TU 257 – Fundamentals of Data Science

Data Analytics

L3 – Data Exploration & Preparation

Brendan Tierney

- Slightly ahead
- Slightly behind
- Second chance to understand
- In more detail
- Reinforcement

Agenda

- Why do we explore the data
  - Understanding your Data (Good, Bad & Ugly)
- Low hanging fruits
- Preparing Data – You have many tools in the toolbox
  - Missing Values
  - Recoding Values
  - Correlated Data
  - Feature Engineering
  - Imbalanced Data Sets
  - Sampling & Subsetting the Dataset
  - Splitting Data
- Iterative in Nature
  - You won't get it right first time
  - Experiment to see what works

No Magic Solution

A set of tools

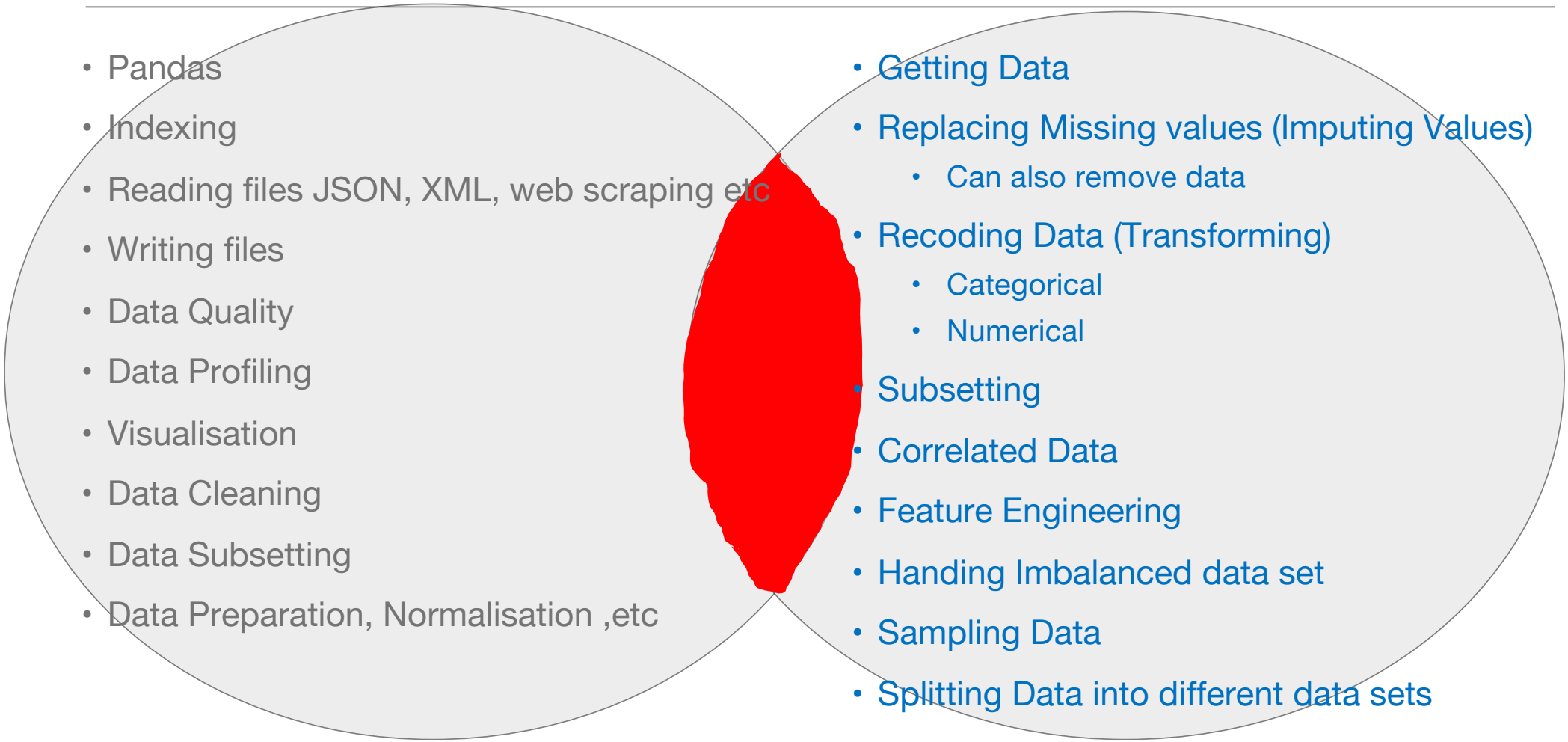
Every Data Set is different

Data Wrangling module vs This Module

- Pandas
- Indexing
- Reading files JSON, XML, web scraping etc
- Writing files
- Data Quality
- Data Profiling
- Visualisation
- Data Cleaning
- Data Subsetting
- Data Preparation, Normalisation ,etc

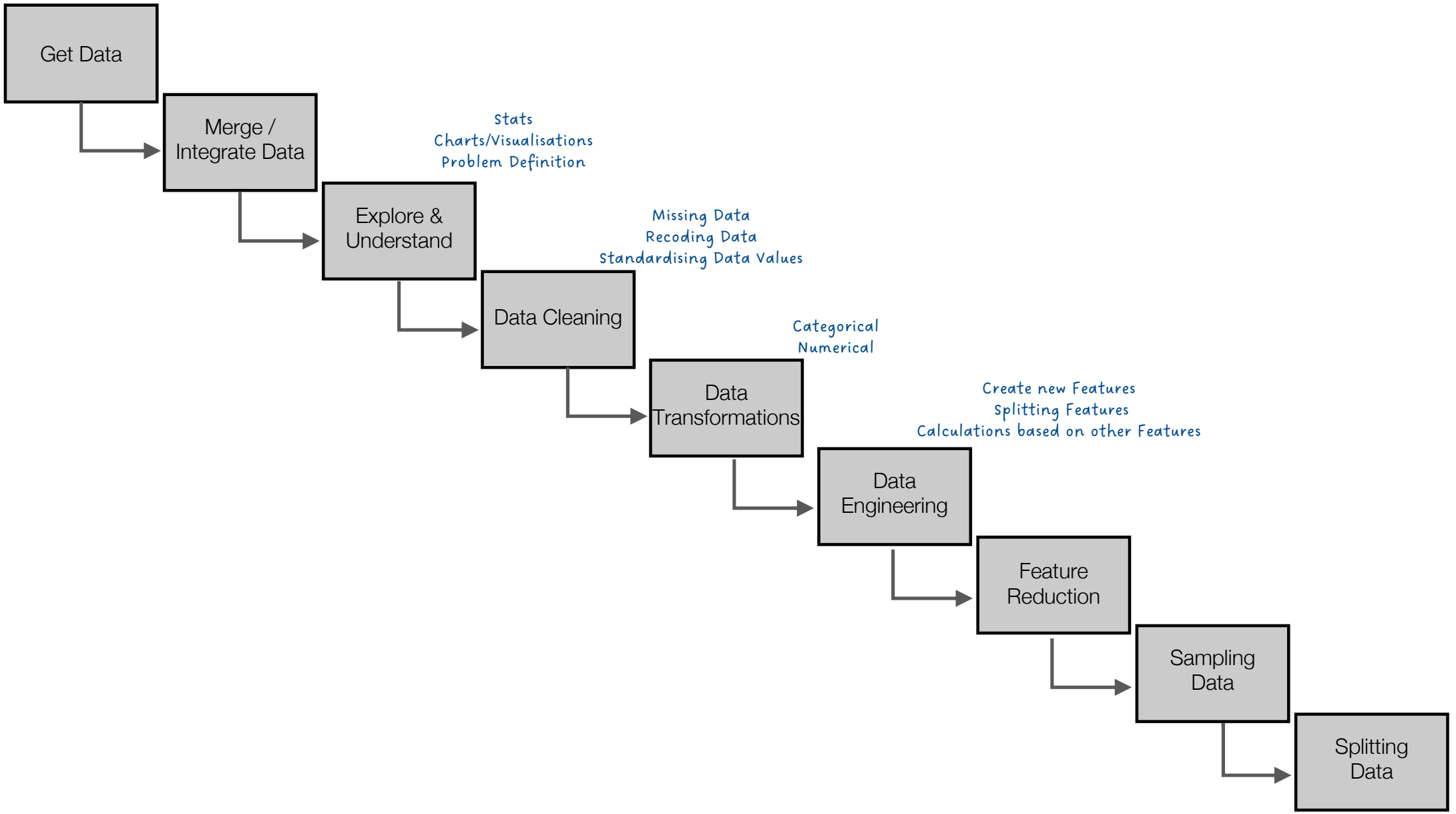
- Getting Data
- Replacing Missing values (Imputing Values)
  - Can also remove data
- Recoding Data (Transforming)
  - Categorical
  - Numerical
- Subsetting
- Correlated Data
- Feature Engineering
- Handing Imbalanced data set
- Sampling Data
- Splitting Data into different data sets

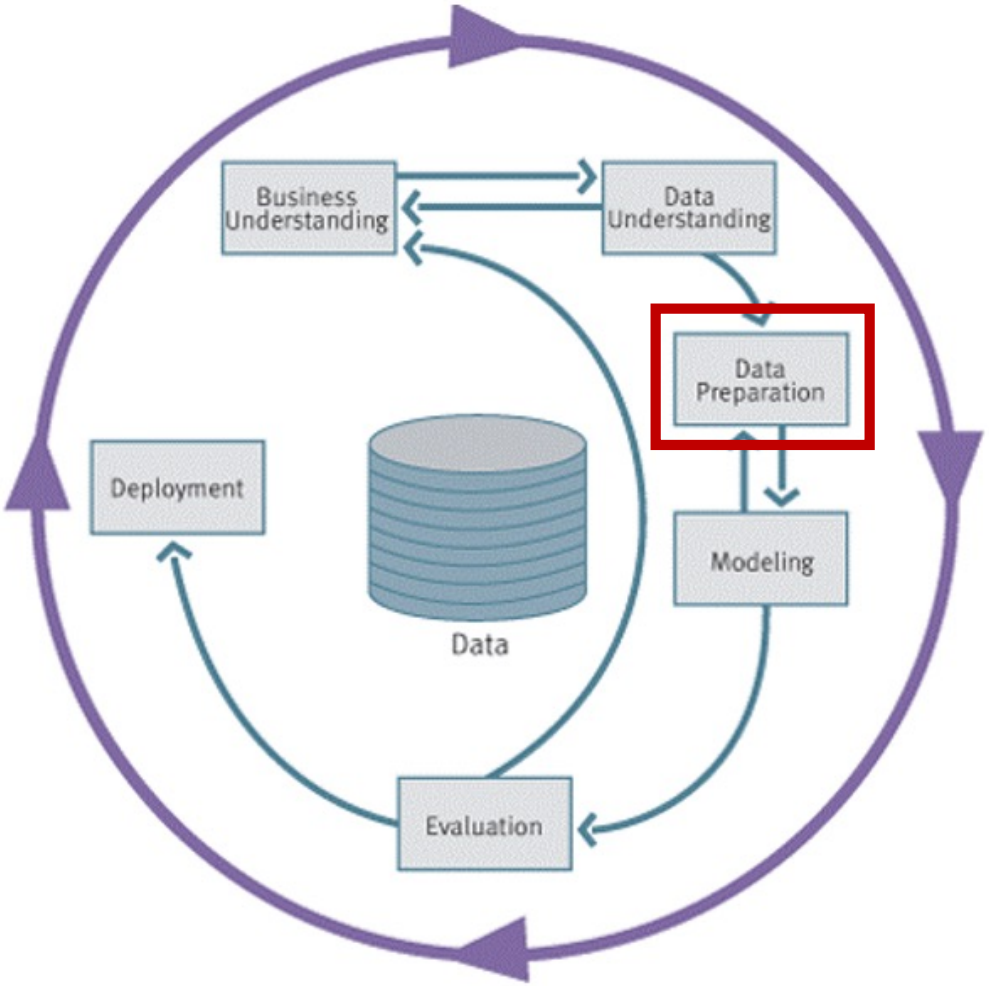
Data Wrangling module vs This Module



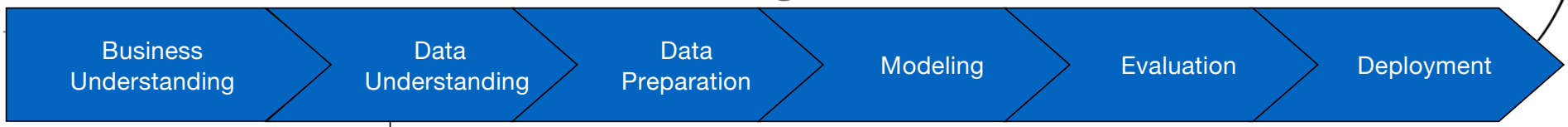
- Pandas
- Indexing
- Reading files JSON, XML, web scraping etc
- Writing files
- Data Quality
- Data Profiling
- Visualisation
- Data Cleaning
- Data Subsetting
- Data Preparation, Normalisation ,etc

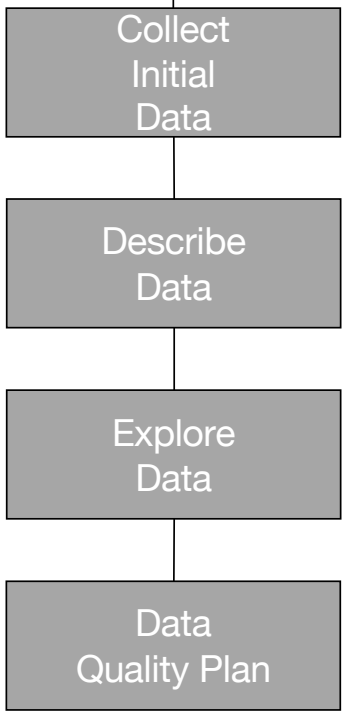
- Getting Data
- Replacing Missing values (Imputing Values)
  - Can also remove data
- Recoding Data (Transforming)
  - Categorical
  - Numerical
- Subsetting
- Correlated Data
- Feature Engineering
- Handing Imbalanced data set
- Sampling Data
- Splitting Data into different data sets


Get Data
Merge / Integrate Data
Stats
Charts/Visualisations
Problem Definition
Explore & Understand
Missing Data
Recoding Data
Standardising Data Values
Data Cleaning
Categorical
Numerical
Data Transformations
Create new Features
Splitting Features
Calculations based on other Features
Data Engineering
Feature Reduction
Sampling Data
Splitting Data


Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Data

Phase 2 – Data Understanding

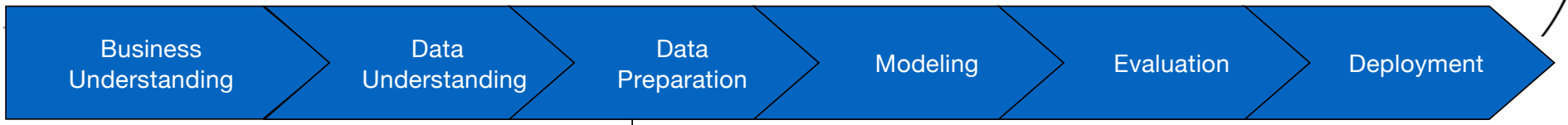


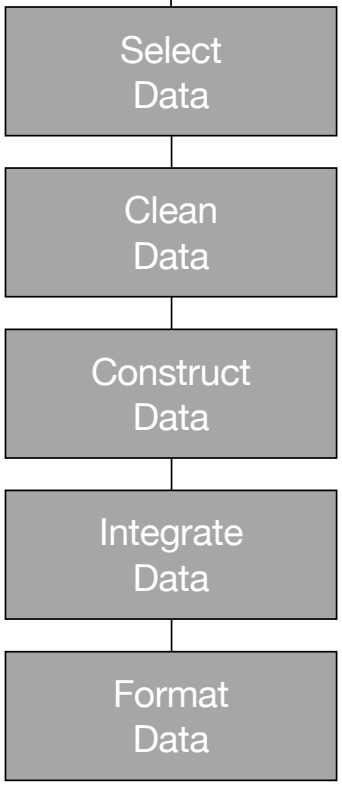


- Acquire the data
- Explore the data (query & visualization)
- Verify the quality

- Starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data or to detect interesting subsets to form hypotheses for hidden information.

Phase 3 – Data Preparation





Data preparation:

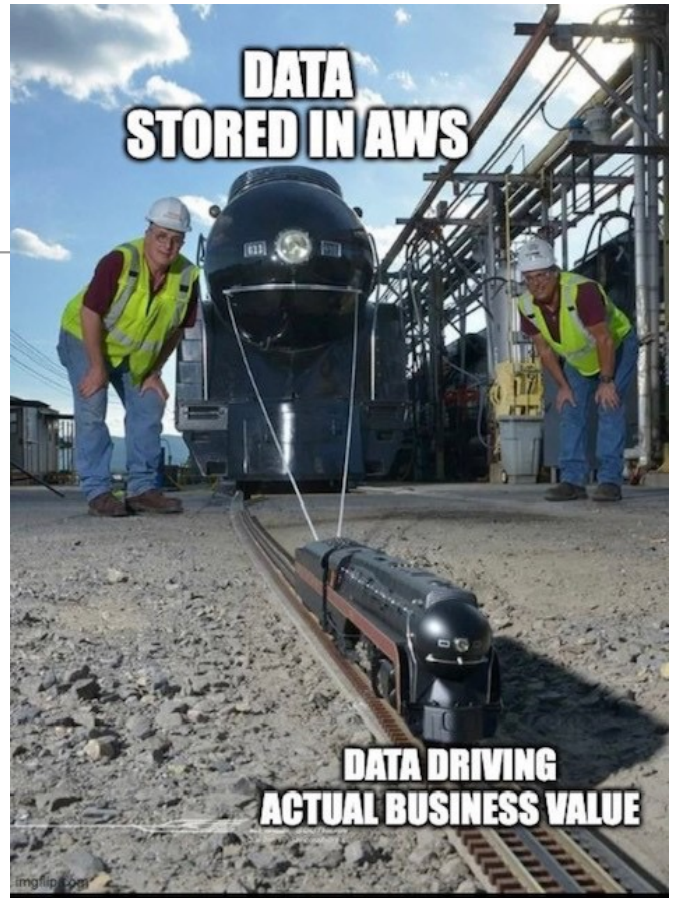
- Collection
- Assessment
- Consolidation and Cleaning
  - table links, aggregation level, missing values, etc
- Data selection
  - active role in ignoring non-contributory data?
  - outliers?
  - Use of samples
  - visualization tools
- Transformations - create new variables

- Covers all activities to construct the final dataset from the initial raw data. Data preparation tasks are likely to be performed multiple times and not in any prescribed order. Tasks include table, record and attribute selection as well as transformation and cleaning of data for modeling tools.

Why do we explore the data ?

Need to know and understand you Data

- What Data do you have? (easy access to)
- What Data would you like to have?
- Not always possible to have all the data you want !
- You will never need ALL the data – Big Data isn't a problem
  - What is Big Data Really!



- Get to know your data
- Data Audit
- Data Quality Report

Remember

- Automated Data Exploration?
- Using Different Pandas functions?

- What you have vs What you don't have
- How clean is the data vs how dirty is it
- How consistent is the data vs how inconsistent it is
  - Particularly when you merge data from different sources
- Is data available from one Data Source vs How many different Data Sources
  - In most cases, Data will come from different Data Sources
  - Different Data Sources => Differences in Data formats, values, representations
    - All of these need to be cleaned up, to make it consistent.
- You'd think data in your Enterprise is clean, consistent, good quality, etc
  - It generally isn't !

Data Audit & Quality Report

- A data quality report includes tabular reports that describe the characteristics of each feature (column)
- The tabular reports are accompanied by data visualizations:
  - A histogram for each continuous feature.
  - A bar plot for each categorical feature.

ID	Type	Inc.	Marital Status	Num Clmnts.	Injury Type	Hospital Stay	Claim Amnt.	Total Claimed	Num Claims	Num Soft Tiss.	% Soft Tiss.	Claim Amt Rcvd.	Fraud Flag
1	CI	0		2	Soft Tissue	No	1,625	3250	2	2	1.0	0	1
2	CI	0		2	Back	Yes	15,028	60,112	1		0	15,028	0
3	CI	54,613	Married	1	Broken Limb	No	-99,999	0	0	0	0	572	0
4	CI	0		4	Broken Limb	Yes	5,097	11,661	1	1	1.0	7,864	0
5	CI	0		4	Soft Tissue	No	8869	0	0	0	0	0	1
6	CI	0		1	Broken Limb	Yes	17,480	0	0	0	0	17,480	0
7	CI	52,567	Single	3	Broken Limb	No	3,017	18,102	2	1	0.5	0	1
8	CI	0		2	Back	Yes	7463	0	0	0	0	7,463	0
9	CI	0		1	Soft Tissue	No	2,067	0	0	0	0	2,067	0
10	CI	42,300	Married	4	Back	No	2,260	0	0	0	0	2,260	0
⋮													
300	CI	0		2	Broken Limb	No	2,244	0	0	0	0	2,244	0
301	CI	0		1	Broken Limb	No	1,627	92,283	3	0	0	1,627	0
302	CI	0		3	Serious	Yes	270,200	0	0	0	0	270,200	0
303	CI	0		1	Soft Tissue	No	7,668	92,806	3	0	0	7,668	0
304	CI	46,365	Married	1	Back	No	3,217	0	0		0	1,653	0
⋮													
458	CI	48,176	Married	3	Soft Tissue	Yes	4,653	8,203	1	0	0	4,653	0
459	CI	0		1	Soft Tissue	Yes	881	51,245	3	0	0	0	1
460	CI	0		3	Back	No	8,688	729,792	56	5	0.08	8,688	0
461	CI	47,371	Divorced	1	Broken Limb	Yes	3,194	11,668	1	0	0	3,194	0
462	CI	0		1	Soft Tissue	No	6,821	0	0	0	0	0	1
⋮													
491	CI	40,204	Single	1	Back	No	75,748	11,116	1	0	0	0	1
492	CI	0		1	Broken Limb	No	6,172	6,041	1		0	6,172	0
493	CI	0		1	Soft Tissue	Yes	2,569	20,055	1	0	0	2,569	0
494	CI	31,951	Married	1	Broken Limb	No	5,227	22,095	1	0	0	5,227	0
495	CI	0		2	Back	No	3,813	9,882	3	0	0	0	1
496	CI	0		1	Soft Tissue	No	2,118	0	0	0	0	0	1
497	CI	29,280	Married	4	Broken Limb	Yes	3,199	0	0	0	0	0	1
498	CI	0		1	Broken Limb	Yes	32,469	0	0	0	0	16,763	0
499	CI	46,683	Married	1	Broken Limb	No	179,448	0	0		0	179,448	0
500	CI	0		1	Broken Limb	No	8,259	0	0	0	0	0	1

(a) Continuous Features

Feature	Count	% Miss.	Card.	Min	1^{st} Qrt.	Mean	Median	3^{rd} Qrt.	Max	Std. Dev.
Income	500	0.0	171	0.0	0.0	13,740.0	0.0	33,918.5	71,284.0	20,081.5
Num Claimants	500	0.0	4	1.0	1.0	1.9	2	3.0	4.0	1.0
Claim Amount	500	0.0	493	-99,999	3,322.3	16,373.2	5,663.0	12,245.5	270,200.0	29,426.3
Total Claimed	500	0.0	235	0.0	0.0	9,597.2	0.0	11,282.8	729,792.0	35,655.7
Num Claims	500	0.0	7	0.0	0.0	0.8	0.0	1.0	56.0	2.7
Num Soft Tissue	500	2.0	6	0.0	0.0	0.2	0.0	0.0	5.0	0.6
% Soft Tissue	500	0.0	9	0.0	0.0	0.2	0.0	0.0	2.0	0.4
Amount Received	500	0.0	329	0.0	0.0	13,051.9	3,253.5	8,191.8	295,303.0	30,547.2
Fraud Flag	500	0.0	2	0.0	0.0	0.3	0.0	1.0	1.0	0.5

(a) Categorical Features

Feature	Count	% Miss.	Card.	Mode	Mode Freq.	Mode %	2^{nd} Mode	2^{nd} Mode Freq.	2^{nd} Mode %
Insurance Type	500	0.0	1	CI	500	1.0	–	–	–
Marital Status	500	61.2	4	Married	99	51.0	Single	48	24.7
Injury Type	500	0.0	4	Broken Limb	177	35.4	Soft Tissue	172	34.4
Hospital Stay	500	0.0	2	No	354	70.8	Yes	146	29.2

Plus charts/graphs of each feature

Data Audit & Quality Report

- Have you seen something like this before?

- Automated Data Exploration – Remember example and exercises from last week

- Explore the Data
- What patterns do you see?
- Can you connect any Domain Knowledge to the patterns you are seeing?
  - Nothing magical with doing this
  - Nobody or thing will tell you the meaning of these patterns
  - It is your job to make the connections
- Start Building a Picture of what is happening in the Data & what work needs to be done to the data
  - Iterate to improve your understanding

Low hanging fruits

Pick the Low Hanging Fruits

- All too often people look to use advanced and unnecessary algorithms, tools, approaches etc. Before doing the simple and basic things
- Most of the valuable Data Insights can be achieved by examining the data
- See what patterns you have
- Can you related to Domain Knowledge or Certain Events
- Discuss with Business Users
- Can be used to Verify existing Knowledge & Identify new Knowledge

- Can/Will be iterative
  - Can generate more questions from Business for you to explore in Data
  - Can identify more questions for Business to clarify and verify

Pick the Low Hanging Fruits

- Identify Patterns
- Can these be converted into Rules
- Can you prioritise these Rules
- Create IF…THEN
- Eg. Car Insurance industry – Claims
  - Age profile
  - Driving experience
  - Type of Car
  - Value of Car
  - Job
  - Service History
  - Payment Schedule

By analysing this data you can identify certain patterns in the data?

Based on your Domain Knowledge, can you identify possible Rules/Patterns?

These would need to be verified with the Business.

- Data Sets available online vs Real World
- Most/All Data Sets online you get are Clean-ish
- Most of what we will cover in remaining slides are examples of things you need to look out for.
  - Even if the Data Set seems clean, you still need to check and verify it.
- In Real World data sets, you can spend a lot of time merging and cleaning the data (and using what is covered in remaining slides)
  - 80+% of project time

- Know your Tool/Product
- Most Enterprise BI/DM/ML/AI solutions – Do these tasks/steps automatically
  - Automate the boring stuff
- Most FREE products don't - R, Python, Julia, Spark, etc
  - Needs to be manually coded! Boring! Wastes a lot of time! Expensive for Businesses
  - Mistakes can happen

Preparing Data – You have many tools in the toolbox

Your Toolbox

Bricks

Cement

Concrete

Sand

Reinforcement

Glass

Plastic

Wood

Tiles

Your Toolbox

From Basics
to Experienced
Takes Time
& Practice


Bricks


Cement


Concrete


Sand


Reinforcement


Glass


Plastic


Wood


Tiles

Explore your Data

Last Week Lab Exercise

Exercise 2 - Data Understanding & Exploring – Manually with code

- Python Panda dataframes is you main tool for storing, processing and analysing data
- Can use other Python libraries to analyse data in Pandas dataframes
- RTFM
  - https://pandas.pydata.org/docs/
  - https://pandas.pydata.org/docs/user_guide/index.html
  - Check the documentation before googling for an answer
    - Google makes you Stupid!
- Install

```
pip3 install pandas
```

Additional Learning Resouces
10 Minutes to Pandas
Pandas Tutorials

TU258 - Lab 2-2 - Data Exploration

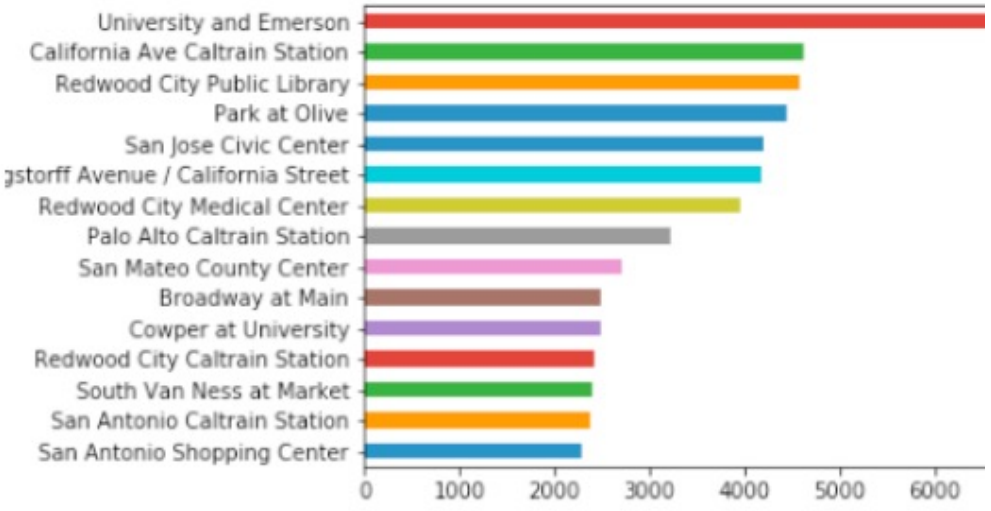
This lab gives an example of using Pandas dataframes for analysing data

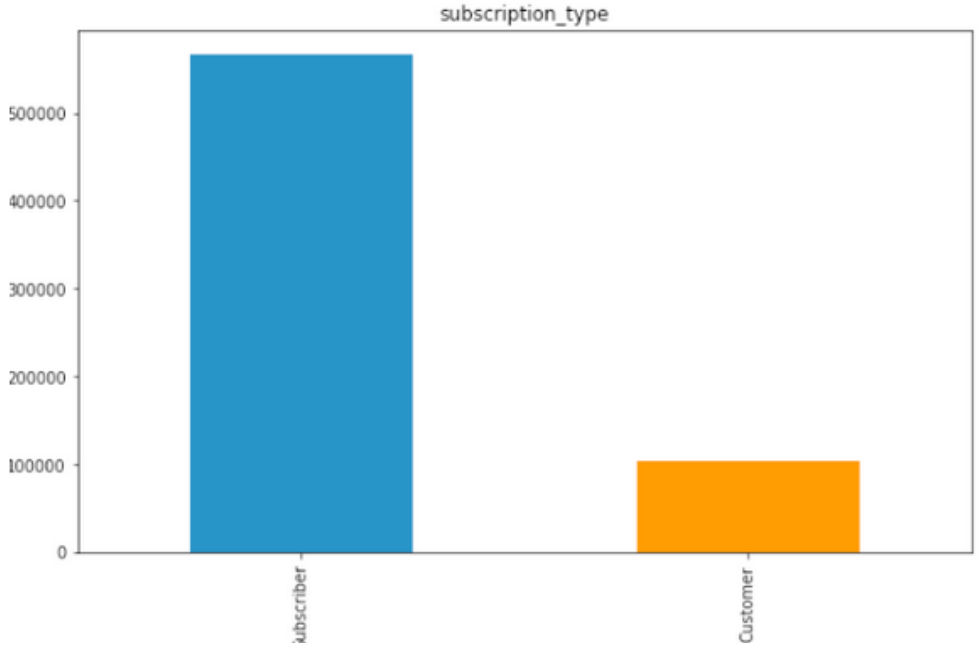
In [2]:
```
#import pandas
import pandas as pd
```

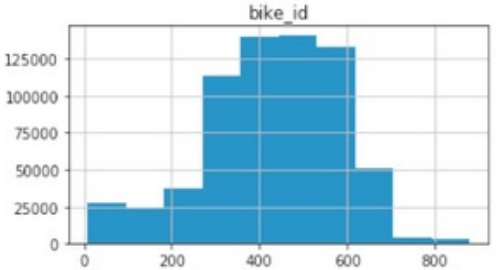
In [3]:
```
#reading a CSV File into a Panda
videoReview = pd.read_csv('/Users/brendan.tierney/Downloads/Video_Games_Sales_as_at_2
```

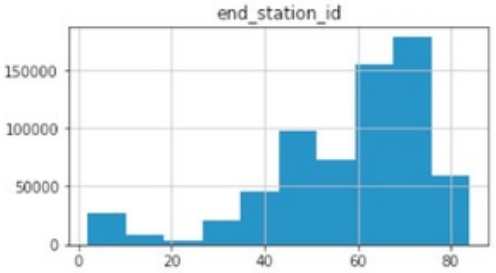
In [5]:
```
print('# print first 3 rows')
print(videoReview[:3])
```

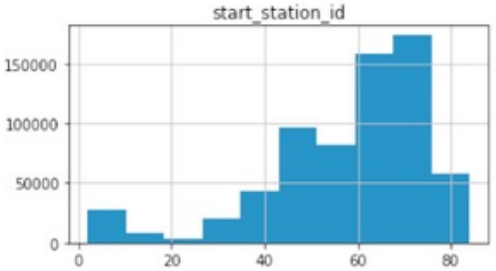
```
# print first 3 rows
                 Name Platform  Year_of_Release     Genre Publisher  NA_Sales  \
0         Wii Sports      Wii           2006.0    Sports  Nintendo     41.36
1  Super Mario Bros.      NES           1985.0  Platform  Nintendo     29.08
2     Mario Kart Wii      Wii           2008.0    Racing  Nintendo     15.68

   EU_Sales  JP_Sales  Other_Sales  Global_Sales  Critic_Score  Critic_Count  \
0     28.96      3.77         8.45         82.53          76.0          51.0
1      3.58      6.81         0.77         40.24           NaN           NaN
2     12.76      3.79         3.29         35.52          82.0          73.0

  User_Score  User_Count Developer Rating
0          8       322.0  Nintendo      E
1        NaN         NaN       NaN    NaN
2        8.3       709.0  Nintendo      E
```

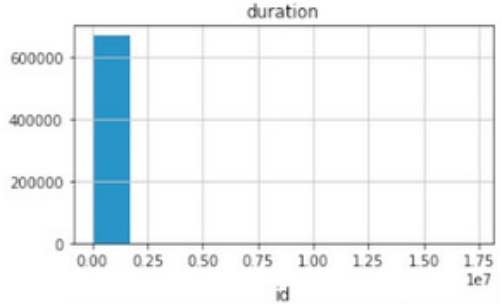

University and Emerson
California Ave Caltrain Station
Redwood City Public Library
Park at Olive
San Jose Civic Center
gstorff Avenue / California Street
Redwood City Medical Center
Palo Alto Caltrain Station
San Mateo County Center
Broadway at Main
Cowper at University
Redwood City Caltrain Station
South Van Ness at Market
San Antonio Caltrain Station
San Antonio Shopping Center
0
1000
2000
3000
4000
5000
6000

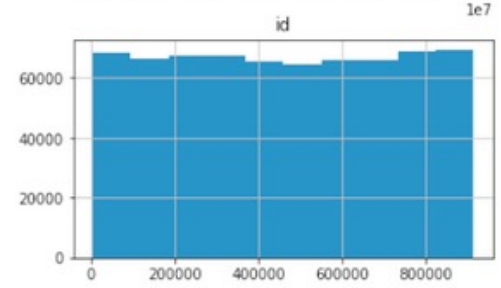

subscription_type
500000
400000
300000
200000
100000
0
ubscriber
Customer


bike_id
125000
100000
75000
50000
25000
0
0
200
400
600
800


end_station_id
150000
100000
50000
0
0
20
40
60
80

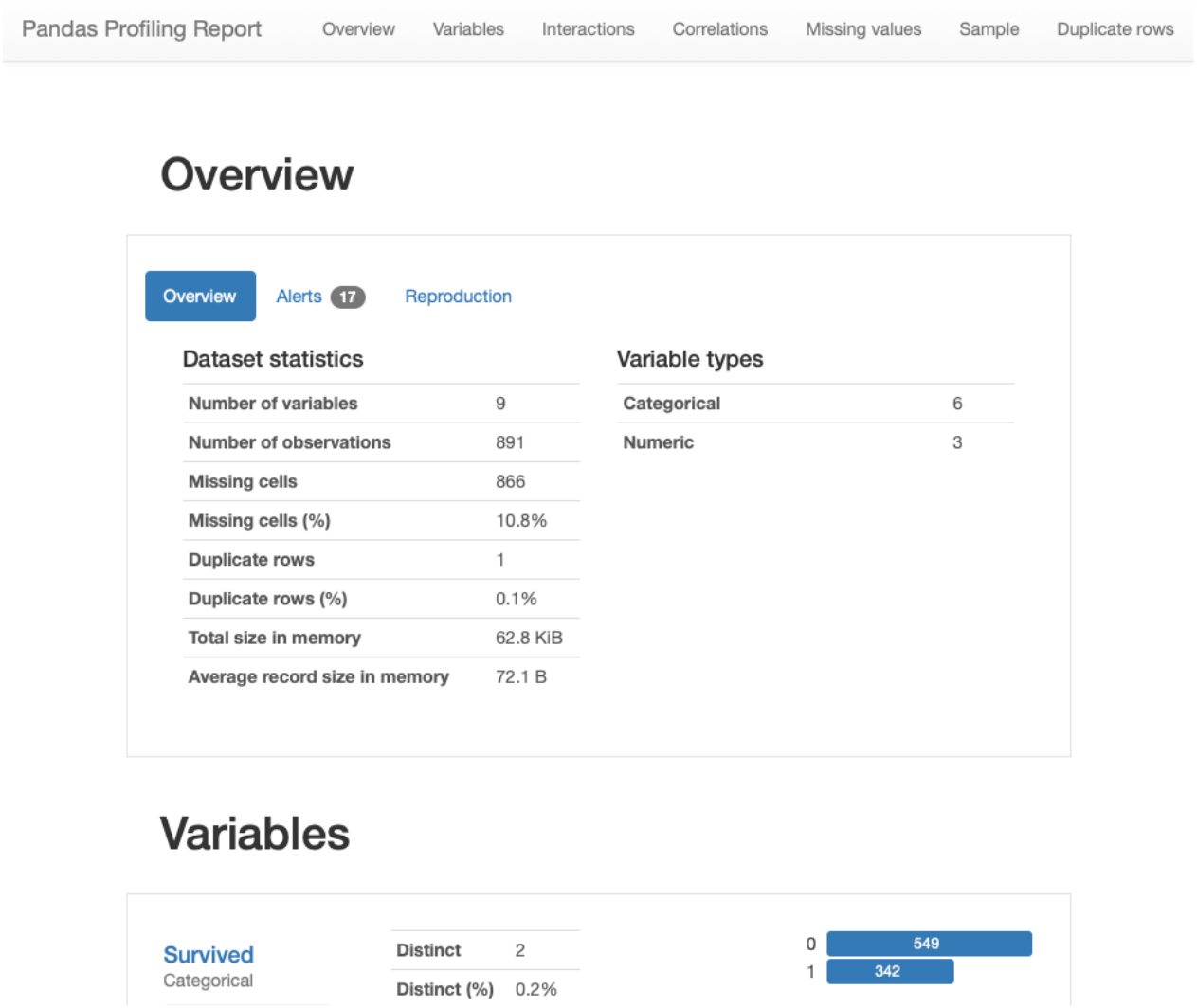

start_station_id
150000
100000
50000
0
0
20
40
60
80


duration
600000
400000
200000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e7


id
60000
40000
20000
0
0
200000
400000
600000
800000

- We also have automated data exploration/profiling

```
import pandas_profiling as pp

df2.profile_report()
```



Handing Missing Data

Data can be Dirty

- Data in the real world is dirty
  - **Incomplete**: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., occupation=“”
    - Comes from
      - N/A data value when collected
      - Different consideration between the time when the data was collected and when it is analyzed
      - Human/hardware/software problems
      - N/A may have a meaning
  - **Noisy**: containing errors or outliers
    - e.g., Salary=“-10”
    - Comes from
      - Collection
      - Entry
      - Transmission
  - **Inconsistent**: containing discrepancies in codes or names
    - e.g., Age=“42” Birthday=“03/07/1997”
    - e.g., Was rating “1,2,3”, now rating “A, B, C”
    - e.g., discrepancy between duplicate records
    - Comes from
      - Different data sources
      - Functional dependency violation

Missing Data

- Reasons for missing values
  - Information is not collected
    (e.g., people decline to give their age and weight)
  - Attributes may not be applicable to all cases
    (e.g., annual income is not applicable to children)
    - E.g. many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to:
  - Equipment malfunction
  - Inconsistent with other recorded data and thus deleted
  - Data not entered due to misunderstanding
  - Certain data may not be considered important at the time of entry
  - Not registering history or changes of the data
- Handling missing values
  - Eliminate Data Objects
  - Estimate Missing Values
  - Ignore the Missing Value During Analysis
  - Replace with all possible values (weighted by their probabilities)

Missing Data

- Your data set could have missing data
- These will be missing values in the features/columns
- Algorithms don’t like missing values
- We need to fix these
- How
  - If a Row/Record has a lot of missing values -> Delete it
    - As it will be just filled with defaults (see below)

  - If a Feature/Column has a lot of missing values -> Drop it, as it won’t contribute to any analysis
  - Calculate (Impute) possible missing value -> Mean, Mode, Median
  - Calculate (Impute) possible missing value -> Based on some calculation of one or more Features/Columns
  - Replace with some default value

Fixing Missing Data

```
#Replace missing values with Zero
#Does this for all Columns in dataframe
df = df.fillna(0)

#Or for one Column
# replace 'DataFrame Column' with name of column
df['DataFrame Column'] = df['DataFrame Column'].fillna(0)

#example - replace with default value
df.country.fillna("Unknown")

#Check to see if a column contains nulls
df[pd.isnull(df.country)]
```

Fixing Missing Data

```
#Alternatively, we may have a non-null value that we would like to replace
df.taster_twitter_handle.replace("@kerinokeefe", "@kerino")


df = df.replace(['NO DATA','N/A', 0, ''], np.nan)
df = df.replace(['NO DATA','N/A', 0, ''], "Unknown")
```

The replace() method is worth mentioning here because it's handy for replacing missing data which is given some kind of sentinel value in the dataset: things like "Unknown", "Undisclosed", "Invalid", and so on.

Fixing Missing Data

Method 1: Fill NaN Values **in One Column** with Mean

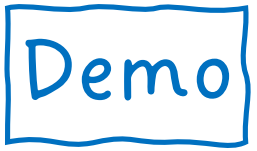
```
df['col1'] = df['col1'].fillna(df['col1'].mean())
```

Method 2: Fill NaN Values in **Multiple Columns** with Mean

```
df[['col1', 'col2']] = df[['col1', 'col2']].fillna(df[['col1', 'col2']].mean())
```

Method 3: Fill NaN Values in **All Columns** with Mean

```
df = df.fillna(df.mean())
```



Recoding Data (Transformations)
Categorical & Numerical

Cleaning Data

- Data set may include data objects that are duplicates, or almost duplicates of one another
  - Major issue when merging data from heterogeous sources
- Can also be same data item entered in a different way
  - Free format text
- Examples:
  - Same person with multiple email addresses
- Data cleaning
  - Process of dealing with duplicate data issues

Exercise – Get out a Pen and Paper / type up the following

Data Cleaning Problem

- Write the full work address for Brendan Tierney. (I'll call it out and you can write it down)
    - Type up what I've called out

- When you have finished -> Put it in the Chat window
- Share with class
- Do you get the same as everyone else?

Data Cleaning Problem

- Write the full work address for Brendan Tierney

brendan tierney
school of computing science
technological university dublin
grangegorman
dublin 7
ireland

Brendan Tierney
School of Computing Science
Technological University Dublin
Grangegorman
Dublin 7
Ireland

Brendan Tierney,
School of Computing Science,
Technological University Dublin,
Grangegorman,
Dublin 7,
Ireland.

Brendan Tierney,
School of Computing Science,
Technological University Dublin,
Grangegorman,
D7,
Ireland.

Brendan Tierney,
School of Computing Science,
Technological University Dublin,
Grangegorman,
Dublin 7,
Ire.

There are lots and lots of other combinations. All of these are different.

Converting Data

- Lots and Lots of possibilities here
- We will only cover some of these now
    - Others will be covered in Data Wangling module
    - Others we will covered throughout the semester in this module
- Generally – Data Analytics algorithms do not like attributes/features in Text format
    - Convert to numerical
- Numerical Data -> Some formatting is needed

Converting Text Data to Numerical Data

- Some examples include:
  - A "*pet*" variable with the values: "*dog*" and "*cat*".
  - A "*color*" variable with the values: "*red*", "*green*" and "*blue*".
  - A "*place*" variable with the values: "first", "*second*" *and* "*third*".

- Possible approaches
  - One Hot Coding
  - Find/Replace (Manual Coding)
  - Label Encodings

Let's have a look at an example of each

- Everyone has the own favourite way of doing these things
- New approaches, libraries, code etc are coming available all the time
- The above are quick and easy for you to get started

One-Hot Coding

- Creates a new attribute/feature for each distinct value
  - If a categorical variable has 5 values, then 5 new attributes/features will be created

```
import pandas as pd
import numpy as np
import matplotlib as plt
videoReview =
pd.read_csv('/Users/brendan.tierney/Downloads/Video_Games_Sales_as_at_22_Dec_2016.csv')
videoReview.head(10)
```

	Name	Platform	Genre	Publisher	User_Score	Developer	Rating
0	Wii Sports	Wii	Sports	Nintendo	8	Nintendo	E
1	Super Mario Bros.	NES	Platform	Nintendo	NaN	NaN	NaN
2	Mario Kart Wii	Wii	Racing	Nintendo	8.3	Nintendo	E
3	Wii Sports Resort	Wii	Sports	Nintendo	8	Nintendo	E
4	Pokemon Red/Pokemon Blue	GB	Role-Playing	Nintendo	NaN	NaN	NaN
5	Tetris	GB	Puzzle	Nintendo	NaN	NaN	NaN
6	New Super Mario Bros.	DS	Platform	Nintendo	8.5	Nintendo	E
7	Wii Play	Wii	Misc	Nintendo	6.6	Nintendo	E
8	New Super Mario Bros. Wii	Wii	Platform	Nintendo	8.4	Nintendo	E
9	Duck Hunt	NES	Shooter	Nintendo	NaN	NaN	NaN

One-Hot Coding

```
#apply one-hot-coding to all the categorical variables
# and create a new dataframe to store the results

df2 = pd.get_dummies(df)
df2.head(10)
```

	Name_ Tales of Xillia 2	Name_.hack//Infection Part 1	Name_.hack//Mutation Part 2	Name_.hack//Outbreak Part 3	Name_007 Racing	Name_007: Quantum of Solace	Name_007: The World is not Enough	Name_1 vs. 100	Name_10 Minute Solution	Name_100 All-Time Favorites	...	Develo
0	0	0	0	0	0	0	0	0	0	0	...	
2	0	0	0	0	0	0	0	0	0	0	...	
3	0	0	0	0	0	0	0	0	0	0	...	
6	0	0	0	0	0	0	0	0	0	0	...	
7	0	0	0	0	0	0	0	0	0	0	...	
8	0	0	0	0	0	0	0	0	0	0	...	
11	0	0	0	0	0	0	0	0	0	0	...	
13	0	0	0	0	0	0	0	0	0	0	...	
14	0	0	0	0	0	0	0	0	0	0	...	
15	0	0	0	0	0	0	0	0	0	0	...	

10 rows × 8138 columns

Lots of additional attributes/features
Lots containing Zeros

Might not be suitable for data sets with a large number of distinct categorical attributes

The original data set had 7 attributes/columns/features

Demo

Find/Replace (Manual Coding)

- You can create a new attribute/feature and write code to do mapping to numeric

```
df['Rating'].value_counts()
```

```
E       3949
T       2939
M       1560
E10+    1411
EC         8
K-A        3
RP         2
AO         1
Name: Rating, dtype: int64
```

- The last 4 values listed have very small number of occurrences.
- We will group these into having one value/category

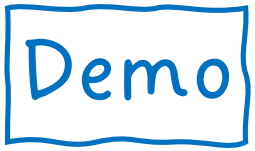
```
find_replace = {"Rating" : {"E": 1, "T": 2, "M": 3, "E10+": 4, "EC": 5, "K-A": 5, "RP": 5, "AO": 5}}
df.replace(find_replace, inplace=True)
df.head(10)
```

Using same data set as previous example

Find/Replace (Manual Coding)

	Name	Platform	Genre	Publisher	User_Score	Developer	Rating
0	Wii Sports	Wii	Sports	Nintendo	8	Nintendo	1
2	Mario Kart Wii	Wii	Racing	Nintendo	8.3	Nintendo	1
3	Wii Sports Resort	Wii	Sports	Nintendo	8	Nintendo	1
6	New Super Mario Bros.	DS	Platform	Nintendo	8.5	Nintendo	1
7	Wii Play	Wii	Misc	Nintendo	6.6	Nintendo	1
8	New Super Mario Bros. Wii	Wii	Platform	Nintendo	8.4	Nintendo	1
11	Mario Kart DS	DS	Racing	Nintendo	8.6	Nintendo	1
13	Wii Fit	Wii	Sports	Nintendo	7.7	Nintendo	1
14	Kinect Adventures!	X360	Misc	Microsoft Game Studios	6.3	Good Science Studio	1
15	Wii Fit Plus	Wii	Sports	Nintendo	7.4	Nintendo	1

- You have greater control over what is created
- Fewer attributes/features needed
- More understanding of the data needed, and on how data can be grouped
- More coding needed



Label Encodings

- With this technique where each distinct value in a categorical variable is converted to a number.
- In this scenario you don't get to pick the numeric value assigned to the value. It is system determined.
- Our categorical variables are of 'object' data type. We need to convert to a category data type.
- In this example 'Platform' has a large-ish number of values and we want a quick way of converting them we can illustrate this by creating a new variable.

```
df["Platform_Category"] = df["Platform"].astype('category')
df.dtypes
```

- Now convert this new variable to numeric.

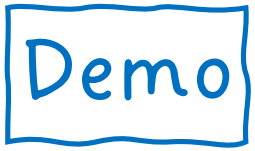
```
df["Platform_Category"] = df["Platform_Category"].cat.codes
df.head(20)
```

	Name	Platform	Genre	Publisher	User_Score	Developer	Rating	Platform_Category
0	Wii Sports	Wii	Sports	Nintendo	8	Nintendo	1	12
2	Mario Kart Wii	Wii	Racing	Nintendo	8.3	Nintendo	1	12
3	Wii Sports Resort	Wii	Sports	Nintendo	8	Nintendo	1	12
6	New Super Mario Bros.	DS	Platform	Nintendo	8.5	Nintendo	1	2
7	Wii Play	Wii	Misc	Nintendo	6.6	Nintendo	1	12
8	New Super Mario Bros. Wii	Wii	Platform	Nintendo	8.4	Nintendo	1	12
11	Mario Kart DS	DS	Racing	Nintendo	8.6	Nintendo	1	2
13	Wii Fit	Wii	Sports	Nintendo	7.7	Nintendo	1	12
14	Kinect Adventures!	X360	Misc	Microsoft Game Studios	6.3	Good Science Studio	1	14
15	Wii Fit Plus	Wii	Sports	Nintendo	7.4	Nintendo	1	12
16	Grand Theft Auto V	PS3	Action	Take-Two Interactive	8.2	Rockstar North	3	8
17	Grand Theft Auto: San Andreas	PS2	Action	Take-Two Interactive	9	Rockstar North	3	7
19	Brain Age: Train Your Brain in Minutes a Day	DS	Misc	Nintendo	7.9	Nintendo	1	2
23	Grand Theft Auto V	X360	Action	Take-Two Interactive	8.1	Rockstar North	3	14
24	Grand Theft Auto: Vice City	PS2	Action	Take-Two Interactive	8.7	Rockstar North	3	7
26	Brain Age 2: More Training in Minutes a Day	DS	Puzzle	Nintendo	7.1	Nintendo	1	2
28	Gran Turismo 3: A-Spec	PS2	Racing	Sony Computer Entertainment	8.4	Polyphony Digital	1	7
29	Call of Duty: Modern Warfare 3	X360	Shooter	Activision	3.4	Infinity Ward, Sledgehammer Games	3	14
32	Call of Duty: Black Ops	X360	Shooter	Activision	6.3	Treyarch	3	14
34	Call of Duty: Black Ops II	PS3	Shooter	Activision	5.3	Treyarch	3	8

- The number assigned to the Platform_Category variable is based on the alphabetical ordering of the values in the Platform variable. For example,

```
df.groupby("Platform")["Platform"].count()
```

```
Platform
3DS      226
DC        14
DS      1255
GBA      517
GC       469
PC       760
PS       201
PS2     1478
PS3      947
PS4      252
PSP      543
PSV      149
Wii      997
WiiU     105
X360    1041
XB       733
XOne     186
Name: Platform, dtype: int64
```



Label Encodings – Using SciKit-Learn

- SciKit-Learn has a number of functions to help with data encodings. The first one we will look at is the 'fit_transform' function.

- This will perform a similar task to what we have seen in a previous example

```
#Let's use the fit_tranforms function to encode the Genre variable

from sklearn.preprocessing import LabelEncoder


le_make = LabelEncoder()

df["Genre_Code"] = le_make.fit_transform(df["Genre"])

df[["Genre", "Genre_Code"]].head(10)
```

	Genre	Genre_Code
0	Sports	10
2	Racing	6
3	Sports	10
6	Platform	4
7	Misc	3
8	Platform	4
11	Racing	6
13	Sports	10
14	Misc	3
15	Sports	10

Demo

What about Numerical Data?

- Normalization is a technique often applied as part of data preparation for machine learning.
- The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.
- For example, assume your input dataset contains one column with values ranging from 0 to 1, and another column with values ranging from 10,000 to 100,000. The great difference in the *scale* of the numbers could cause problems when you attempt to combine the values as features during modeling
- There are several approaches to this (experimentation needed to determine), common ones are:
    - Min-Max Scaling: Subtract the min value from each column's highest value and divide by the range.
    - Standardization Scaling: Subtracting the mean of each observation and dividing by the standard deviation

What about Numerical Data

- Example Data
  - Salaries
  - Temperatures
  - Prices
  - Salaries
  - Sales
  - Populations

- All have ranges from Zero to some large number
- Convert to common scale to allow the algorithms to compare
  - Otherwise the attributes/features with larger numbers will have a greater influence to those attributes with small variables
  - The algorithms may use these attributes incorrectly, and the models produced may not be the best

	Country	Age	Salary	Purchased
0	France	44.0	27000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

data.head(10)
# <-- See results to the left

scaler = MinMaxScaler()
data[['Age', 'Salary']] = scaler.fit_transform(data[['Age', 'Salary']])

data
```

	Country	Age	Salary	Purchased
0	France	44.0	27000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

data.head(10)
# <-- See results to the left

scaler = MinMaxScaler()
data[['Age', 'Salary']] = scaler.fit_transform(data[['Age', 'Salary']])

data

#       See results to the right -->
```

	Country	Age	Salary	Purchased
0	France	0.739130	0.000000	No
1	Spain	0.000000	0.375000	Yes
2	Germany	0.130435	0.482143	No
3	Spain	0.478261	0.607143	No
4	Germany	0.565217	NaN	Yes
5	France	0.347826	0.553571	Yes
6	Spain	NaN	0.446429	No
7	France	0.913043	0.928571	Yes
8	Germany	1.000000	1.000000	No
9	France	0.434783	0.714286	Yes

Demo

Lots of functions available

- Lots of options
- Remember to Keep It Simple
- Start Simple and with the basics
- As your knowledge and experience grows, explore some of the others

`sklearn.preprocessing` : Preprocessing and Normalization

'he `sklearn.preprocessing` module includes scaling, centering, normalization, binarization and imputation methods.

Iser guide: See the Preprocessing data section for further details.

`preprocessing.Binarizer` ([threshold, copy])	Binarize data (set feature values to 0 or 1) according to a threshold
`preprocessing.FunctionTransformer` ([func, ...])	Constructs a transformer from an arbitrary callable.
`preprocessing.KBinsDiscretizer` ([n_bins, ...])	Bin continuous data into intervals.
`preprocessing.KernelCenterer` ()	Center a kernel matrix
`preprocessing.LabelBinarizer` ([neg_label, ...])	Binarize labels in a one-vs-all fashion
`preprocessing.LabelEncoder`	Encode labels with value between 0 and n_classes-1.
`preprocessing.MultiLabelBinarizer` ([classes, ...])	Transform between iterable of iterables and a multilabel format
`preprocessing.MaxAbsScaler` ([copy])	Scale each feature by its maximum absolute value.
`preprocessing.MinMaxScaler` ([feature_range, copy])	Transforms features by scaling each feature to a given range.
`preprocessing.Normalizer` ([norm, copy])	Normalize samples individually to unit norm.
`preprocessing.OneHotEncoder` ([n_values, ...])	Encode categorical integer features as a one-hot numeric array.
`preprocessing.OrdinalEncoder` ([categories, dtype])	Encode categorical features as an integer array.
`preprocessing.PolynomialFeatures` ([degree, ...])	Generate polynomial and interaction features.
`preprocessing.PowerTransformer` ([method, ...])	Apply a power transform featurewise to make data more Gaussian-like.
`preprocessing.QuantileTransformer` ([...])	Transform features using quantiles information.
`preprocessing.RobustScaler` ([with_centering, ...])	Scale features using statistics that are robust to outliers.
`preprocessing.StandardScaler` ([copy, ...])	Standardize features by removing the mean and scaling to unit variance
`preprocessing.add_dummy_feature` (X[, value])	Augment dataset with an additional dummy feature.
`preprocessing.binarize` (X[, threshold, copy])	Boolean thresholding of array-like or scipy.sparse matrix
`preprocessing.label_binarize` (y, classes[, ...])	Binarize labels in a one-vs-all fashion
`preprocessing.maxabs_scale` (X[, axis, copy])	Scale each feature to the [-1, 1] range without breaking the sparsity.
`preprocessing.minmax_scale` (X[, ...])	Transforms features by scaling each feature to a given range.
`preprocessing.normalize` (X[, norm, axis, ...])	Scale input vectors individually to unit norm (vector length).
`preprocessing.quantile_transform` (X[, axis, ...])	Transform features using quantiles information.
`preprocessing.robust_scale` (X[, axis, ...])	Standardize a dataset along any axis
`preprocessing.scale` (X[, axis, with_mean, ...])	Standardize a dataset along any axis
`preprocessing.power_transform` (X[, method, ...])	Power transforms are a family of parametric, monotonic transformations that are applied to make data more Gaussian-like.

Correlated Data

Correlated Data

- Correlation explains how one or more variables are related to each other.
- If two variables are closely correlated, then we can predict one variable from the other

- Examples
  - Income and Expenditure of an individual
  - Demand and price of a commodity
  - Age and Insurance Claims

- Why is this important?
  - Can help clean up the data set
  - Reduce/Remove variables that are highly correlated
    - As they indicate the same thing, meaning, outcome, etc

Correlated Data

- A supermarket is beginning to offer a new line of organic products. The supermarket's management would like to determine which customers are likely to purchase these products.

- The supermarket has a customer loyalty program. As an initial buyer incentive plan, the supermarket provided coupons for the organic products to all of their loyalty program participants and has now collected data that includes whether or not these customers have purchased any of the organic products.

Name	Model Role	Measurement Level	Description
CUSTID	ID	Nominal	Customer loyalty identification number
GENDER	Input	Nominal	M = male, F = female, U = unknown
DOB	Input	Interval	Date of birth
EDATE	Input	Unary	Date extracted from the daily sales data base
AGE	Input	Interval	Age, in years
AGEGRP1	Input	Nominal	Age group 1
AGEGRP2	Input	Nominal	Age group 2
TV_REG	Input	Nominal	Television region
NGROUP	Input	Nominal	Neighborhood group
NEIGHBORHOOD	Input	Nominal	Type of residential neighborhood
LCDATE	Input	Interval	Loyalty card application date
LTIME	Input	Interval	Time as loyalty card member
ORGANICS	Input	Interval	Number of organic products purchased
BILL	Input	Interval	Total amount spent
REGION	Input	Nominal	Geographic region
CLASS	Input	Nominal	Customer loyalty status: tin, silver, gold, or platinum
ORGYN	Input	Binary	Organics purchased? 1 = Yes, 0 = No
AFFL	Input	Interval	Affluence grade on a scale from 1 to 30

Correlated Data

- A supermarket is beginning to offer a new line of organic products. The supermarket's management would like to determine which customers are likely to purchase these products.

- The supermarket has a customer loyalty program. As an initial buyer incentive plan, the supermarket provided coupons for the organic products to all their loyalty program participants and has now collected data that includes whether or not these customers have purchased any of the organic products.

Name	Model Role	Measurement Level	Description
CUSTID	ID	Nominal	Customer loyalty identification number
GENDER	Input	Nominal	M = male, F = female, U = unknown
DOB	Input	Interval	Date of birth
EDATE	Input	Unary	Date extracted from the daily sales data base
AGE	Input	Interval	Age, in years
AGEGRP1	Input	Nominal	Age group 1
AGEGRP2	Input	Nominal	Age group 2
TV_REG	Input	Nominal	Television region
NGROUP	Input	Nominal	Neighborhood group
NEIGHBORHOOD	Input	Nominal	Type of residential neighborhood
LCDATE	Input	Interval	Loyalty card application date
LTIME	Input	Interval	Time as loyalty card member
ORGANICS	Input	Interval	Number of organic products purchased
BILL	Input	Interval	Total amount spent
REGION	Input	Nominal	Geographic region
CLASS	Input	Nominal	Customer loyalty status: tin, silver, gold, or platinum
ORGYN	Input	Binary	Organics purchased? 1 = Yes, 0 = No
AFFL	Input	Interval	Affluence grade on a scale from 1 to 30

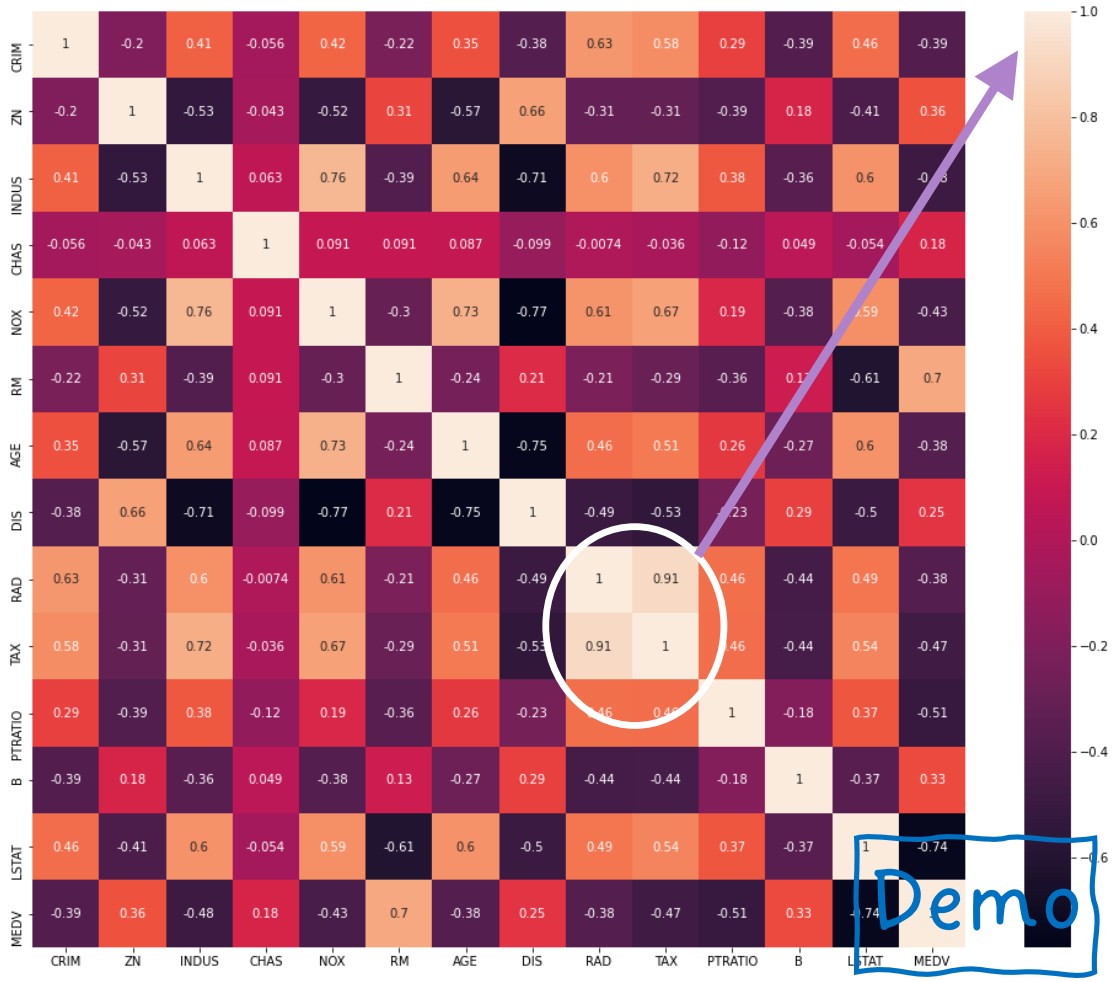
ORGANICS and ORGYN are highly correlated

```
import pandas as pd

#read in the data set
data = pd.read_csv('/Users/brendan.tierney/Dropbox/4-Datasets/boston-housing-dataset.csv')

corr_matrix=data.corr()
corr_matrix
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEI
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455621	-0.3883
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995	0.3604
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800	-0.4837
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929	0.1752
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879	-0.4273
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808	0.6953
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339	-0.3769
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496996	0.2499
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488676	-0.3816
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543993	-0.4685

```
import pandas as pd

#read in the data set
data = pd.read_csv('/Users/brendan.tierney/Dropbox/4-Datasets/boston-housing-dataset.csv')

corr_matrix=data.corr()
corr_matrix
```

```
import seaborn as sn

fig = plt.subplots(figsize=(17,14))
sn.heatmap(corr_matrix, annot=True)
```



"tax" and "rad" columns are highly correlated with value of 0.91(positive correlation).

Some of the features are negatively correlated and their correlation value is negatively high. Such as "lstat" vs "medv", "dis" vs "indus", "dis" vs "age".

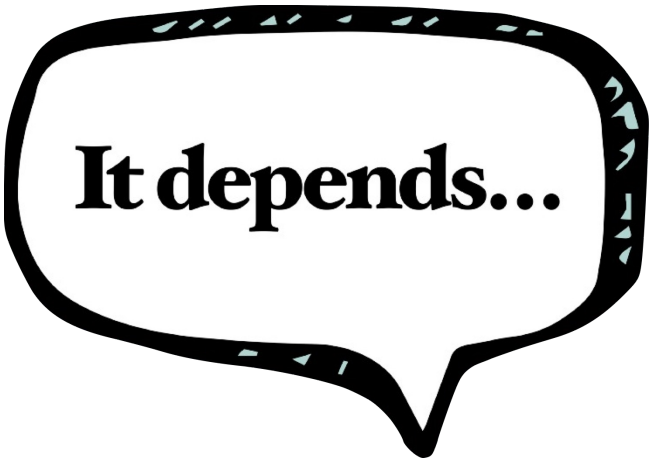
From the above observations, we can conclude that there is a hidden covariation between the tax rate and the index of accessibility to radial highways.

We can derive a relationship that if the house accessibility to highways is high then the full-value property-tax rate will also be higher.
This may sound like a genuine thing, because of the houses with high price generally nearer to market, good amenities, highways, etc.

Splitting Data, Sampling and Final Steps

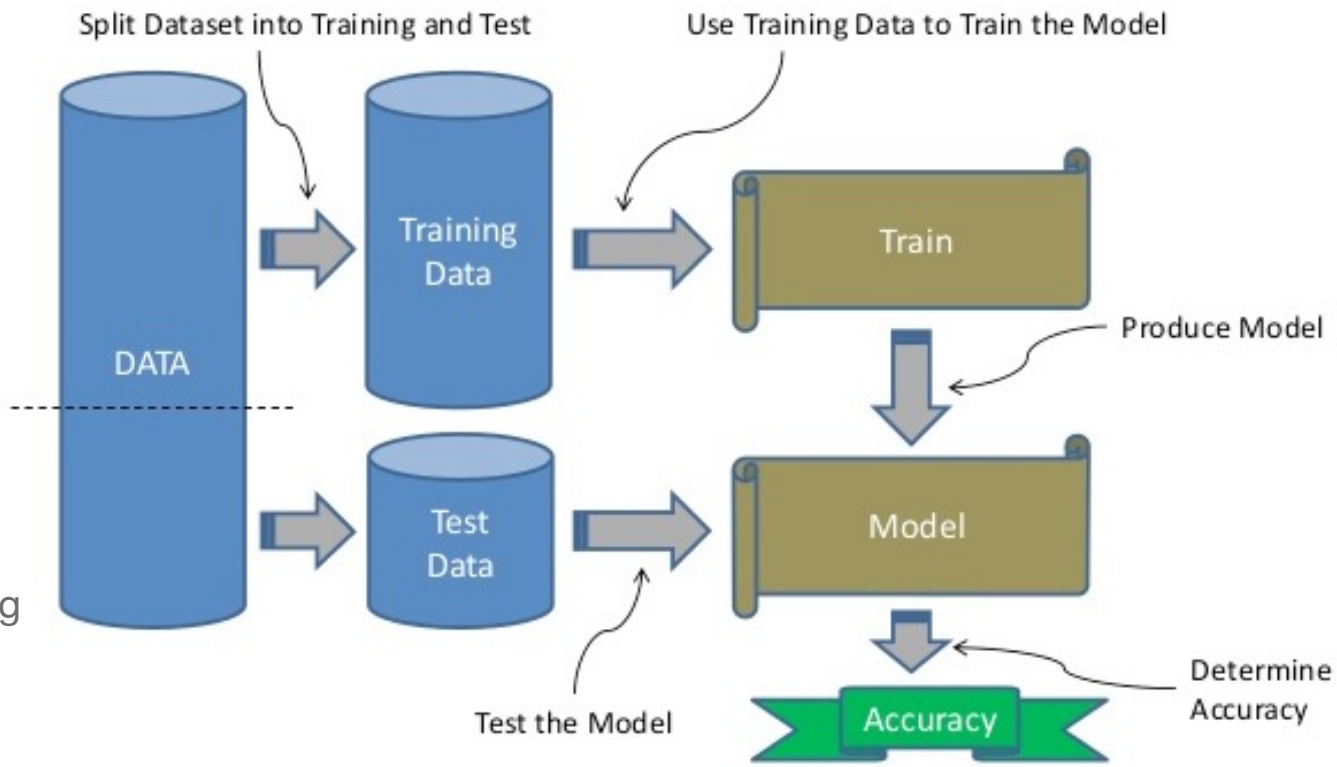
Are we nearly there?

- Yes we are :-)
- The finals steps are really dependent on what you will do next
    - What visualisations
    - What Tools
    - What Languages
    - What Algorithms
    - What APIs, web services, functions



For Supervised Machine Learning

- e.g. Classification, Regression
- Training Data set
  - Used to create the ML Model
  - Representative of entire data
- Testing Data set
  - Use for testing the ML Model
  - How good is it at making predictions
  - How many predictions are correct
  - How many predictions it got wrong
- 70-30 split
- 60-40 split



Example – train_test_split

- Will split the data set into 2 different dataframes
- Needs to have one of test_size or train_size defined, in range 0 to 1.0

```
from sklearn.model_selection import train_test_split

training_data, testing_data = train_test_split(df, test_size=0.2, random_state=25)
```

- 2 Data Sets are created
  - Test Data Set will contain 20% of records, as defined by parameter
  - Training Data Set will contain 80% of records, calculated by default from test_size parameter

- Limitation might result in Training and Test not having representative numbers of each Target value

Example – Using proportional sampling

- Stratified Sampling
- Divide the population into subgroups or into strata, and instances are sampled from each stratum to guarantee that the test set is representative of the entire population.
- Stratified sampling is different from simple random sampling, which involves the random selection of data from the entire population so that each possible sample is equally likely to occur.
- A random sample is taken from each stratum in direct proportion to the size of the stratum compared to the population, so each possible sample is equally likely to occur.

```
from sklearn.model_selection import train_test_split

training_data, testing_data = train_test_split(df, test_size=0.2, random_state=25, stratify=y)
```

Example – Using proportional sampling

```
#After the data is prepared, we can now divide the data into the Training and Test data
sets.
import random
random.seed(3434200)

# use Stratified sampling to divide the data
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=10, test_size = 0.2, random_state=18)

for train_index, test_index in split.split(df_new, df_new['y']):
   train_set = df_new.loc[train_index]
   test_set = df_new.loc[test_index]
```

Stratified Sampling ensure proportional representation in each sub-group

```
#Calculate the distributions for each of the Target variables. We have an imbalanced data set.

df['y'].value_counts() # dataset is imbalanced with majority of class label as "no".

no      36548
yes      4640

df['y'].value_counts()/len(df)   #calculate percentages

no      0.887346
yes     0.112654

#Check the sizes of the Training and Test data sets.
train_set['y'].value_counts()
0    29238
1     3712

train_set['y'].value_counts()/len(train_set)
0    0.887344
1    0.112656

#Check the Test data set
test_set['y'].value_counts()
0    7310
1     928

test_set['y'].value_counts()/len(test_set)
0    0.887351
1    0.112649
```

Stratified Sampling ensure proportional representation in each sub-group

For Un-Supervised Machine Learning

- e.g. Clustering, Association Rules Analysis, Text Mining, Forecasting, etc

- Nothing more to do
- All done
- Just pass the data set (dataframe) into the algorithms

Imbalanced Data Sets

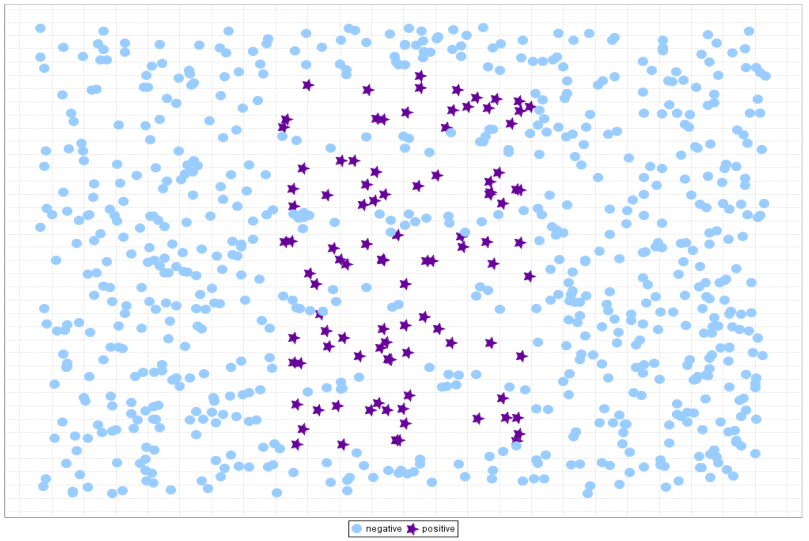
For Classification & Regression Algorithms

- For Classification & Regression Algorithms
- They look to learn how the attributes and their values contribute towards the Target attribute
- Need to have enough examples or records of each value of Target attribute

- Split the available data into a *training set* and a *test set*

- Train the classifier in the training set and evaluate based on the test set
- A couple of drawbacks
  - We may not have enough data
  - We may happen upon an *unfortunate split*

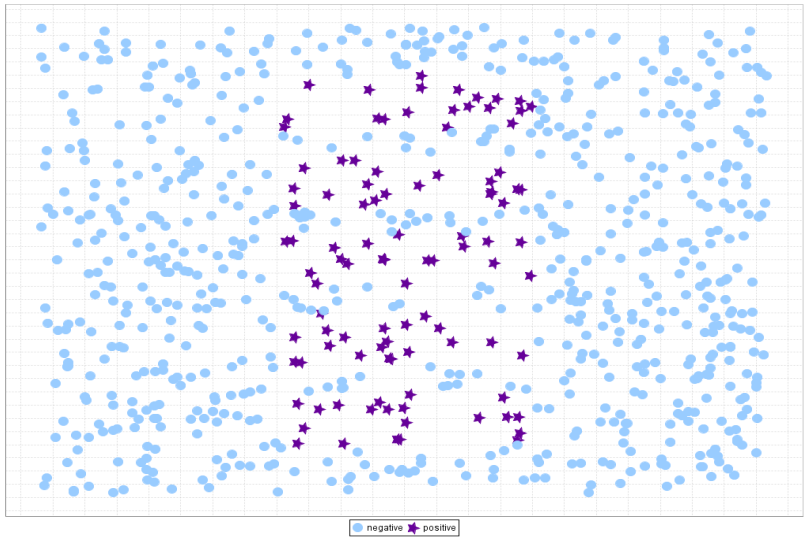
Imbalanced Data Sets



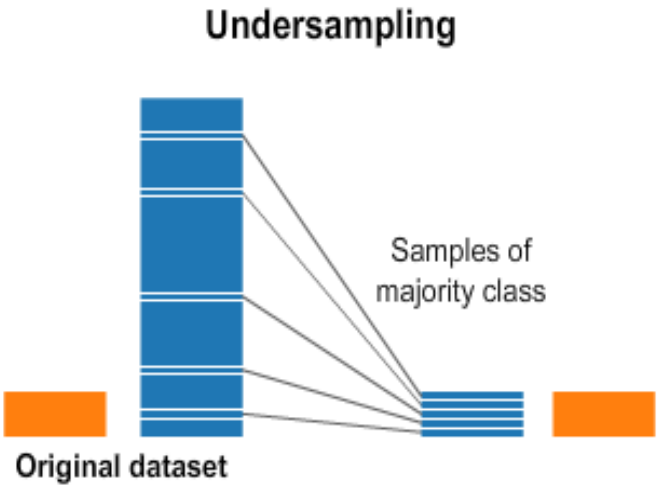
- One Target value has more than the other
- Not a problem if difference is small
- If we use the data => a biased model
  - It will be good at predicting one of the Target values
  - And rubbish at predicting the other value

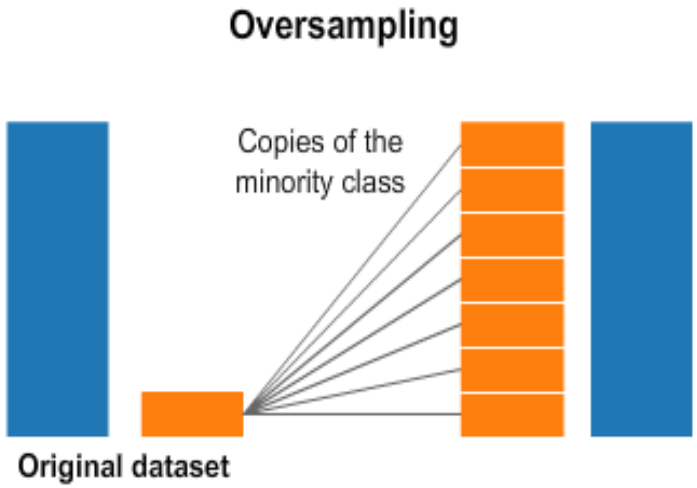
Imbalanced Data Sets

- Typically, One Target value has more than the other
  - Very common in real world
  - Typically want to predict the smaller “thing”
- Not a problem if differences in Target attribute (in number of records) is small
- If we use the Imbalanced data set => a biased model
  - It will be good at predicting one of the Target values (most number of records)
  - And rubbish at predicting the other value (lower number of records



How to manage Imbalanced Data Sets

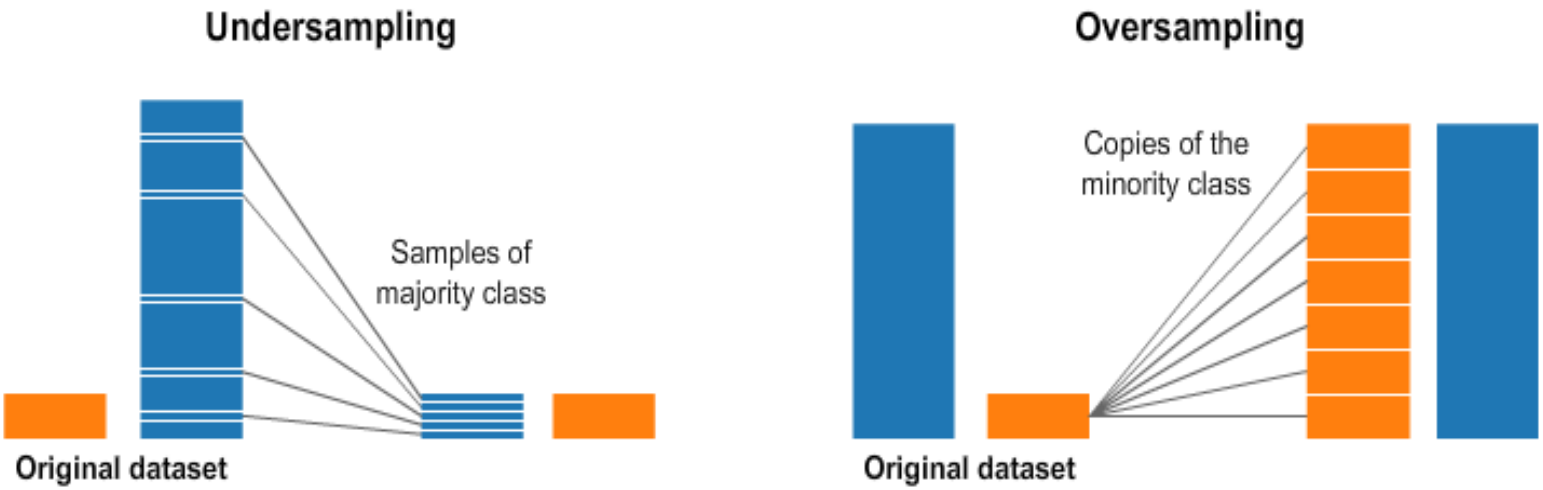




- Under Sample
  - Remove records from data set
  - Random (but how random is it really)
    - How can you ensure representative samples remain.
    - May end up removing important cases

Issues with reducing the size of data set

How to manage Imbalanced Data Sets



- Over Sample
  - Repeat/Duplicate records - Quick and Easy, by has Limitations (biases, no-variances, etc)
  - SMOTE: Synthetic Minority Over-sampling Technique
  - take a sample from the dataset, and consider its *k* nearest neighbors (in feature space). To create a synthetic data point, take the vector between one of those *k* neighbors, and the current data point. Multiply this vector by a random number *x* which lies between 0, and 1. Add this to the current data point to create the new, synthetic data point. Allows for possible data variations, within current ranges.

```
#Import the data set

import pandas as pd
columns = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = pd.read_csv('/Users/brendan.tierney/Dropbox/4-Datasets/pima-indians-diabetes.csv', names=columns)
data.head()
```

	preg	plas	pres	skin	test	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

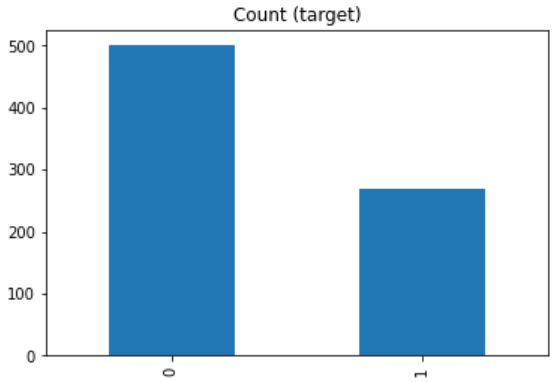
```
data.shape
```

```
(768, 9)
```

```
data['class'].value_counts()
```

```
0    500
1    268
Name: class, dtype: int64
```

```
#print bar chart
data['class'].value_counts().plot(kind='bar', title='Count (target)');
```



Down Sampling - Majority Class - Using Random Sampling

```
count_class_0, count_class_1 = data['class'].value_counts()

# Divide by class
df_class_0 = data[data['class'] == 0] #majority class
df_class_1 = data[data['class'] == 1] #minority class

print('Class = 0 ', df_class_0.shape[0])
print('Class = 1 ', df_class_1.shape[0])
```

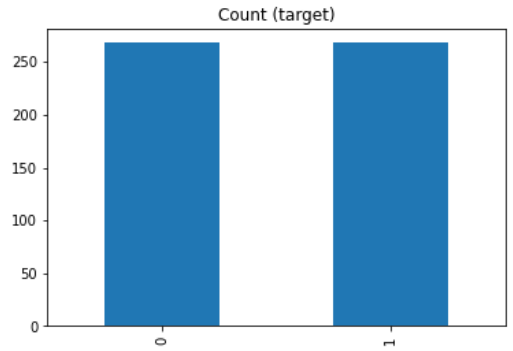
```
Class = 0  500
Class = 1  268
```

```
# Sample Majority class (y=0, to have same number of records as minority calls (y=1)
df_class_0_under = df_class_0.sample(count_class_1)

df_class_0_under.shape
```

```
(268, 9)
```

```
# join the dataframes containing y=1 and y=0
df_test_under = pd.concat([df_class_0_under, df_class_1])

print('Random under-sampling:')
print(df_test_under['class'].value_counts())
print("Num records = ", df_test_under.shape[0])

df_test_under['class'].value_counts().plot(kind='bar', title='Count (target)');
```

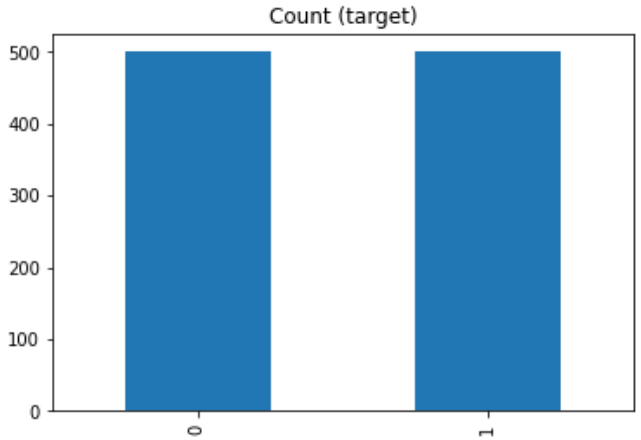
```
Random under-sampling:
0    268
1    268
Name: class, dtype: int64
Num records =  536
```



Over sampling the minority call y=0 using SMOTE

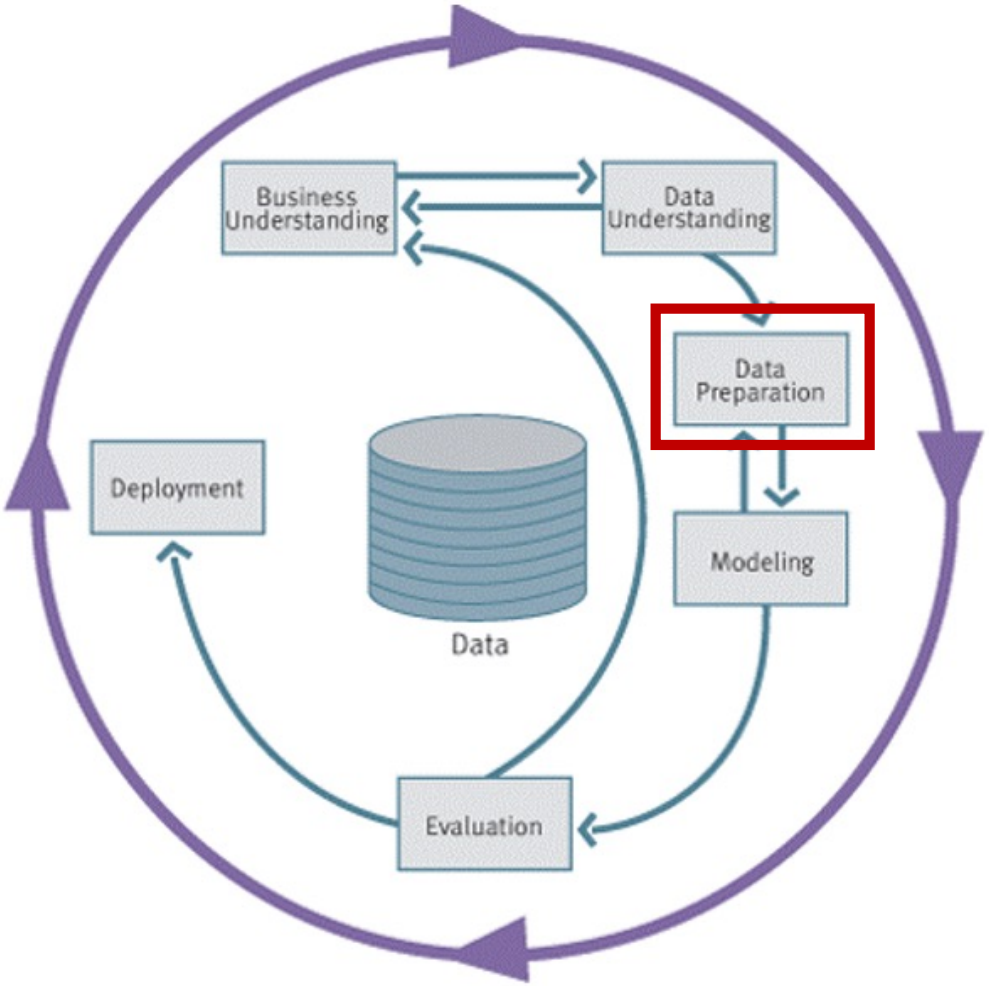
```
from imblearn.over_sampling import SMOTE

print(data['class'].value_counts())
X = data.drop('class', axis=1)
Y = data['class']

sm = SMOTE(random_state=42)
X_res, Y_res = sm.fit_resample(X, Y)

df_smote_over = pd.concat([pd.DataFrame(X_res), pd.DataFrame(Y_res, columns=['class'])], axis=1)

print('SMOTE over-sampling:')
print(df_smote_over['class'].value_counts())

df_smote_over['class'].value_counts().plot(kind='bar', title='Count (target)');
```

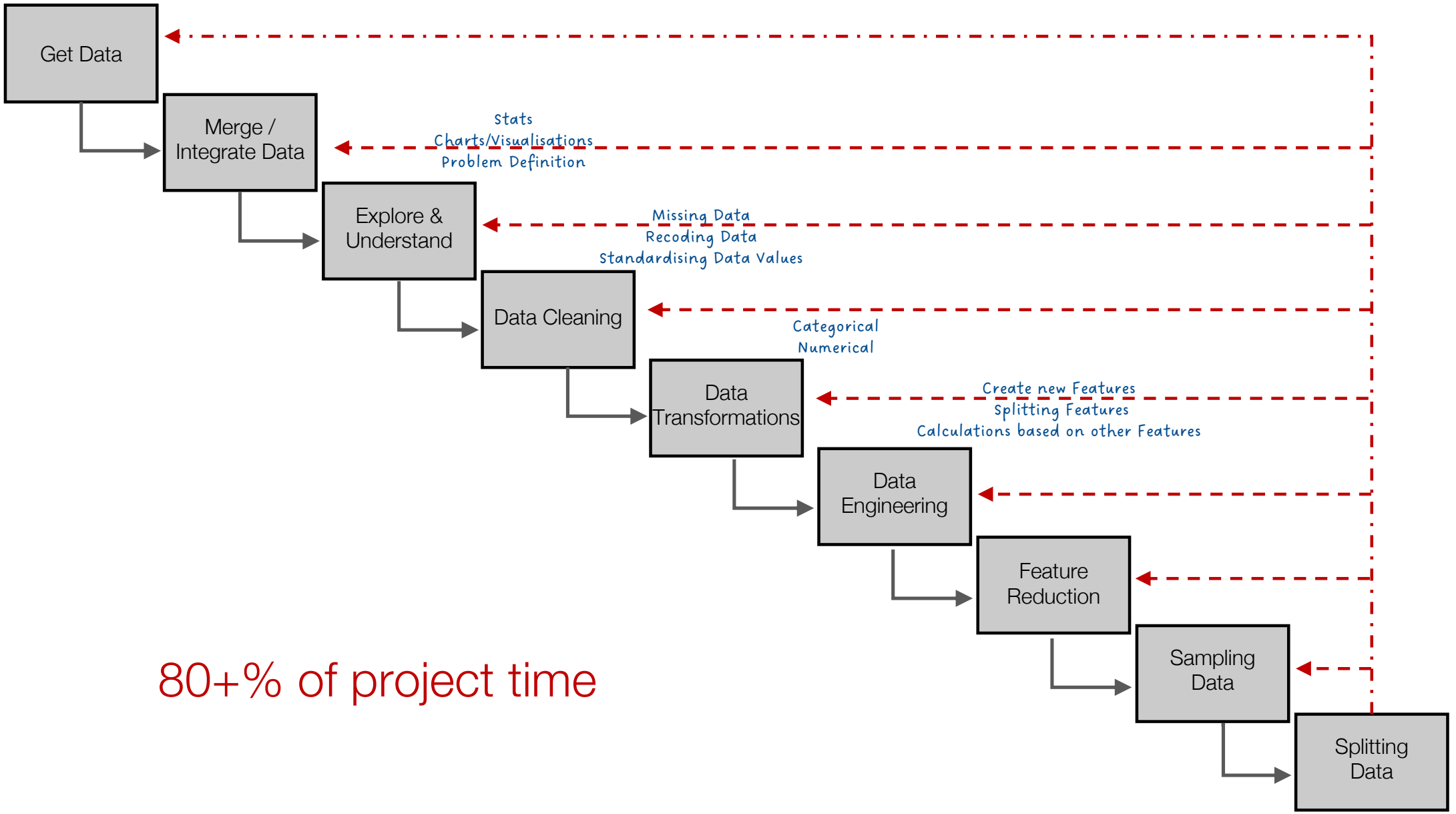
```
0    500
1    268
Name: class, dtype: int64
SMOTE over-sampling:
0    500
1    500
Name: class, dtype: int64
```



Demo

Iterative in Nature


Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Data



80+% of project time

Your Toolbox

- Lots of possible tasks to do
- Sometimes it can be a little confusing what you need to do
- Keep it Simple. Iterate. Built upon. Learn
- Start simple and learn/experiment
- We haven’t covered all possibilities
- But we have covered more than enough to get started
- Build and Learn from each example

Any Questions ?

What Now/Next ?

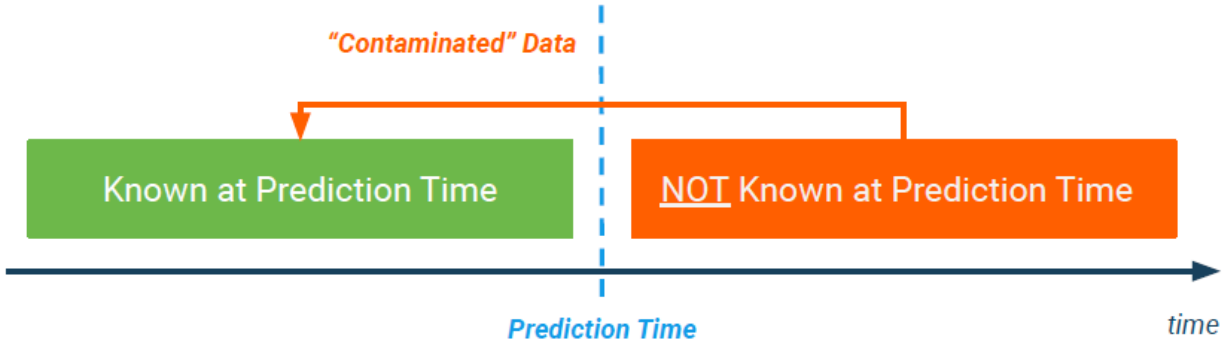
One more thing…

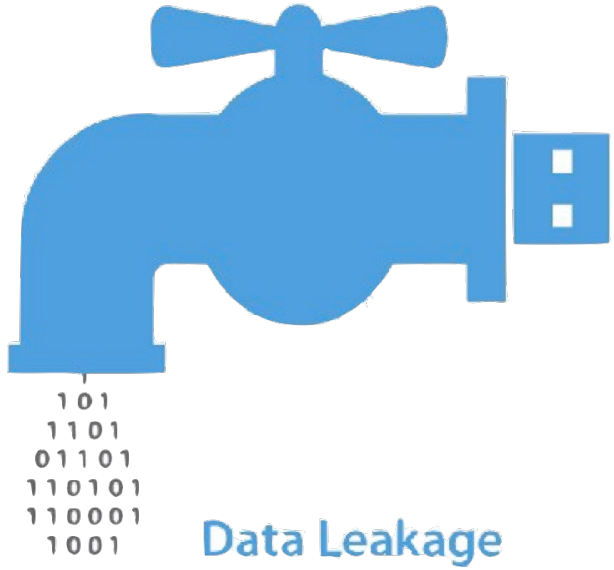
Data Leakage

- Data leakage is when information from outside the training dataset is used to create the model.
- Your Data Set contains features which were created after the Event your data set represents.
- Your Data Set contains records that represent an Event.
  - Details of Customer leading up to them Buying a Product
  - Customer details leading up to them being involved in a Marketing Campaign
- Examples
  - How long did it take for the Customer to Buy their next product
  - Did the customer Buy the product
  - How long were they on the Call
    - the longer the call the more likely (correlated) to buying the product
    - The shorter the call means they didn't buy

Data Leakage

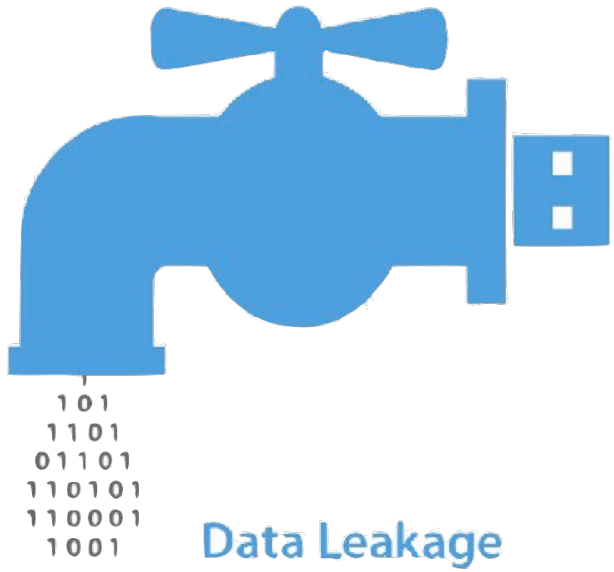
- Data Leakage can be applied to many different types of problems
- For Data Science/Mining etc it refers to data quality issue
- It can affect the Analysis, Data Understanding, Visualisations, Models, Interoperation of results, etc
- (Target) Data Leakage happens if you train your model on information that would NOT be available at the point in time when you use the model in production to score newly collected data





Data Leakage

- Less common if you are building the data set yourself
- Can be very common in publicly available Data Sets.
- Always check the Meta-Data / Data Dictionary.
  - And any other information provided for the Data Set.
- Remove these features from the Data Set.
- These features Might appear as Strong indicators in Machine Learning models.



Data Leakage

- For example, imagine you want to predict who will get sick with pneumonia.

got_pneumonia	age	weight	male	took_antibiotic_medicine	...
False	65	100	False	False	...
False	72	130	True	False	...
True	58	100	False	True	...

Data Leakage - Example

- Bank Marketing Data Set - https://archive.ics.uci.edu/ml/datasets/bank+marketing

1 - age (numeric)
2 - job : t ype of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown')
related with the last contact of the current campaign:
8 - contact: contact communication type (categorical: 'cellular','telephone')
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - duration: last contact duration, in seconds (numeric).
other attributes:
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):
21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

Do we have an example of Data Leakage in this Data Set?

Example the information and come up with a possible features

Data Leakage - Example

- Bank Marketing Data Set - https://archive.ics.uci.edu/ml/datasets/bank+marketing

1 - age (numeric)
2 - job : t ype of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown')
related with the last contact of the current campaign:
8 - contact: contact communication type (categorical: 'cellular','telephone')
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
other attributes:
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):
21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

```
df = df.drop('duration', axis=1)
```

Your Toolbox

- Lots of possible tasks to do
- Sometimes it can be a little confusing what you need to do
- Keep it Simple. Iterate. Built upon. Learn
- Start simple and learn/experiment
- We haven’t covered all possibilities
- But we have covered more than enough to get started
- Build and Learn from each example

Any Questions ?

What Now/Next ?