AWS

MACHINE LEARNING

CERTIFICATION

DOMAIN #2: EXPLORATORY DATA ANALYSIS (24% EXAM)

INTRODUCTION

AWS ML CERTIFICATION EXAM DOMAINS

Domain	% of Examination
Domain 1: Data Engineering	20%
Domain 2: Exploratory Data Analysis	24%
Domain 3: Modeling	36%
Domain 4: Machine Learning Implementation and Operations	20%
TOTAL	100%

Source: https://d1.awsstatic.com/training-and-certification/docs-ml/AWS%20Certified%20Machine%20Learning%20-%20Specialty_Exam%20Guide%20(1).pdf

DOMAIN #2 OVERVIEW: WHERE ARE WE NOW?!

SECTION #5: JUPYTER NOTEBOOKS, SCIKIT LEARN, PYTHON PACKAGES, AND DISTRIBUTIONS

- Introduction
- Jupyter Notebooks and Scikit Learn
- Python Packages (Pandas, Numpy, Matplotlib and Seaborn
- Distributions (Normal, Standard, Poisson, Bernoulli)
- Time Series

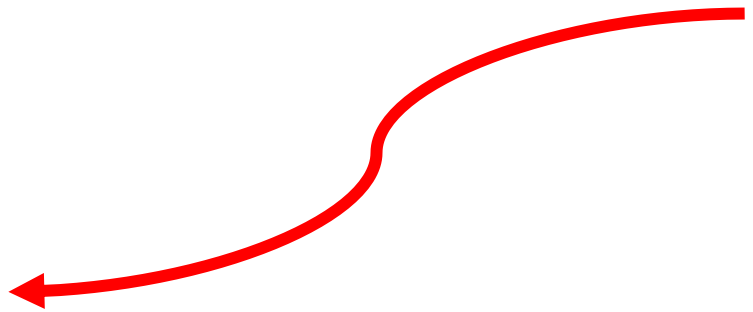
SECTION #6: AMAZON ATHENA, QUICKSIGHT AND ELASTIC MAP REDUCE

- Amazon Athena Features
- Amazon Athena Deep Dive (Security, Cost, and glue integration)
- Amazon QuickSight Features
- Amazon QuickSight (integration with AWS services)
- Amazon QuickSight ML insights and Use Cases
- Elastic Map Reduce (EMR)
- Apache Hadoop with EMR
- Apache Spark with EMR

DOMAIN #1 OVERVIEW:

SECTION #7: FEATURE ENGINEERING

- Introduction to Feature Engineering
  - Amazon SageMaker GroundTruth
  - Feature Selection
  - Scaling
  - Imputation
  - Outliers
  - One Hot Encoding
  - Binning
  - Log Transformation
  - Shuffling, Feature Splitting, Unbalanced Datasets
  - Text Feature Engineering overview
  - Bag of words, punctuation, and dates (easy ones!)
  - Term Frequency Inverse Document Frequency (TF-IDF)
  - N-Grams (Unigram vs. Bigram vs. Trigram)
  - Orthogonal Sparse Bigram (OSB)
  - Cartesian Product Transformation

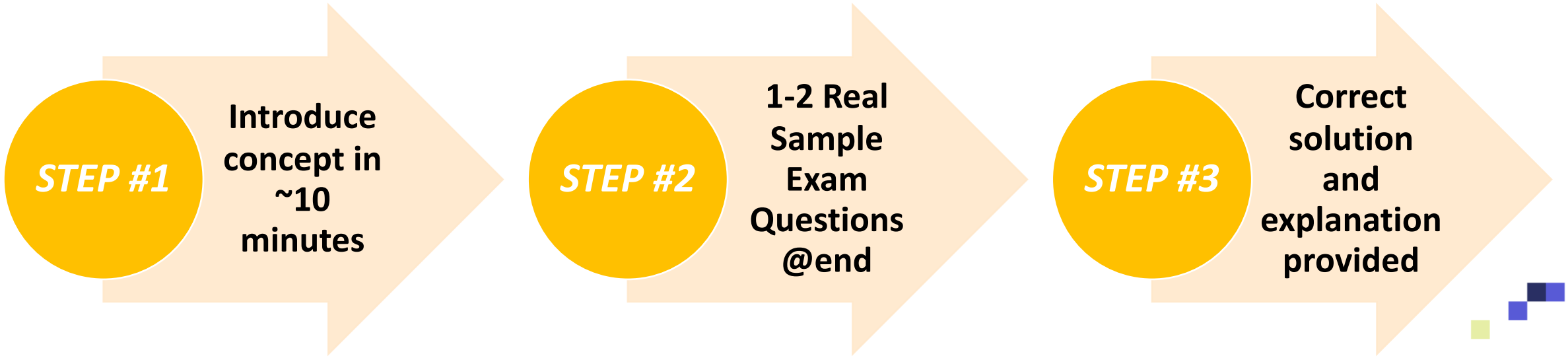


LECTURE DESIGN

- We know how hard it is to study for an exam especially if you have a busy schedule.
- This course is designed to be extremely on point and optimized to pass the exam.

No boring content. Zero unnecessary information.

- Here's the lecture structure that we will follow:

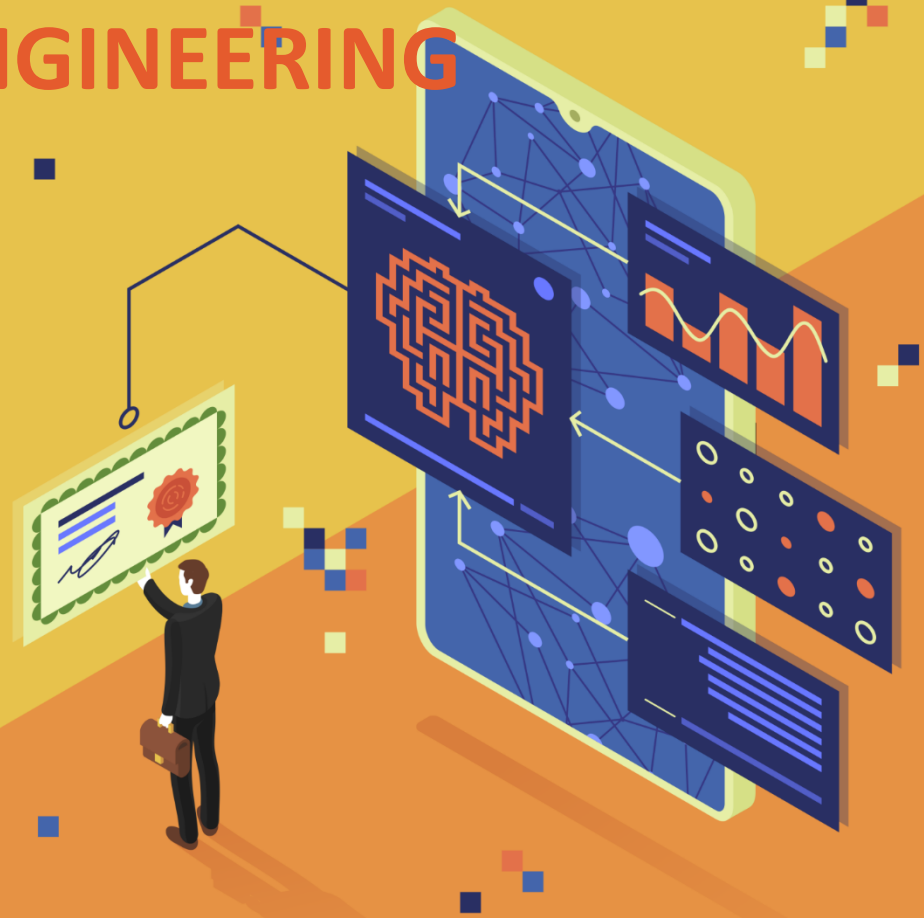


RECALL OUR MINI CHALLENGE AND PRIZE!

- For those of you who will successfully complete the entire section and watch the videos till the end, they will receive a valuable prize!

https://pixabay.com/illustrations/trophy-win-prize-transparent-1414791/

INTRODUCTION TO FEATURE ENGINEERING



WHAT IS FEATURE ENGINEERING?

- Machine Learning Algorithms require training data to train.
- Feature engineering is a critical task that data scientists have to perform prior to training the AI/ML models.
- As a data scientist, you may need to:
  - Highlight important information in the data
  - Remove/isolate unnecessary information (e.x.: outliers).
  - Add your own expertise and domain knowledge to the alter the data.
- Feature engineering is an art of introducing new features that weren't existing before.
- Data scientists spend 80% of their time performing feature engineering.
- The remaining 20% is the easy part which includes training the model and performing hyperparameters optimization.
- Performing proper feature engineering is crucial to improve AI/ML model performance.

Photo Credit: https://pixabay.com/illustrations/network-data-memory-data-collection-4478146/

FEATURE ENGINEERING: PROPER QUESTIONS TO ASK?

- As a data scientist, you need to answer the following questions:
  - ***What are the capabilities of the ML model I have?***
  - ***Which features should I select?***
  - ***Can I add my domain knowledge to use less features?***
  - ***Can I come up with new features from the data I have at hand?***
  - ***What should I put in the missing data locations?***

- It is important to choose features that are most relevant to the problem.
- Adding new features that are unnecessary will increase the computational requirements needed to train the model (curse of dimensionality).
- There are numerous techniques that could be used to reduce the number of features (compress/encode the data) such as Principal Component Analysis (PCA) – This will be covered in the next sections.

FEATURE ENGINEERING: EXAMPLE

- Let's take a look at this data and see what's wrong with it!

ENTIRE COLUMN
REQUIRES ENCODING

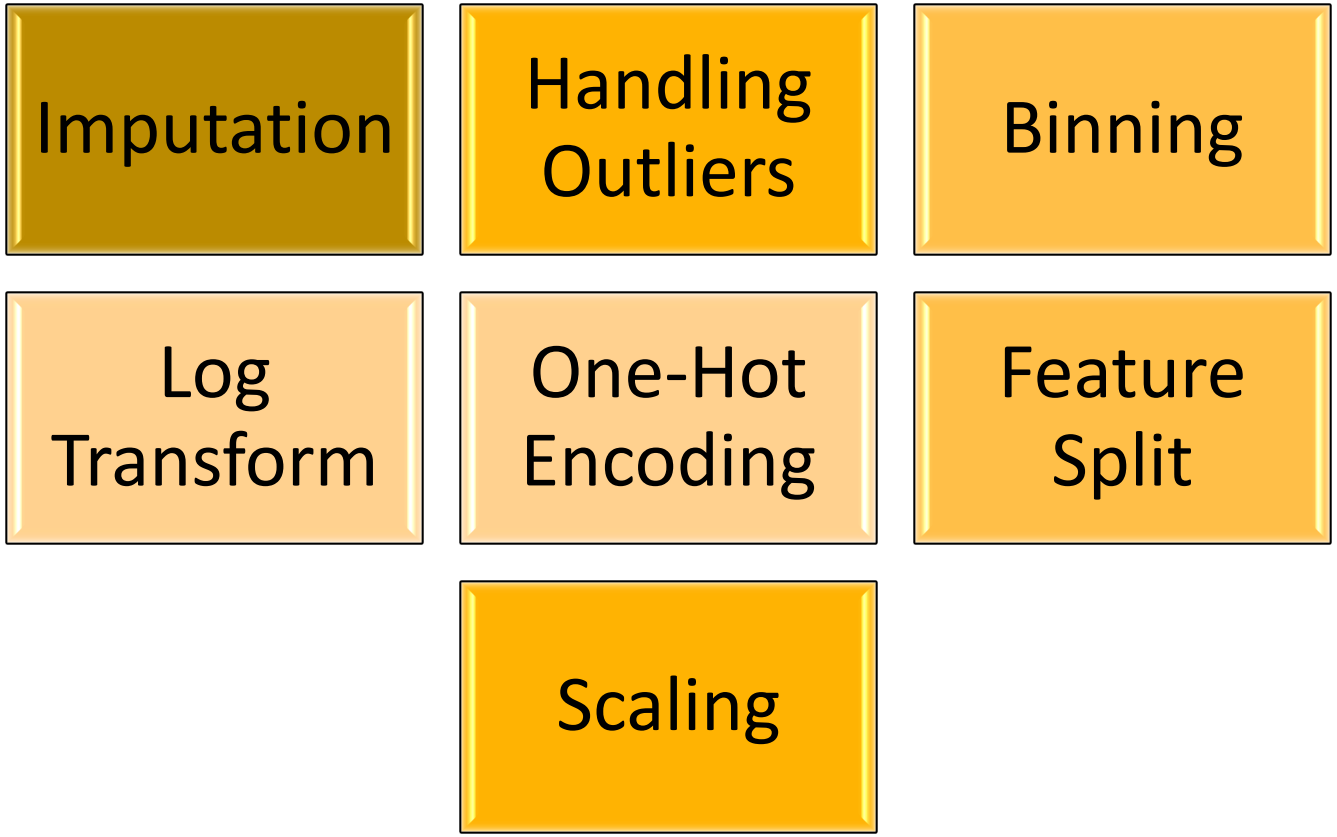
CUSTOMER ID	CUSTOMER NAME	LOCATION	CLICK ON AD?
1	Steve	USA	Yes
2	Mitch	Canada	1
3	Chanel	France	0
4	Bird		1
5	Cynthia	Netherlands	0
6	Chanel	France	0

MISSING INFORMATION

REQUIRES FORMATTING

DUPLICATE ENTRY

FEATURE ENGINEERING TECHNIQUES



FEATURE ENGINEERING: TOOLS

SAGEMAKER AND JUPYTER NOTEBOOKS (ADHOC)

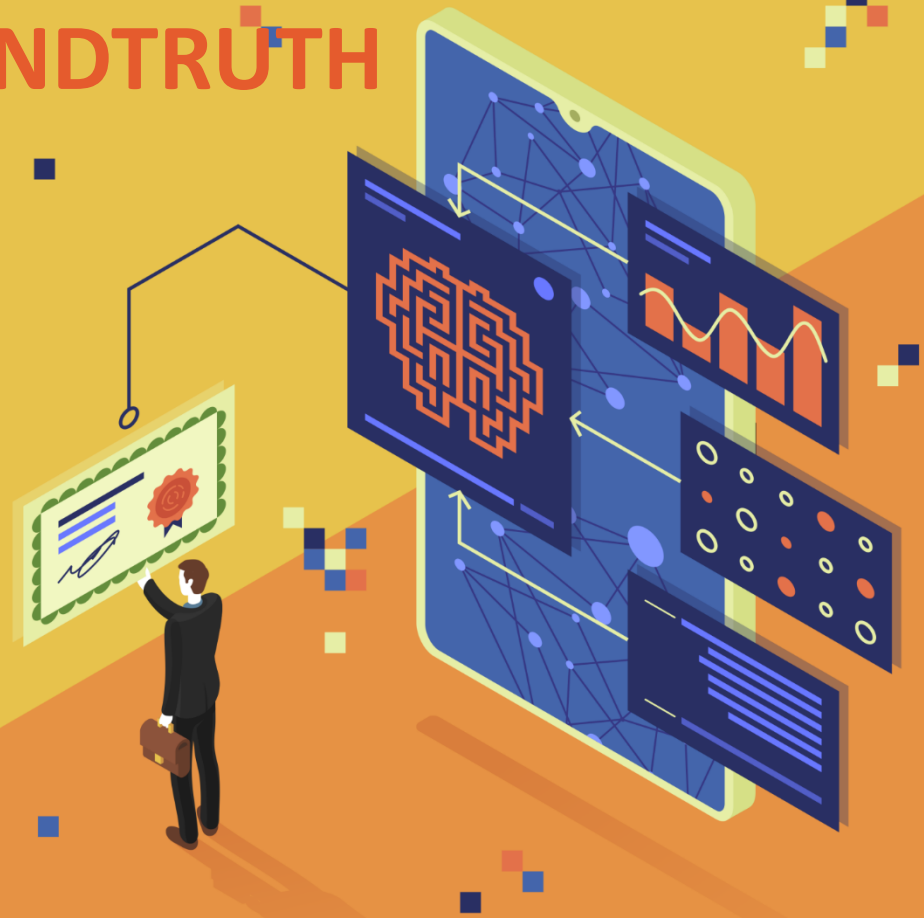
GLUE ETL JOB (REPEATABLE OR REUSABLE APPLICATIONS)

JUPYTER NOTEBOOKS

AMAZON SAGEMAKER

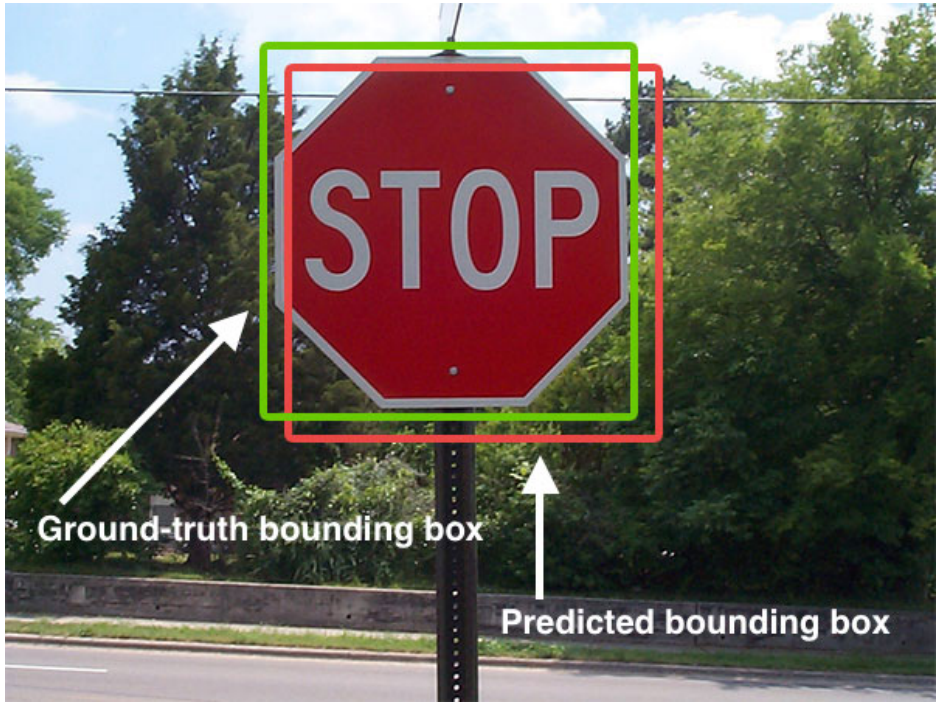
AWS GLUE

AMAZON SAGEMAKER GROUNDTRUTH



SAGEMAKER GROUND TRUTH

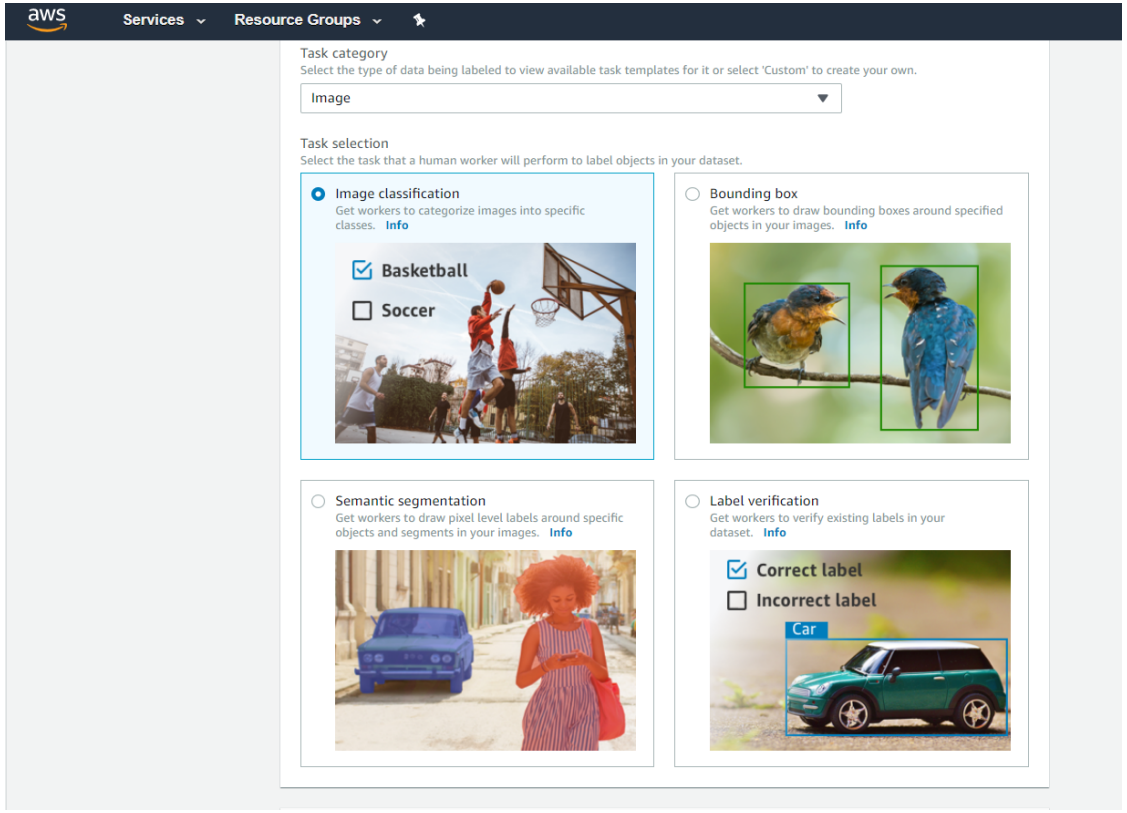
- In order to perform supervised training, labelled input/output data is required.
- Data labeling is a very expensive and time consuming job.
- Amazon SageMaker Ground Truth allows for creating datasets quickly and at low cost.
- Amazon Sagemaker provide developers and data scientists access to thousands of labelers around the globe.
- SageMaker ground truth can reduce the cost by 70% using automatic labeling.
- Automatic labeling works by training ground truth using the data that has already been labelled manually.



https://commons.wikimedia.org/wiki/File:Intersection_over_Union_-_object_detection_bounding_boxes.jpg

SAGEMAKER GROUND TRUTH

- Amazon SageMaker can save time and cost by using auto labeling service. The model learns from experience and continuously lean from human-labelled data.
- If model has high confidence in the label, it will automatically label raw data (produce final product).
- If the model has low confidence in the label, it will pass the image to human labellers for review.
- This dataset are then fed back to the model to improve and learn from experience (so it becomes better next time and learn from its mistakes which will lead to higher confidence next time.



SAGEMAKER GROUND TRUTH: BENEFITS

MASSIVE COST SAVINGS

- SageMaker Ground Truth can result in 70% savings by automating the labeling process.
- This is achieved by using machine learning to automatically label images.
- Human labelers are needed only if the confidence is not high.

ACCESS TO THOUSANDS OF LABELERS WORLDWIDE

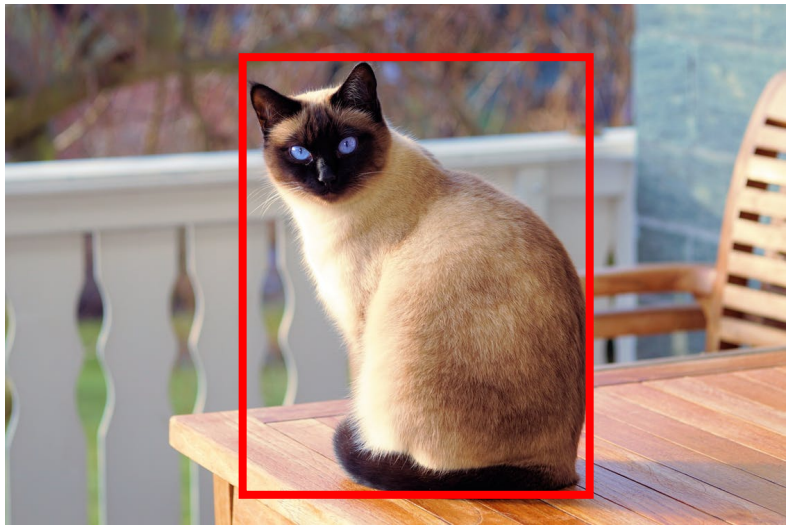
- SageMaker Ground Truth allows you to:
  - (1) Choose your own private labelers.
  - (2) Outsource to labelers outside of your company (500,000 labelers with Amazon Mechanical Turk).
  - (3) For sensitive information, Amazon provides professional labeling service with companies verified by Amazon.

FAST

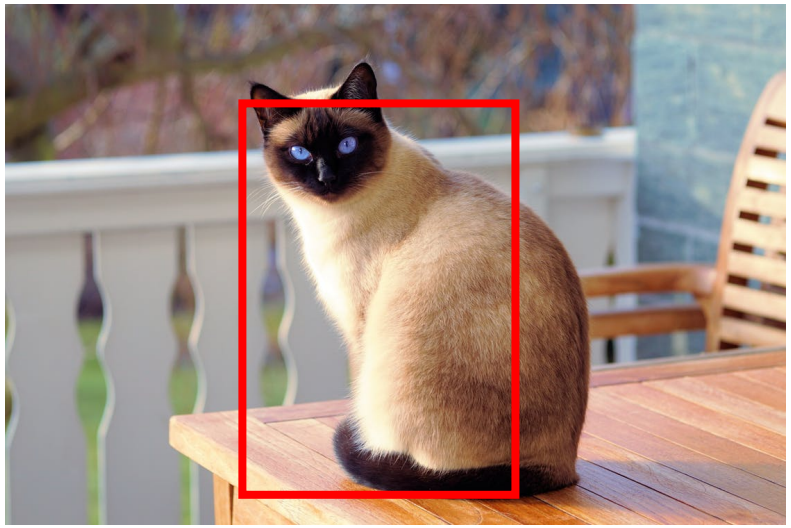
- SageMaker Ground Truth is fast and efficient.
- Labeling services provided manually are being automatically assessed to ensure quality.

SAGEMAKER GROUND TRUTH: LABELING GUIDELINES

- SageMaker Ground Truth allows for data labeling of many kinds such as: images, audio, and text.
- As a user, you can provide guidelines to the human labeler as shown below.



GOOD LABEL



BAD LABEL

https://www.pexels.com/photo/cat-outdoors-326875/

SAGEMAKER GROUND TRUTH: PRICING

- The pricing model is per object.
- It does not matter if the object is labeled automatically or manually.
- Additional cost is paid with Amazon Mechanical Turk.
- The cost differs based on the workflow: (1) classification, (2) bounding box, (3) semantic segmentation.

NUMBER OF LABELED OBJECTS	PRICE PER LABELED OBJECT
Less than 50,000 objects	$0.08
50,000 to 1,000,000 objects	$0.04
Greater than 1,000,000 objects	$0.02

WORKFLOW	SUGGESTED PRICE PER LABELER
Image classification	$0.012
Text classification	$0.012
Bounding box	$0.036
Semantic Segmentation	$0.84

FEATURE SELECTION

FEATURE SELECTION: GET RID OF USELESS DATA

- Feature selection is the process of selecting relevant features only from the available dataset.
- Feature selection enhances model performance by removing the “noise” and therefore enabling the model to focus on important features only.
- Useless data is a type of data that is discrete and has no relationship whatsoever with the output.
- We usually drop the useless features from the dataset before training the model
- Example include: random customer IDs at a store or a bank

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

USELESS INFORMATION

Photo Credit: https://www.maxpixels.net/Not-Useless-Town-Sign-Usefulness-Use-Useful-Road-822236

FEATURE SELECTION: PRINCIPAL COMPONENT ANALYSIS

- PCA is an unsupervised machine learning algorithm that performs dimensionality reductions while attempting at keeping the original information unchanged.
- PCA works by trying to find a new set of features called components.
- Components are composites of the uncorrelated given input features.
- The first component accounts for largest data variability followed by second component and so on.

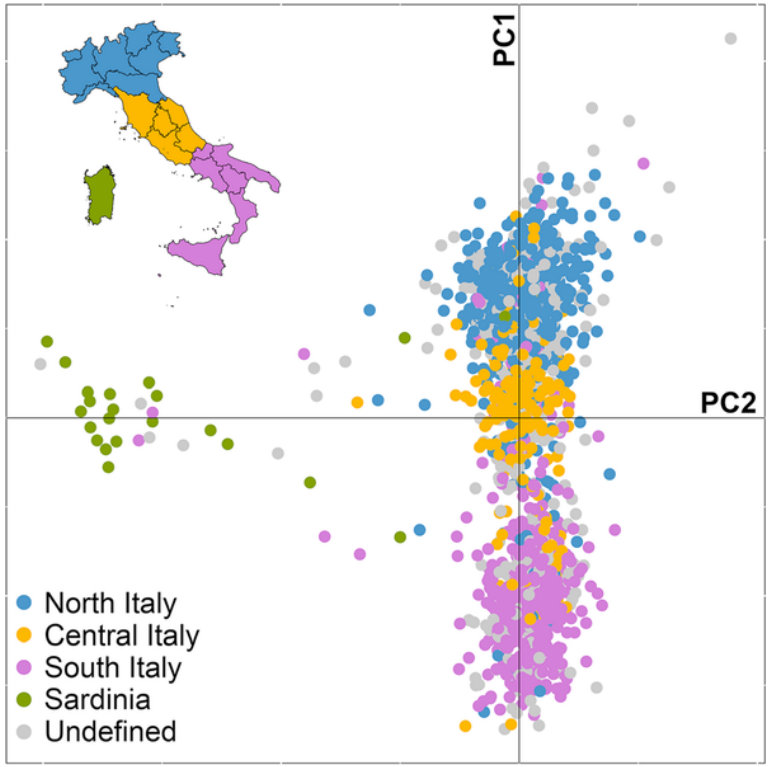
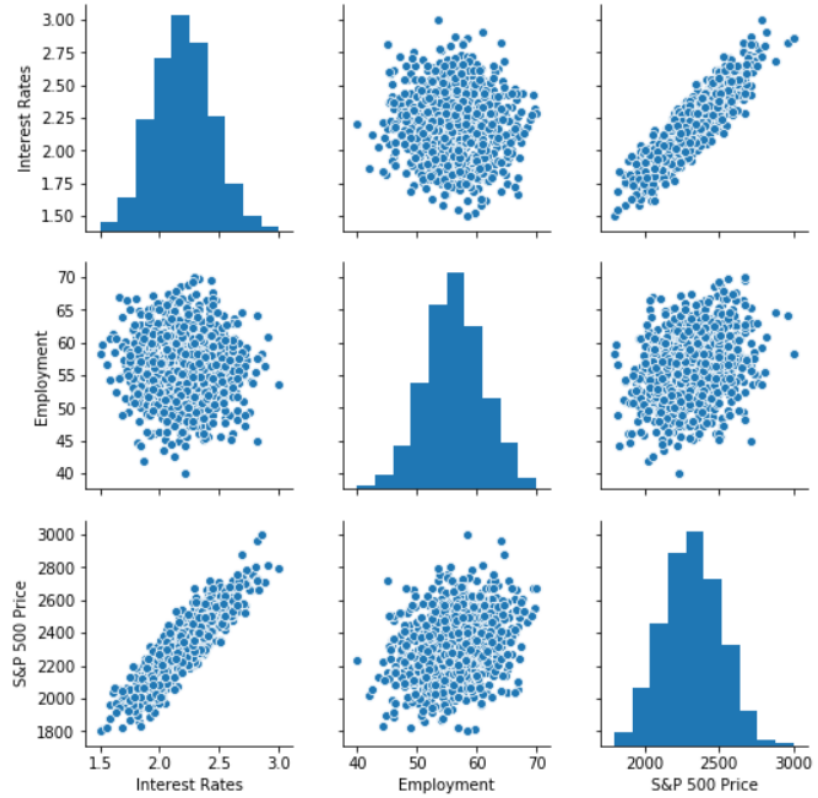


Photo Credit: https://commons.wikimedia.org/wiki/File:Principal_Component_Analysis_of_the_Italian_population.png

SCALING

SCALING

- Feature Scaling (normalization) is required prior to training of Artificial Neural Networks to ensure that features are within the same scale.
- Example: interest rate and employment rates are at a different scale. This will result in one feature dominating the other feature.
- Scikit Learn offers several tools to perform feature scaling.



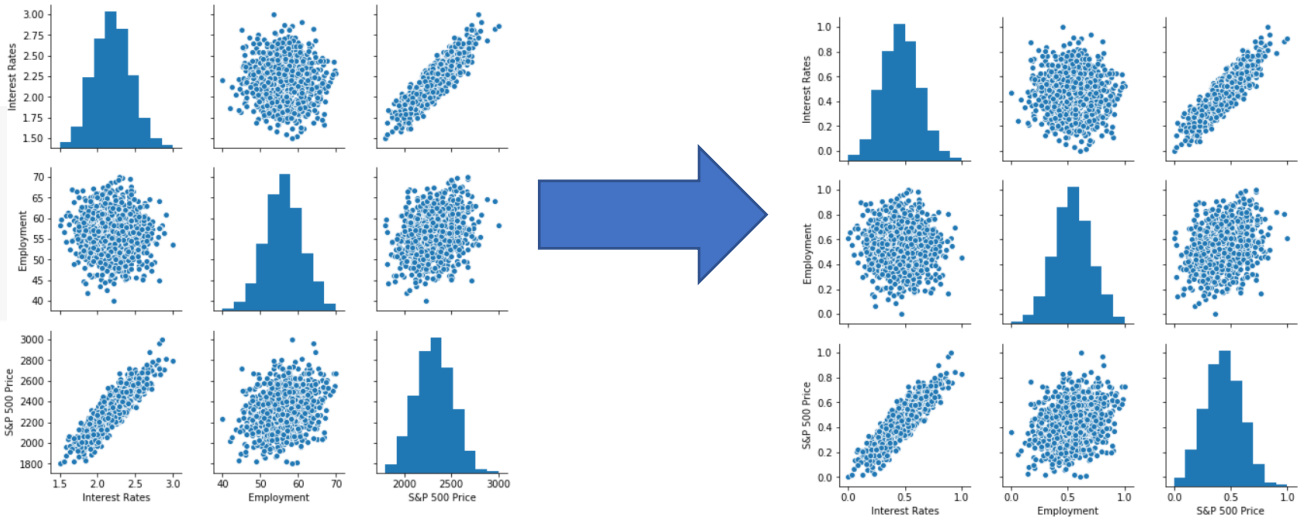
	Interest Rates	Employment	S&P 500 Price
0	1.943859	55.413571	2206.680582
1	2.258229	59.546305	2486.474488
2	2.215863	57.414687	2405.868337
3	1.977960	49.908353	2140.434475
4	2.437723	52.035492	2411.275663
5	2.143637	56.060598	2187.344909
6	2.148647	51.513208	2263.049249
7	2.176184	53.475909	2281.496374
8	2.125352	63.668422	2355.163011
9	2.225682	56.993396	2326.330337
10	1.814688	55.361780	2078.553895
11	2.281897	58.484752	2337.504507
12	2.426738	55.709328	2485.774097

SCALING: NORMALIZATION

- Normalization is conducted to make feature values range from 0 to 1.

$$x' = \frac{x - \min(x)}{\max(x) - min(x)}$$

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
y = scaler.fit_transform(y)
```



SCALING: STANDARDIZATION

- Standardization is conducted to transform the data to have a mean of zero and standard deviation of 1.
- Standardization is also known as Z-score normalization.
- Standardization is preferred over normalization when there are a lot of outliers.
- You can revert back to the original data range by applying inverse standardization.

$$z = \frac{x - \bar{x}}{\sigma}$$

- *Note: You can get rid of outliers all together by using Random Cut Forest Technique!*

SCALING: NORMALIZATION Vs. STANDARIZATION EXAMPLE

ORIGINAL DATASET

	Interest Rates	Employment	S&P 500 Price
0	1.943859	55.413571	2206.680582
1	2.258229	59.546305	2486.474488
2	2.215863	57.414687	2405.868337
3	1.977960	49.908353	2140.434475
4	2.437723	52.035492	2411.275663
5	2.143637	56.060598	2187.344909
6	2.148647	51.513208	2263.049249
7	2.176184	53.475909	2281.496374
8	2.125352	63.668422	2355.163011
9	2.225682	56.993396	2326.330337
10	1.814688	55.361780	2078.553895
11	2.281897	58.484752	2337.504507
12	2.426738	55.709328	2485.774097

QUICK STATS!

	Interest Rates	Employment	S&P 500 Price
count	1000.000000	1000.000000	1000.000000
mean	2.195392	56.254855	2319.999936
std	0.241630	4.862178	193.854745
min	1.500000	40.000000	1800.000000
25%	2.035735	53.029784	2190.447901
50%	2.198214	56.160941	2312.443024
75%	2.359061	59.422633	2455.764327
max	3.000000	70.000000	3000.000000

SCALING: NORMALIZATION

$$x' = \frac{x - \min(x)}{\max(x) - min(x)} = \frac{2206 - 1800}{3000 - 1800} = 0.338$$

S&P 500 PRICE (ORIGINAL)
2206
2319
3000
1800

NORMALIZATION/SCALING

- S&P 500 PRICE (Max)=3000
- S&P 500 PRICE (Min)=1800

S&P 500 PRICE (NORMALIZED)
0.338
0.432
1
0

SCALING: STANDARIZATION

$$z = \frac{x - \bar{x}}{\sigma} = \frac{2206 - 2319}{193.8} = -0.583$$

S&P 500 PRICE (ORIGINAL)
2206
2319
3000
1800

STANDARIZATION

S&P 500 PRICE (STANDARIZED)
-0.583
0
3.513
-2.67

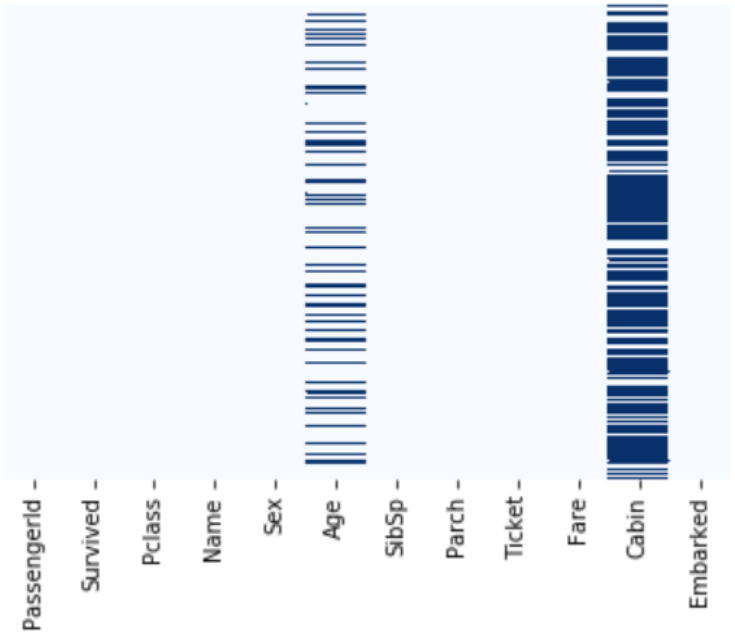
ORIGINAL DATA = MEAN

- S&P 500 PRICE (Mean)=2319
- S&P 500 PRICE (Std)=193.8

IMPUTATION

1. IMPUTATION: BEYOND REPAIR OR IS THERE HOPE?!

- As a data scientist, you will encounter data with missing values “NaN values”.
- Missing data might arise from a sensor or a server that might have broken down
- In order to feed this data a machine learning model, you will need to:
  - Replace the missing NaN values with something which could be mean, median or maximum value!
  - Remove the entire row (not recommended since row might contain valuable information).
- You can obtain the mean by replacing NaN values with feature mean for example.
- Let’s take a look at the Titanic Dataset.

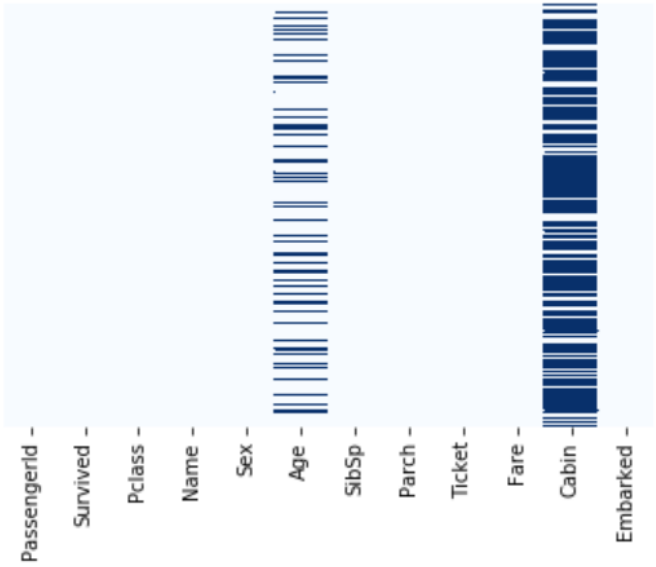


	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

1. IMPUTATION: DROPPING

- You might need to remove the “Cabin” column.
- It seems that there are a ton missing data beyond repair.

```
1  # Let's drop the cabin coloumn
2  training_set.drop('Cabin',axis=1,inplace=True)
```



1. IMPUTATION: REPLACING MISSING VALUES WITH MEAN

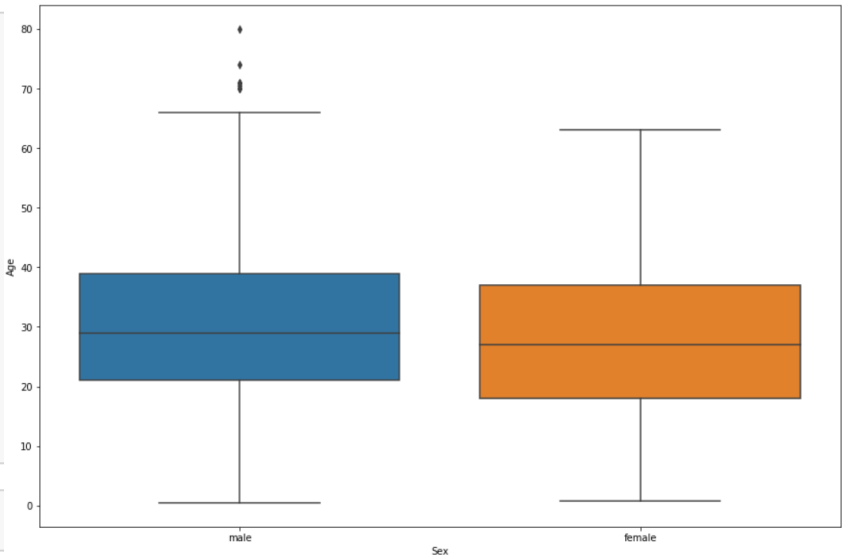
- For the “Age” Column, you might need to obtain the mean and replace missing values with the mean of the “Age” column.
- Another way of improving this, is to replace missing values with the average “age” depending on the “sex” of the passenger.

In [22]:
```
def Fill_Age(data):
    age = data[0]
    sex = data[1]

    if pd.isnull(age):
        if sex is 'male':
            return 29
        else:
            return 25
    else:
        return age

```

In [23]:
```
training_set['Age'] = training_set[['Age','Sex']].apply(Fill_Age,axis=1)
```



1. IMPUTATION: REPLACING MISSING VALUES WITH MEAN

- Replacing missing values with the mean is simple and effective but lack accuracy.
- In the presence of outliers, using the median might lead to better results.
- What about categorical data?
  - You can replace missing value with the most frequently occurred value.
  - If values follows uniform distribution, use “other” instead.

1, 3, 3, **6**, 7, 8, 9

Median = 6

1, 2, 3, **4**, **5**, 6, 8, 9

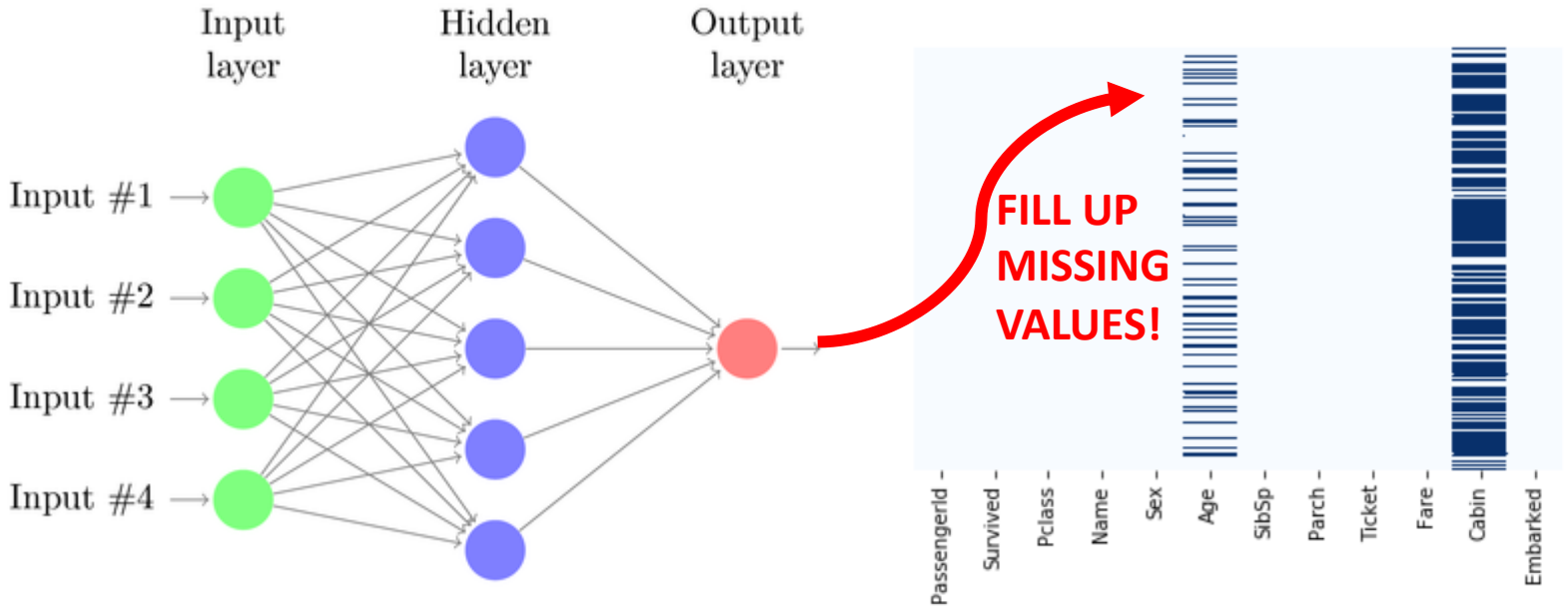
Median = (4 + 5) ÷ 2

= 4.5

https://commons.wikimedia.org/wiki/File:Finding_the_median.png

1. IMPUTATION: REPLACING MISSING VALUES WITH MACHINE LEARNING!

- Another way of handling missing values is by using machine leaning techniques.
- These techniques will lead to most accuracy but they are much more complex than simple “mean replacement”.
- They could work well with numerical and categorical data.
- Deep Learning, KNN or Regression techniques could be used.
- Remember that you can always collect more data.



http://www.texample.net/tikz/examples/neural-network/

1. IMPUTATION: HANDLING MISSING VALUES IN AWS DEEPAR ALGORITHM

- AWS DEEPAR which is a time series forecasting algorithm provided by SageMaker.
- In timeseries, having missing data might have a huge impact on the forecasting accuracy.
- Imputing missing values with zeros, will bias the algorithm towards zero which is not desirable.
- Now DeepAR algorithm can work with missing data (directly in the model) so you do not have to perform any manual feature extraction.
- DeepAR relies on Recurrent Neural Network (deep learning) to perform the imputation resulting in a much accurate results.

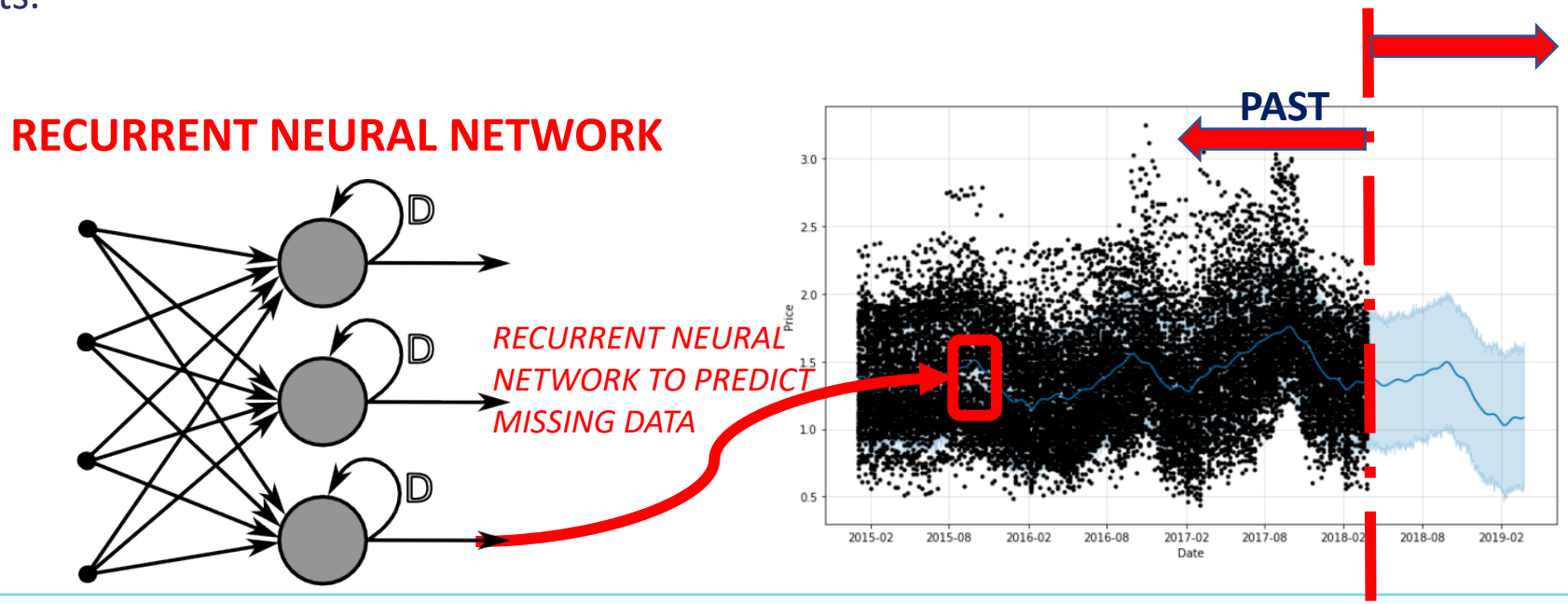
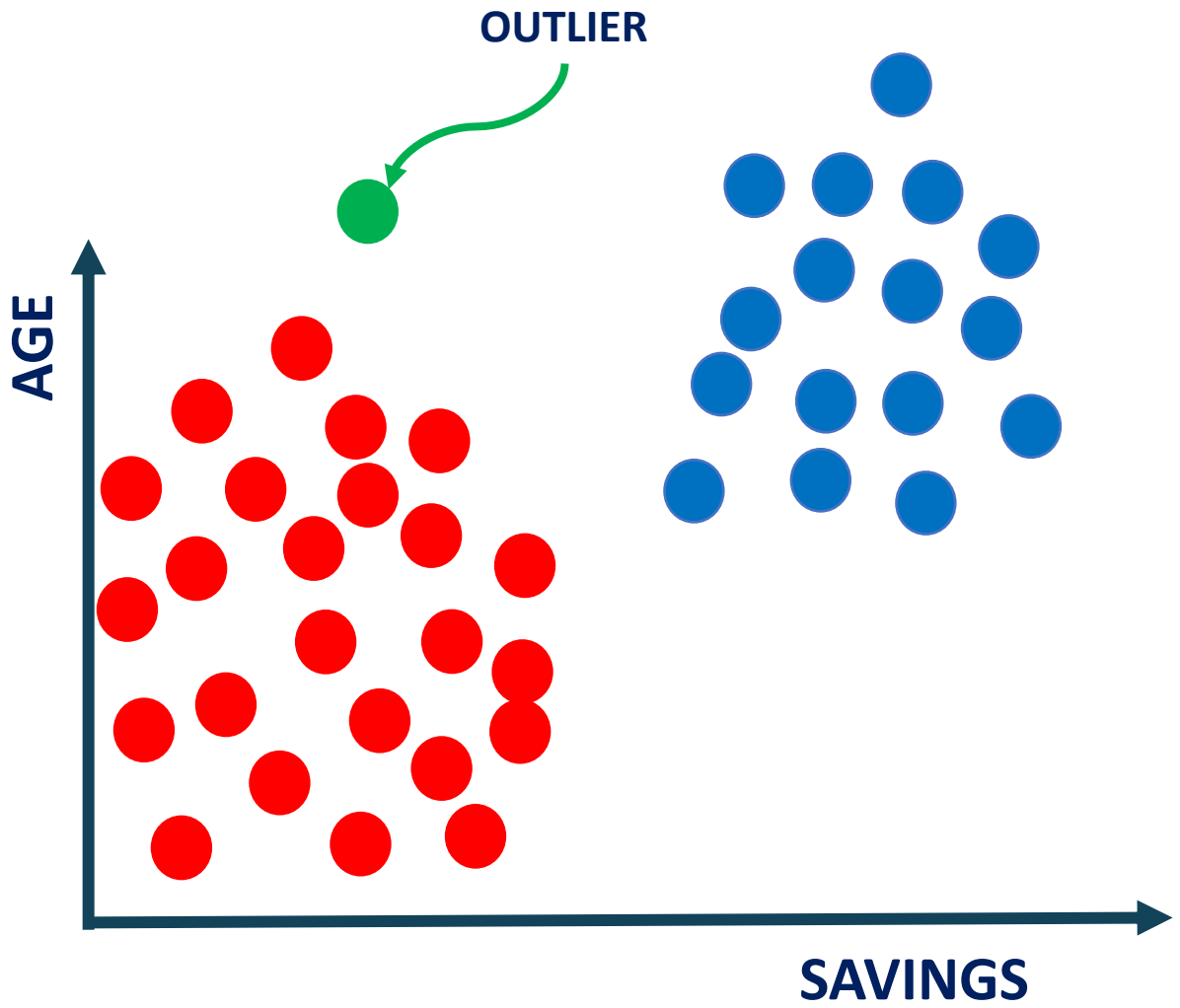


Photo Credit: https://commons.wikimedia.org/wiki/File:RecurrentLayerNeuralNetwork_english.png

OUTLIERS

HOW TO HANDLE DATA OUTLIERS?

- The best way to handle outliers is by performing data visualization.
- The following 2 statistical methods could be used to detect outliers:
  - Standard deviation
  - Percentiles



VARIANCE

- The Variance is defined as the average of the squared differences from the Mean.

- Variance calculation steps:

1. Calculate the mean (average) of all the numbers
2. For each data point, subtract the Mean (calculated in step #1)
3. Square the result to obtain the squared difference.
4. Calculate the average of those squared differences.

$$Variance = \sigma^2 = \frac{1}{N}(\sum_{i=1}^{N}(x_i - \mu)^2$$

STANDARD DEVIATION

- Standard deviation is a measure of the dispersion from the mean
- Represented by the symbol is σ (sigma)
- Standard deviation is the square root of the variance

$$Standard\ Deviation\ (\sigma) = \sqrt{variance}$$

- Standard deviation is used to detect outliers.
- Data points that are greater than 2σ or 3σ or 4σ are considered outliers (tunable parameter).
- AWS Random Cut Forest algorithm is used for outlier detection.
- AWS Random Cut Forest is included in many services such as Amazon Kinesis analytics and QuickSight.

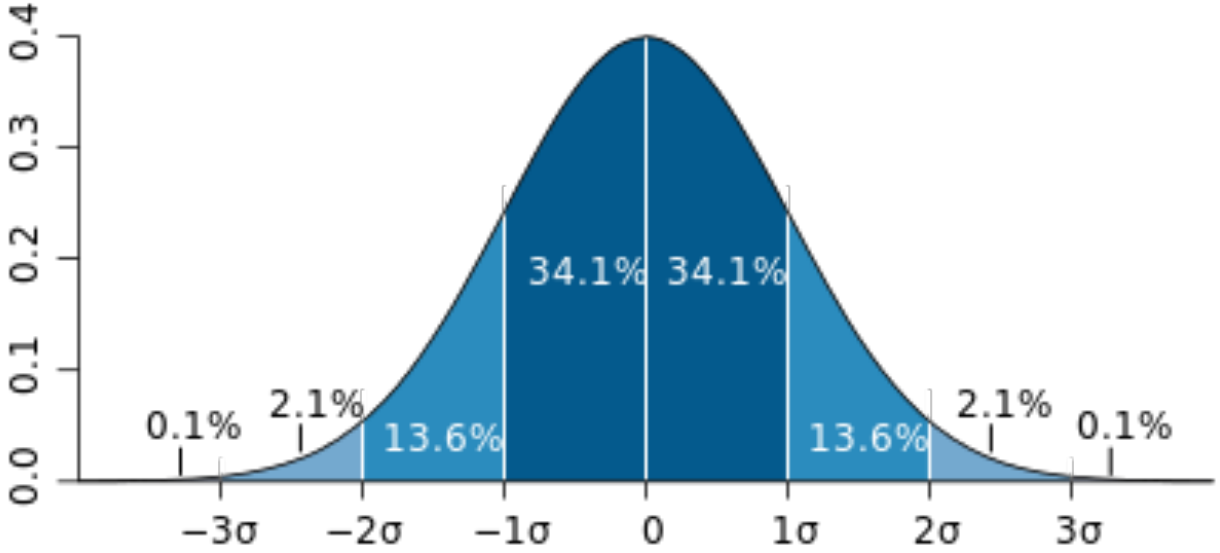


Photo Credit: https://commons.wikimedia.org/wiki/File:Standard_deviation_diagram.svg

PERCENTILES

- Another way of detecting and removing outliers is by using percentiles.
- For example, you can assume that values that are more than 90% percentile will be removed from the data.

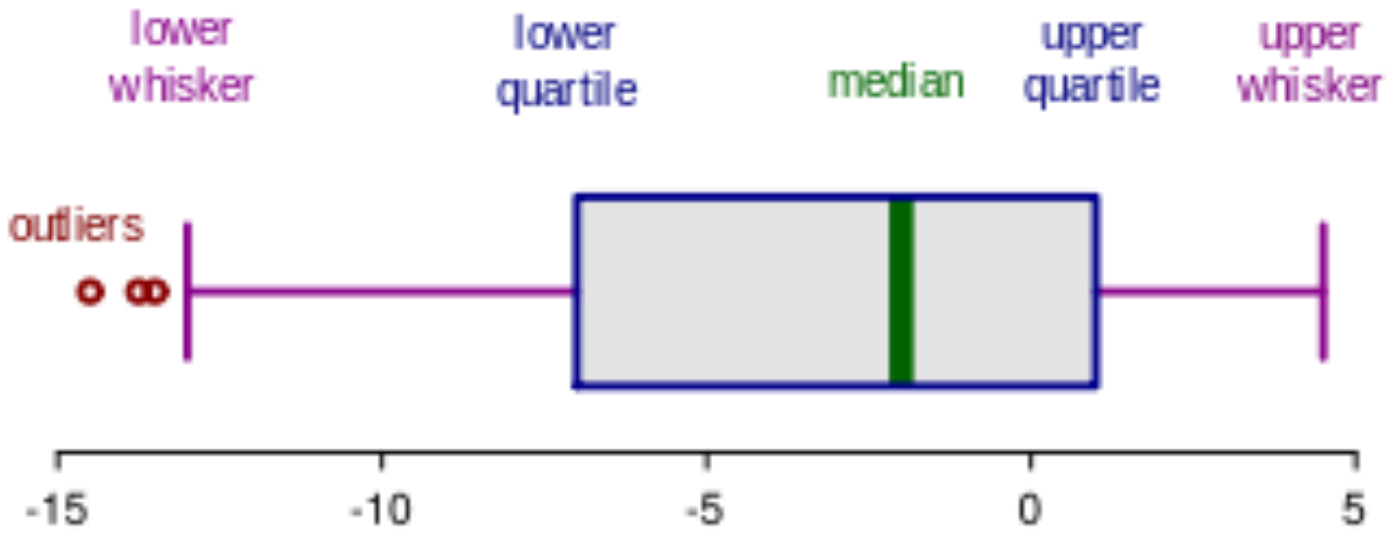


Photo Credit: https://commons.wikimedia.org/wiki/File:Elements_of_a_boxplot_en.svg

HOW TO DEAL WITH OUTLIERS?

- Is Elon musk an an outlier?!

Photo Credit: https://commons.wikimedia.org/wiki/File:Elon_Musk_Royal_Society.jpg

ONE HOT ENCODING

ONE-HOT ENCODING: WHY DO WE NEED IT?

- Can we simply replace colors with integer values?
- The machine learning model will assume that:

$$GREEN > YELLOW > RED$$

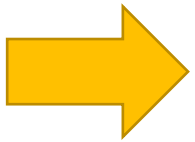
COLOR	ENCODED COLOR
RED	1
RED	1
YELLOW	2
GREEN	3
YELLOW	2

WRONG!

ONE-HOT ENCODING

- One hot encoding is widely used in machine learning.
- It works by converting values such as “color” into columns with 1’s and 0’s in them.
- Since machine learning models deal with numbers, we perform one hot encoding to convert from categorical data into numerical.
- If you have N categories, you will need N-1 binary columns to represent them.

COLOR
RED
RED
YELLOW
GREEN
YELLOW



RED	YELLOW	GREEN
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

ONE-HOT ENCODING: ORDINAL Vs. NOMINAL

- Recall the difference between nominal and ordinal data.
  - **In ordinal data, order is important**
  - **In Nominal data, order is not important.**

NOMINAL

Order of colors doesn't mean anything!

COLOR
RED
RED
YELLOW
GREEN
YELLOW

ORDINAL

Order is important!

- *1 star means poor quality course*
- *5 star means great quality course*

Photo Credit: https://pixabay.com/vectors/rating-stars-system-evaluation-153125/

BINNING

BINNING

- Binning could be used for nonlinear transformation from numeric values to categorical.
- Binning is used when the relationship between numeric feature (input) and output is not linear. (*no monotonic increase or decrease with the target*). Then by doing binning, each categorical feature (bin) might have a linear relationship with the target.
  - Let's assume that you have age as an input and the likelihood to purchase a cars an target (output).
  - Looking at the data, there is no linear relationship between the two.
  - However, by doing binning, you can now convert the numerical values into bins and this will lead to more accurate model.

VALUE	BIN
0-20	Young
20-70	Mid-age
70-100	Old

BINNING

- One of the key advantages of binning is to improve the model robustness and avoid overfitting.
- Binning is also important when data contains a lot of uncertainty.
- Quantile binning is used to bin the data based on its location in the distribution.
- Binning could be used for both numerical and categorical transformations.
- Let’s take a look at a categorical binning example.

VALUE	BIN
Toronto	Ontario
Hamilton	Ontario
Vancouver	British Columbia
Burnaby	British Columbia

QUANTILE BINNING

- **Quantile Binning:** works by assigning same number of observations to each of the bins so each bin will end up having the same number of observations,

Photo Credit: https://pixabay.com/vectors/recycling-container-bin-boxes-41078/
Photo Credit: https://en.m.wikipedia.org/wiki/File:Happy_face.svg
Photo Credit: https://pixabay.com/illustrations/sad-cry-tear-emotion-mood-face-1533965/
Photo Credit: https://publicdomainvectors.org/en/free-clipart/Laughing-crying-face/82663.html

BINNING AND ENCODING

BINNING CRITIREA

VALUE	BIN
0-20	Young
20-60	Mid-age
60-100	Old

AGE
10
50
61
5

BINNING

BIN
Young
Mid-Age
Old
Young

One-Hot Encoding

YOUNG	MID-AGE	OLD
1	0	0
0	1	0
0	0	1
1	0	0

LOG TRANSFORM

LOG TRANSFORM

- Log transform is widely used transformation in feature engineering.
- Log transform must be applied to positive values only to avoid getting error.
- Log transform enables data scientists to deal with skewed dataset.
- Log transform is used when the variables span many orders of magnitude such as “income”.
- Since the majority of incomes are very small and very few incomes are large.

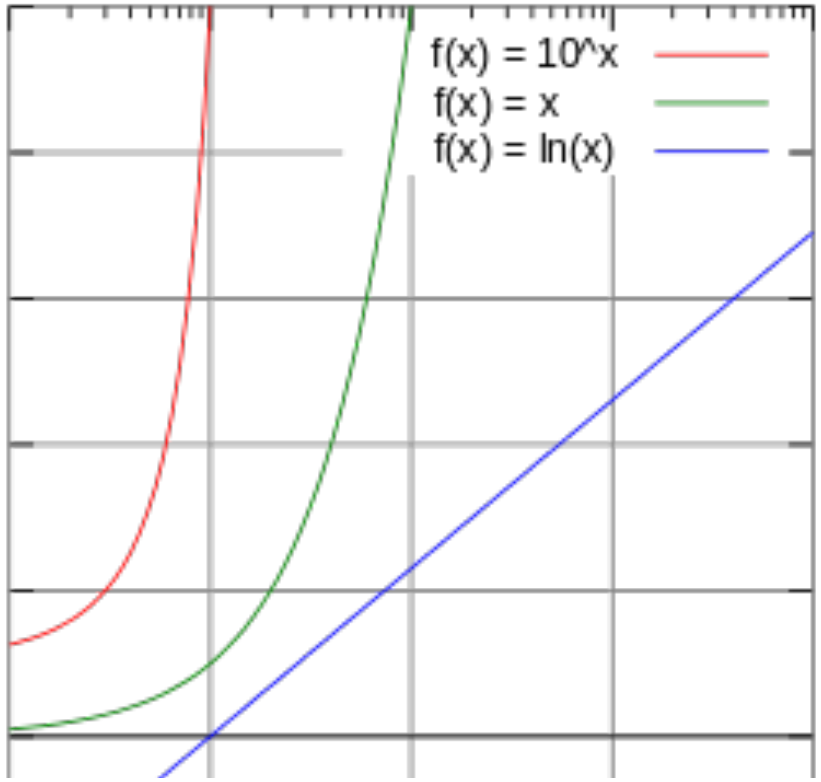


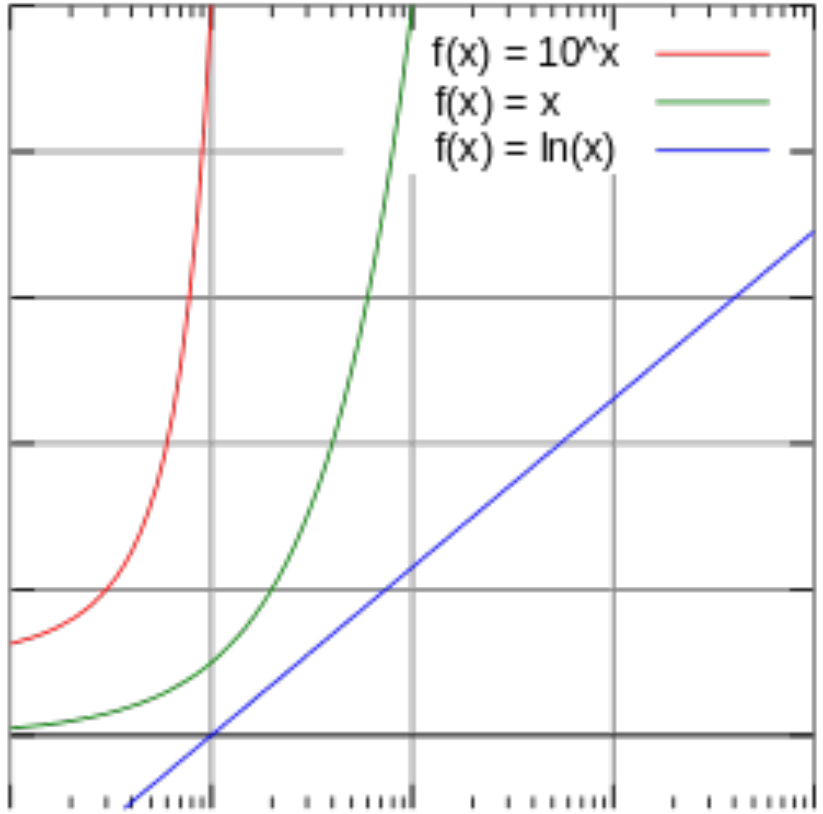
Photo Credit: https://commons.wikimedia.org/wiki/File:Logarithmic_Scales.svg

LOG TRANSFORM

- So this type of data is best deal with in logarithmic scale
- Log transform reduces the negative effect of outliers since it normalizes the magnitude difference.
- Example:

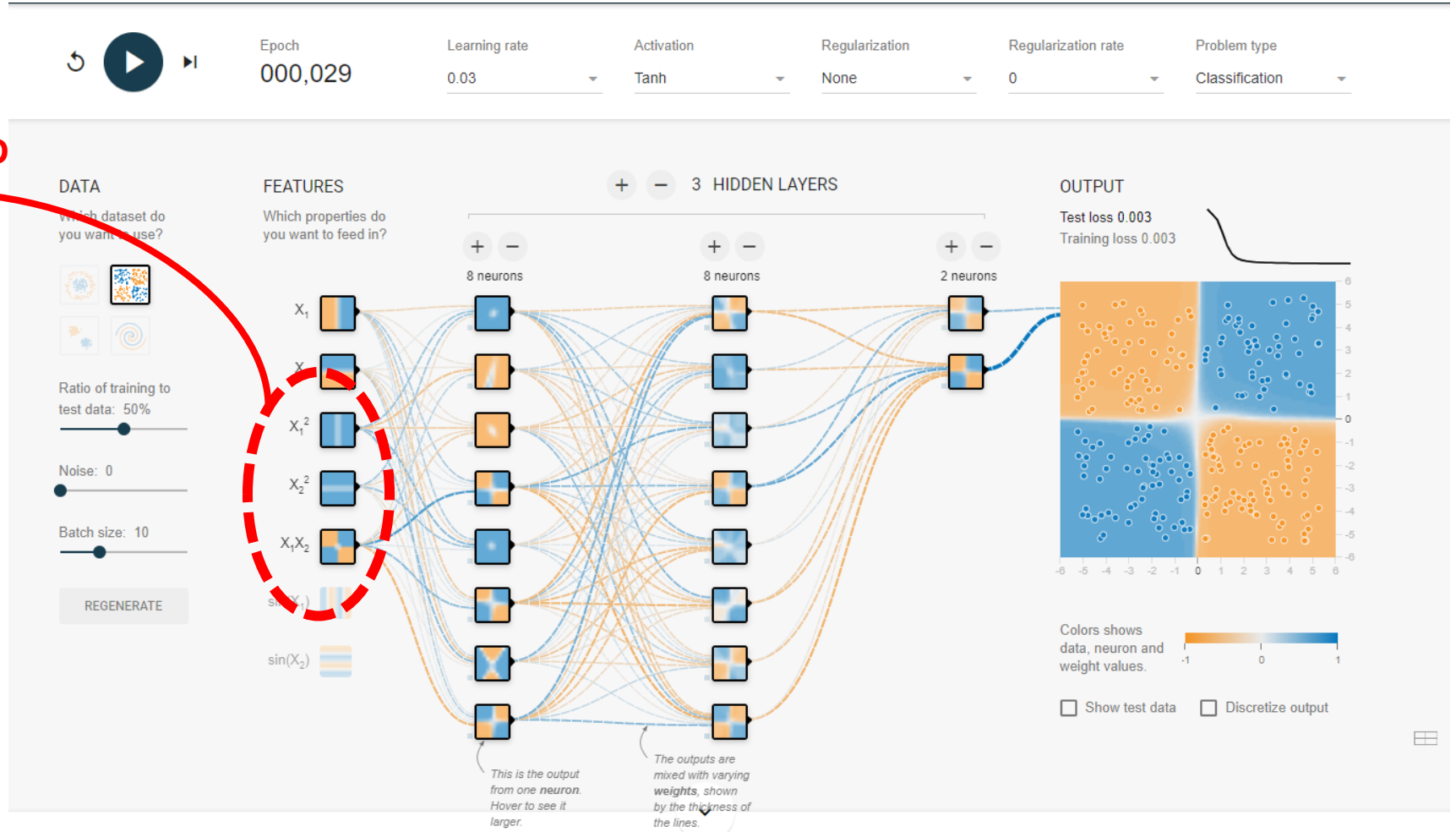
$$log(x^n) = n * log(x)$$
$$log(10^4) = 4 * log(10) = 4$$
$$log(10^3) = 3 * log(10) = 3$$

- Difference between $\boldsymbol{log(10^4) - log(10^3)}$ is much bigger compared to **(4-3)**



ANOTHER TRANSFORMATION EXAMPLE

TRANSFORMED FEATURES



https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.17464&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

DATA SHUFFLING

SHUFFLING

- Shuffling is an important step prior to training of machine learning models.
- This is important to avoid any bias or pattern related to the order of the data.
- Especially since data is divided to training, testing and validation.
- Shuffling ensures:
  - Enhanced ML model quality/performance
  - Reduce tendency to overfit the training data

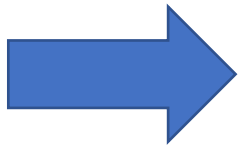
Photo Credit: https://pixabay.com/vectors/shuffle-icon-player-button-outline-2297766/

FEATURE SPLITTING

FEATURE SPLITTING

- Splitting features is used to split one feature into two.
- This might improve the performance of the machine learning model by extracting more information.

MOVIE
Titanic (1997)
Notebook (2004)
A Merry Christmas Match (2019)



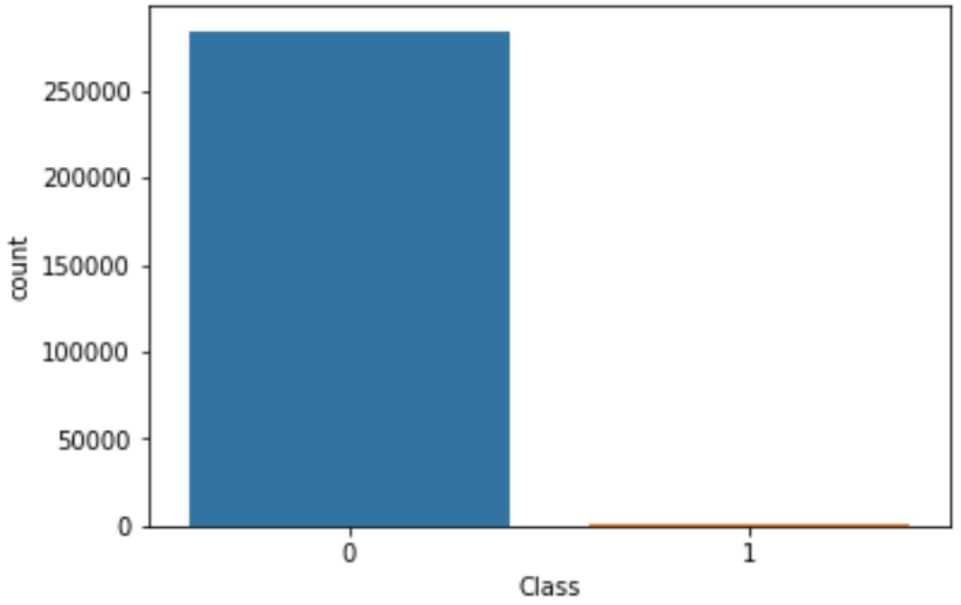
FEATURE #1
Titanic
Notebook
A Merry Christmas Match

FEATURE #2
1997
2004
2019

UNBALANCED DATASET

HOW TO DEAL WITH UNBALANCED DATASET?

- Let's take a look at an example data with unbalanced positive and negative classes.
  - Credit card companies need to have the ability to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.
  - Datasets contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions.
  - The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.
  - Link to the dataset: https://www.kaggle.com/mlg-ulb/creditcardfraud/home



UNBALANCED DATASET: SOLUTIONS?

- You can overcome the unbalanced dataset by doing one of the following:

1. UNDERSAMPLING

- Under sampling is the process of selecting some samples only from the majority class.
- Doing so will remove some of the “unbalance” seen in the data.
- Example: only select some samples that are “non-fraudulent”.

2. OVERSAMPLING

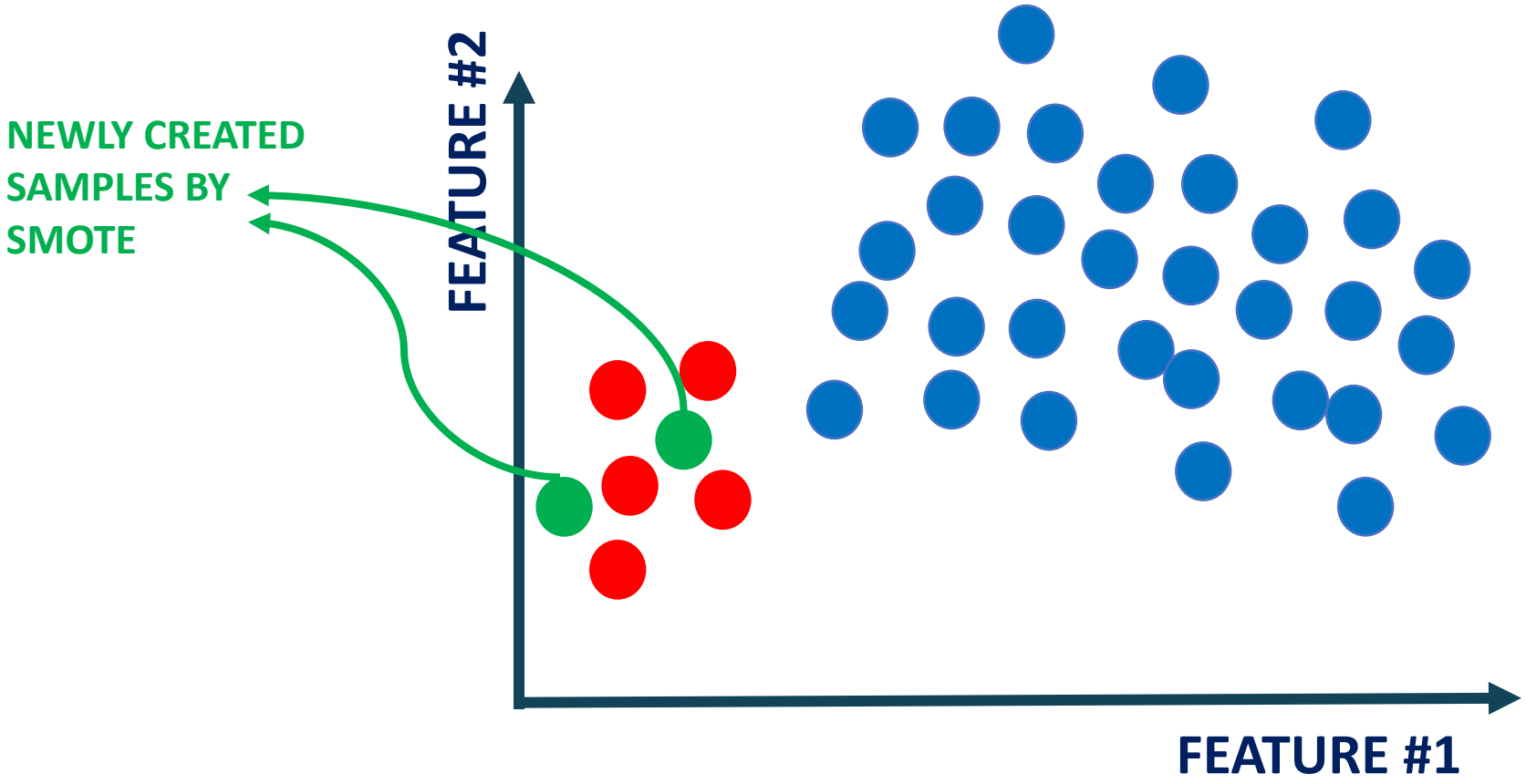
- Oversampling is the process of replicating data from the less representative (minority) class.
- Doing so will help alleviate the “unbalance” seen in the data.
- Example: duplicate samples that are “fraudulent”.

3. GENERATE SYNTHEIC DATASET

- Generate more artificial dataset from the minority class
- Example: Synthetic Minority Oversampling (SMOTE) technique

SYNTHETIC MINORITY OVERSAMPLING (SMOTE) TECHNIQUE

- SMOTE technique works by applying K nearest neighbour to add new dataset.
- For each point from the minority class, SMOTE will calculate k nearest neighbors.
- Then new samples are created based on the oversampling percentage (tunable parameter).



TEXT FEATURE ENGINEERING OVERVIEW

TEXT FEATURE ENGINEERING: OVERVIEW

BAG OF WORDS

- Bag of words is a text feature engineering technique use to tokenize text. Each key includes the word, and each value represents the number of occurrences of that word.

TERM FREQUENCY INVERSE DOCUMENT FREQUENCY (TF-IDF)

- TF-IDF is used to find important keywords in corpus of documents and to ignore less important (more frequent) words.

N-GRAM

- N-Gram is used to create an index of how often words follow one another. It is important for comparing text such as in spam emails detection.

PUNCTUATION REMOVAL

- Getting rid of punctuation from text.

DATES RETRIEVAL

- Extracting date-related information from text.

ORTHOGONAL SPARSE BIGRAMS (OSB)

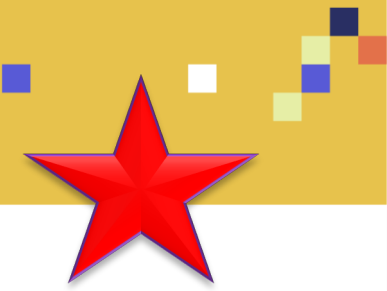
- OSB is used for text transformation by encoding the words in a text along with how many words have been skipped as well (distance between words).

CARTESIAN PRODUCT

- Cartesian transformation works by creating permutations between variables.

TEXT FEATURE ENGINEERING: BAG OF WORDS

BAG OF WORDS



- Bag of words is a text feature engineering technique use to tokenize text.
- Each key includes the word, and each value represents the number of occurrences of that word.

I am happy today. I am learning SageMaker today.

Bag of words = {“I”, “am”, “happy”, “today”, “I”, “am”, “learning”, “SageMaker”, “today”}

WORD	COUNT
I	2
am	2
happy	1
Today	2
Learning	1
Sagemaker	1

TEXT FEATURE ENGINEERING: REMOVE PUNCTUATION

REMOVE PUNCTUATION

- Amazon machine learning uses whitespace as a separator between words (tokens).
- Applying punctuation removal step might be useful in some cases such as bag of words or N-gram transformation.
- Note that lower case transformation could be applied as well.

"Welcome to AML - please fasten your seat-belts!"

{"Welcome", "to", "Amazon", "ML", "-", "please", "fasten", "your", "seat-belts!"}

WITH PUNCTUATIONS

{"Welcome", "to", "Amazon", "ML", "please", "fasten", "your", "seat-belts"}

WITHOUT PUNCTUATIONS

TEXT FEATURE ENGINEERING:
DATE FEATURE ENGINEERING

DATE FEATURE ENGINEERING

- Date feature engineering is used to extract date-related information as follows:

DATE
12/1/2019
1/1/2020

DATE FEATURE EXTRACTION

MONTH	YEAR	DAY
12	2019	01
01	2020	01

THESE FEATURES COULD THEN BE FED INTO A MACHINE LEARNING ALGORITHM

TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF)

TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF)

- TF–IDF stands for "Term Frequency–Inverse Document Frequency" is a numerical statistic used to reflect how important a word is to a document in a collection or corpus of documents.
- TF-IDF is used as a weighting factor during text search processes and text mining.
- The intuition behind the TF-IDF is as follows:
  - if a word appears several times in a given document, this word might be meaningful (more important) than other words that appeared fewer times in the same document.
  - However, if a given word appeared several times in a given document but also appeared many times in other documents, there is a probability that this word might be common frequent word such as 'I' 'am'..etc. (not really important or meaningful!).

https://pixnio.com/objects/books/magnifying-glass-dictionary-document-paper-book-text

TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF)

- TF: Term Frequency is used to measure the frequency of term occurrence in a document:

$$TF(word) = \frac{Number\ of\ times\ the\ 'word'\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

- IDF: Inverse Document Frequency is used to measure how important a term is:

$$IDF(word) = log(\frac{Total\ Number\ of\ Documents}{Number\ of\ documents\ with\ the\ term\ 'word'\ in\ it})$$

- TF-IDF is then calculated as follows:

$$TF - IDF = TF * IDF$$

- TF-IDF generated a matrix of the following shape:

(Number of documents, Number of unique N-Grams)

TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF): EXAMPLE

- Let's assume we have a document that contains 1000 words and the term “John” appeared 20 times.
- The Term-Frequency for the word 'John' can be calculated as follows:

$$TF|john = 20/1000 = 0.02$$

- Let's calculate the IDF (inverse document frequency) of the word 'john' assuming that it appears 50,000 times in a 1,000,000 million documents (corpus).

$$IDF|john = \log(1{,}000{,}000/50{,}000) = 1.3$$

- Therefore the overall weight of the word 'john' is as follows”

$$TF - IDF|john = 0.02 * 1.3 = 0.026$$

TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF): USE CASE

- TF-IDF could be used to develop a search algorithm as follows:
  - Step #1: calculate the TF-IDF for every word present in the corpus
  - Step #2: Once a user enter a word, the corpus is sorted by the TF-IDF for the given word inserted by the user.
  - Step #3: Show the output to user.

In [143]:
```
from sklearn.feature_extraction.text import TfidfTransformer

emails_tfidf = TfidfTransformer().fit_transform(spamham_countvectorizer)
print(emails_tfidf.shape)
```

```
(5728, 37229)
```

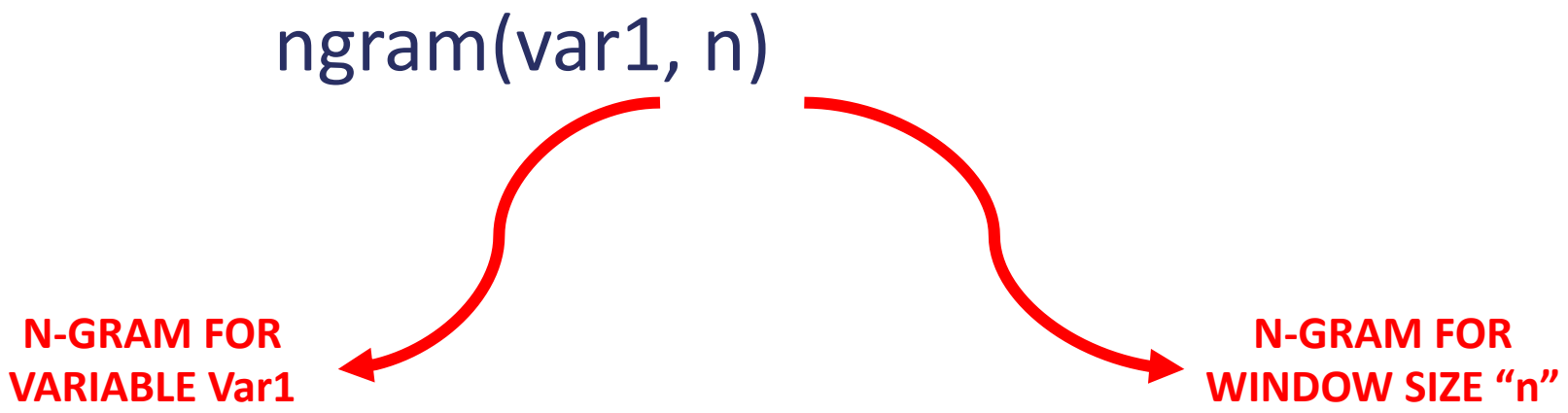
In [144]:
```
print(emails_tfidf[:,:])
# Sparse matrix with all the values of IF-IDF
```

```
  (0, 3638)     0.017223322243491098
  (0, 23369)    0.118508643434226
  (0, 18841)    0.13854196464928686
  (0, 10065)    0.07179540742040964
  (0, 17696)    0.08994844691767893
```

N-GRAM (UNIGRAM Vs. BIGRAM Vs. TRIGRAM) TRANSFORMATION

N-GRAM: INTRODUCTION

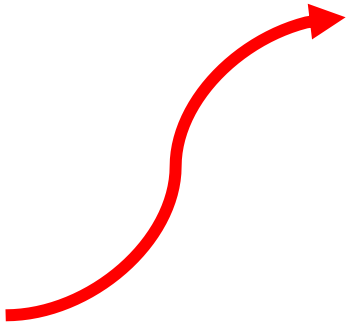
- The n-gram transformation converts text data to strings corresponding to sliding a window of n words.
- This is important to perform machine learning; you can now train a model using a single word or group of words.
- You can select the size of “n” as follows:
  - **Unigram => n=1**
  - **Bigram => n=2**
  - **Trigram => n=3**



N-GRAM: UNIGRAM

- Let’s start with Unigram first: ngram(var1, 1)

NOTE THAT N-GRAM BREAKS THEH RAW INPUT DATA BASED ON WHITESPACE CHARACTERS



“***Let’s have a happy life***"

Unigram: {“Let’s", "have", "a" , “happy" , “life“}

N-GRAM: WHAT ABOUT BI-GRAM AND TRI-GRAM?

- Let’s assume that now we want to train a machine learning model to understand this but with Bigrams and Trigrams as well:

“Let’s have a happy life"

NOTE THAT BIGRAMS AND TRIGRAMS INCLUDE UNIGRAMS AS WELL!

- We can divide this sentence into unigrams, bigrams and Trigrams
  - **Unigram: {**“Let’s", "have", "a" , “happy" , “life“}
  - **Bigram: {**“Let’s have" , "have a" , "a happy" , “happy life“, “Let’s", "have", "a" , “happy" , “life“}
  - **Trigram: {**“Let’s have a" , "have a happy" , "a happy life“, “Let’s have" , "have a" , "a happy" , “happy life“, “Let’s", "have", "a" , “happy" , “life“}

UNIGRAM Vs. BIGRAM Vs. TRIGRAM

- Note that n-grams breaks the raw text based on whitespace.
- Any punctuations will be considered a part of the word. (unless you want to remove the punctuation first)

"red, green, blue"

Bigram:

{“red,” , “green,” , “blue” , “red, green” , “green, blue”}

NOTE THAT THE COMMA IS CONSIDERED A PART OF THE WORD!

UNIGRAM Vs. BIGRAM Vs. TRIGRAM

“Let’s have a happy life”
“Let’s have a better job”

	Let’s	Have	a	happy	life	Better	job	Let’s have	Have a	A happy	Happy life	A better	Better job
Let’s have a happy life													
Let’s have a better job													

ORTHOGONAL SPARSE BIGRAM (OSB)

ORTHOGONAL SPARSE BIGRAM (OSB)

- OSB is used for text transformation and it is an alternative to bi-gram transformation.
- OSB encodes the words in a text along with how many words have been skipped as well (distance between words).
- OSB might perform better than n-grams if text data has large text fields (10 or more words)
- OSB works as follows:
  - Slide window of size n over text.
  - Output every pair of words that includes the first word in the window.
  - Join words with underscore “_” and include another “_” for every skipped token.

CONFUSED? LET’S TAKE A LOOK AT AN EXAMPLE!!

ORTHOGONAL SPARSE BIGRAM (OSB): EXAMPLE

- Example: "The quick brown fox jumps over the lazy dog", and OSBs of size 4.
- The six four-word windows, and the last two shorter windows from the end of the string are shown in the following example, as well OSBs generated from each:

```
"The quick brown fox", {The_quick, The__brown, The___fox}
"quick brown fox jumps", {quick_brown, quick__fox, quick___jumps}
"brown fox jumps over", {brown_fox, brown__jumps, brown___over}
"fox jumps over the", {fox_jumps, fox__over, fox___the}
"jumps over the lazy", {jumps_over, jumps__the, jumps___lazy}
"over the lazy dog", {over_the, over__lazy, over___dog}
"the lazy dog", {the_lazy, the__dog} "lazy dog", {lazy_dog}
```

CARTESIAN PRODUCT TRANSFORMATION

CARTESIAN PRODUCT TRANSFORMATION

- The Cartesian transformation works by creating permutations between variables.
- Cartesian transformation works well if there is an interactions between features.
- Example:
    - purchase_product, education, job
    - 0, university.degree, technician
    - 0, high.school, services
    - 1, university.degree, admin
- There might be interaction between education and job in deciding whether customer would buy a product or not so Cartesian product might work well in this case.
    - purchase_product, education_job_interaction
    - 0, university.degree_technician
    - 0, high.school_services
    - 1, university.degree_admin

CARTESIAN PRODUCT TRANSFORMATION

- The Cartesian transformation works with text as well as follows:

Textbook	Title	Binding	Cartesian product of no_punct(Title) and Binding
0	Deep Learning: basics, applications	Hardcover	{“Deep_Hardcover", “Learning_Hardcover", “basics_Hardcover", “applications_Hardcover"}
1	Machine Learning Practical	Softcover	{“Machine_Softcover", “Learning_Softcover", “Practical_Softcover"}