similar to Word2Vec

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

Object2Vec :



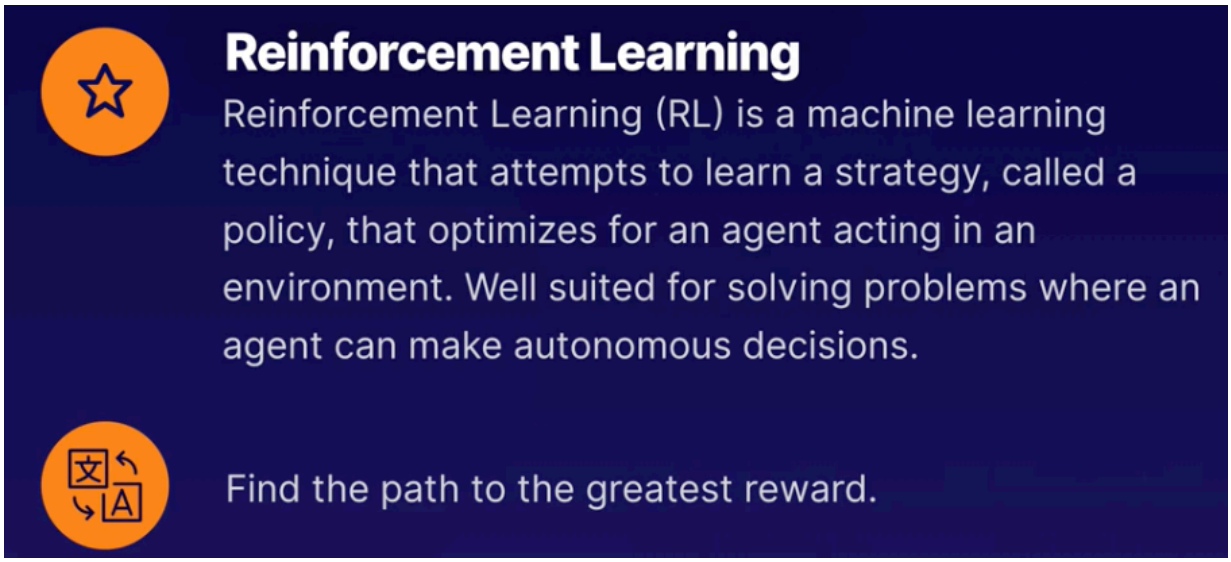
Use Cases:



Reinforcement Learning

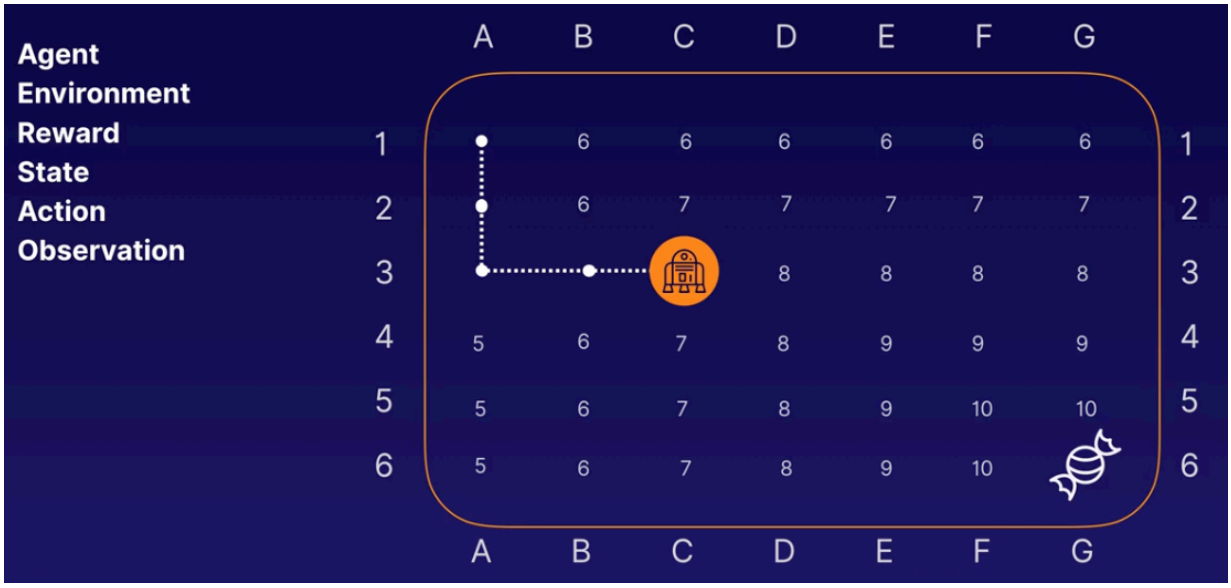
We can use 2 strategies:
- positive reinforcement
- negative reinforcement

Goal is to maximize reward
Reward is based on the behavior of a subject/agent

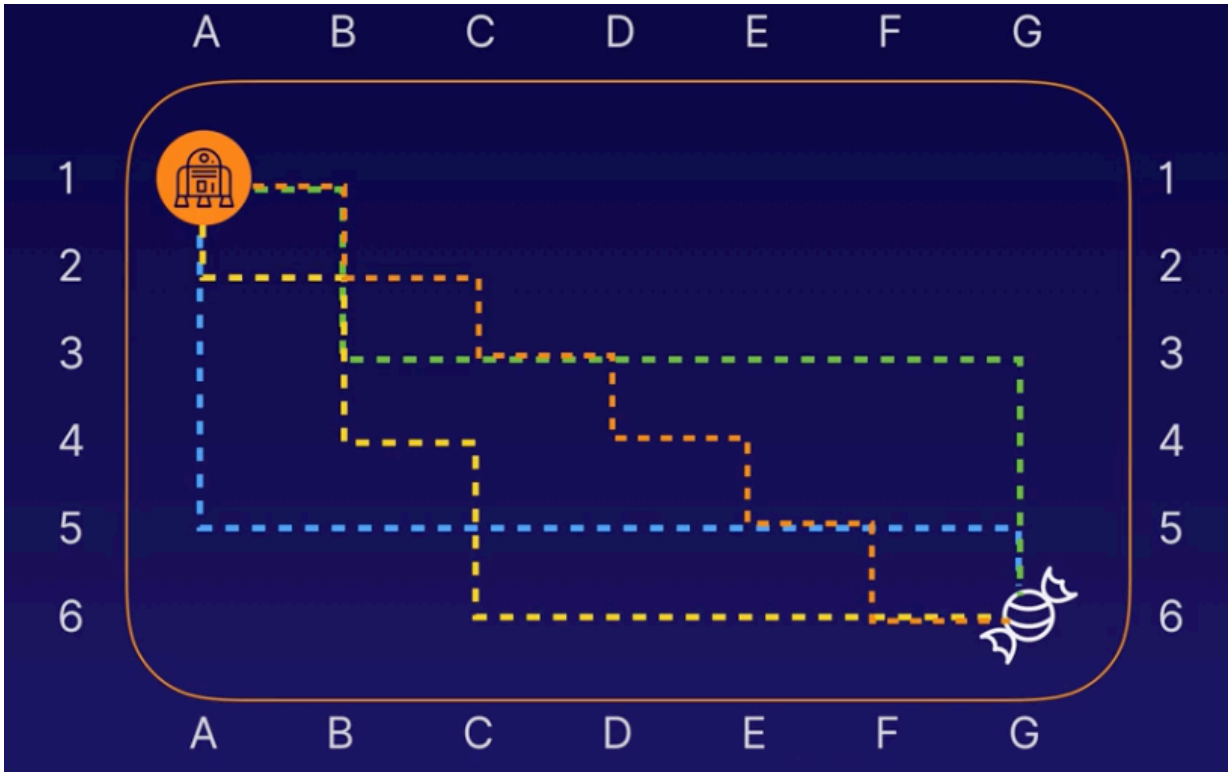


Markov Decision Process: MDP

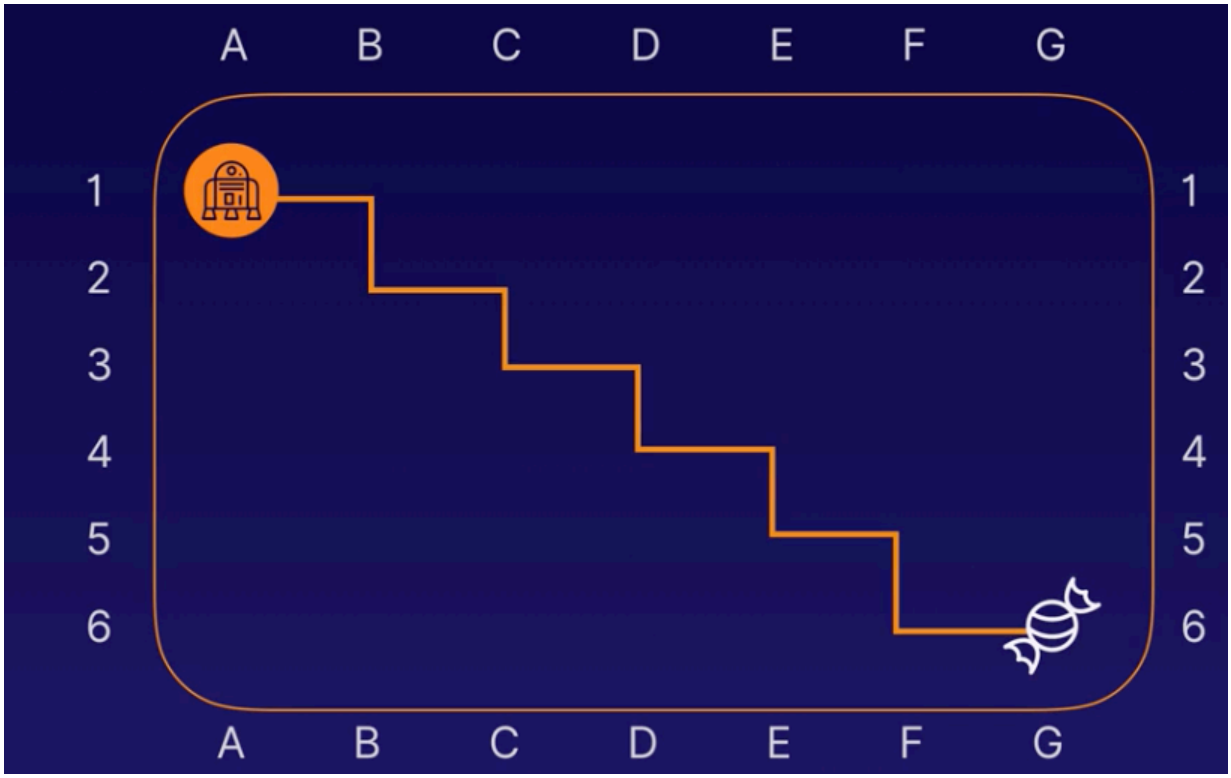
An **Agent** is placed into an **Environment**
Goal is to get to candy (to note a goal is not required in reinforcement learning) => Reward
Current **State** (no need to know how we got there)
Then we have **actions** that an Agent can perform (go up, down, right,...
Agent access **Observation**.



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



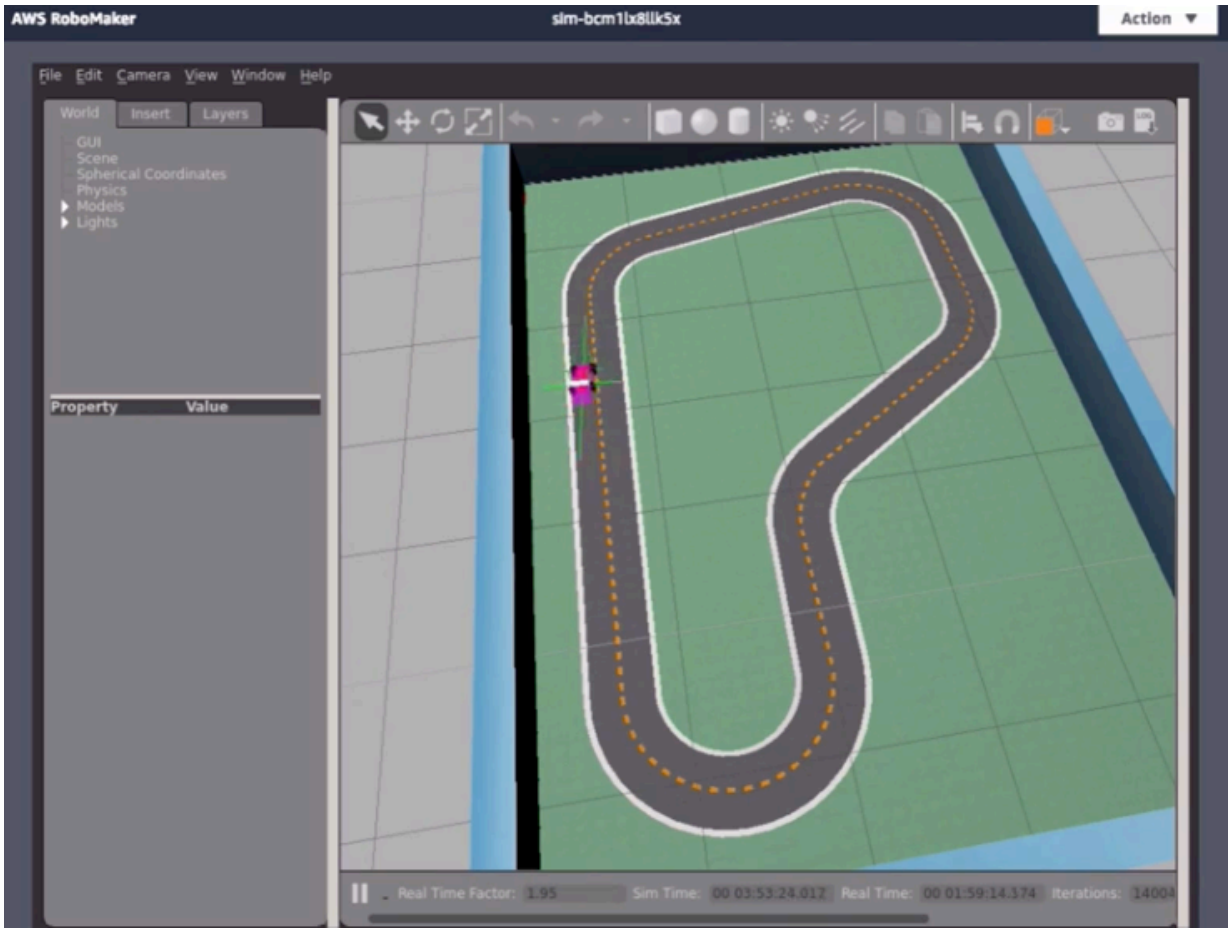
Optimal path;



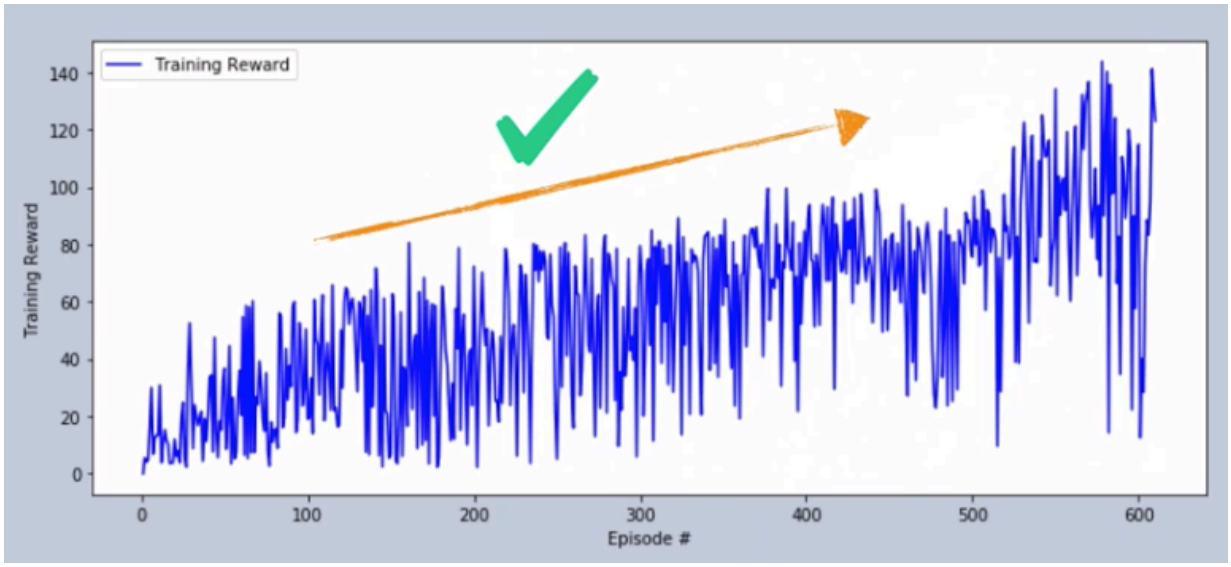
In developing that optimal path, it will learn lessons that will help him make those decision in the future, given a similar objective.
This is called the **Policy** - the decision making part of the reinforcement learning model.

A good example in AWS: Deepracer project

Example of simulation

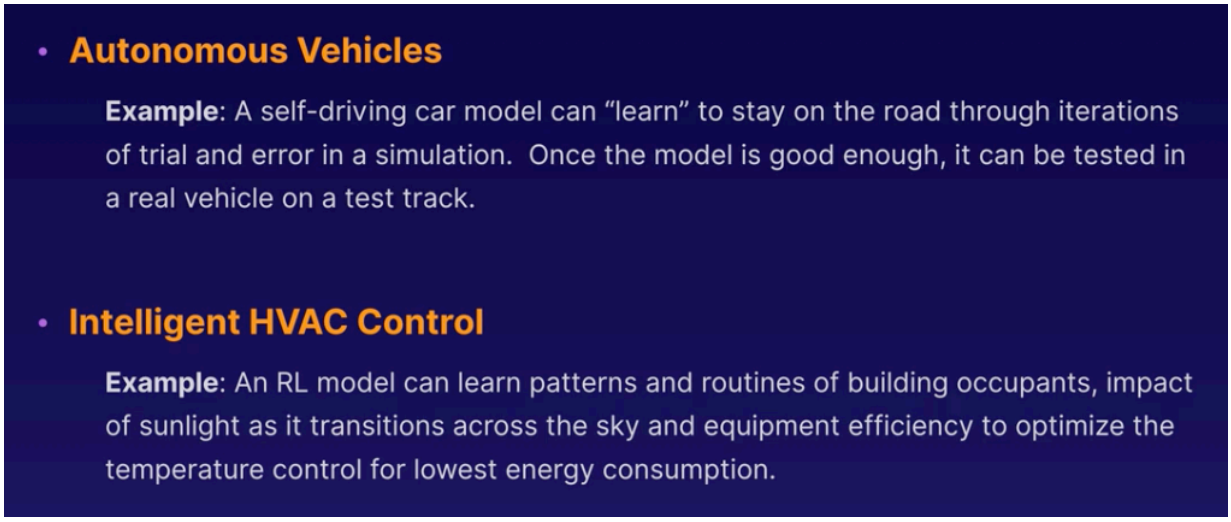


Car learns that if it stays closer to the center of track, it will get a higher reward
Graph of cumulative reward



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

Use cases of Reinforcement learning:



Forecasting

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

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

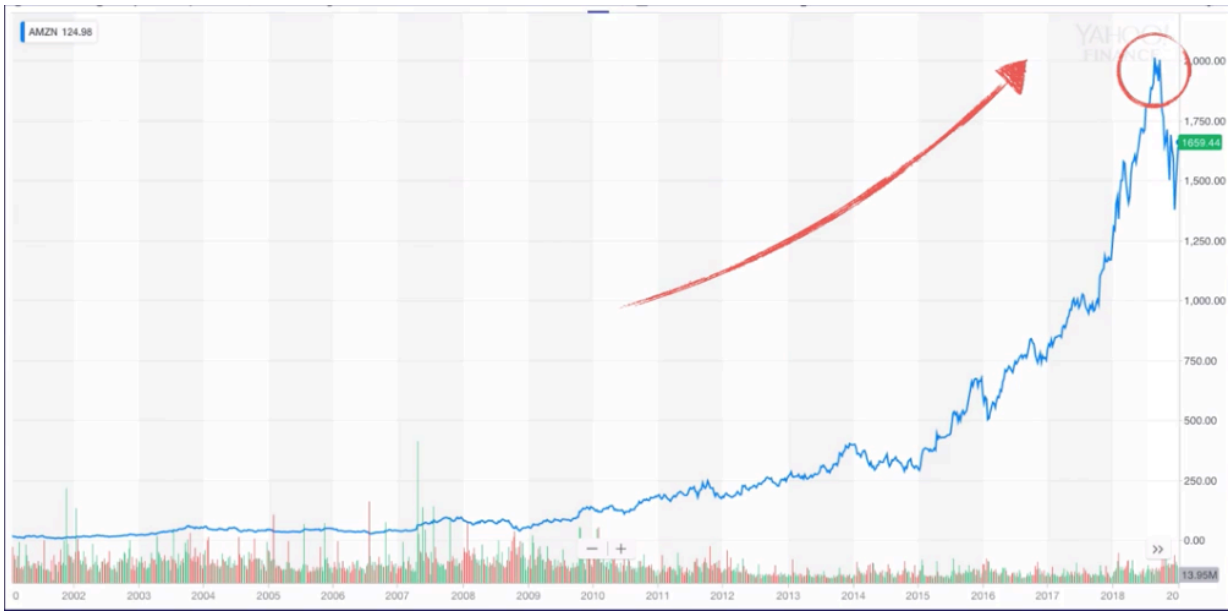
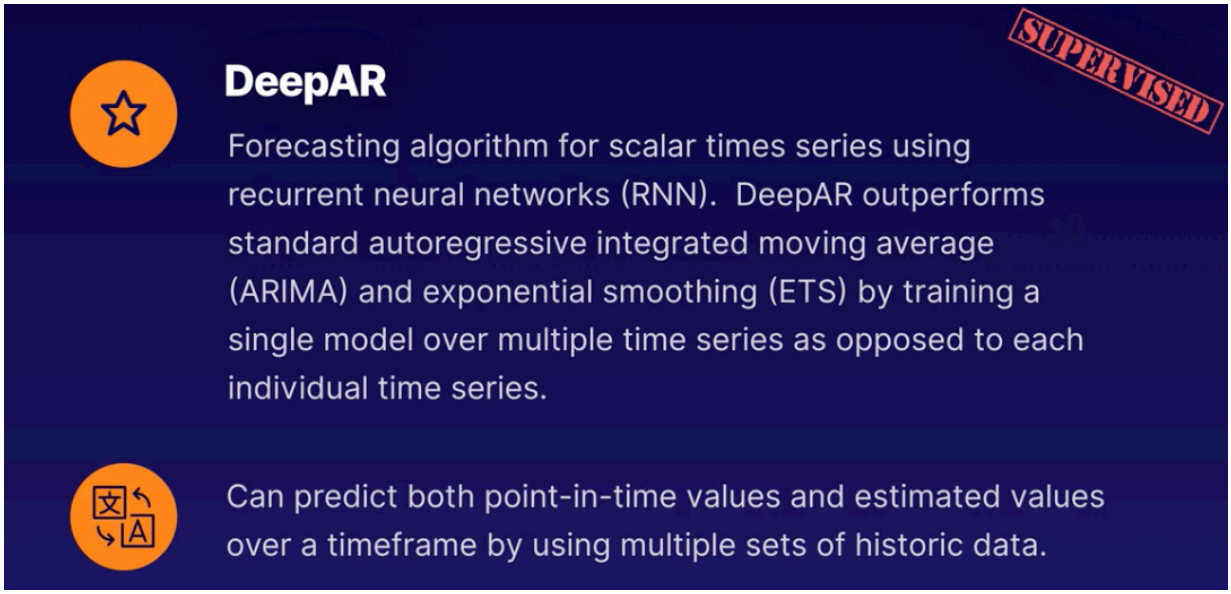


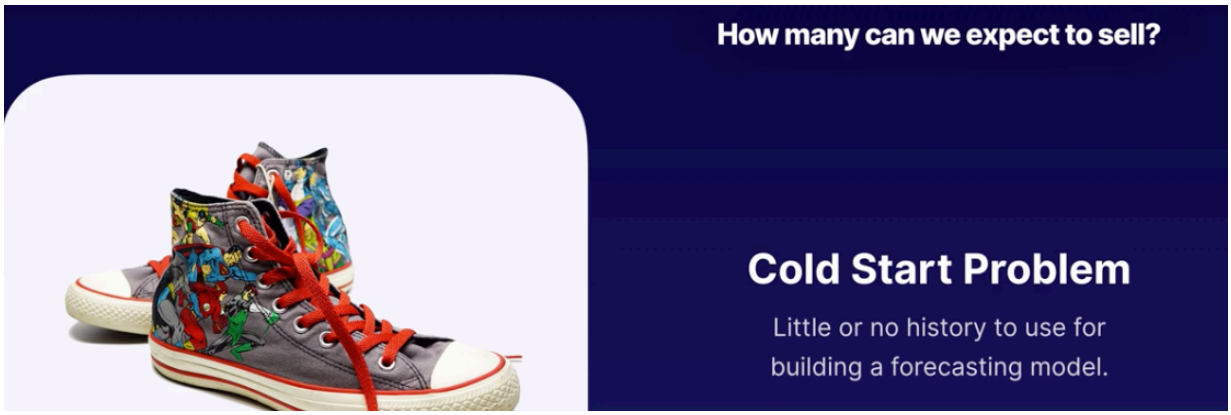
Illustration of: Past performance is not indicative of future results

SM forecasting algo: DeepAR

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



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



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.)
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.
3. **Must supply at least 300 observations**
   DeepAR requires a minimum number of observations across all time series.
4. **You must supply some hyperparameters**
   Context Length, Epochs, Prediction Length and Time Frequency are required hyperparameters.
5. **Automatic Evaluation of the Model**
   Uses a backtest after training to evaluate the accuracy of the model automatically.

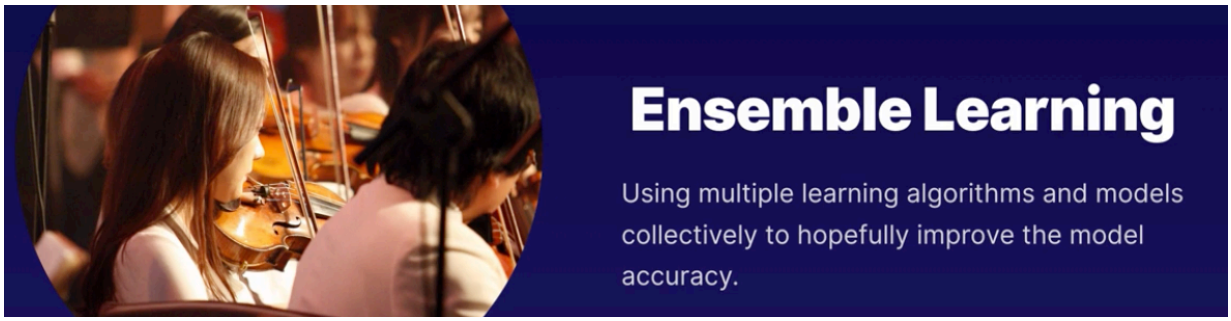
- **Forecasting New Product Performance**

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

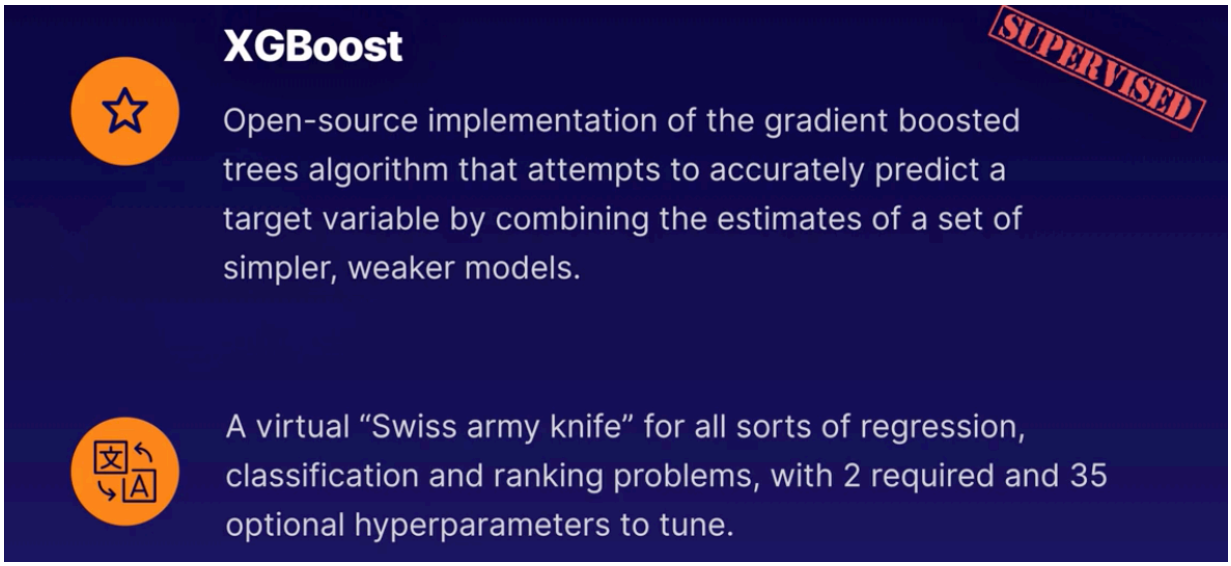
- **Predict Labor Needs for Special Events**

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

Ensemble Learning

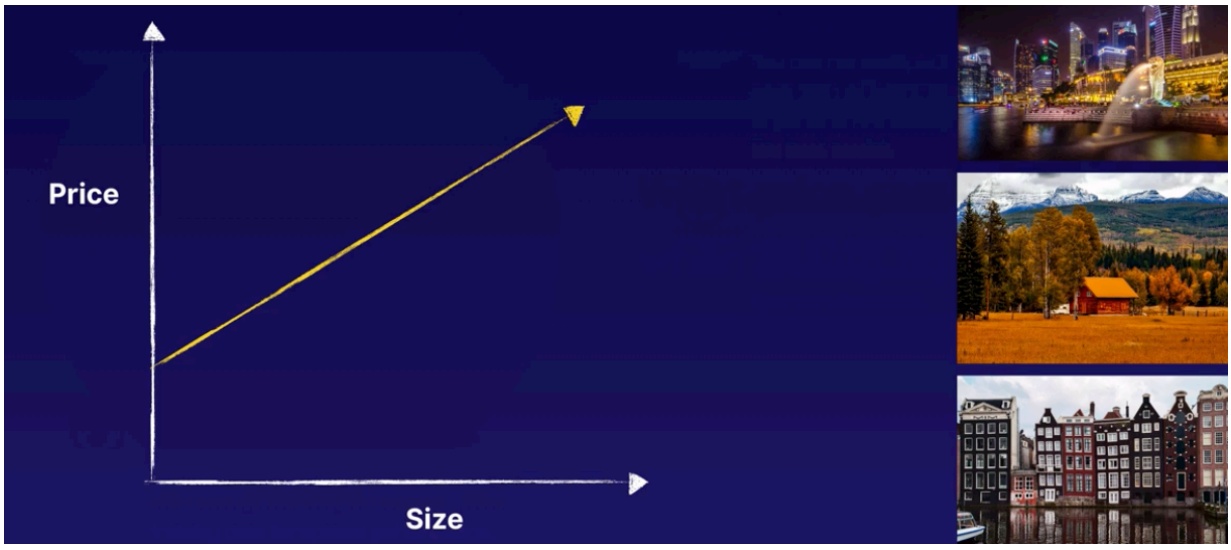


SM Extreme Gradient Boosting - SGBoost
Swiss Army knife



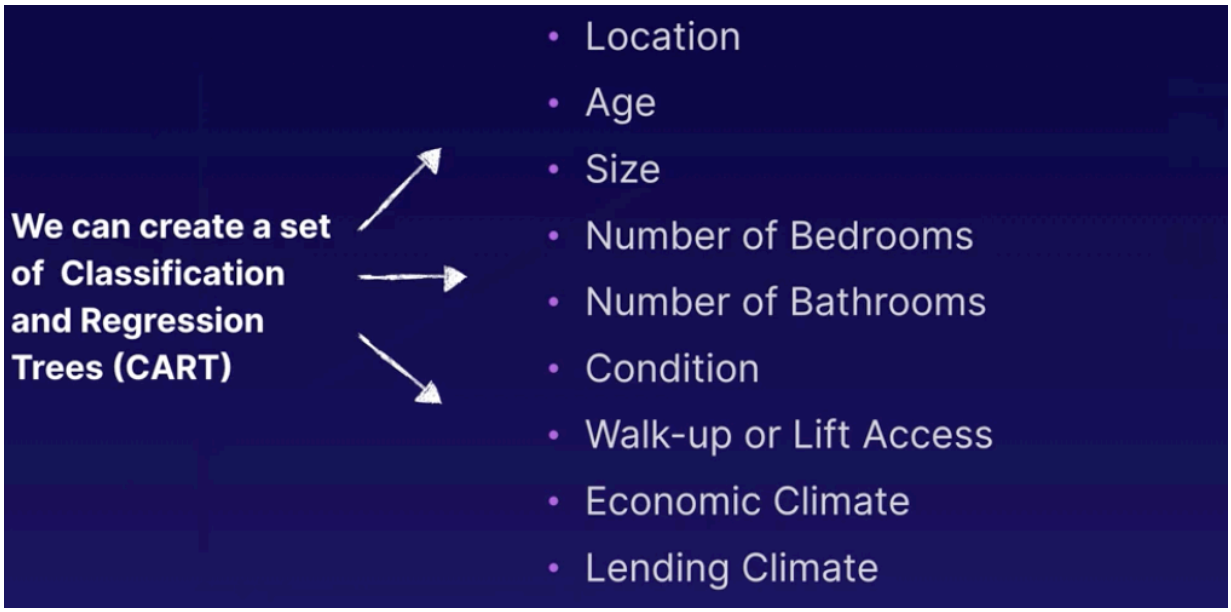
Example - price house when selling it

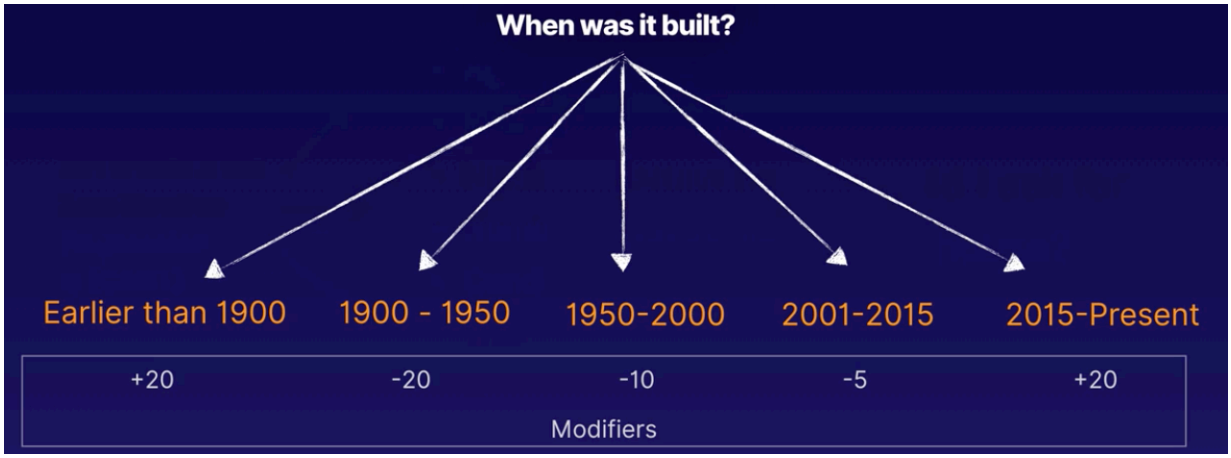
Price is not just related to size



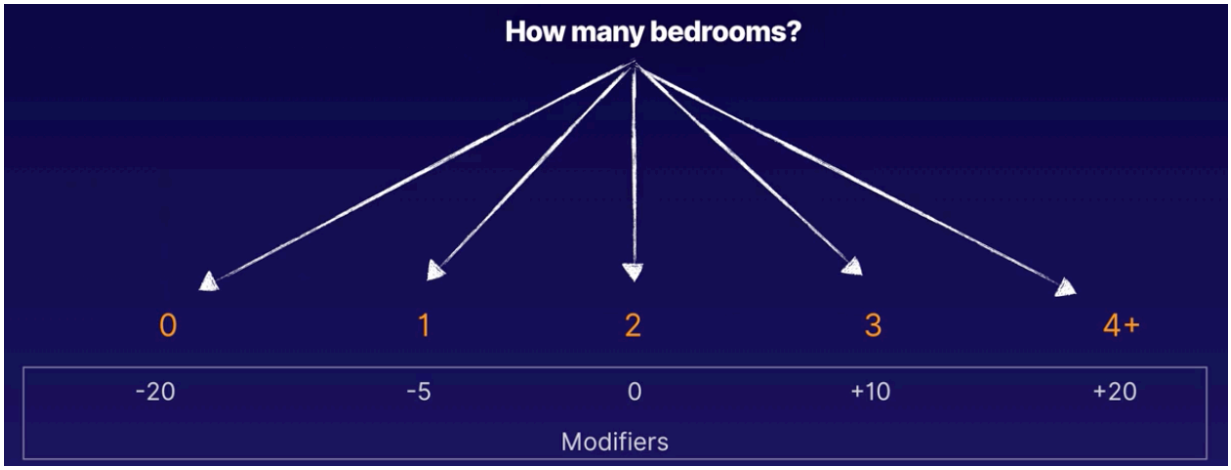
Need to account for many other things to have an accurate predictor.
In a decision Tree Ensemble, we can create a set of Classification and Regression

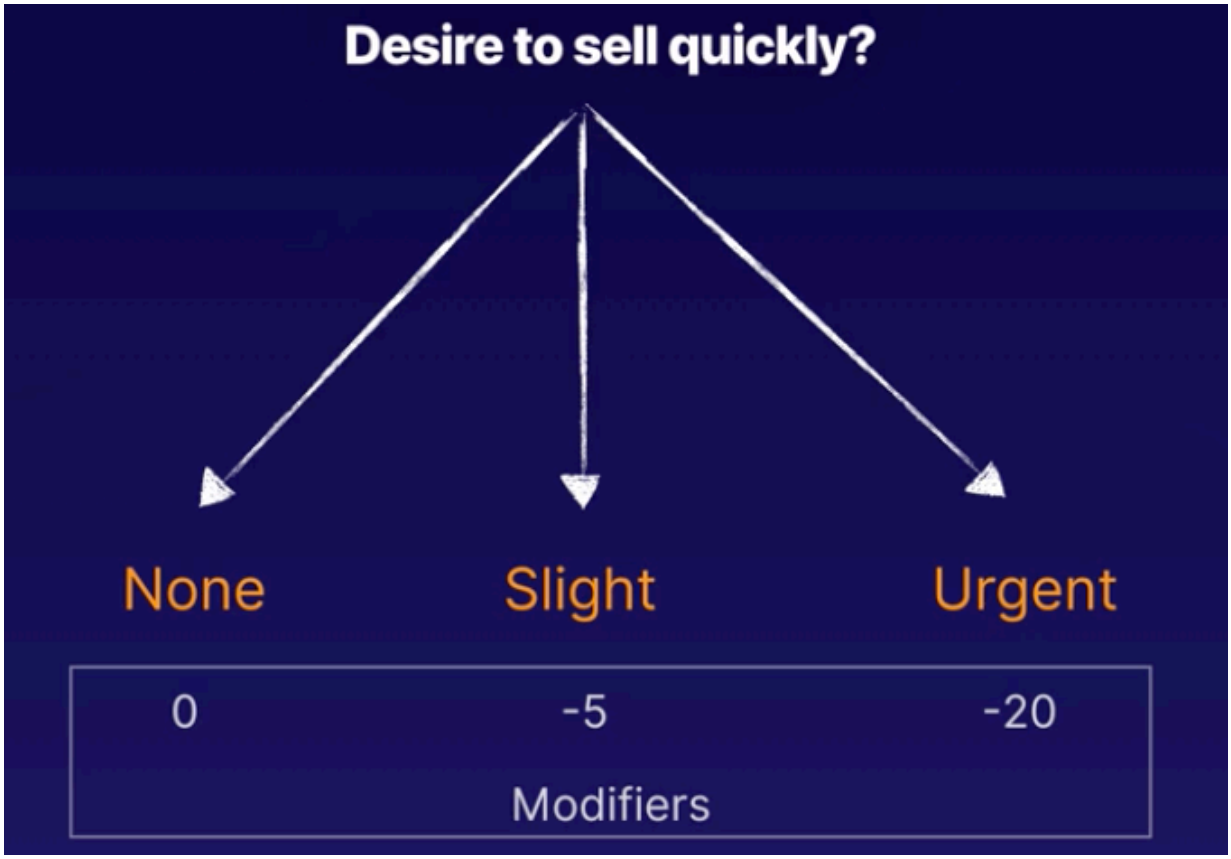
Trees, known as **CART**



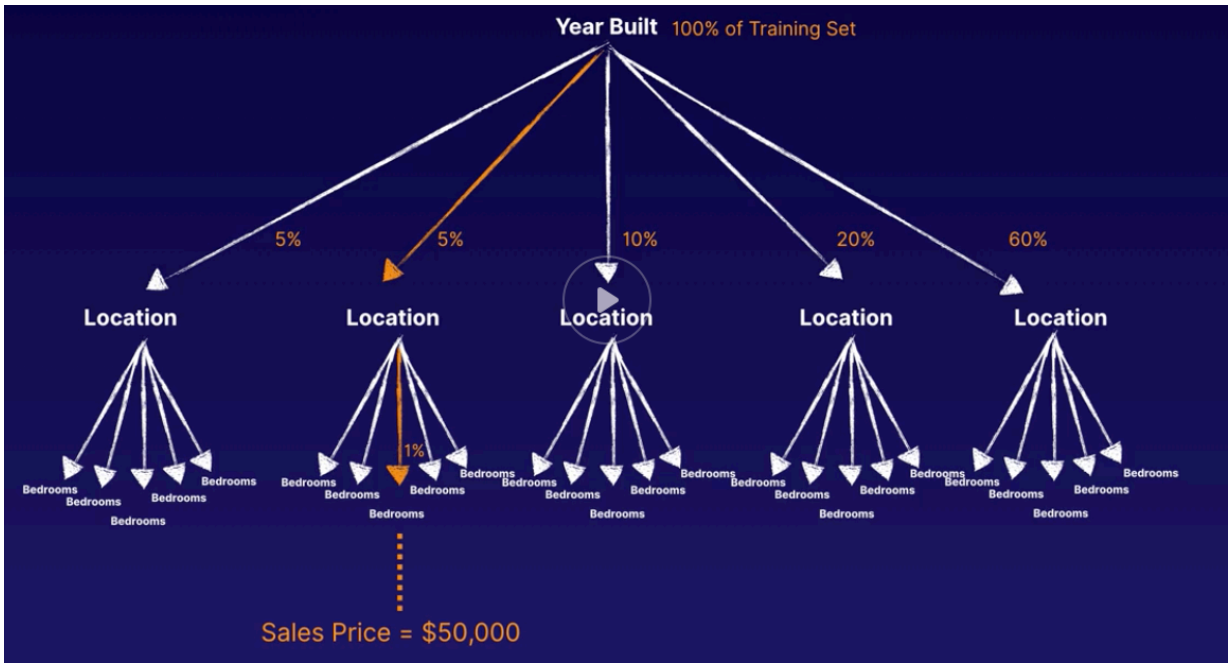


Add modifiers:
If house is very old, it could be desirable => +20





We then have all these trees tacked one on the other



Each tree has a modifier

=> 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.