

## CloudGuru - 3 - Data Preparation

<https://acloud.guru/course/aws-certified-machine-learning-specialty/learn/f399041c-e040-9d9a-d60c-35317e0e9a97/844cc92f-b2b7-dcb4-e7b8-5e38306d3205/watch?backUrl=~2Fcourses>

Data preparation is the process of transforming a dataset using different techniques to prepare it for model training and testing.

Data preparation is usually the most time consuming step of ML



Typical data preparation example:

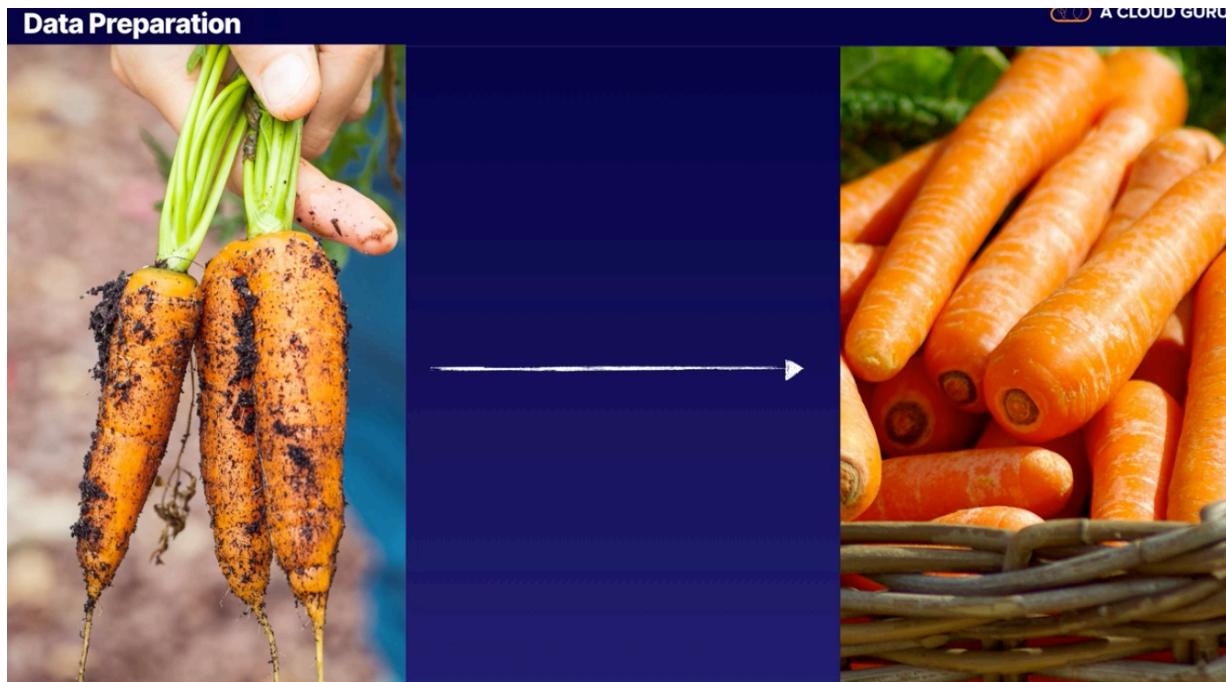
- Formatting: some values may not be standard like "No" instead of a 0/1 integer
- Missing values
- Duplicates - they can skew results (remove them)
- Invalid Data
- Encoding - when an ML algorithm does not accept text based values or categorical values => need to encode into 0 or 1 (or numerical value)

ID	Name	Evil	Affiliation
1	Luke	No	Rebels
2	Leia	NULL	REB
3	Han	0	Rebels
4	Vadar	1	Empire
5	Han	0	Rebels
6	Jabba the Hutt	1	
7	Greedo	0	Bounty Hunter

Legend:

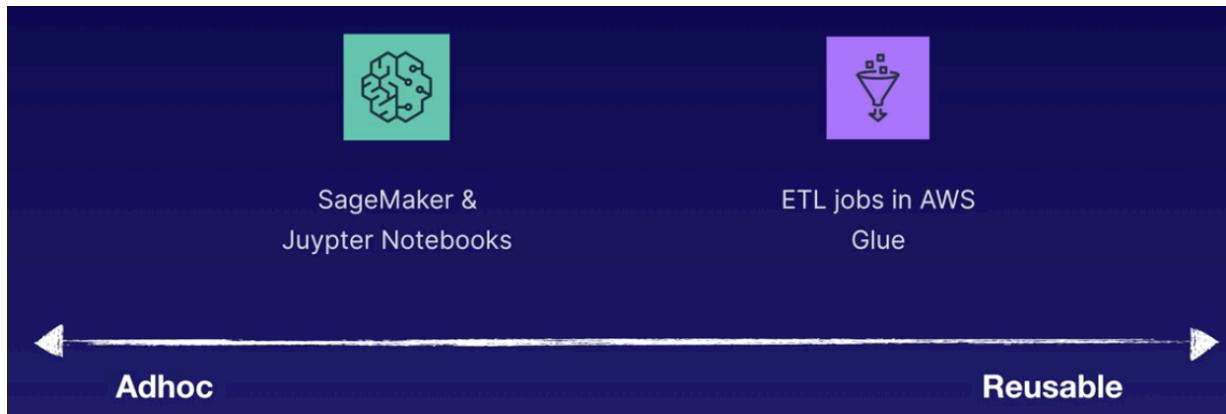
- Formatting: No, NULL
- Duplicates: Han (row 3, 5), Jabba the Hutt (row 6)
- Encoding: 0, 1
- Missing Values: None
- Invalid Data: None

In summary, data preparation is to take unprepared, dirty data and transform it to something useful, that we can consume in our model.



There are a few tools in AWS that can help with data transformation:

- SageMaker (and Jupyter notebooks)
- AWS Glue (that is an ETL tool)



### Categorical Encoding

**Categorical Encoding** is the process of manipulating categorical variables when ML algorithms expect numerical values as inputs  
=> it's about changing category values in our datasets to numbers.

## **categorical variable = categorical feature = discrete feature**

Historical reference: Alan Turing - managed to decipher message on the "enigma" machine from the German

**Alan Turing**



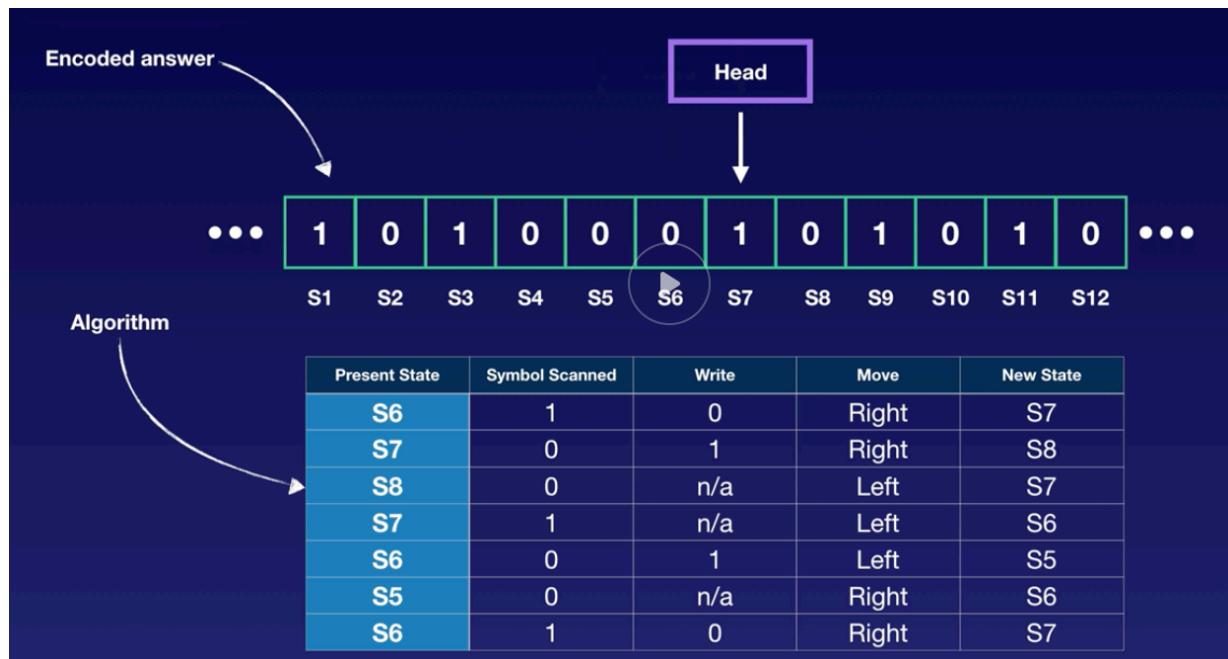
Museo della Scienza e della Tecnologia "Leonardo da Vinci" [CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>)]

# Alan Turing

Educator, Mathematician, PhD (1912–1954)

An English scientist in mathematics and cryptology who helped intercept coded messages and decipher the messages. Founding father of computer science and artificial intelligence.

He proved any algorithm can be defined with a virtual tape encodings in 0 and 1  
He is one of the founding fathers of CS and AI

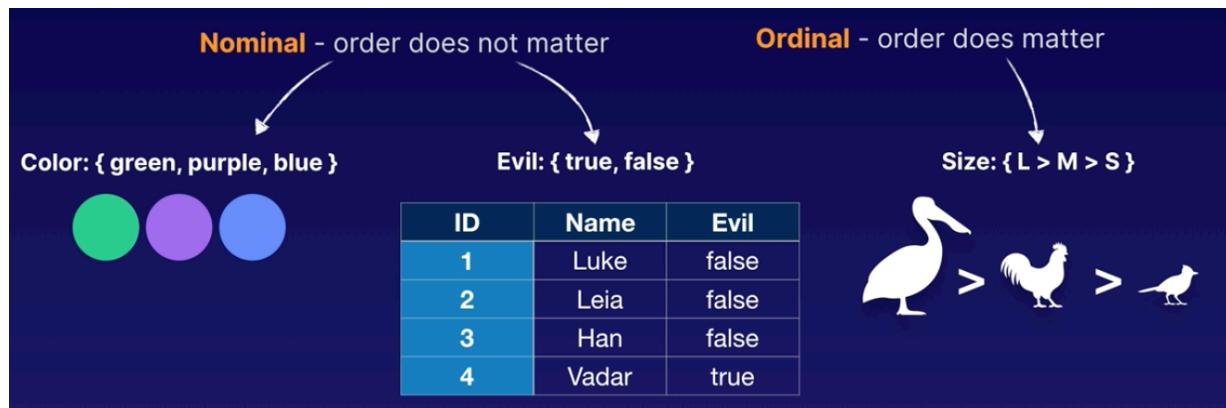


When is it a good time to encode?

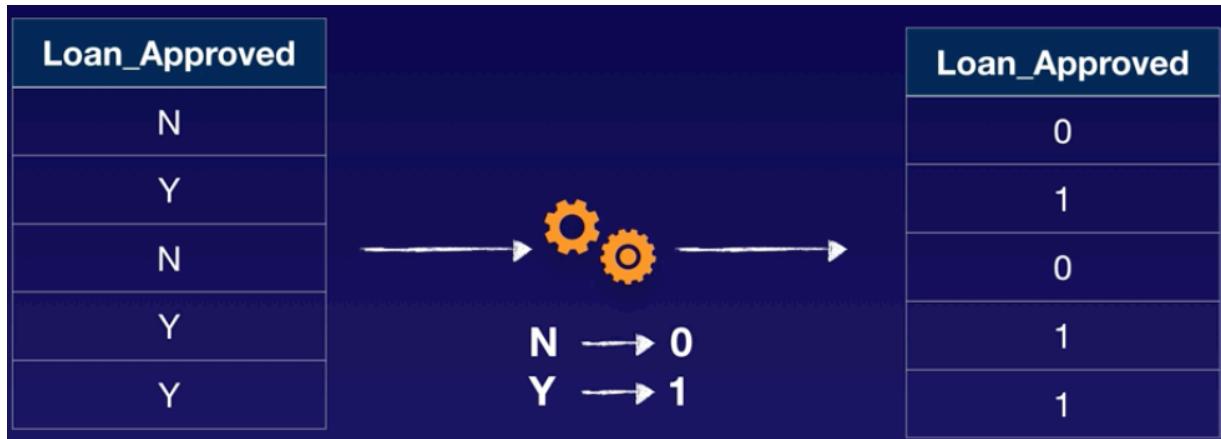
Problem	Algorithm	Encoding
Predicting the price of a home	Linear Regression	Encoding necessary
Determine whether given text is about sports or not	Naive Bayes	Encoding not necessary
Detecting malignancy in radiology images	Convolutional Neural Network	Encoding necessary

### Nominal Vs Ordinal:

- Nominal: order does not matter
- Ordinal: order DOES matter

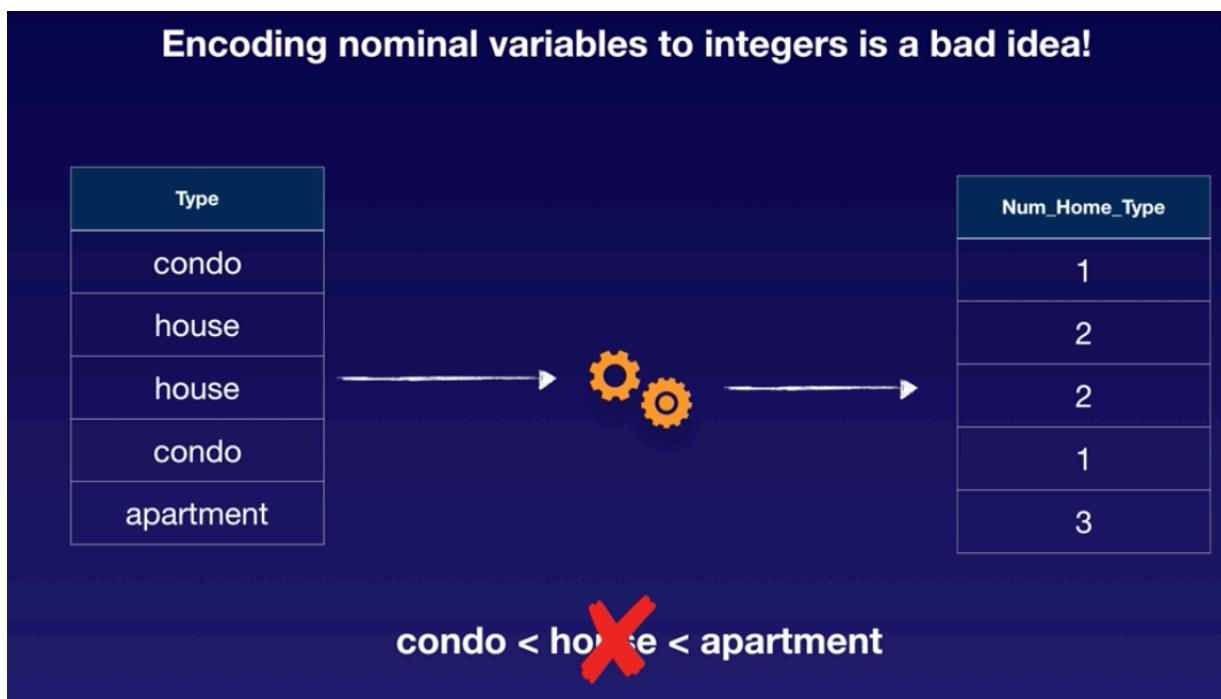


Encoding categorical features



### Encoding nominal values into integers is a bad idea

In the below example, a model could interpret the house incorrectly.



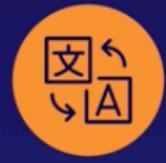
Use **One-hot encoding** instead

=> adds new features for each new category and assign a 0 if the observation does not include that category, or 1 if it does



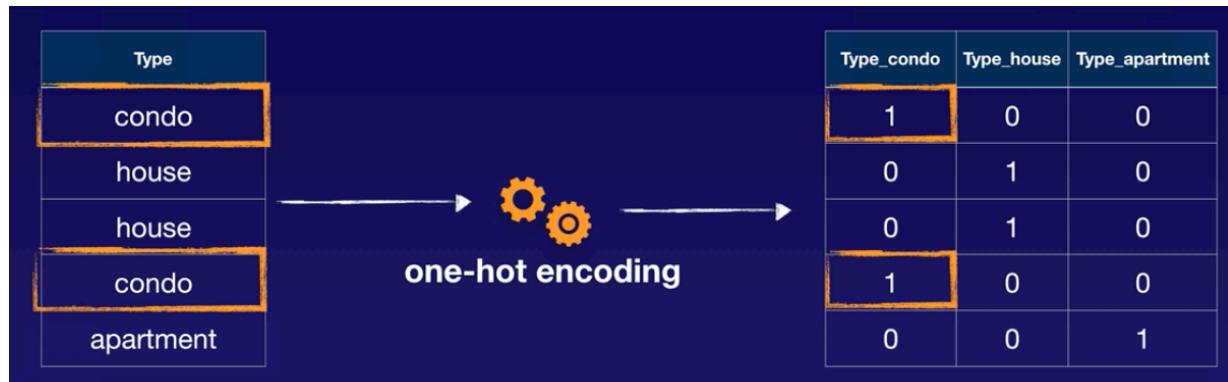
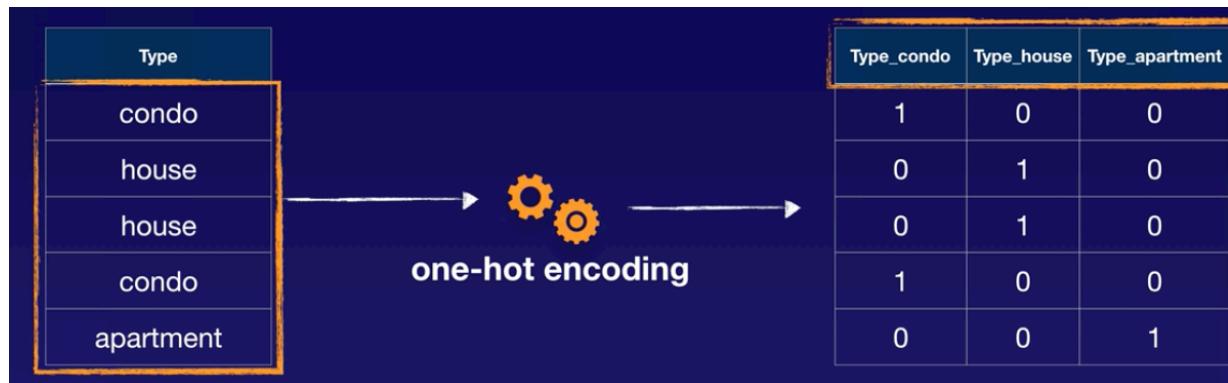
# One-hot Encoding

Transforms nominal categorical features and creates new binary columns for each observation.



Adding columns to your dataset of 1's and 0's.

Here is the house type with One-Hot encoding



**One-Hot encoding is not goof for many features** though - it will grow exponentially

There are other techniques like **Grouping** or **Mapping rare values**

<b>1</b>	<b>One-hot or not?</b> One-hot encoding is not always a good choice when there are many, many categories.	<b>2</b>	<b>Grouping</b> Using techniques like grouping by similarity could create fewer overall categories before encoding.	<b>3</b>	<b>Mapping Rare Values</b> Mapping rare values to "other" can help reduce overall number of new columns created.
----------	--	----------	--	----------	---

## Categorical Encoding Summary

A CLOUD GU...

<b>1</b>	<b>ML Algorithm Specific</b> In general, categorical encoding is used when the ML algorithm can not support categorical data.	<b>3</b>	<b>There is No "Golden Rule"</b> There is no "golden rule" on how to encode your categories (or transform your data in general).
<b>2</b>	<b>Text into Numbers</b> We must find a way to turn text attributes into numeric attributes within our datasets.	<b>4</b>	<b>Many Different Approaches</b> There are many different approaches and each approach can have a different impact on the outcome of your analysis.

## Text Feature Engineering

=> Transforming text within our data so ML Lagos can better analyze it.  
In other words, it's about splitting text into bide size pieces

Examples of text data (or corpus data): white paper, dialog between 2 people, newspaper clipping

AAPS PharmSciTech Vol. 11, No. 1, February 2000 (1–2000)  
DOI: 10.1208/ps011004-00062

## Editorial

### Theme: Pharmaceutical Thermal Processing

Guest Editors: Feng Zhang and Michael A. Repta\*

## Pharmaceutical Thermal Processing

Feng Zhang,<sup>1,2</sup> and Michael A. Repta<sup>3</sup>

published online 9 February 2000

Thermal processing has flourished as a novel technology to prepare a wide range of dosage forms systems over the past decade. During this time, significant progress has been made in the understanding of the mechanical energy into the thermal energy of the system and its effect on pharmaceutical products. This special issue of *AAPS PharmSciTech* will employ thermal processing experience. Kishimoto<sup>1</sup> describes the use of the hot melt extrusion technique to prepare solid oral dosage forms. This theme issue of *AAPS PharmSciTech* provides a broad overview of the state-of-the-art in pharmaceutical thermal processing. The scientific approaches to develop drug delivery systems using thermal processing are presented in this special issue.

Marin provided a historical and technical review of two basic concepts as a continue for thermal processing in the pharmaceutical industry. The first concept is the history of thermal processing.<sup>2</sup> From his personal opinions, Marin believes that the pharmaceutical industry needs what occurred for the pharmaceutical industry to move forward. The second concept is the history of the development of pharmaceutical thermal processing in the production of pharmaceutical dosage forms. Marin also described the development of pharmaceutical thermal processing in quality-by-design approach and process analytical technology in pharmaceutical industry.<sup>3</sup>

Huang et al. discussed a number of formulation and process variables that influence the physical properties of flexible delivery multilayer pills with shell determined by the temperature of the outermost shell.<sup>4</sup> The authors characterized the phase behavior and morphology of shell-controlled capsules with different compositions of the outermost shell. The results showed that the outermost shell temperature and state of gel or amorphous solid dispersion design in the clinical application of the pharmaceutical product were important. Zhou et al. studied the effect of the temperature of the outermost shell on the crystallization of the innermost shell.<sup>5</sup> The results showed that the crystallization of the innermost shell was influenced by the temperature of the outermost shell.

The article by Chen et al. contains 15 original research articles. Huang et al. propose use of the each extraction procedure for the preparation of a polyfyllanthrin gel base cream.<sup>6</sup>

The article by Li et al. contains 15 original research articles. Bhuagale et al. propose use of each extraction procedure for the preparation of a polyfyllanthrin gel base cream.<sup>7</sup>

<sup>1</sup>College of Pharmacy, University of Texas at Austin, 2000 University Station, Stop C5600, Austin, TX 78712-0560; <sup>2</sup>Department of Pharmaceutical Sciences, Donghua University, Shanghai 201600, China; <sup>3</sup>College of Pharmacy, DePaul University, Chicago, IL 60614-2972, USA

\* To whom correspondence should be addressed. (e-mail: longfeng.zhang@utexas.edu)

modified screw design was used and parameter such as feeding rate, barrel temperature, and screw speed were used to obtain the desired product. Zhou et al. proposed a new extraction procedure to prepare Eudragit FL-based gels for colon targeting delivery system.<sup>8</sup> The results showed that the extraction procedure and lipid in enhancing the drug release was reported. Yu et al. proposed a new extraction procedure to prepare Eudragit L-based capsules by conjugating high-pressure homogenization and melt extrusion.<sup>9</sup> The results showed that the high-pressure homogenization and melt extrusion can produce more flexible delivery multilayer pills with shell determined by the temperature of the outermost shell.

Wang et al. proposed a new extraction procedure to prepare capsules characterized the phase behavior and morphology of shell-controlled capsules with solid dispersions composed of the outermost shell.<sup>10</sup> The results showed that the outermost shell temperature and state of gel or amorphous solid dispersion design in the clinical application of the pharmaceutical product were important. Zhou et al. studied the effect of the temperature of the outermost shell on the crystallization of the innermost shell.<sup>11</sup> The results showed that the crystallization of the innermost shell was influenced by the temperature of the outermost shell.

The article by Chen et al. contains 15 original research articles. Bhuagale et al. propose use of the each extraction procedure for the preparation of a polyfyllanthrin gel base cream.<sup>12</sup> The results showed that the crystallization of the innermost shell was influenced by the temperature of the outermost shell. Zhou et al. proposed a new extraction procedure to prepare capsules with shell determined by the temperature of the outermost shell.<sup>13</sup> The results showed that the crystallization of the innermost shell was influenced by the temperature of the outermost shell.

The article by Li et al. contains 15 original research articles. Bhuagale et al. propose use of each extraction procedure for the preparation of a polyfyllanthrin gel base cream.<sup>14</sup> The results showed that the crystallization of the innermost shell was influenced by the temperature of the outermost shell.

The article by Chen et al. contains 15 original research articles. Bhuagale et al. propose use of each extraction procedure for the preparation of a polyfyllanthrin gel base cream.<sup>15</sup> The results showed that the crystallization of the innermost shell was influenced by the temperature of the outermost shell.

The article by Li et al. contains 15 original research articles. Bhuagale et al. propose use of each extraction procedure for the preparation of a polyfyllanthrin gel base cream.<sup>16</sup> The results showed that the crystallization of the innermost shell was influenced by the temperature of the outermost shell.

APS

1 © 2000 American Association of Pharmaceutical Scientists

Other example: street address extraction from a long set string

## Example:

**Raw Text:** { "123 Main Street, Seattle, WA 98101" }



Address	City	State	Zip
123 Main Street	Seattle	WA	98101

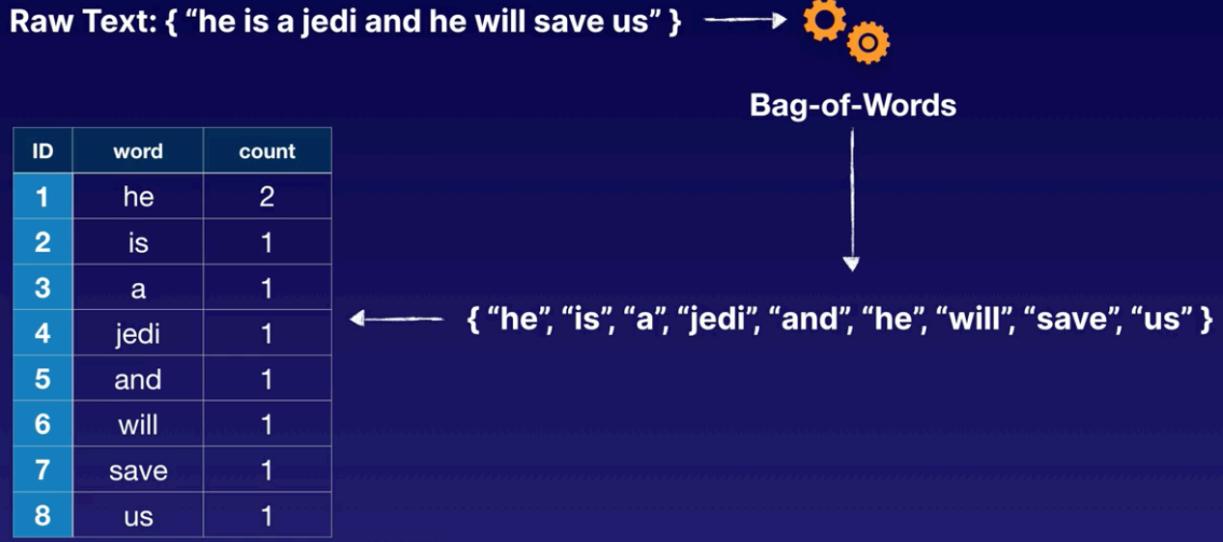
**Bag-of-Words:** Tokenizes raw text and creates statistical representation of the text.

In other words it breaks up text by white space into single words.  
=> It's the simple method to break down text into single tokens.

From this we can create word counts and see the different distribution of those words are produced/used

Example:

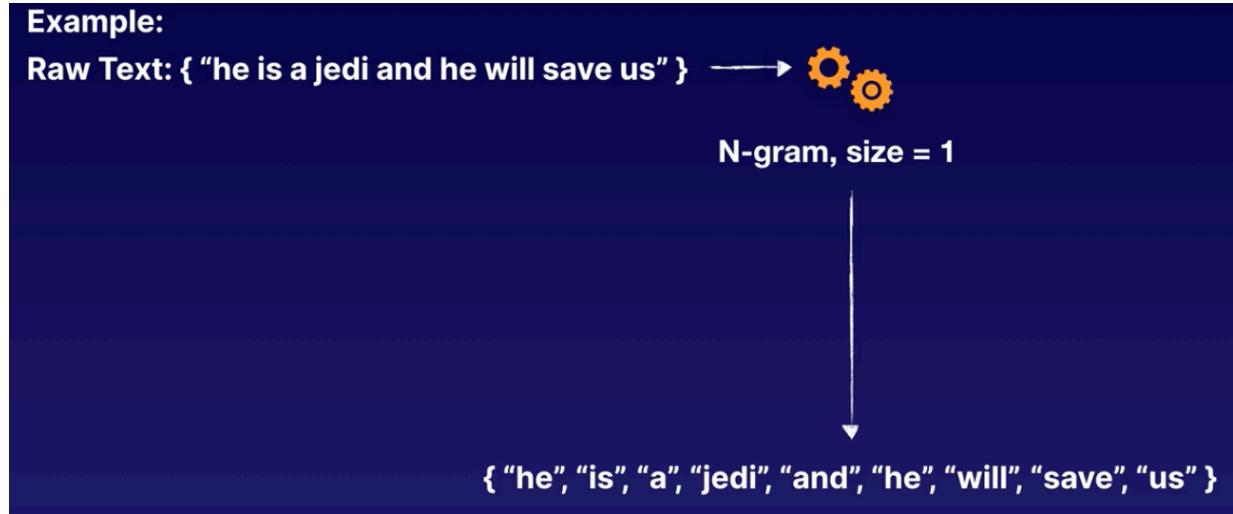
**Example:**



**N-Gram:** builds of Bag-of-Words and produces groups of words of n size  
In other words it breaks up text by white space into groups of words.

Example

N-gram, size = 1

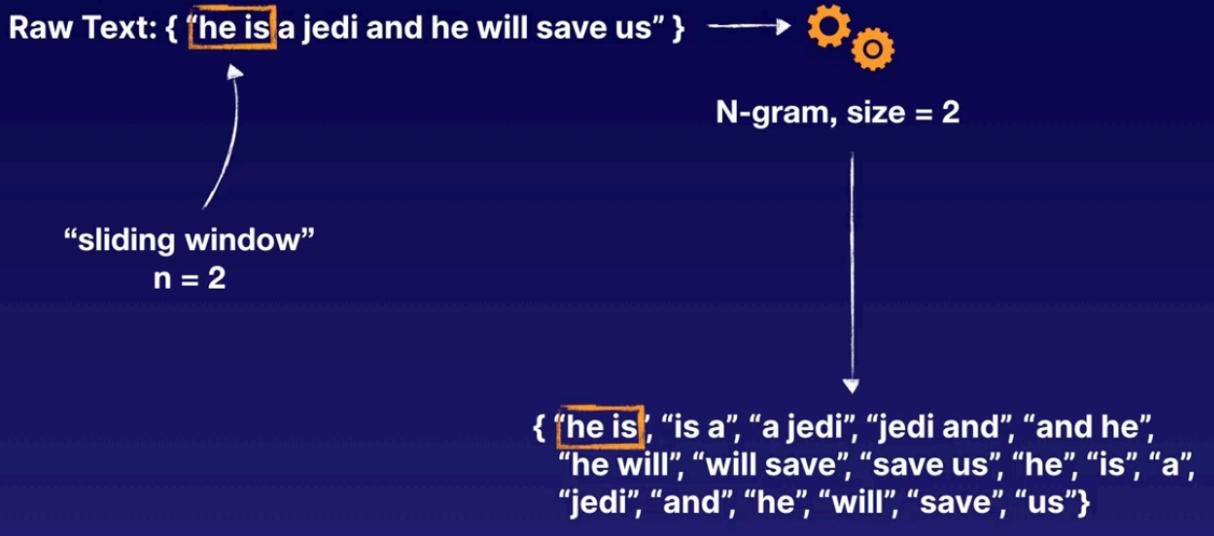


N-gram, size = 2

We use a window of 2 words and slides the window to tokenize all the pairs of 2 words.

To note it can also

### Example:



To note it can also create n-grams of lower size (size = 1)

{ “he is”, “is a”, “a jedi”, “jedi and”, “and he”,  
“he will”, “will save”, “save us”, “he”, “is”, “a”,  
“jedi”, “and”, “he”, “will”, “save”, “us” }

When is that useful?

Spam filter to find pairs filter like “you win”, “click here”

Unigram = 1 word token

Bigram = 2 word tokens

Trigram = 3 word tokens

**Orthogonal Sparse Bigram (OSB)** - similar to n-gram

Creates groups of words of size n and outputs every pair of words that includes the first word.

In other words, it creates groups of 2 words that always include the first word.

## Orthogonal Sparse Bigram

When two things are independent of each other

scattered or thinly distributed

2-gram or two words

**OSB is not “better” or “worse” than N-Gram,  
it’s just another technique to use...**

Example:

OSB with size = 4

Uses “\_” where there is a word or white space

**Example:**

Raw Text: { “he is a jedi and he will save us” } → 

OSB, size = 4

{ “he\_is”, “he\_a”, “he\_jedi” }

1 underscore

2 underscores

3 underscores

**Example:**

Raw Text: { “he is a jedi and he will save us” } → 

OSB, size = 4

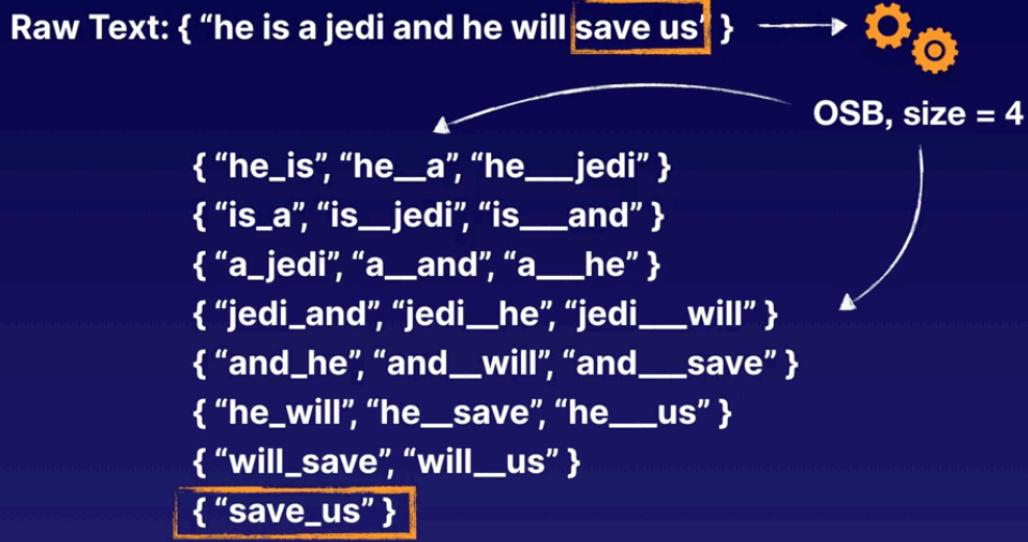
{ “he\_is”, “he\_a”, “he\_jedi” }

{ “is\_a”, “is\_jedi”, “is\_and” }

{ “a\_jedi”, “a\_and”, “a\_he” }

{ “jedi\_and”, “jedi\_he”, “jedi\_will” }

### Example:

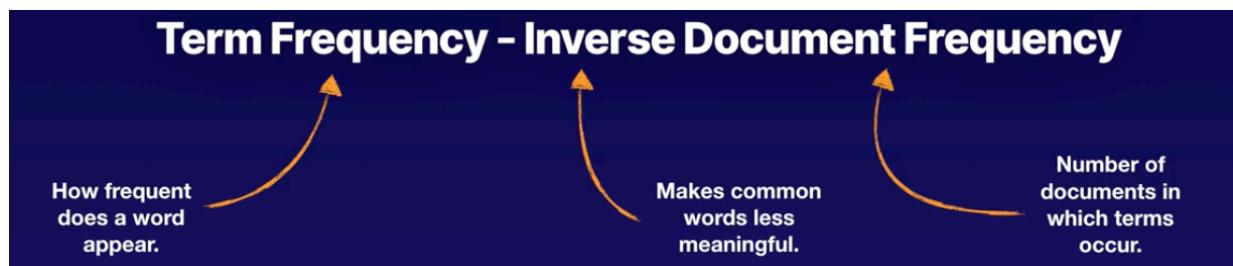


N-grams and OSB are good way to tokenize a text.

But sometimes we need to find importance of words and maybe reduce importance of common words too like next technique.

### TF-IDF: Term Frequency - Inverse Document Frequency

Represents how important a word or words are to a given set of text by providing appropriate weights to terms that are common and less common in the text. In other words, it shows us the popularity or a word or words in text data by making common words like "the" or "and" less important.



Example:

Example:



tf-idf

Document 1

word	count
the	3
force	1
Luke	1
Skywalker	1
a	2

Document 2

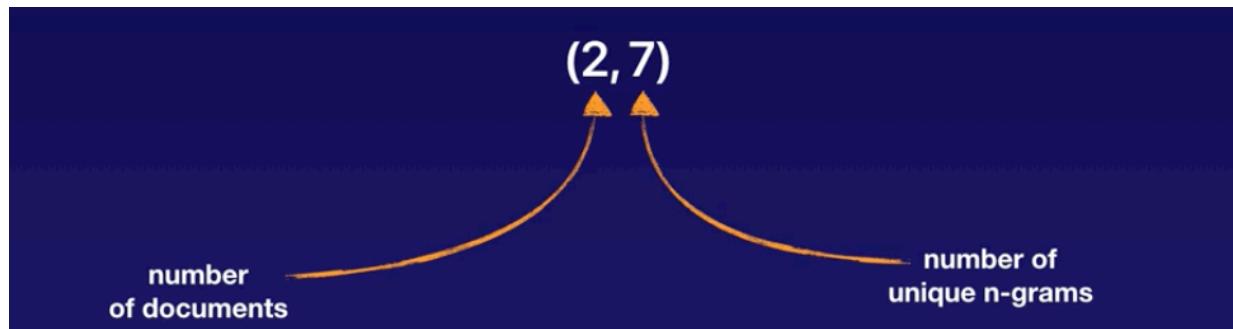
word	count
the	2
jedi	1
a	1
empire	1

Since the tokens “the” and “a” showed up in **BOTH** documents many times, these are deemed as **less important**.

In exam, we can ask to vectorize a document with TF-IDF

It is show as (number of documents, number of unique n-grams)

Example:



**QUESTION 8**

A term frequency-inverse document frequency (tf-idf) matrix using both unigrams and bigrams is built from a text corpus consisting of the following two sentences: { Hello world } and { Hello how are you }. What are the dimensions of the tf-idf vector/matrix?

(2, 9)

(2, 5)

(5, 9)

(2, 6)

(2, 10)

Sorry!

**Correct Answer**

There are 2 sentences (or corpus data we are vectorizing) with 5 unique unigrams ('are', 'hello', 'how', 'world', 'you') and there are 4 unique bigrams ('are you', 'hello how', 'hello world', 'how are'). So the vectorized matrix would be (2, 9).

**What to use when?**

Problem	Transformation	Why
Matching phrases in spam emails	N-Gram	Easier to compare whole phrases like "click here now" or "you're a winner".
Determining the subject matter of multiple PDFs	Tf-idf Orthogonal Sparse Bigram	Filter less important words in PDFs. Find common word combinations repeated throughout PDFs.

Remember that:

- N-gram can be used to math whole words or phrases
- TF-IDF helps us determine which words are more important in document than others, and help us filter out common words like "and" or "the"
- OSB - find common words combination or bi-grams

Before using these Lagos, it is probably a good thing to remove punctuation and use lower casing

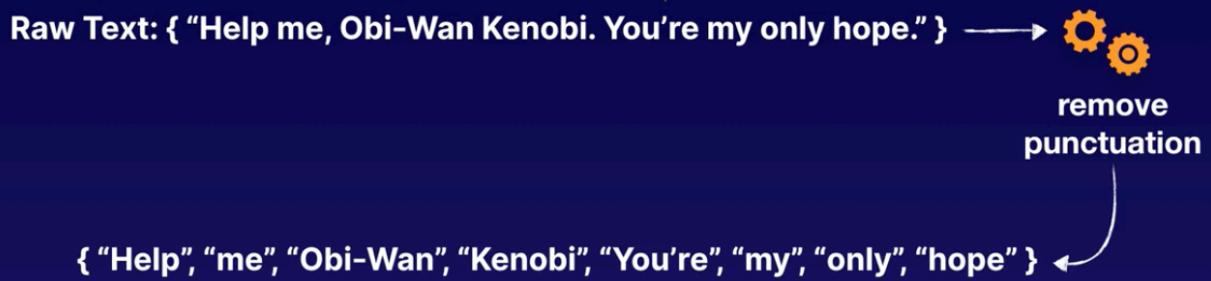
### • Remove Punctuation

Sometimes removing punctuation is a good idea if we do not need them.

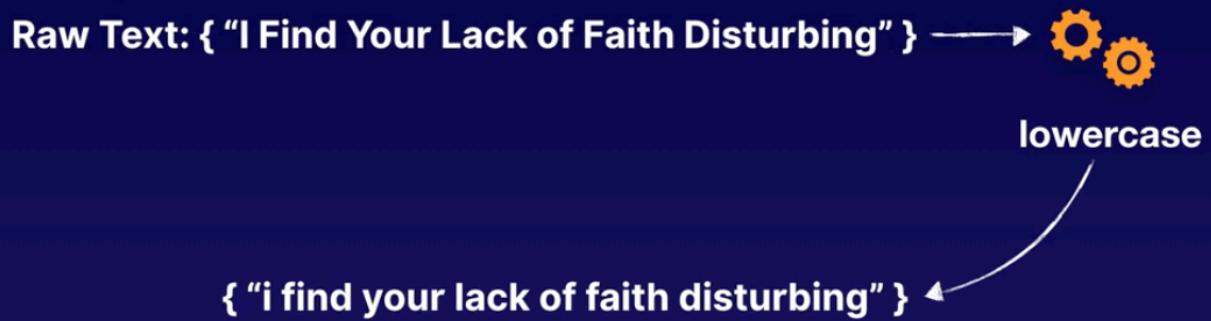
### • Lowercase Transformation

Using lowercase transformation can help standardize raw text.

**Example:**



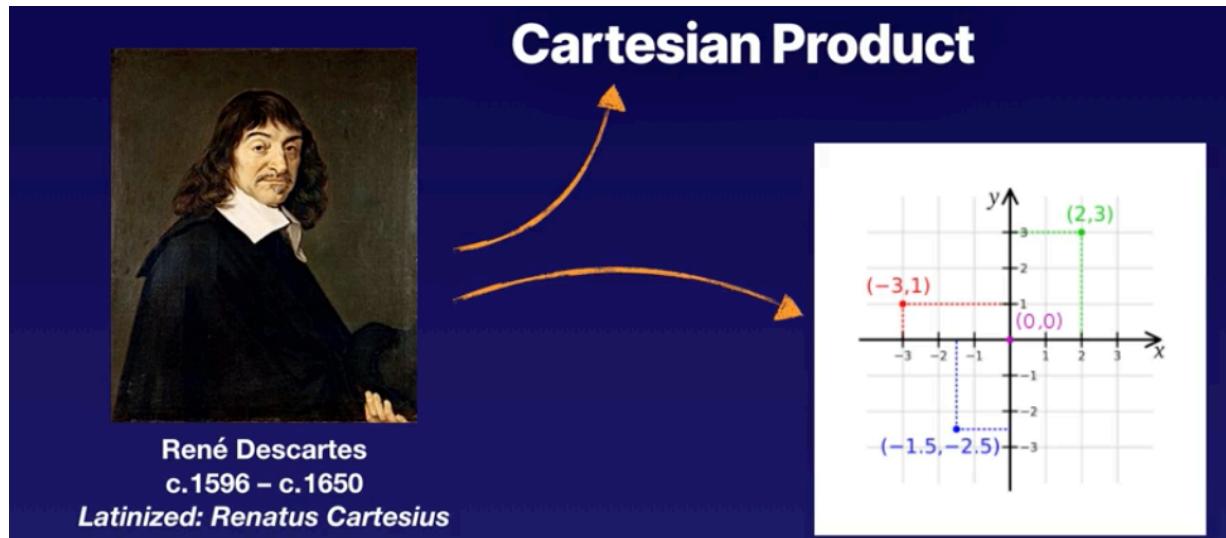
**Example:**



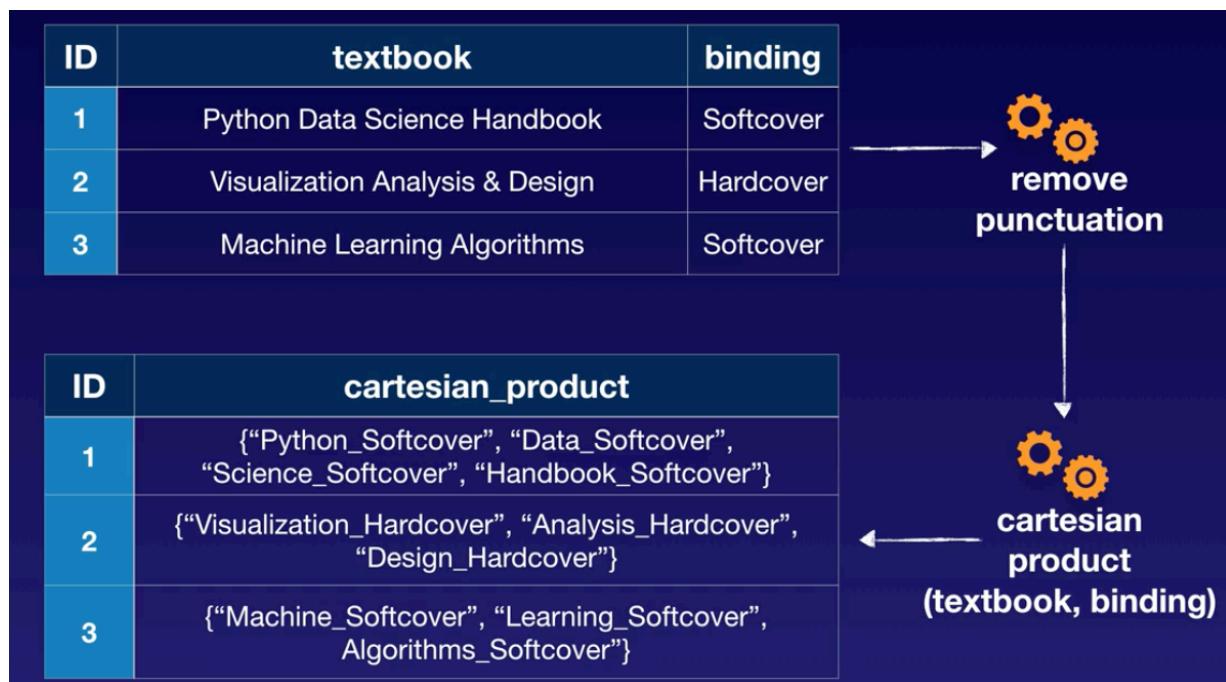
**Cartesian product Transformation** - creates a new feature from the combination

of two or more text or categorical values  
 In other words, it combines sets of words together

Cartesian product comes from Rene Descartes.



Example of Cartesian product Example, using "\_" to concatenate the words

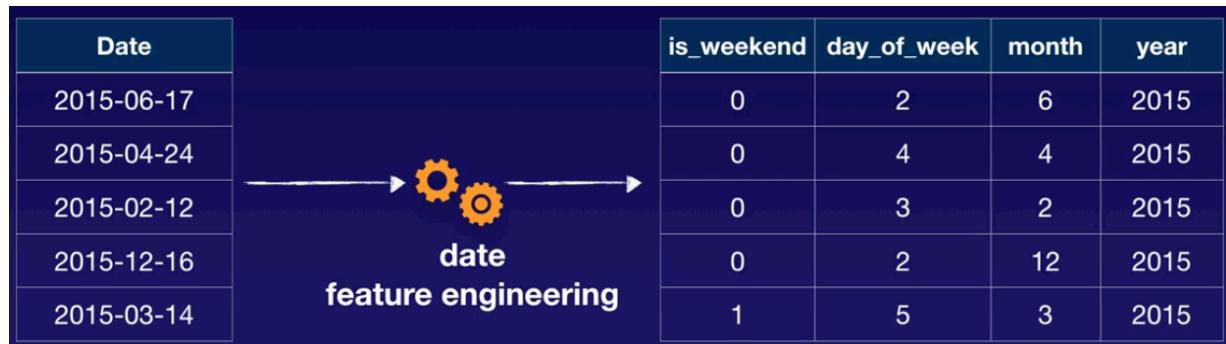


This can be used to make recommendations

### Feature Engineering Dates.

Dates come in many format - need to translate them into useful info

DATE FORMATS	EXTRACTED INFORMATION
• 2013-02-08T09	• Was it a weekend? Was it a week day?
• 6 Mar 17 21:22 UT	• Was the date the end of a quarter?
• Mon 06 Mar 2017 21:22:23 z	• What was the season?
• 09/28/2019	• Was the day a holiday?
	• Was it during business hours?
	• Was the World Cup taking place on this date?



With new features, we can now determine more important correlation or info about our observations.

### Recap of Text Feature Engineering:

Technique	Function
N-Gram	Splits text by whitespace with window size $n$
Orthogonal Sparse Bigram (OSB)	Splits text by keeping first word and uses delimiter with remaining whitespaces between second word with window size $n$
Term Frequency - Inverse Document Frequency (tf-idf)	Helps us determine how important words are within multiple documents
Removing Punctuation	Removes punctuation from text
Lowercase	Lowercase all the words in the text
Cartesian Product	Combines words together to create new feature
Feature Engineering Dates	Extracts information from dates and creates new features

### Numerical Feature Engineering

It's about transforming numeric values within our data so ML Lagos can better analyze them.

In other words, it's about changing numeric values in our datasets so they are easier to work with.

2 techniques:

- **Feature scaling**: take large numbers and scale them down to lower numbers and avoid skewing results
- **Binning**: takes numeric values and group them together in groups or bins.

### • **Feature Scaling**

Changes numeric values so all values are on the same scale.

- Normalization
- Standardization

### • **Binning**

Changes numeric values into groups or buckets of similar values.

- Quantile Binning aims to assign the same number of features to each bin.

## ***scaling = feature scaling = normalization***

### Normalization/Scaling.

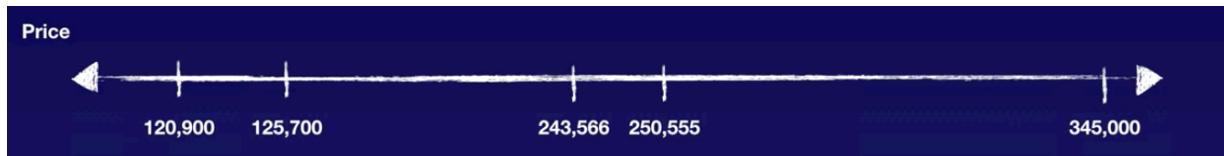
Pb - will a loan be approved.



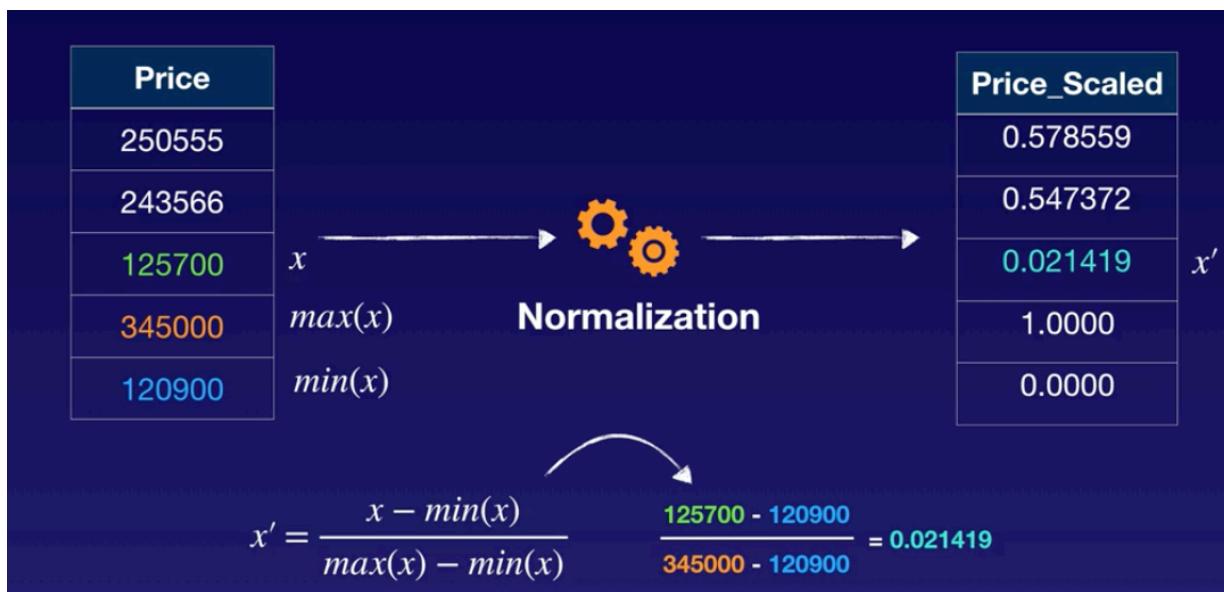
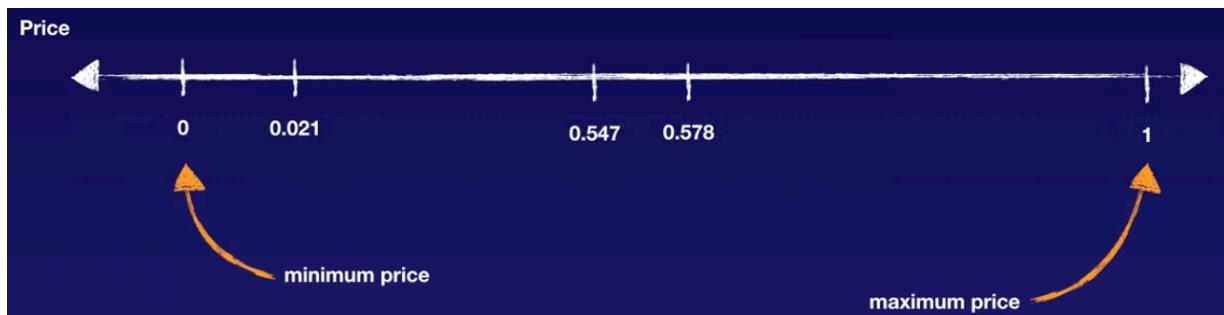
### **Problem**

Will a home loan be approved?

ID	Type	Bedrooms	Area	Pool_Size	Price	Loan_Approved
1	condo	2	2432	S	250555	N
2	house	4	2988	L	243566	Y
3	house	3	1877	N	125700	N
4	condo	5	3876	M	345000	Y
5	apartment	2	1250	N	120900	Y



Changed to:

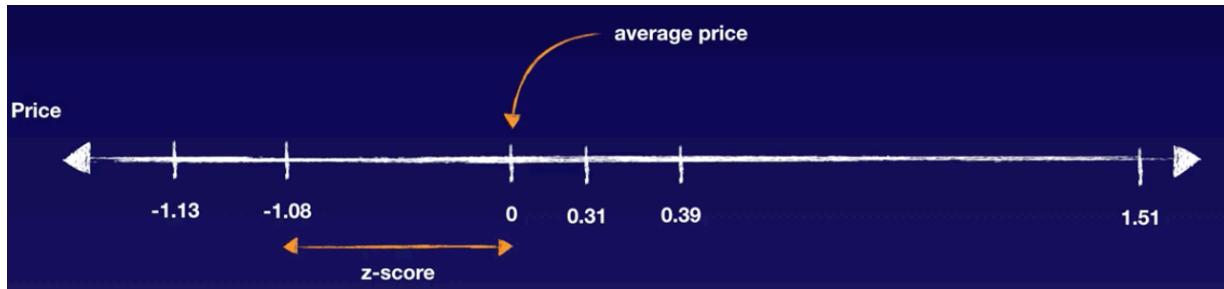


**Normalization** is the most common and easiest technique to scale down values.  
But pb is that **outliers** can throw off normalizers.

## Outliers can throw off normalization!

Another type of feature scaling is **Standardization**

Put avg on 0 and put z-score that takes into account stdev and smoothen the values so outliers are not as important



==> If outliers are a pb, use **Standardization** over **Normalization**

- **Normalization** is on a scale from 0 to 1
- **Standardization** uses the avg as 0 and other values scaled by their z-score

Scaling summary:

**1**

## Scaling Features

Is required for many algorithms like linear/non-linear regression, clustering, neural networks, and more. Scaling features depends on the algorithm you use.

**3**

## Standardization

Rescales values by making the values of each feature in the data have zero mean and is much less affected by outliers.

**2**

## Normalization

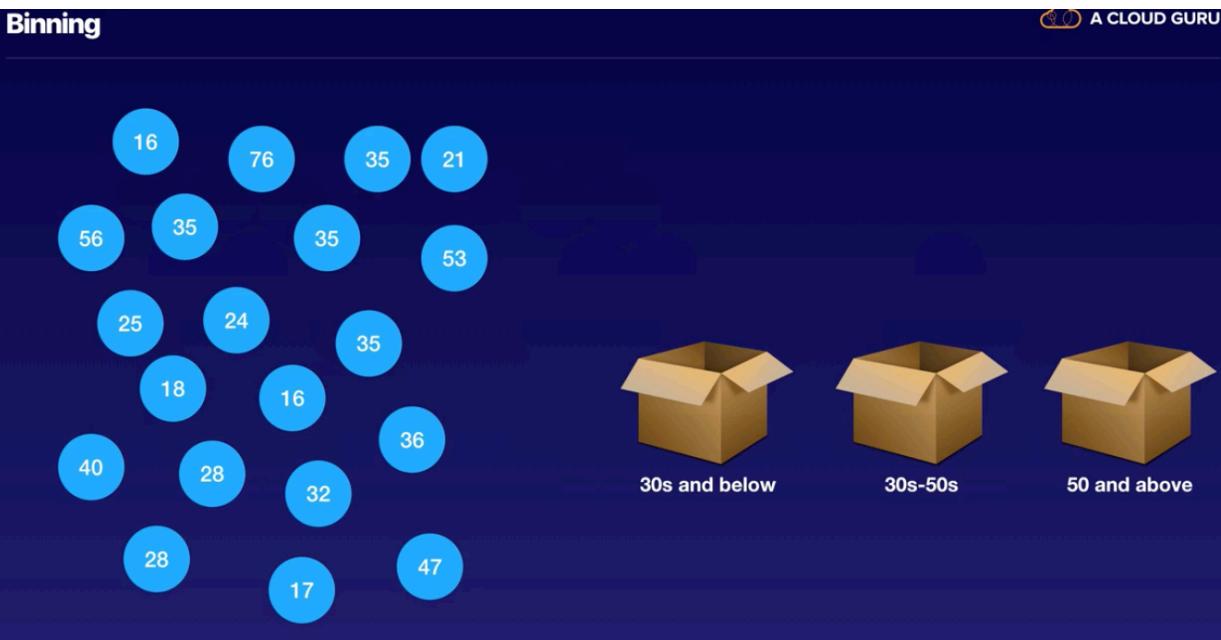
Rescales values from 0 to 1 but doesn't handle outliers very well.

**4**

## Translating Back

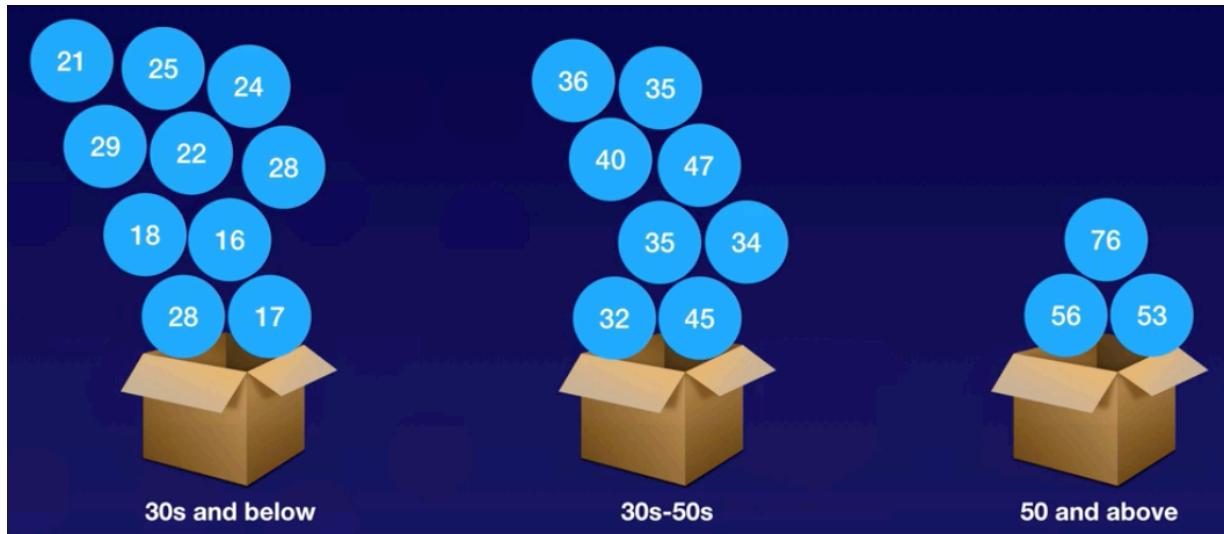
When you're done you can always scale back the data to the original representation.

## Binning



Example of age

Since we are not concerned about someone aged 50 or 51, we use bins and now we can use these as categorical variables



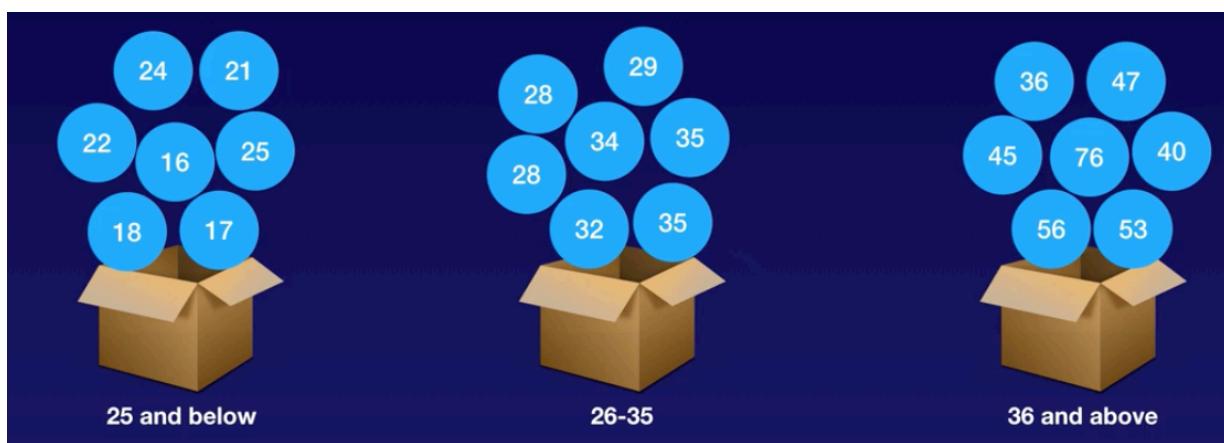
Pb - we can end up with dat beings queued

**We can end up with irregular bins which are not uniform...**

This is where we can apply:

Quantile Binning = “equals parts” + “groups”

Range of values now changes and we have an equal number of points



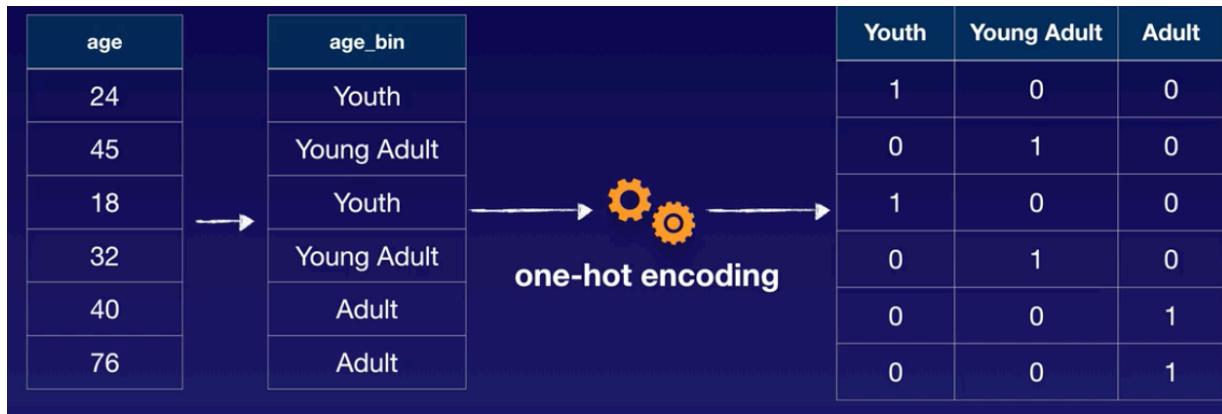
Now we can apply categorical variables to each bin: Youth, Young Adult and Adult

Youth

Young Adult

Adult

Then we can apply One-hot encoding



In Summary:

1	Binning	2	Quantile Binning	3	Optimum Number of Bins
	Is used to group together values to reduce the effects of minor observation errors.		Bins values into equal number of bins.		Depends on the characteristics of the variables and its relationship to the target. This is best determined through experimentation.

In summary:

Technique	Function
Normalization	From 0 to 1 0 - minimum value 1 - maximum value
Standardization	0 is the average value is the z-score
Quantile Binning	creates equal number of bins

## Other Feature Engineering



# **Image Feature Engineering**

Extracting useful information from images before using them with ML algorithms.

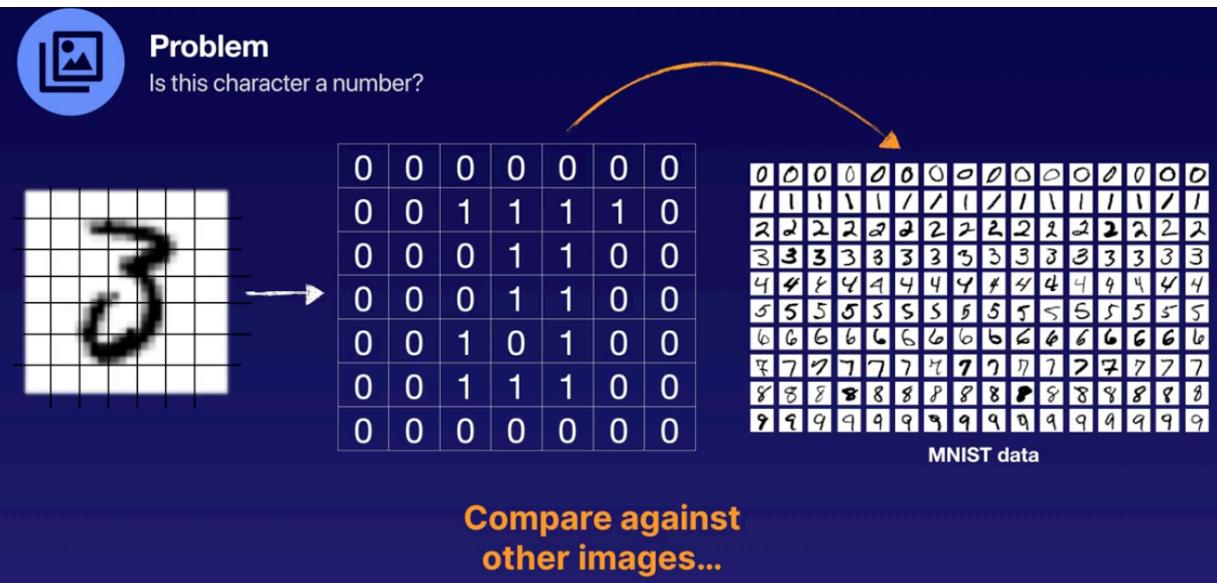


## Transforming images to find out useful information.



Pb: is this character a number?

Use a grid and compare it to other grid of values from the MNIST dataset



# Audio Feature Engineering

Extracting useful information from sounds and audio before using them with ML algorithms.



## Transforming audio data into something useful.





### Built-in algorithms from Amazon.

#### **Input:**

- File
- Pipe

File: AWS will copy the data from S3 to the EBS volume or whoever the algorithm runs

Pipe: stream from s3 to the instance => quicker startup => create tensor

Tensor is a multi dimensional array

File	Pipe
<ul style="list-style-type: none"> <li>• Loads all of the data from S3 directly onto the training instance volumes.</li> <li>• CSV</li> <li>• JSON</li> <li>• Parquet</li> <li>• Image files (.png or .jpg)</li> </ul>	<ul style="list-style-type: none"> <li>• Your datasets are streamed directly from Amazon S3.</li> <li>• recordIO-protobuf (creates tensor)</li> </ul>

Sample python code to transform data from s3 to a tensor (protobuf)

```

from sagemaker.amazon.common import write_numpy_to_dense_tensor
import io
import boto3

bucket = 'bucket-name' # Use the name of your s3 bucket here
data_key = 'kmeans_lowlevel_example/data'
data_location = 's3://{}{}'.format(bucket, data_key)

# Convert the training data into the format required by the algorithm
buf = io.BytesIO()
write_numpy_to_dense_tensor(buf, train_set[0], train_set[1])
buf.seek(0)

# location to upload to recordIO-protobuf data
boto3.resource('s3').Bucket(bucket).Object(data_key).upload_fileobj(buf)

```

**Feature Engineering is like a layered cake. Usually there are **multiple** layers of transformations done to properly **prepare** your **data**.**



### Handling Missing values



### **Missing Data**

Missing data can be represented in many different ways (*null*, *NaN*, *NA*, *None*, etc.) and handling missing values is an important data preparation step.

Before filling in null values, need to understand why are values missing in the first place?

**Missingness Mechanisms:**

- **MAR:** Missing at Random
- **MCAR:** Missing completely at Random
- **MNAR:** Missing not at Random

1	2	3
<p><b>Missing at Random (MAR)</b></p> <p>Missing at random means that the propensity for a data point to be missing is not related to the missing data, but it is related to some of the observed data.</p>	<p><b>Missing Completely at Random (MCAR)</b></p> <p>The fact that a certain value is missing has nothing to do with its hypothetical value and with the values of other variables.</p>	<p><b>Missing not at Random (MNAR)</b></p> <p>Two possible reasons are that the missing value depends on the hypothetical value or missing value is dependent on some other variable's value.</p>

## How to handle missing values?

- Supervised Learning with an algorithm like K-nearest neighbor or fuzzy-kmeans
- Mean
- Median
- Mode: most common value
- Drop the rows

Technique	Why this works	Ease of Use
Supervised learning	Predicts missing values based on the values of other features	Most difficult, can yield best results
Mean	The average value	Quick and easy, results can vary
Median	Orders values then chooses value in the middle	Quick and easy, results can vary
Mode	Most common value	Quick and easy, results can vary
Dropping rows	Removes missing values	Easiest but can dramatically change datasets

Replacing data is known as **imputation**

Technique	Why this works	Ease of Use
Supervised learning	Predicts missing values based on the values of other features	Most difficult, can yield best results
Mean	The average value	Quick and easy, results can vary
Median	Orders values then chooses value in the middle	Quick and easy, results can vary
Mode	Most common value	Quick and easy, results can vary
Dropping rows	Removes missing values	Easiest but can dramatically change datasets

**Replacing data is known as imputation**

## Feature Selection

=> Select the most relevant features from dataset to prevent over-complicating the analysis, resolving potential inaccuracies, and removes irrelevant features or repeated information.

In other words, it's deciding what to keep and what to get rid of

Example

 **Problem**  
Can old dogs learn new tricks?

ID	age	fur_color	born_month	born_month_num	breed	trick_learned
1	2	brown	July	7	terrier	0
2	4	white	August	8	shepard	1
3	3	golden	June	6	terrier	1
4	7	brown	September	8	collie	1
5	2	black	June	6	retriever	0
6	6	black	May	5	collie	1

# Feature selection is an **intuitive step** humans take to reduce the number of features.

## PCA - Principal Component Analysis

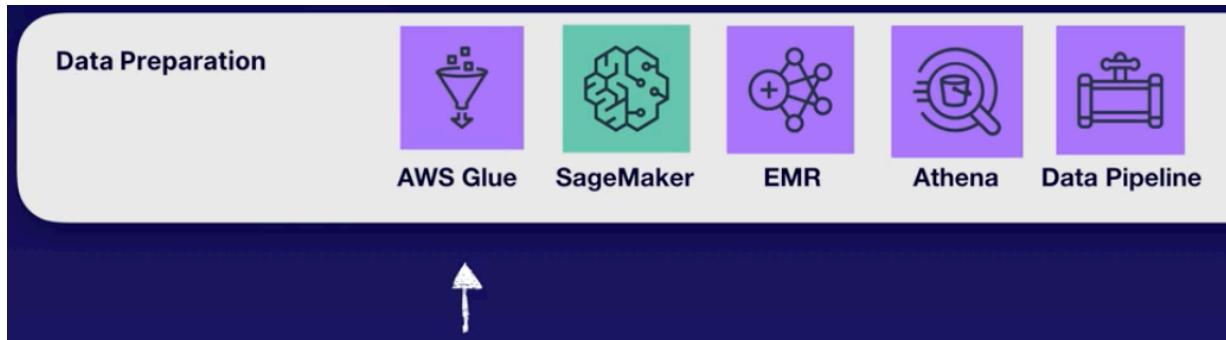
Similar to feature selection but it is an unsupervised ML technique that reduces the number of features while still retaining as much information as possible. In other words, it reduces the total number of features in a dataset.

Problem	Technique	Why
Data is too large due to the large number of features	Principle Component Analysis (PCA)	Algorithm that reduces total number of features
Useless features that do not help solve ML problem	Feature Selection	Remove features that do not help solve the problem

## Data Preparation Tools

- **AWS Glue** - most commonly used to transform data. Fully managed ETL  
<https://docs.aws.amazon.com/glue/latest/dg/how-it-works.html>  
AWS Glue uses other AWS services to orchestrate your ETL (extract, transform, and load) jobs to build data warehouses and data lakes and generate output streams. AWS Glue calls API operations to transform your data, create runtime logs, store your job logic, and create notifications to help you monitor your job runs. The AWS Glue console connects these services into a managed application, so you can focus on creating and monitoring your ETL work  
You don't need to create the infrastructure for an ETL tool because AWS Glue does it for you. When resources are required, to reduce startup time, AWS Glue uses an instance from its warm pool of instances to run your workload.  
With AWS Glue, you create jobs using table definitions in your Data Catalog. Jobs consist of scripts that contain the programming logic that performs the

transformation. You use triggers to initiate jobs either on a schedule or as a result of a specified event. You determine where your target data resides and which source data populates your target. With your input, AWS Glue generates the code that's required to transform your data from source to target. You can also provide scripts in the AWS Glue console or API to process your data.



## Data Sources

AWS Glue supports the following data sources:

- Data stores
  - Amazon S3
  - Amazon Relational Database Service (Amazon RDS)
  - Third-party JDBC-accessible databases
  - Amazon DynamoDB
- Data streams
  - Amazon Kinesis Data Streams
  - Apache Kafka

## Data Targets

AWS Glue supports the following data targets:

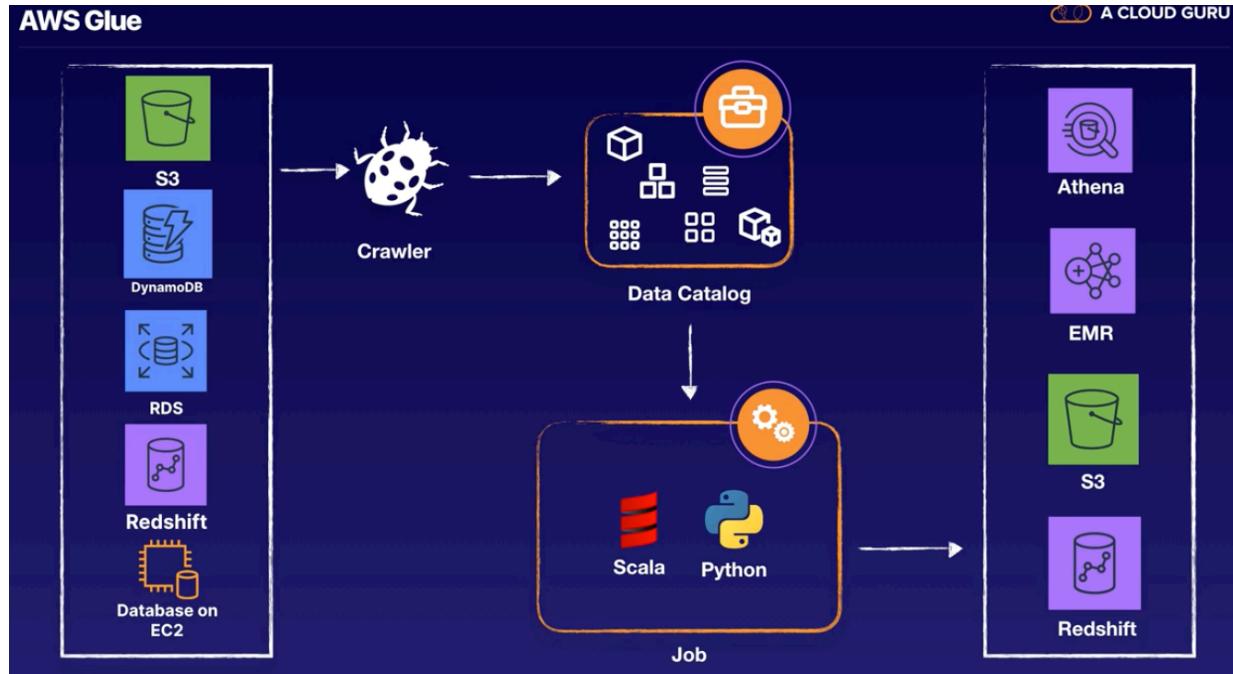
- Data stores
  - Amazon S3
  - Amazon Relational Database Service (Amazon RDS)
  - Third-party JDBC-accessible databases

Use one of the services built into AWS Glue: S3, DynamoDB, RDS, RedShift  
Set up a crawler, determines structure/schema. It then creates a data catalog (metadata of our data)

Once data catalog is created, we can set up jobs and run Scala or python jobs => categorical encoding, feature engineering, selection,...

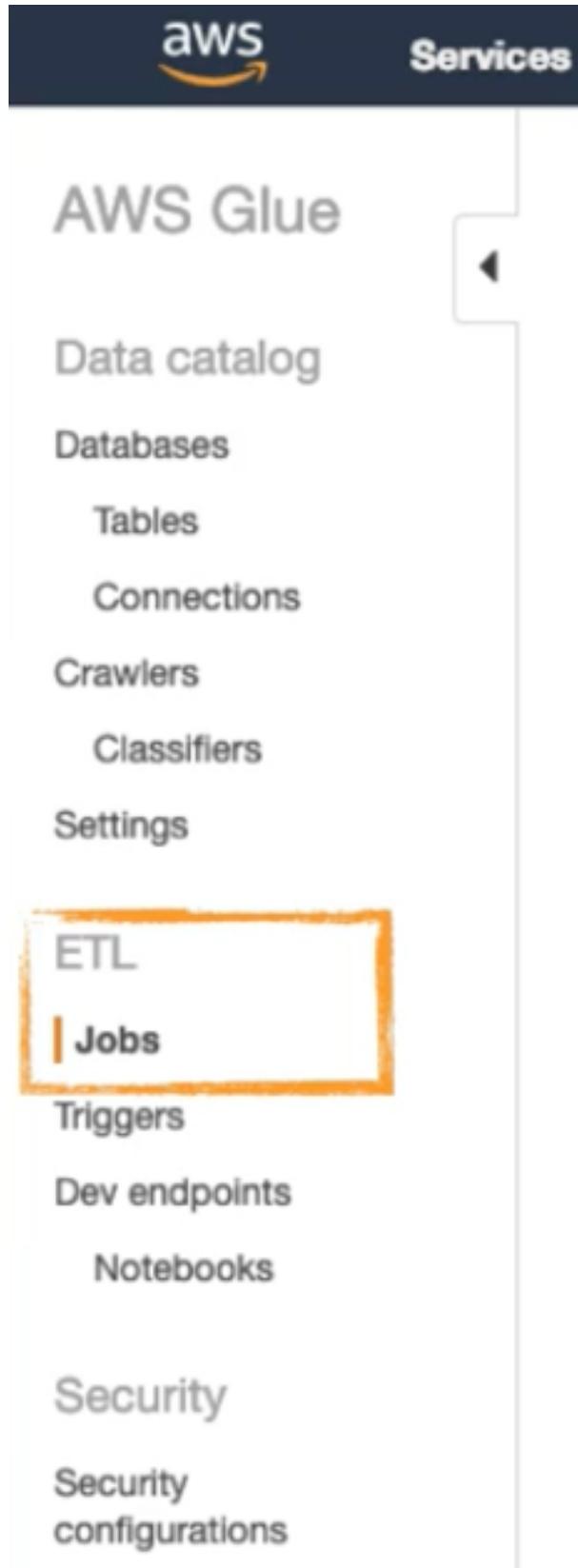
Jobs can run on demand, on schedule or when an AWS service is triggered (like S3 trigger)

Once data is transformed, it can then be put in a data target: S3, EMR, RedShift, Athena and use within SageMaker

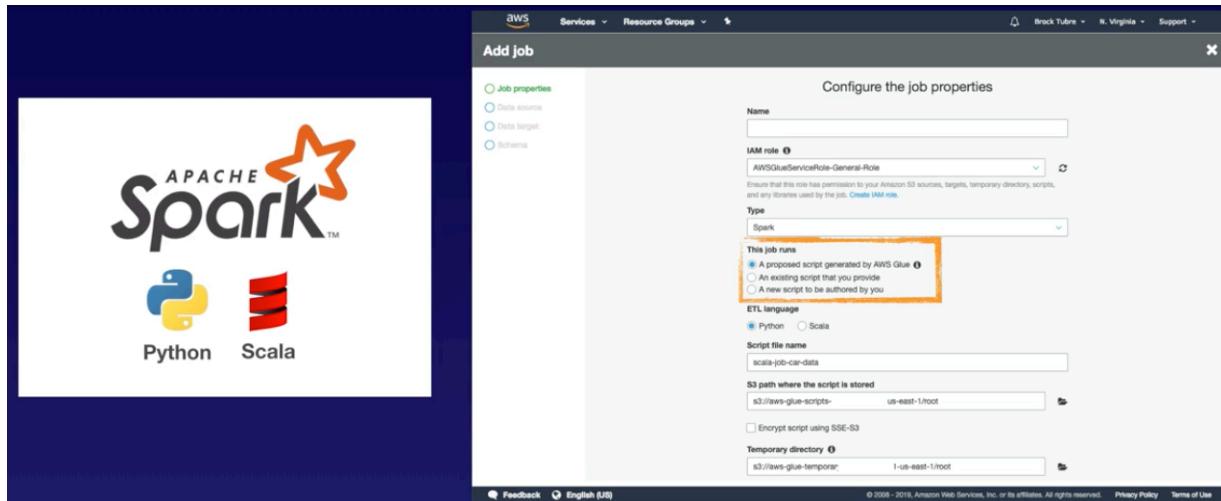


AWS Glue in console:

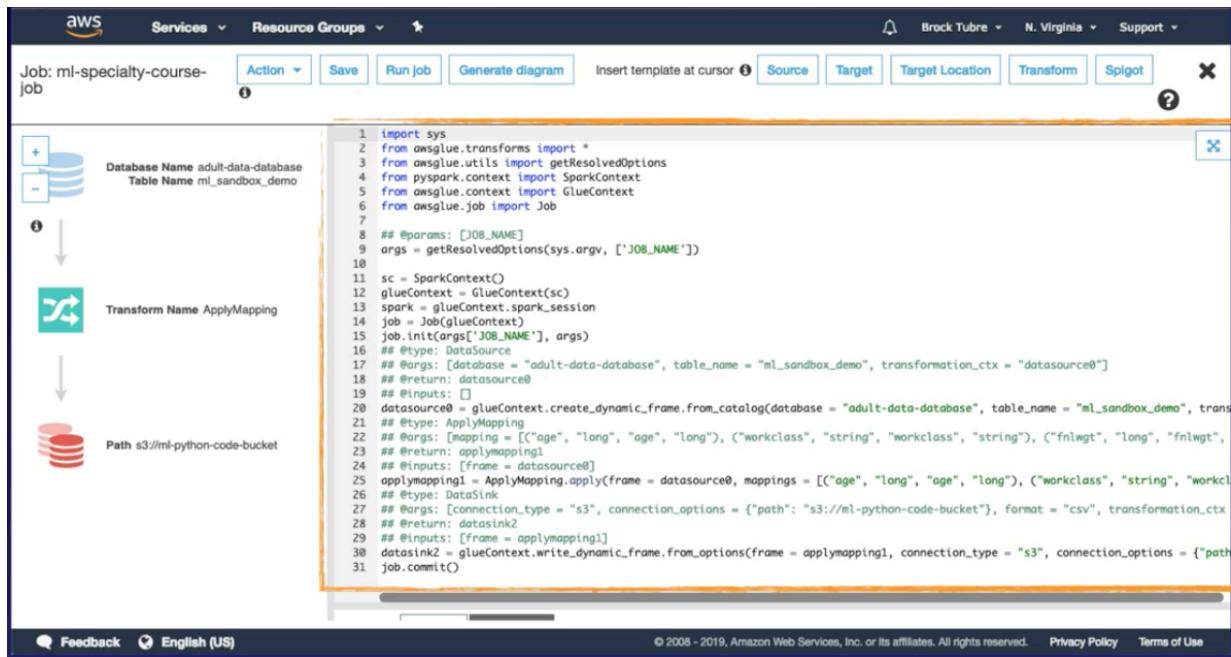
- Data Catalog: Database + crawler (classify the data source, infer the schema from the source,...). From the crawler, it creates tables in the DB
- ETL: jobs run to populate the tables



A job can be in Spark (default)  
We can then use Python or Scala for ETL language



If we let AWS Glue generate the code for us, it will look like this:



To note we can use built in Transforms to add to our scripts

## AWS Glue PySpark Transforms Reference

AWS Glue has created the following transform Classes to use in PySpark ETL operations.

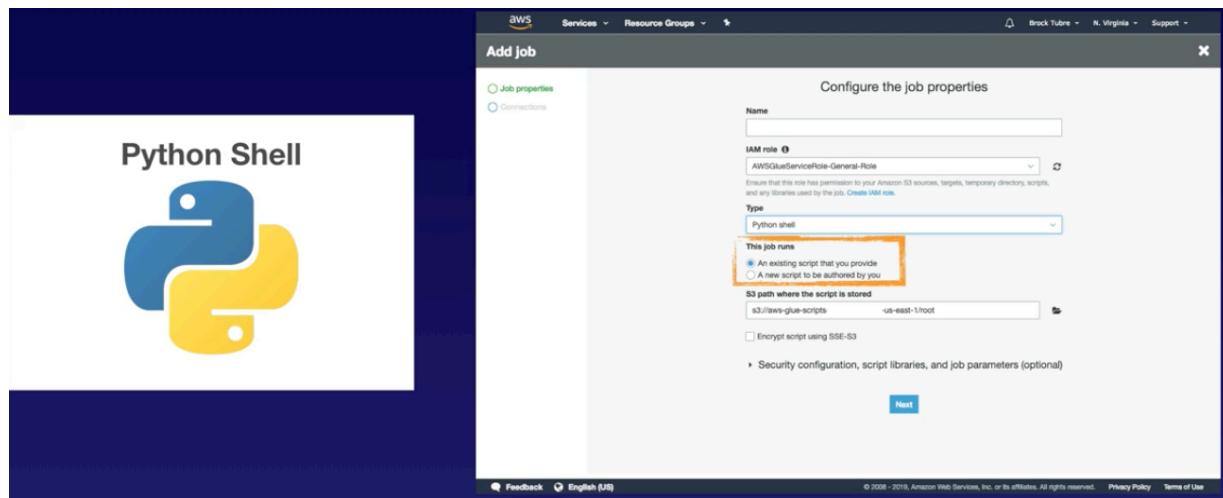
- [GlueTransform Base Class](#)
- [ApplyMapping Class](#)
- [DropFields Class](#)
- [DropNullFields Class](#)
- [ErrorsAsDynamicFrame Class](#)
- [Filter Class](#)
- [Join Class](#)
- [Map Class](#)
- [MapToCollection Class](#)
- [Relationalize Class](#)
- [RenameField Class](#)
- [ResolveChoice Class](#)
- [SelectFields Class](#)
- [SelectFromCollection Class](#)
- [Spigot Class](#)
- [SplitFields Class](#)
- [SplitRows Class](#)
- [Unbox Class](#)
- [UnnestFrame Class](#)

### Output Formats

| [avro](#)  
[csv](#)  
[ion](#)  
[grokLog](#)  
[json](#)  
[orc](#)  
[parquet](#)  
[xml](#)

The job will create "py-spark" score for us.

If we are more comfortable with traditional Python and libraries like numpy, pandas, scikit learn, we can use "Python Shell" as a job



AWS Glue

Data catalog

Databases

Tables

Connections

Crawlers

Classifiers

Settings

ETL

- Jobs**
- Triggers
- Dev endpoints
- Notebooks
- Security
- Security configurations
- Tutorials
- Add crawler
- Explore table
- Add job

Jobs

A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

Add job Action Filter by attributes

Name	Type	ETL language	Script location	Last modified	Job bookmark
new-test-job-python-s...	Python shell		s3://aws-glue-scripts-...	6 March 2019 5:43 PM...	Disable

History Details Script

```

1 import boto3
2 import pandas as pd
3 import io
4 from sklearn.preprocessing import MinMaxScaler
5
6 bucket='ml-sandbox-demo'
7 data_key = 'car_data.csv'
8
9 s3 = boto3.client('s3')
10 obj = s3.get_object(Bucket=bucket, Key=data_key)
11 df = pd.read_csv(io.BytesIO(obj['Body'].read()))
12
13 df.replace({'Make' : {'BMW':'Corvette'}}, inplace=True)
14 df.replace({'Make' : {'Ford':'Mustang'}}, inplace=True)
15 df.replace({'Make' : {'Toyota':'Camero'}}, inplace=True)
16
17 # Clean up any missing data
18 df.drop(columns=['Market Category'], inplace=True)
19 engine_hp_mean = df['Engine HP'].mean()

```

Edit script

Feedback English (US)

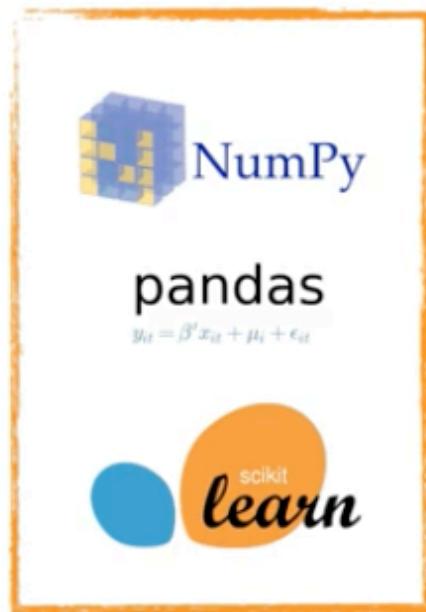
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Built-in libraries to transform our data:

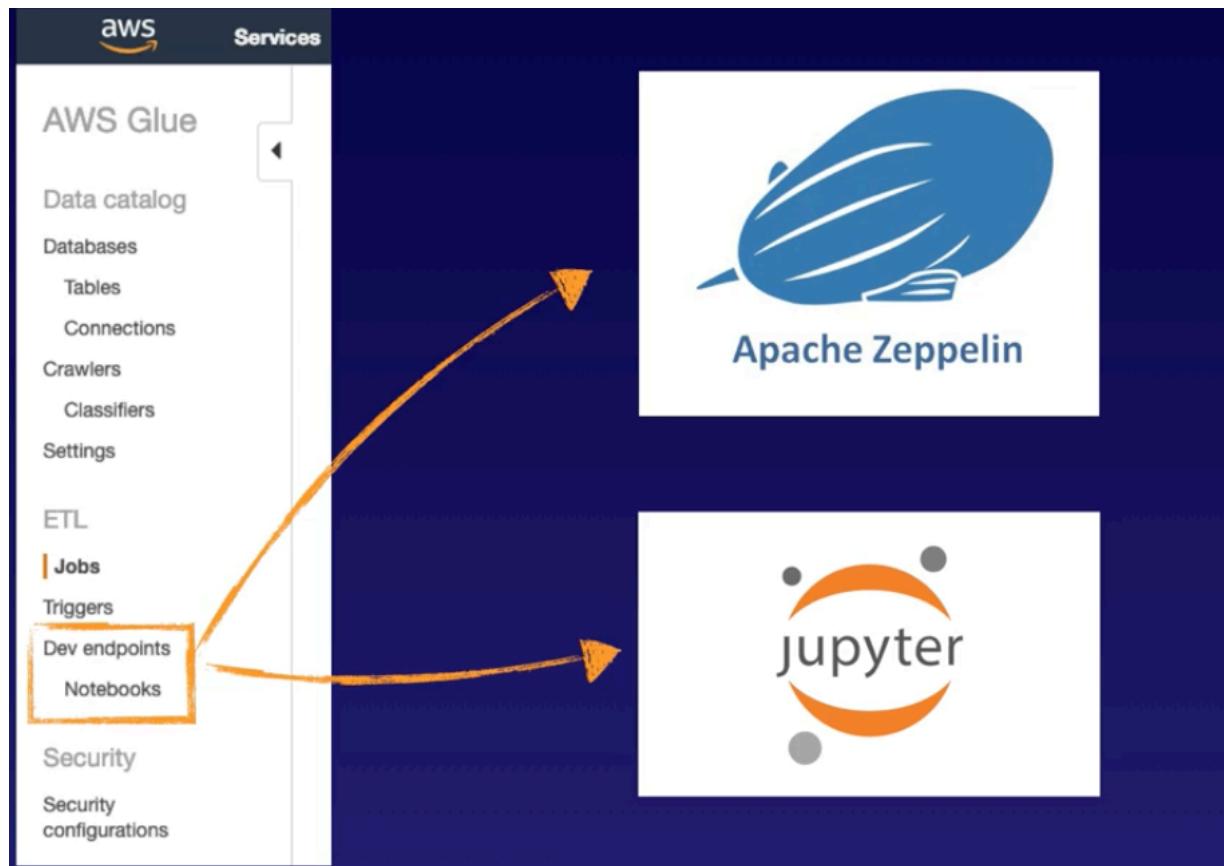
## Supported Libraries for Python Shell Jobs

The environment for running a Python shell job supports the following libraries:

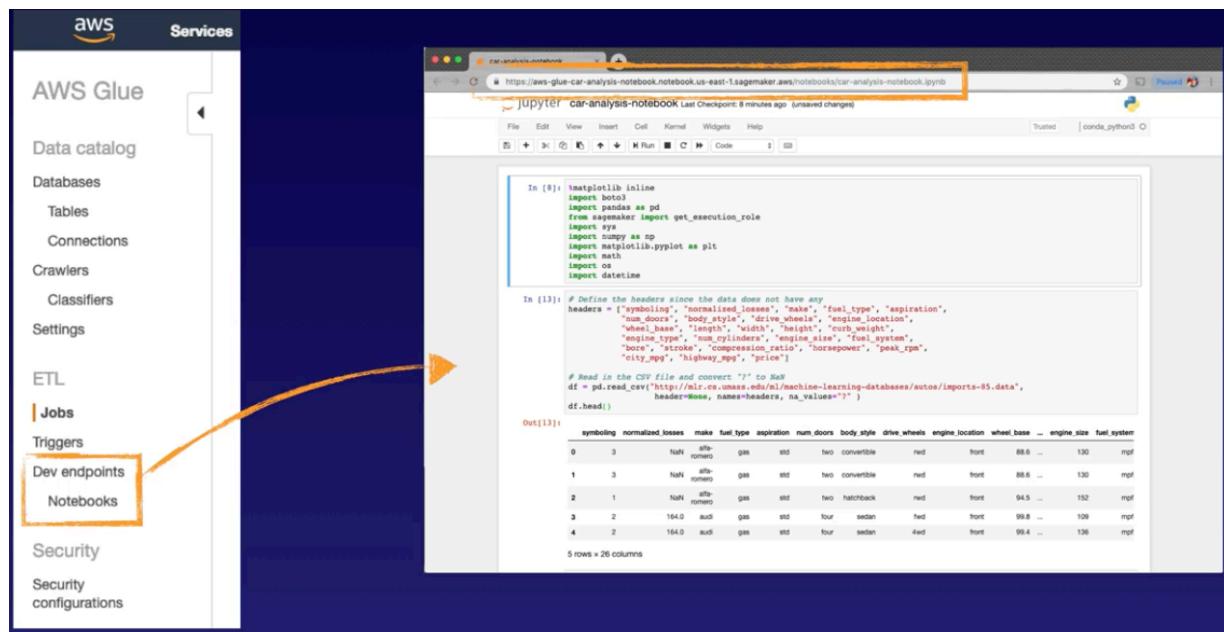
- Boto3
- collections
- CSV
- gzip
- multiprocessing
- NumPy
- pandas
- pickle
- re
- SciPy
- sklearn
- sklearn.feature\_extraction
- sklearn.preprocessing
- xml.etree.ElementTree
- zipfile



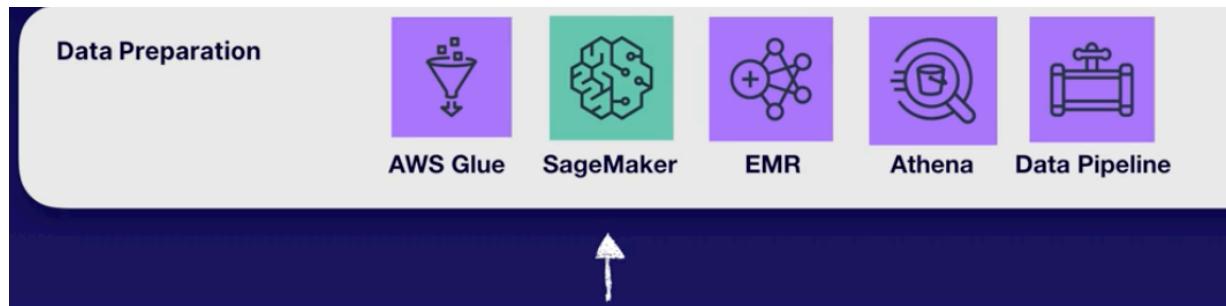
Lastly, AWS Glue allows us to create Zeppelin and Jupiter notebooks to do ad-hoc transformations



The notebook will be hosted as part of SageMaker:



## Other Data Preparation tool (and more than that): SageMaker



SageMaker allows us to create Jupiter notebooks directly integrated with the service.

The screenshot shows the Amazon SageMaker dashboard. The left sidebar includes options for Ground Truth, Notebook, Training, and Inference. The 'Notebook' section is highlighted with an orange box. Within the 'Notebook' section, 'Labeling jobs' and 'Notebook instances' are also highlighted with orange boxes. The main 'Overview' section features four cards: 'Ground Truth', 'Notebook', 'Training', and 'Inference'. The 'Notebook' card is the primary focus, containing sub-options for 'Labeling jobs' and 'Notebook instances'. Below the 'Notebook' card, there are buttons for 'Training jobs', 'Hyperparameter tuning jobs', 'Models', 'Endpoints', and 'Batch transform jobs'. At the bottom of the dashboard, there is a 'Recent activity' section with a note about no recent activity.

From the notebooks, we can use many of the python libraries most common

```

In [26]: %matplotlib inline
import boto3
import pandas as pd
from sagemaker import get_execution_role
import sys
import numpy as np
import matplotlib.pyplot as plt
import math
import os
import datetime

In [27]: role = get_execution_role()
bucket="ml-sandbox-demo"
data_key = 'car_data.csv'
data_location = 's3://{}{}'.format(bucket, data_key)

df = pd.read_csv(data_location)

In [28]: df.replace({'Make': {'BMW': 'Corvette'}}, inplace=True)
df.replace({'Make': {'Ford': 'Mustang'}}, inplace=True)
df.replace({'Make': {'Toyota': 'Camaro'}}, inplace=True)

# Ford
# Toyota
# Corvette
df = pd.DataFrame({ 'price': df['MSRP'],
                     'car': df['Make'],
                     'date': df['Year']})

df = df[df['car'].isin(['Corvette', 'Mustang', 'Camaro'])]
df['car'] = df['car'].astype('category')
df.head()

```

Out[28]:

	car	date	price
0	Corvette	2011	46135
1	Corvette	2011	40650
2	Corvette	2011	36350
3	Corvette	2011	26440

We can also install other modules with package managers like conda or pip

**conda**

```

$ h-4.29 conda install scikit-learn
Proceed (y/n)? y

Downloading and Extracting Packages
numpy-base-1.16.2          4.4 MB
conda-4.4.8                 1.7 MB
certifi-2018.3.13.g         131 KB
scikit-learn-0.20.3         5.7 MB
openblas-0.2.0rc2           3.2 KB
mkl-service-1.1.1            132 KB
mkl-1.1.1                   100B
scipy-1.2.1                  17.7 MB
Preparing transaction: done
Verifying transaction: done
Resolving transaction: done
Executing transaction: done
sh-4.29

```

**pip**

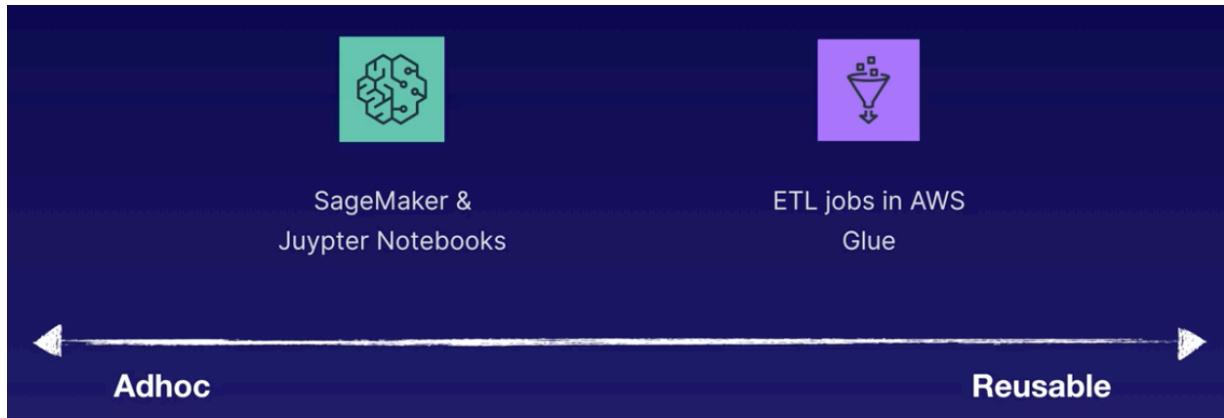
```

$ h-4.29 pip install scikit-learn
Collecting scikit-learn
  Downloading https://files.pythonhosted.org/packages/fe/82/c0d6339d11b820dd088599ac0bbfbbebd8ca4/scikit-learn-0.20.3-py3-none-any.whl (14.4MB)
    100% |████████████████████████████████| 14.4MB 10.0MB/s
Requirement already satisfied: numpy<1.17, >=1.16 in ./anaconda3/envs/JupyterSystemEnv/lib/python3.6/site-packages (from scikit-learn)
Requirement already satisfied: mkl-service<1.1.1, >=1.0.0 in ./anaconda3/envs/JupyterSystemEnv/lib/python3.6/site-packages (from scikit-learn)
Requirement already satisfied: openblas<0.3.17, >=0.3.16 in ./anaconda3/envs/JupyterSystemEnv/lib/python3.6/site-packages (from scikit-learn)
Requirement already satisfied: mkl<1.1.1, >=1.0.0 in ./anaconda3/envs/JupyterSystemEnv/lib/python3.6/site-packages (from scikit-learn)
Requirement already satisfied: certifi<2019.3.9, >=2018.3.13 in ./anaconda3/envs/JupyterSystemEnv/lib/python3.6/site-packages (from scikit-learn)
Requirement already satisfied: scipy<1.3.3, >=1.2.1 in ./anaconda3/envs/JupyterSystemEnv/lib/python3.6/site-packages (from scikit-learn)
Installing collected packages: scikit-learn
Successfully installed scikit-learn-0.20.3
sh-4.29

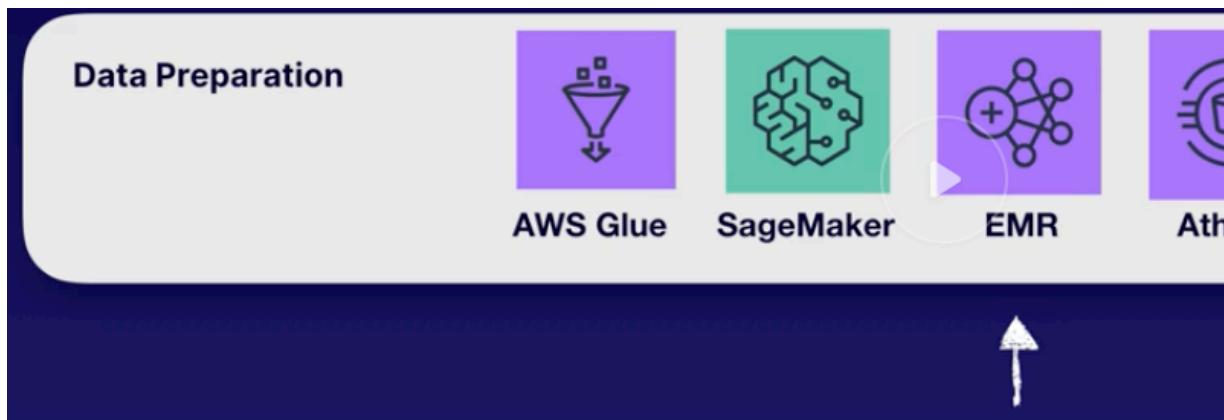
```

So use AWS Glue for long standing or repetitive transformations to do on schedule or when actions are required

Use Jupiter notebooks for quick transformation or ad-hoc transformation on the data before it feeds the model.



### Other data transformation tool: EMR



EMR is a fully managed Hadoop cluster that runs on multiple EC2 instances. It allows us to use different frameworks like Tensorflow, spark, flume, Jupiter, Monet,...



It's more work to set up transformation than SageMaker.  
It is great at scaling petabytes of data.

To note sageMaker and EMR can integrate together. We can use SageMaker SDK to connect it to an existing EMR cluster.

So if data is set up in EMR cluster, we can use Apache Spark to integrate directly with SageMaker.

Least amount of effort = AWS Glue. Since it is a fully managed service, we don't need to set it up like we would have to do with EMR.

When creating cluster within EMR, we can use frameworks lthat we want to include in the cluster

And since Jupiter is built into EMR, we can use the notebooks to transform the data and use services like Hive or Apache Spark to run the ETL jobs

Create Cluster - Advanced Options

Software Configuration

Release emr-5.20.0

<input checked="" type="checkbox"/> Hadoop 2.8.5	<input type="checkbox"/> Zeppelin 0.8.0	<input type="checkbox"/> Livy 0.5.0
<input checked="" type="checkbox"/> JupyterHub 0.9.4	<input type="checkbox"/> Tez 0.9.1	<input type="checkbox"/> Flink 1.6.2
<input type="checkbox"/> Ganglia 3.7.2	<input type="checkbox"/> HBase 1.4.8	<input type="checkbox"/> Pig 0.17.0
<input checked="" type="checkbox"/> Hive 2.3.4	<input checked="" type="checkbox"/> Presto 0.214	<input type="checkbox"/> ZooKeeper 3.4.13
<input checked="" type="checkbox"/> MXNet 1.3.1	<input type="checkbox"/> Sqoop 1.4.7	<input checked="" type="checkbox"/> Mahout 0.13.0
<input type="checkbox"/> Hue 4.3.0	<input type="checkbox"/> Phoenix 4.14.0	<input type="checkbox"/> Oozie 5.0.0
<input checked="" type="checkbox"/> Spark 2.4.0	<input type="checkbox"/> HCatalog 2.3.4	<input checked="" type="checkbox"/> TensorFlow 1.12.0

The screenshot shows a Jupyter notebook interface running on an AWS EMR cluster. The title bar says "jupyter ml-course-sandbox-notebook (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Kernel. A toolbar below has icons for New, Open, Save, Run, Cell, Code, and Help.

A box highlights the "Supported Kernels" section:

- PySpark
- PySpark3
- Python3
- Spark
- SparkR

The code cell (In [8]) contains Python code for calculating pi using PySpark:

```
import sys
from random import random
from operator import add

from pyspark import SparkContext

if __name__ == "__main__":
    ...
    Usage: pi [partitions]
...
sc = SparkContext(appName="PythonPi")
partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2
n = 100000 * partitions

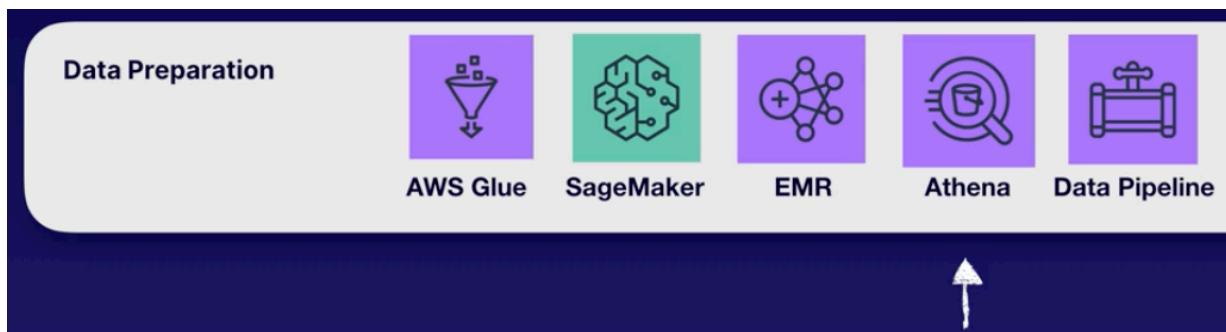
def f(_):
    x = random() * 2 - 1
    y = random() * 2 - 1
    return 1 if x ** 2 + y ** 2 < 1 else 0

count = sc.parallelize(xrange(1, n + 1), partitions).map(f).reduce(add)
print "Pi is roughly %f" % (4.0 * count / n)
sc.stop()
```

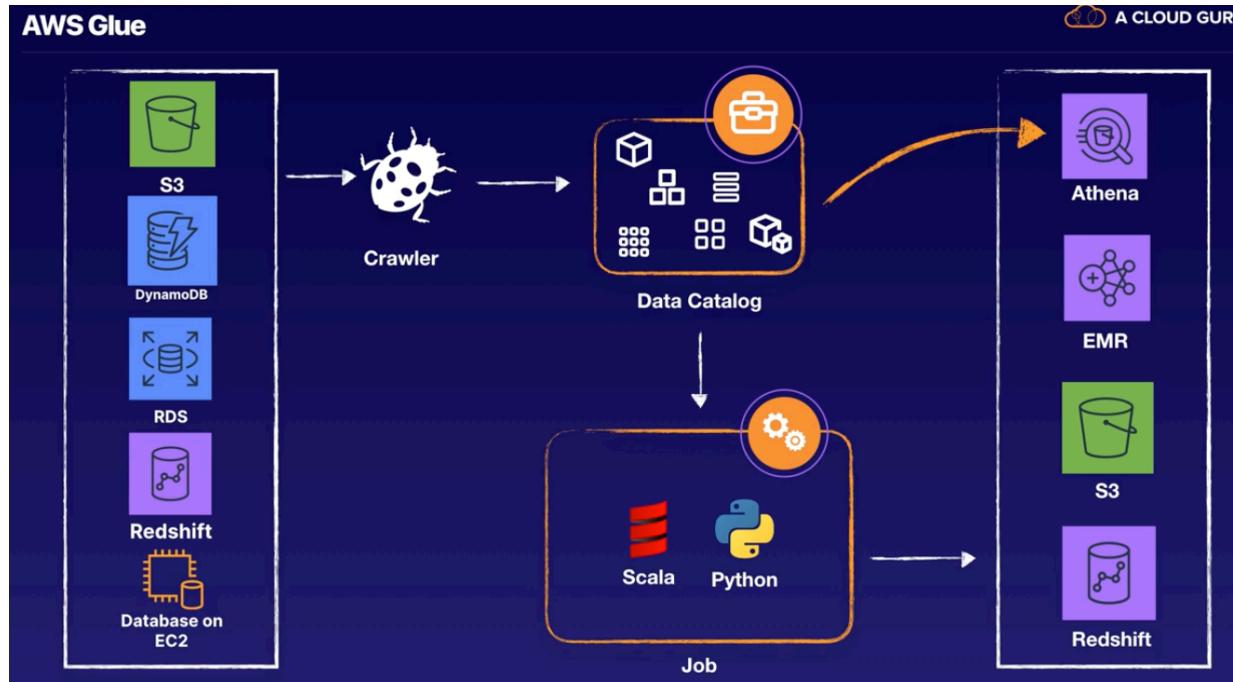
The status bar at the bottom shows "In [ ]:"

## Other Data preparation tool: Athena

To run SQL queries on S3 data. Fully managed server less service.



We can set a data catalog with AWS Glue, then query it with Athena



Screenshot of the AWS Athena Query Editor interface. The top navigation bar shows 'Athena' selected, along with 'Services', 'Resource Groups', and other options. The main area has tabs for 'Query Editor' (selected), 'Saved Queries', 'History', and 'AWS Glue Data Catalog'. A 'Workgroup : primary' dropdown is also present. On the left, the 'Database' section shows 'adult-data-database' selected, with a table named 'ml\_sandbox\_demo' containing 1 table and 8 views. The 'Tables (1)' section lists columns: age, workclass, fnlwgt, education, education\_num, marital\_status, occupation, relationship, race, sex, and capital\_gain. The 'Views (0)' section indicates no views have been created. In the center, a 'New query 1' editor window contains the following SQL query:

```

1 SELECT * FROM ml_sandbox_demo
2 WHERE sex = 'Male'
3 AND age > 39 limit 10;

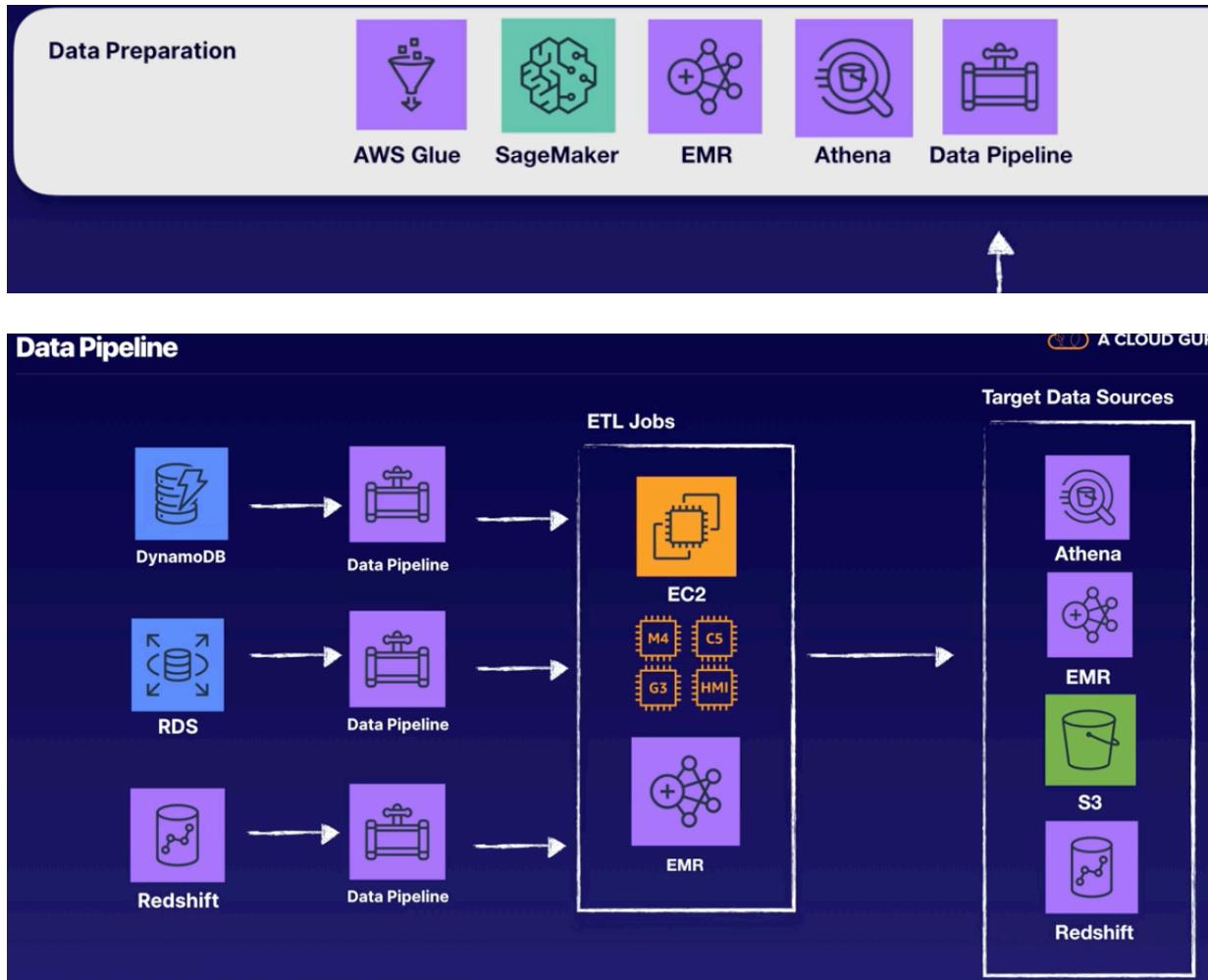
```

Below the query editor are buttons for 'Run query', 'Save as', and 'Create'. A status message says '(Run time: 1.68 seconds, Data scanned: 1.12 MB)'. At the bottom, there are 'Format query' and 'Clear' buttons, and a note: 'Use Ctrl + Enter to run query, Ctrl + Space to autocomplete'. The bottom section displays the 'Results' of the query, showing 8 rows of data:

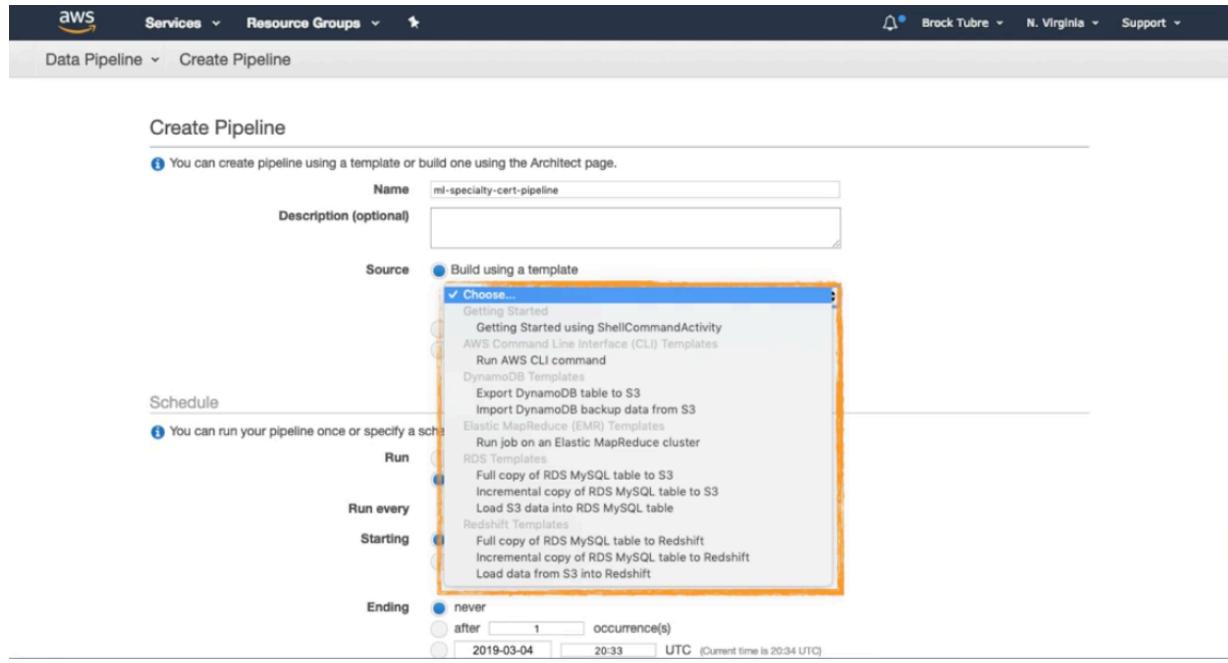
	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain
1	50	Self-emp-not-inc		Bachelors		Married-civ-spouse	Exec-managerial	Husband	White	Male	
2	53	Private		11th		Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	
3	52	Self-emp-not-inc		HS-grad		Married-civ-spouse	Exec-managerial	Husband	White	Male	
4	42	Private		Bachelors		Married-civ-spouse	Exec-managerial	Husband	White	Male	
5	40	Private		Assoc-voc		Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	
6	40	Private		Doctorate		Married-civ-spouse	Prof-specialty	Husband	White	Male	
7	43	Private		11th		Married-civ-spouse	Transport-moving	Husband	White	Male	
8	56	Local-gov		Bachelors		Married-civ-spouse	Tech-support	Husband	White	Male	

## Last data transform: Data Pipeline

Process and move data between different AWS compute services : from Dynamo, RDS, Redshift to Athena, EMR, S3,...



When we want to use Java or Java script, we can load that into EC2 and use the dAta Pipeline to feed these algorithms



## Which service to use?

Most cases: AWS Glue with python or Scala

We can also use Athena for simple queries against S3 if we know the data

If we want to use a language outside of python or Scala, use the data pipeline

Datasource	Data Preparation Tool	Why
S3, Redshift, RDS, DynamoDB, On Premise DB	AWS Glue	Use Python or Scala to transform data and output data into S3
S3	Athena	Query data and output results into S3
EMR	PySpark/Hive in EMR	Transform petabytes of distributed data and output data into S3
RDS, EMR, DynamoDB, Redshift	Data Pipeline	Setup EC2 instances to transform data and output data into S3