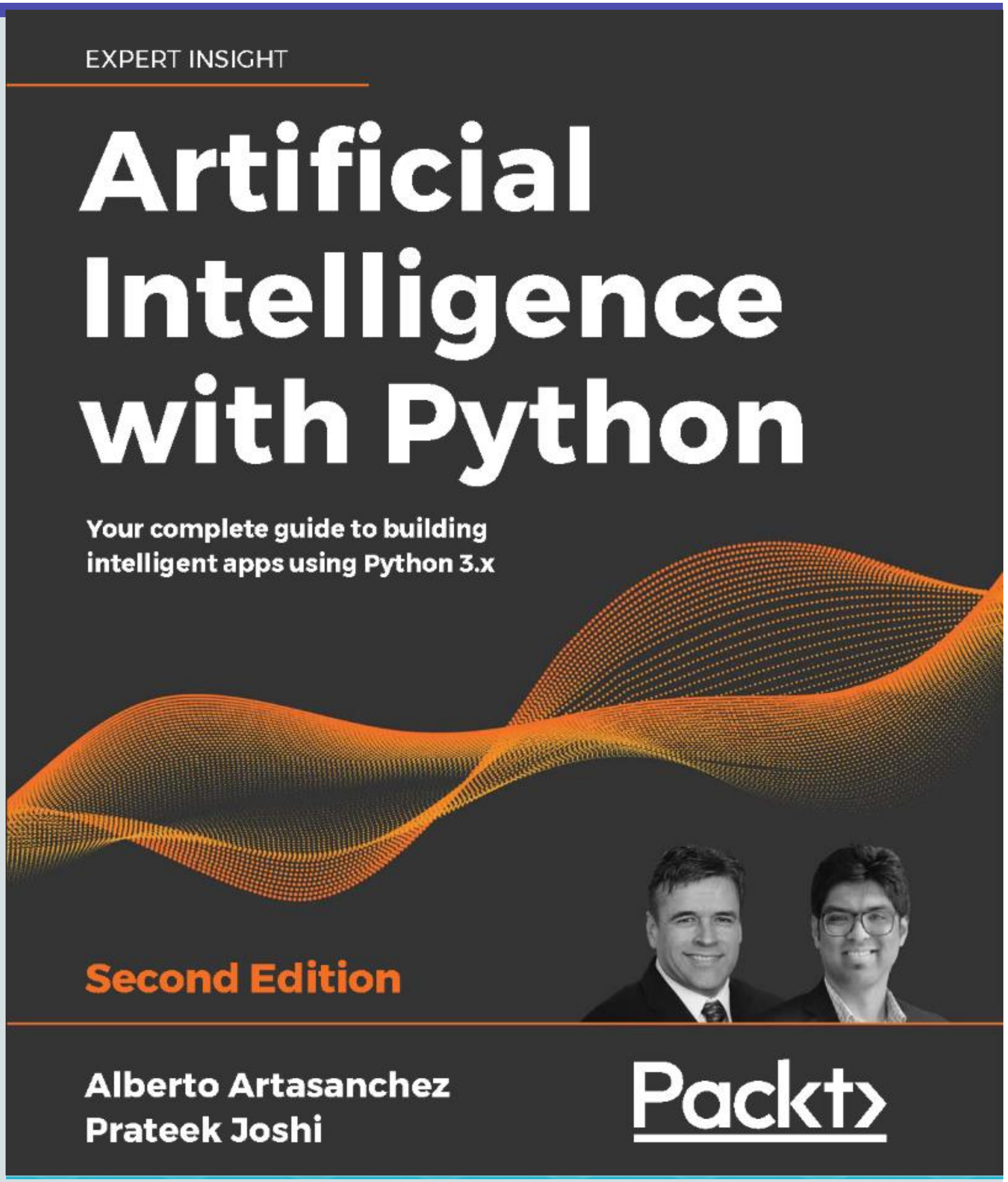
University of South-Eastern Norway
School of Business

Lecture 2: Machine Learning Pipelines and Feature Selection

Sinuo Wu
Course: AI for Business Applications (AI3000)



Quiz

- Enter room number or Scan QR code

Today

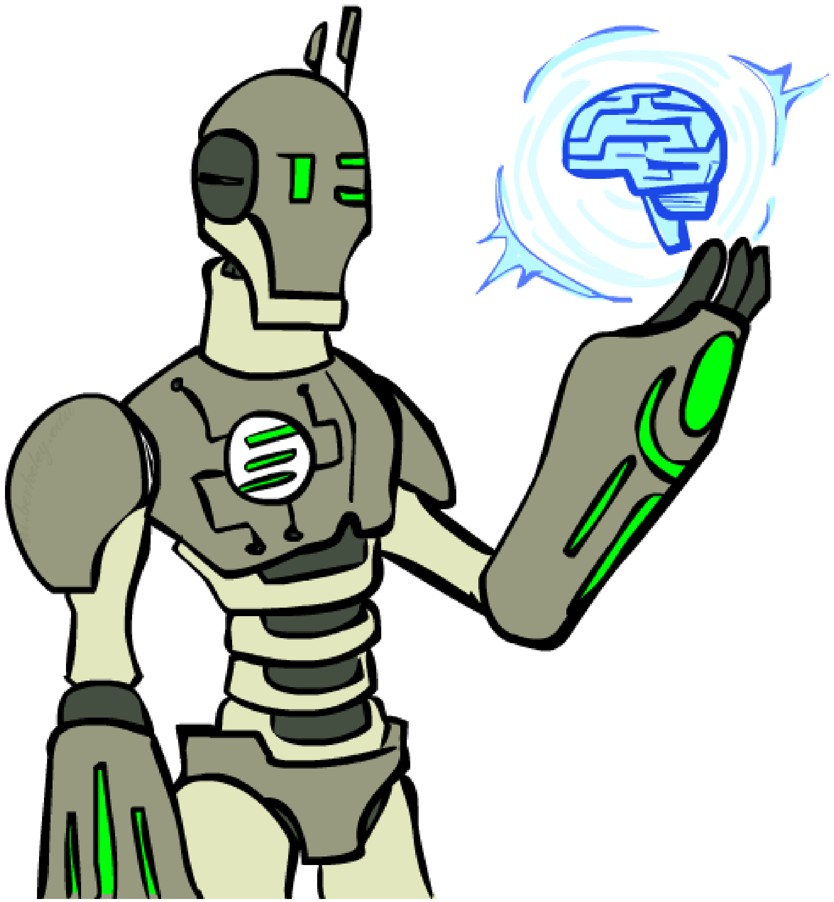
- **Machine Learning Pipelines**

Problem definition

Data preparation

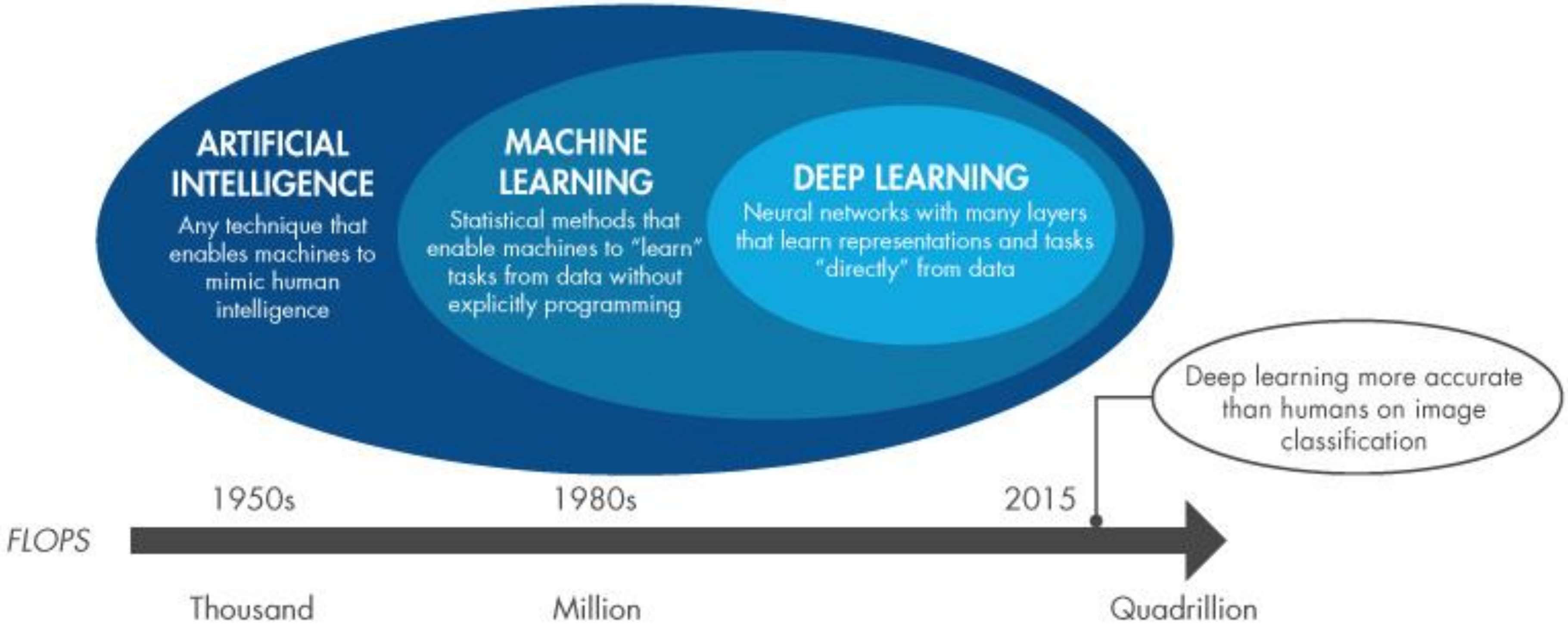
Model training

- **Feature Selection**
- **Feature Engineering**



Machine Learning Pipeline

Machine Learning



Why ML

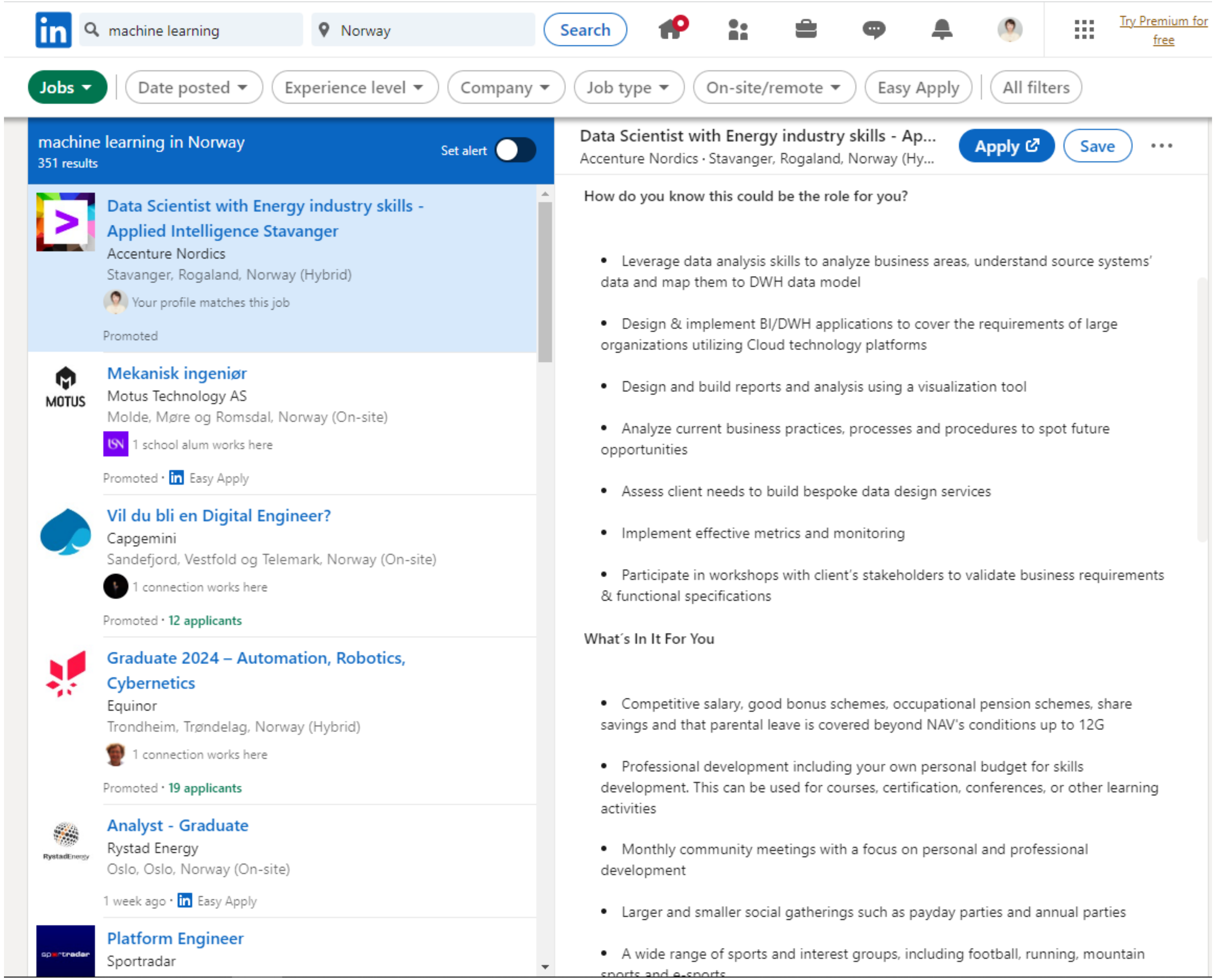
Develop strengths in the job market

If AI can replace some basic coding jobs, then what can not be replaced?

The one which creates complex AI models!

Machine Learning

Deep Learning



ML Pipeline

- A machine learning pipeline is a way to codify and automate the workflow it takes to produce a machine learning model.
- Model training is only a small piece of the machine learning process.
- Creating successful machine learning systems involves a lot more than choosing between a random forest model and a support vector machine model.

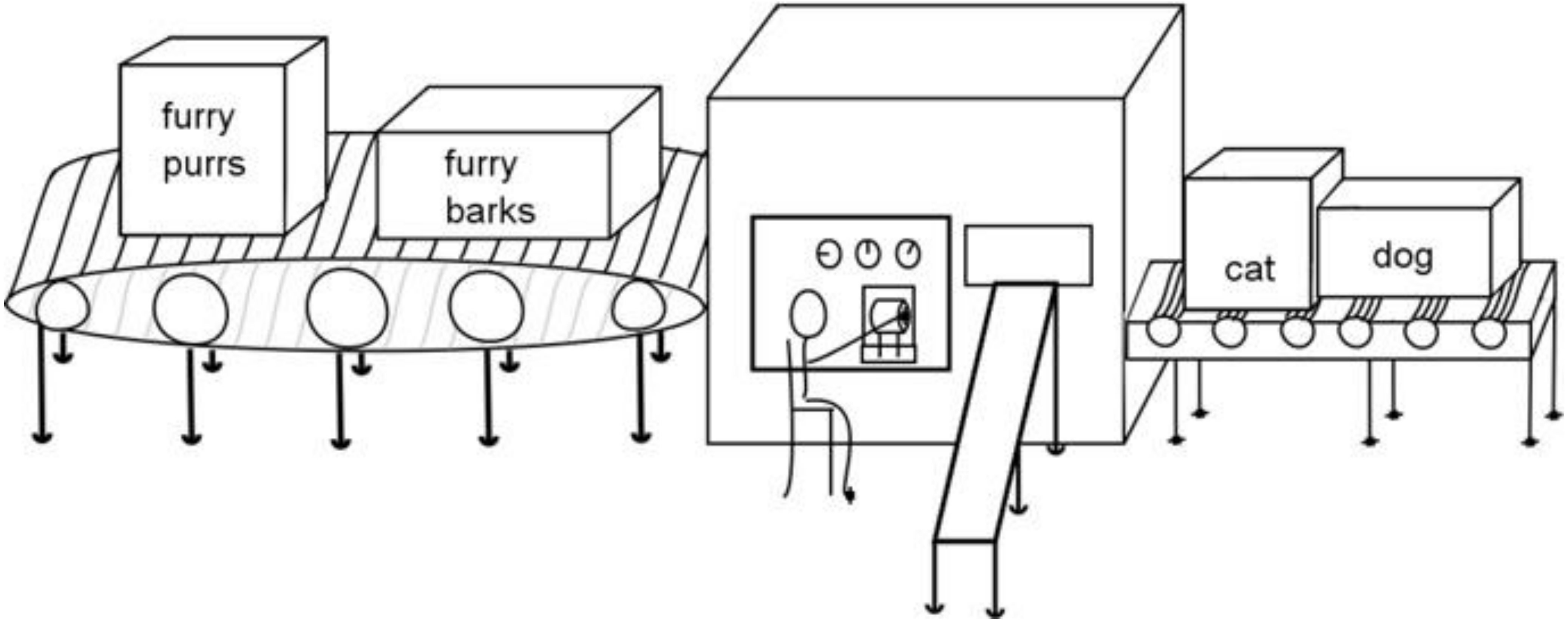


Figure 1.2 Descriptions go in. Categories or other values come out.
We can adjust the machine to improve the relationship between the inputs and outputs.

Tools that data pipelines commonly leverage

- Hadoop
- Spark
- Spark Streaming
- Kafka
- Azure
- AWS
- Google Cloud Platform
- R
- SAS
- Databricks

Python

Steps in a ML pipeline

Real-time? Expected traffic? How often it needs to be updated?

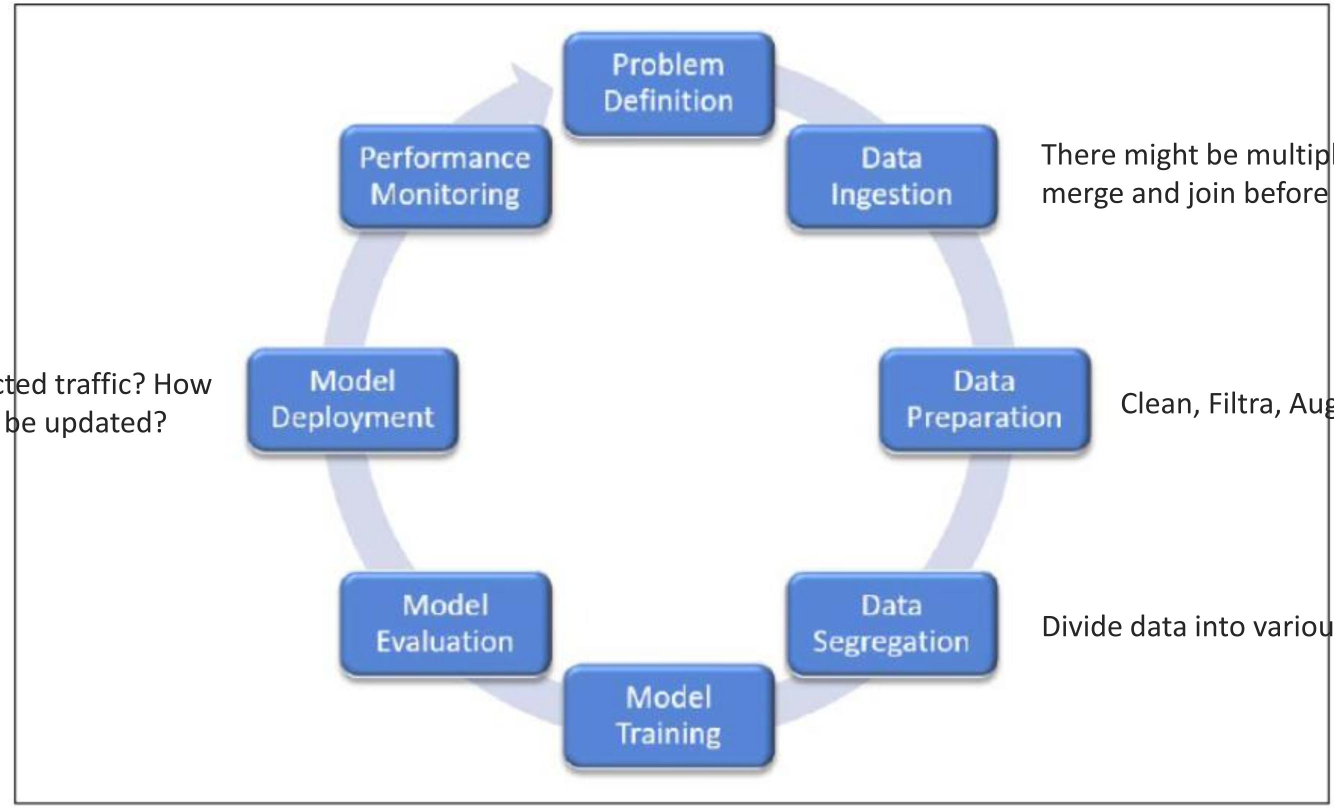


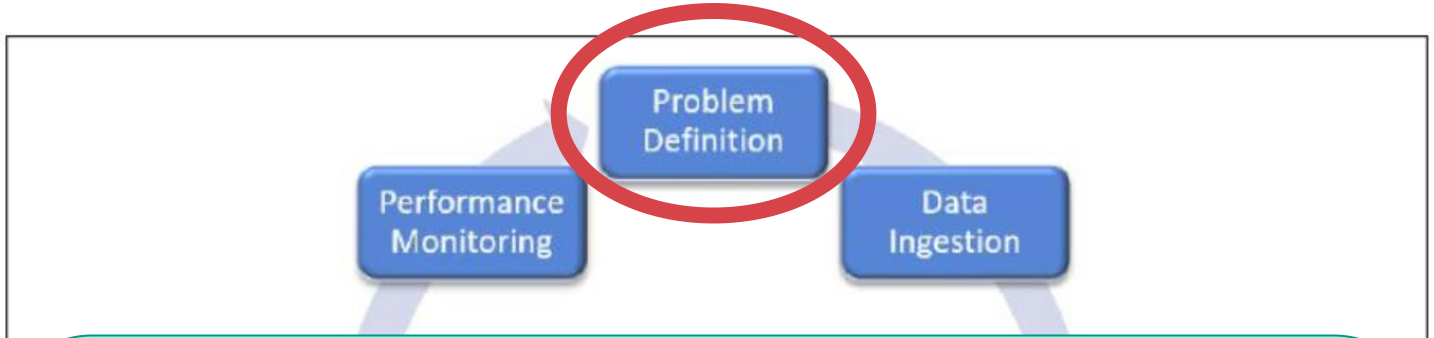
Figure 1: The machine learning pipeline

There might be multiple sources that we need to merge and join before feeding them to the model.

Clean, Filtra, Augment, etc.

Divide data into various categories for difference purpose

Problem Definition



"Bob spent years planning, executing, and optimizing how to conquer a hill. Unfortunately, it turned out to be the wrong hill."

Figure 1: The machine learning pipeline

Problem definition

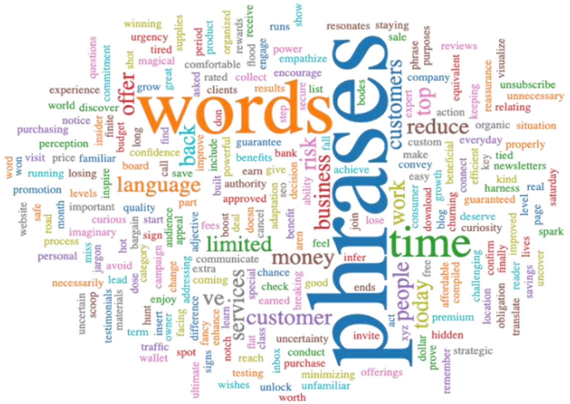
- For a given loan, will it default or not?
- When will the loan default?
- How much money will be received from a given loan?
- What will be the profit made on a given loan?
- What will be the profit made on a given loan without using disallowed input features?

Data Ingestion

- **What is data** → A collection of fact

2016→ 50 billion IoT
By 2020 → 1.7 megabytes per person per second
https://analyticsweek.com/big-data-facts/

https://www.wnycstudios.org/podcasts/radiolab/articles/love-n

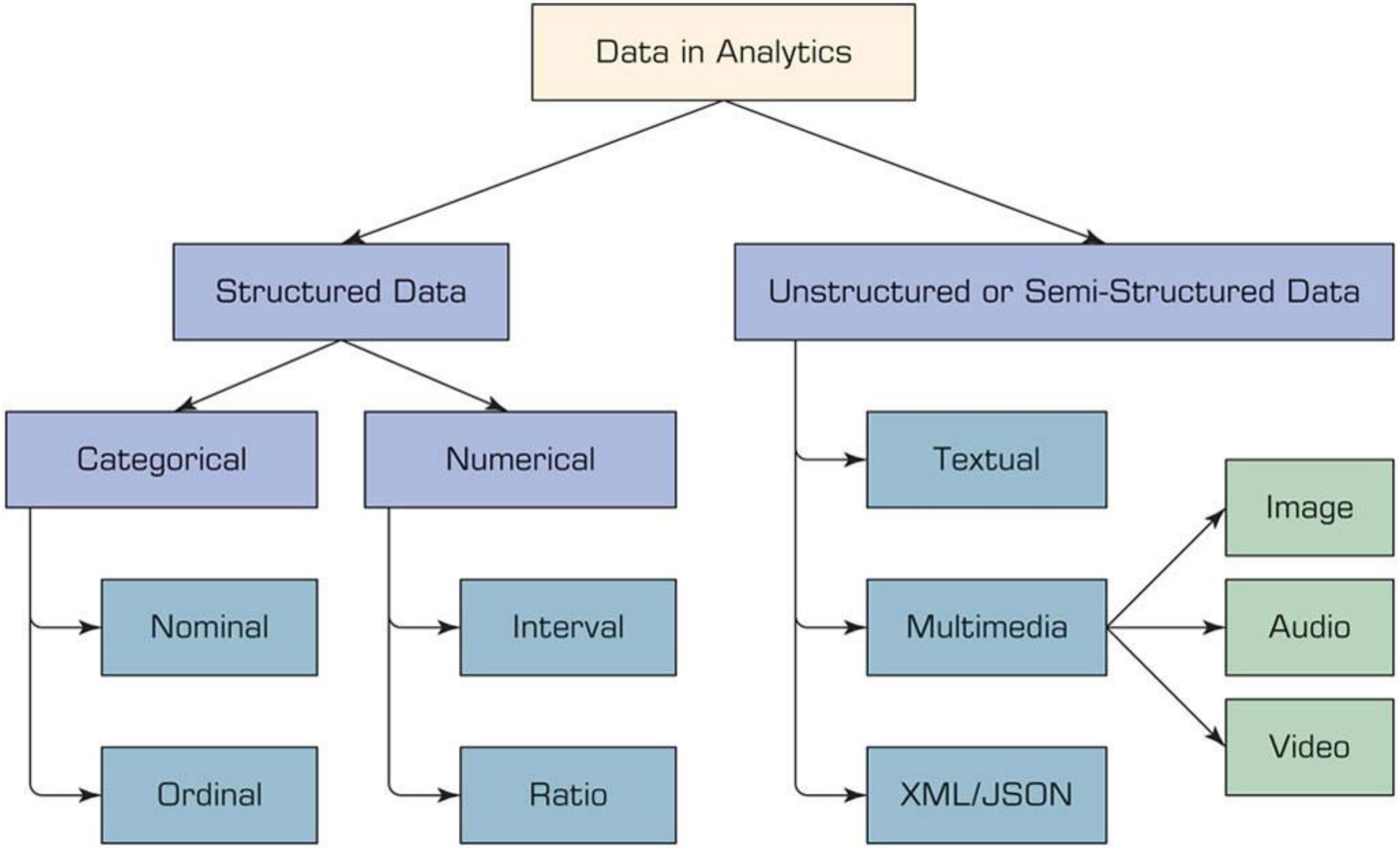


https://www.wordstream.com/

https://twitter.com/ibm_in/status/756097248149184513

Data Ingestion

- **Taxonomy of Data**



Data Ingestion

- **Get the right data source!**
    - What data provider or vendor should we use? Can they be trusted?
    - How will it be ingested? Hadoop, Impala, Spark, just Python, and so on?
    - Should it be stored as a file or in a database?
    - What type of database? Traditional RDBMS, NoSQL, graph.
    - Should it even be stored? If we have a real-time feed into the pipeline, it might not even be necessary or efficient to store the input.
    - What format should the input be? Parquet, JSON, CSV.

Scikit-learn datasets

Toy Dataset

scikit-learn comes with a few small standard datasets that do not require to download any file from some external website.

They can be loaded using the following functions:

`load_boston`(*[, return_X_y])	DEPRECATED: `load_boston` is deprecated in 1.0 and will be removed in 1.2.
`load_iris`(*[, return_X_y, as_frame])	Load and return the iris dataset (classification).
`load_diabetes`(*[, return_X_y, as_frame, scaled])	Load and return the diabetes dataset (regression).
`load_digits`(*[, n_class, return_X_y, as_frame])	Load and return the digits dataset (classification).
`load_linnerud`(*[, return_X_y, as_frame])	Load and return the physical exercise Linnerud dataset.
`load_wine`(*[, return_X_y, as_frame])	Load and return the wine dataset (classification).
`load_breast_cancer`(*[, return_X_y, as_frame])	Load and return the breast cancer wisconsin dataset (classification).

```
from sklearn.datasets import load_iris
data = load_iris(return_X_y=True)
print ("data:\n", data)
```

Scikit-learn datasets

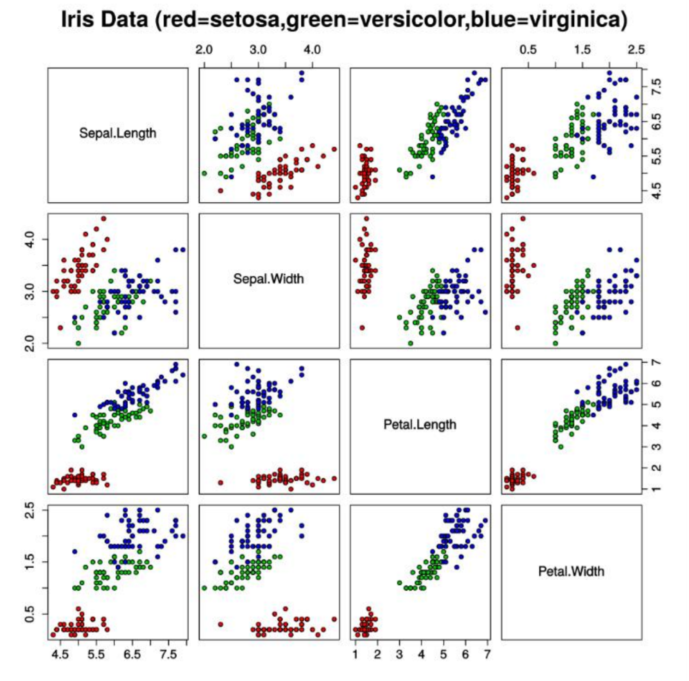
Real world datasets

scikit-learn provides tools to load larger datasets, downloading them if necessary.

They can be loaded using the following functions:

`fetch_olivetti_faces`(*[, data_home, ...])	Load the Olivetti faces data-set from AT&T (classification).
`fetch_20newsgroups`(*[, data_home, subset, ...])	Load the filenames and data from the 20 newsgroups dataset (classification).
`fetch_20newsgroups_vectorized`(*[, subset, ...])	Load and vectorize the 20 newsgroups dataset (classification).
`fetch_lfw_people`(*[, data_home, funneled, ...])	Load the Labeled Faces in the Wild (LFW) people dataset (classification).
`fetch_lfw_pairs`(*[, subset, data_home, ...])	Load the Labeled Faces in the Wild (LFW) pairs dataset (classification).
`fetch_covtype`(*[, data_home, ...])	Load the covertype dataset (classification).
`fetch_rcv1`(*[, data_home, subset, ...])	Load the RCV1 multilabel dataset (classification).
`fetch_kddcup99`(*[, subset, data_home, ...])	Load the kddcup99 dataset (classification).
`fetch_california_housing`(*[, data_home, ...])	Load the California housing dataset (regression).

Fisher’s Iris Dataset



Data Ingestion examples

- Stock prices,
  - the price of the stock the previous day
  - interest rates,
  - company earnings,
  - news headlines.

- Restaurant daily sales
  - the previous day's sales
  - Day of the week,
  - holiday or not holiday,
  - rain or no rain,
  - daily foot traffic

Data Preparation

Missing Values!

- Data cleansing
- Filtration
- Aggregation
- Augmentation
- Consolidation Storage

Missing Values

N/A or 0000

- Do nothing
- Imputation using **median values**
- Imputation using **the most frequent value**

What if we want to replace the missing value with a constant value instead of the median value? Try it!

Imputation

Using median values

```
import pandas as pd

data = pd.read_csv("train.csv", nrows=10)
X = data.iloc[:, 0:20]
Y = data.iloc[:, -1]

data_new = data.fillna(0)
data_new = data.fillna(data.median())
print(data)
print(data_new)
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
0	842	0	2.2	0	1	0	NaN
1	1021	1	0.5	1	0	1	53.0
2	563	1	0.5	1	2	1	41.0
3	615	1	2.5	0	0	0	10.0
4	1821	1	1.2	0	13	1	44.0
5	1859	0	0.5	1	3	0	22.0
6	1821	0	1.7	0	4	1	10.0
7	1954	0	0.5	1	0	0	24.0
8	1445	1	0.5	0	0	0	53.0
9	509	1	0.6	1	2	1	9.0

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
0	842	0	2.2	0	1	0	24.0
1	1021	1	0.5	1	0	1	53.0
2	563	1	0.5	1	2	1	41.0
3	615	1	2.5	0	0	0	10.0
4	1821	1	1.2	0	13	1	44.0
5	1859	0	0.5	1	3	0	22.0
6	1821	0	1.7	0	4	1	10.0
7	1954	0	0.5	1	0	0	24.0
8	1445	1	0.5	0	0	0	53.0
9	509	1	0.6	1	2	1	9.0

Imputation

Using common values

```
    index   color
0       0   green
1       1  yellow
2       2     NaN
3       3     red
4       4  purple
5       5     red
6       6     red
7       7  purple
8       8     NaN
9       9     red
10     10  yellow
11     11     NaN
12     12   black
13     13   white
```

```
    index   color
0       0   green
1       1  yellow
2       2     red
3       3     red
4       4  purple
5       5     red
6       6     red
7       7  purple
8       8     red
9       9     red
10     10  yellow
11     11     red
12     12   black
13     13   white
```

```python
import pandas as pd

data = pd.read_csv("dataset.csv")
print(data)

data["color"].fillna(data["color"].value_counts().idxmax(), inplace=True)
print(data)
```

Other issues with data

- Duplicate records or values

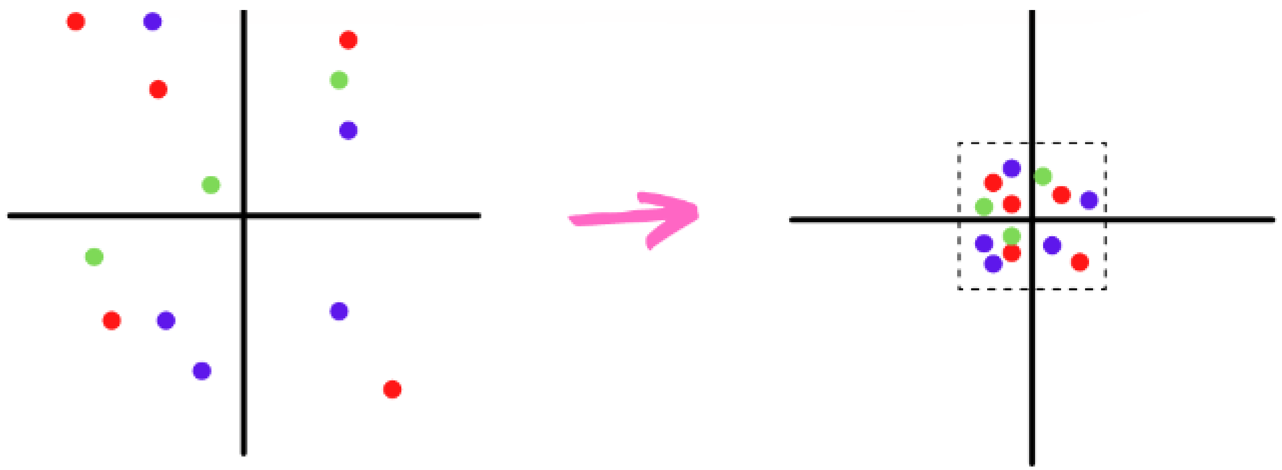
- Example:
  - Same person with multiple email addresses

- ++Clean the data

Other issues with data

- Duplicate records or values
- Feature scaling

- 15 kg → 0
- 100 kg → 1



Other issues with data

- Duplicate records or values
- Feature scaling
- Inconsistent values

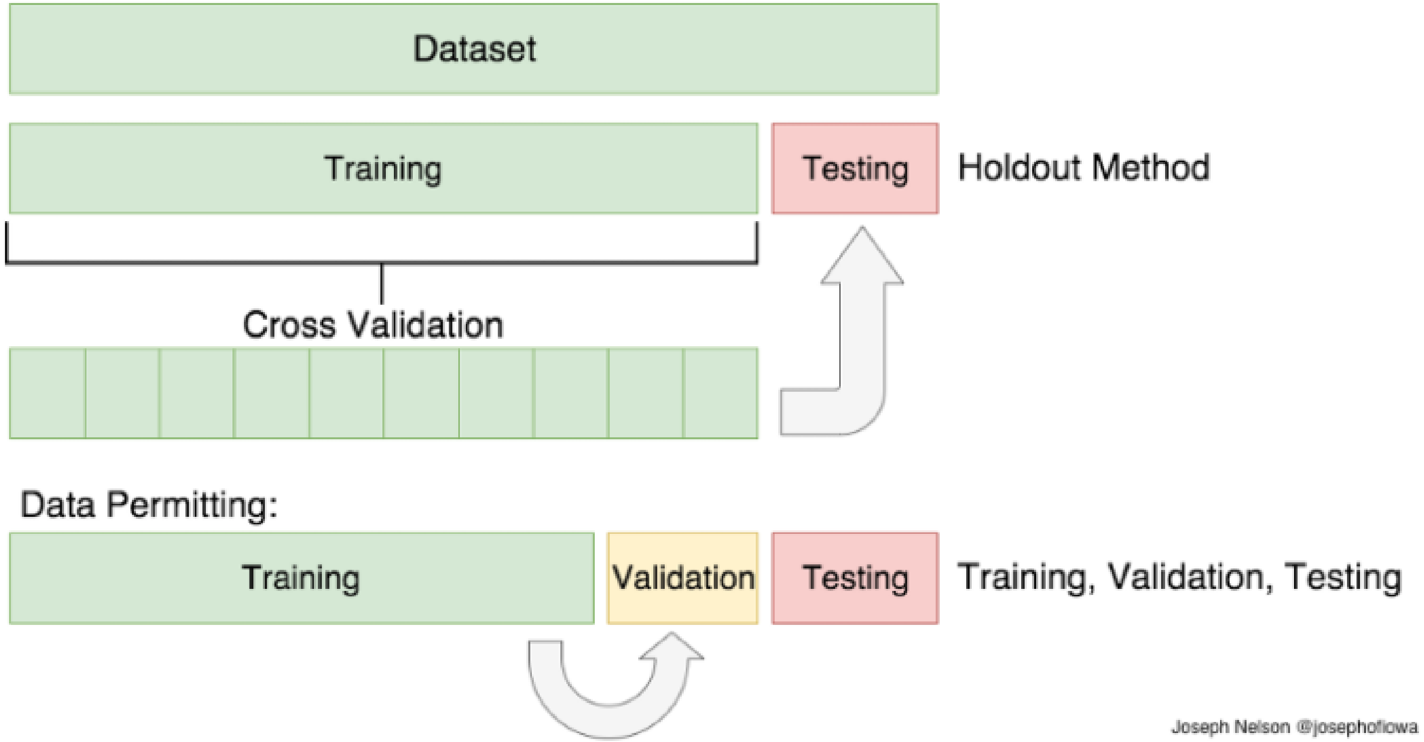
Fifth Avenue
Fifth Ave
Fifth Av
Fifth Av.

Other issues with data

- Duplicate records or values
- Feature scaling
- Inconsistent values
- Inconsistent date formatting

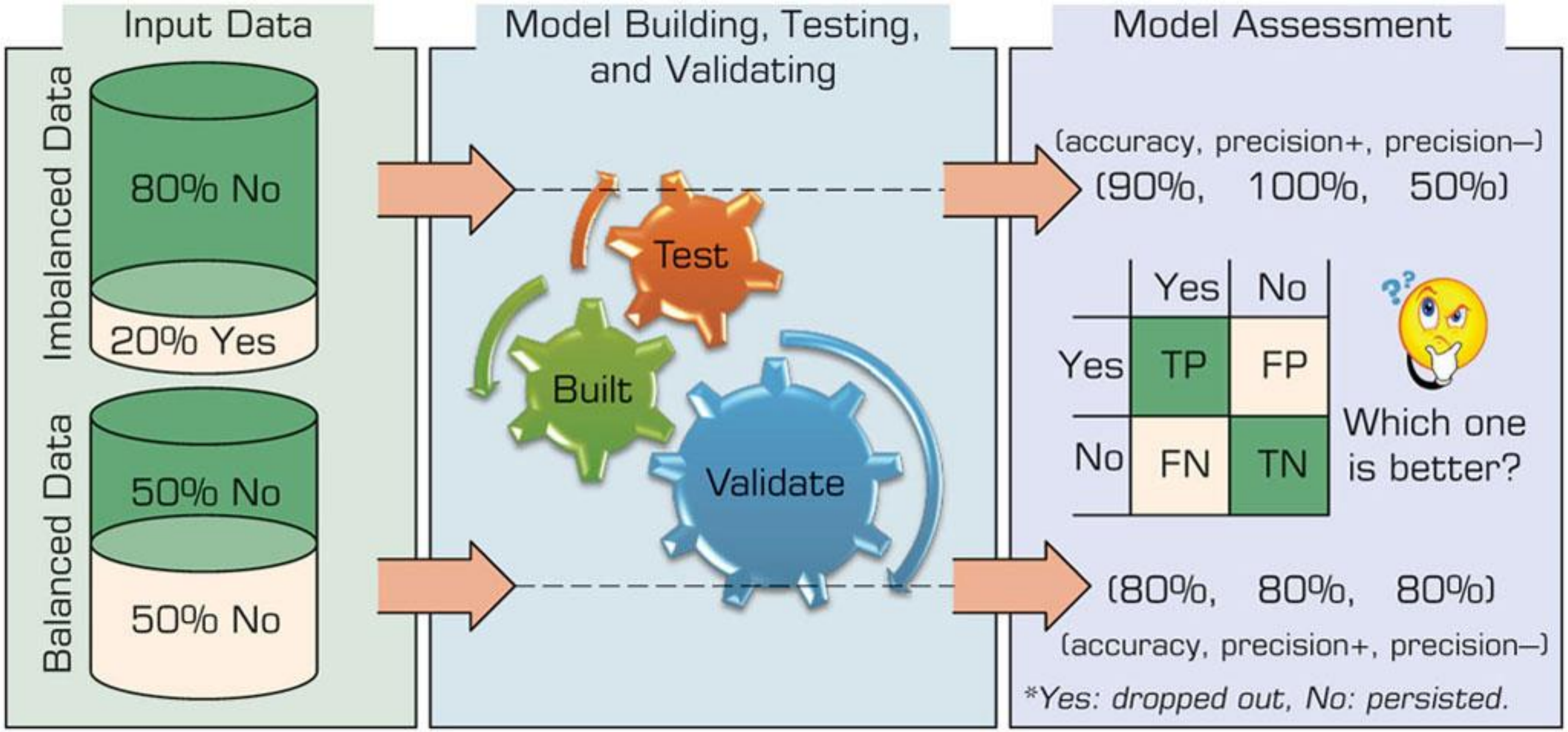
11/1/2016
11/01/2016
11/1/16
Nov 1 16
November 1st, 2016

Data Segregation



Training, validation, testing dataset

Model Training



Have a Break!

Feature Selection

Feature Selection

Lower the amount of input features

“ In most datasets, it is common for a few features to be responsible for the majority of the information signal and the rest of the features are just mostly noise.”

-- Alberto & Prateek (2020)

Every Group Project - Barmen Declaration (1934) edition.

Feature Selection

- Shorten training times
- Simplify models and make them easier to interpret
- Enhances testing set performance by reducing overfitting

Feature Selection

- For a model to produce accurate results, you need to make sure it’s using the *right* data. Feature selection is how you ensure your model is focused on the data with the most predictive power and is not distracted by data that won’t impact decision making. Precise feature selection will result in a faster, more efficient, more interpretable model.

- **If you have a lot of domain knowledge**, use machine learning and manually select the important features of your data.

- **If you have limited domain knowledge**, try automatic feature selection techniques such as neighborhood component analysis or use a deep learning algorithm (what we will learn soon) for feature selection.

- **If your data has lots of features**, use principal component analysis with machine learning to reduce dimensionality.

What Data Should You Include?

- For example, a medical researcher wants to make sense of a large amount of patient data. There could be thousands of features from patient stats, from the characteristics of a disease to DNA traits to environmental elements. If you have a solid understanding of the data, select the features you think will be the most influential and start with a machine learning algorithm. If you have high-dimensional data, try dimensionality reduction techniques such as principal component analysis (PCA) to create a smaller number of features to try to improve results.

Inspirations

https://www.youtube.com/watch?v=P8ERBy91Y90

Techniques

- **Feature Importance** ***Use ExtraTreesClassifier & matplotlib

Provide a score for each feature in a dataset, can be used for selecting important features.

- **Univariate Selection** ***Use SelectKBest & chi2

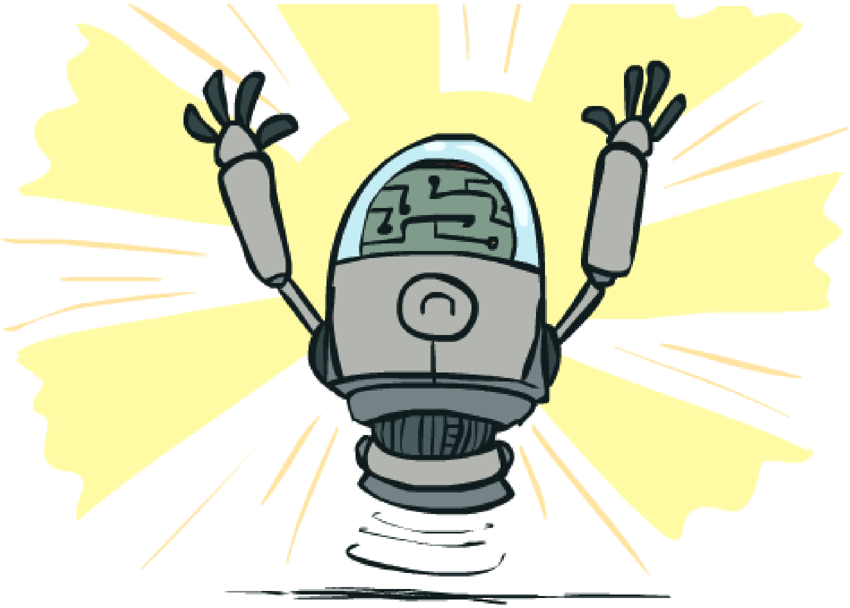
Provide a score for each feature in a dataset, can be used to determine which features have the strongest correlation to the output variable.

- **Correlation Heatmaps** *Import seaborn & matplotlib

Provide a matrix to show the relationship between the different values of the features. A heatmap makes it easy to identify which features are more correlated to the target variable.

Practice

TRY THE APPROACHES



Report

One representative from each group report for:

1. What approach have you applied

2. What is your results

3. Explain your results

Approach – Feature Importance

Task: Select the top 3 important features from the given data

```
import pandas as pd
from sklearn.ensemble import ExtraTreesClassifier
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv("train.csv")
X = data.iloc[:, 0:20]
X = X.fillna(X.median())
Y = data.iloc[:, -1]
Y = Y.fillna(Y.median())

model = ExtraTreesClassifier()
model.fit(X,Y)
print (model.feature_importances_)
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind="barh")
plt.show()
```

Approache – Univariate Selection

Task: Select features that have strong correlation to the output variable

```
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

data = pd.read_csv("train.csv")
X = data.iloc[:,0:20]
X = X.fillna(X.median())
Y = data.iloc[:,-1]
Y = Y.fillna(Y.median())
bestfeatures = SelectKBest(score_func=chi2, k=5)
fit = bestfeatures.fit(X,Y)
dfscores = pd.DataFrame(fit.scores_)

dfcolumns = pd.DataFrame(X.columns)
scores = pd.concat([dfcolumns,dfscores], axis=1)
scores.columns = ["specs","score"]
print(scores.nlargest(5,"score"))
```

Approach – Correlation Heatmaps

Task: Select features that are most correlated to the target variable

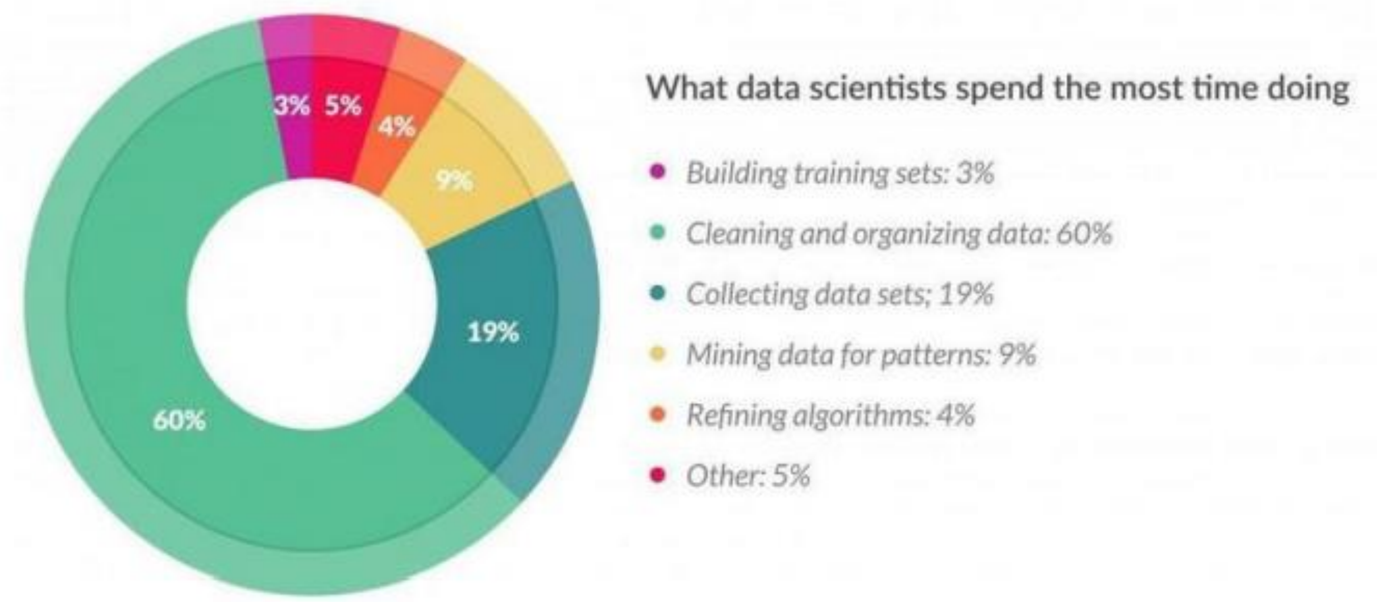
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv("train.csv")
X = data.iloc[:,0:20]
Y = data.iloc[:,-1]

correlation_matrix = data.corr()
top_corr_features = correlation_matrix.index
plt.figure(figsize=(20,20))
g = sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")

plt.show()
```

Have a Break!

Feature Engineering

Feature selection vs engineering

- With feature selection → you remove variables.
- In feature engineering → you create new variables to enhance the model.



Feature engineering

- **Steps**

Brainstorm about which features are relevant;

Decide what features might improve the model performance;

Create new features and determine if you should add them to the model performance (if not, drop them);

Go back to step 1 until the performance of the model meets expectations.

- **Data science techniques (We will discuss three of them)**

Imputation
Outlier management
One-hot encoding
Log transform
Scaling
Data manipulation

Example

1. **Patient Demographics:**
   Age: Age can be a critical factor in many healthcare predictions.   Gender: Gender-based differences might impact various health conditions.
   Ethnicity: Some health conditions are more prevalent in certain ethnic groups.
2. **Medical History and Diagnoses:**
   Previous Diagnoses: Previous health conditions can indicate risk factors or potential complications.
   Family Medical History: Genetic predispositions to certain diseases.
3. **Vital Signs:**
   Blood Pressure: An important indicator of cardiovascular health.
   Heart Rate: Can be related to various cardiac conditions.
   Temperature: Can indicate fever or other anomalies.
4. **Laboratory Results:**
   Blood Tests: Levels of glucose, cholesterol, hemoglobin, etc.
   Biomarkers: Specific proteins or substances that indicate certain diseases.
5. **Medications and Treatments:**
   Medication History: The types of medications a patient is on can provide insights into their conditions.
   Treatment Plans: Previous and ongoing treatments can impact a patient's health.
6. **Symptoms and Observations:**
   Symptoms: Self-reported symptoms can be valuable indicators.
   Physical Examinations: Physician observations about the patient's physical condition.
7. **Lifestyle Factors:**
   Diet: Dietary habits can impact various health conditions.
   Exercise: Physical activity levels can influence overall health.
   Smoking and Alcohol: Lifestyle choices can have significant health implications.

Healthcare Application

Imputation

Using median values

```
import pandas as pd

data = pd.read_csv("train.csv", nrows=10)
X = data.iloc[:, 0:20]
Y = data.iloc[:, -1]

data_new = data.fillna(0)
data_new = data.fillna(data.median())
print(data)
print(data_new)
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
0	842	0	2.2	0	1	0	NaN
1	1021	1	0.5	1	0	1	53.0
2	563	1	0.5	1	2	1	41.0
3	615	1	2.5	0	0	0	10.0
4	1821	1	1.2	0	13	1	44.0
5	1859	0	0.5	1	3	0	22.0
6	1821	0	1.7	0	4	1	10.0
7	1954	0	0.5	1	0	0	24.0
8	1445	1	0.5	0	0	0	53.0
9	509	1	0.6	1	2	1	9.0

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
0	842	0	2.2	0	1	0	24.0
1	1021	1	0.5	1	0	1	53.0
2	563	1	0.5	1	2	1	41.0
3	615	1	2.5	0	0	0	10.0
4	1821	1	1.2	0	13	1	44.0
5	1859	0	0.5	1	3	0	22.0
6	1821	0	1.7	0	4	1	10.0
7	1954	0	0.5	1	0	0	24.0
8	1445	1	0.5	0	0	0	53.0
9	509	1	0.6	1	2	1	9.0

Imputation

Using common values

	index	color
0	0	green
1	1	yellow
2	2	NaN
3	3	red
4	4	purple
5	5	red
6	6	red
7	7	purple
8	8	NaN
9	9	red
10	10	yellow
11	11	NaN
12	12	black
13	13	white

	index	color
0	0	green
1	1	yellow
2	2	red
3	3	red
4	4	purple
5	5	red
6	6	red
7	7	purple
8	8	red
9	9	red
10	10	yellow
11	11	red
12	12	black
13	13	white

```
import pandas as pd

data = pd.read_csv("dataset.csv")
print(data)

data["color"].fillna(data["color"].value_counts().idxmax(), inplace=True)
print(data)
```

One-hot encoding

```python
from sklearn.preprocessing import OneHotEncoder
import numpy as np

# Define the categories
categories = ['Human', 'Penguin', 'Octopus', 'Alien']

# Create the OneHotEncoder instance
encoder = OneHotEncoder(categories=[categories])

# Fit and transform the data
data = [['Human'], ['Penguin'], ['Octopus'], ['Alien']]
encoded_data = encoder.fit_transform(data).toarray()

# Print the encoded data
print(encoded_data)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

One-hot encoding example

Transfer text to statistical data

[“Life is short, I like python” ,
Life is too long, I hate python“]

Feature names: ['hate', 'is', 'life', 'like', 'long', 'python', 'short', 'too']

```
from sklearn.feature_extraction.text import CountVectorizer
# Input
# Create an instance
# Fit and transform the text data
# Get the feature names
# Convert the sparse matrix to an array for better visualization
# Print the feature names and the one-hot encoded representation
```

```
[[0 1 1 1 0 1 1 0]
 [1 1 1 0 1 1 0 1]]
```

Scaling

What is Scaling?

90	2	10	40
60	4	15	45
75	3	13	46

1.	0.	0.	0.
0.	1.	1.	0.83
0.5	0.5	0.6	1.

A technique for standardizing the range of features in a dataset

Scaling

Why scaling?

milage	liters	consimtime	target
14488	7.153469	1.673904	2
26050	1.441871	0.805124	1
75136	13.14739	0.428964	1
38344	1.669788	0.134296	1
72993	10.14174	1.032955	1
35948	6.830792	1.213192	3
42666	13.27637	0.54388	3
67497	8.631577	0.749278	1
35483	12.27317	1.503053	3
50242	3.723498	0.831917	1

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

This sample presents three eigenvalues of three different objects related to dating, namely: annual flight mileage, weekly consumption of liters of ice cream, and the proportion of time spent playing games.

The last column represents the three types evaluated by women, with 1 denoting dislike, 2 indicating general liking, and 3 representing high liking.

Since researchers consider these three characteristics to be equally significant, we need to use data preprocessing techniques to standardize the different range of the data and convert them to a common range.

Scaling

Normalization

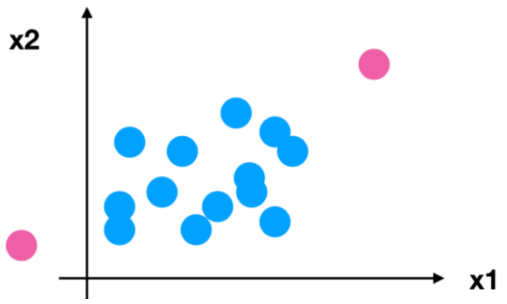
$$X' = \frac{x - min}{max - min} \qquad X'' = X' * (mx - mi) + mi$$

```python
from sklearn.preprocessing import MinMaxScaler
import pandas as pd

def minmax_demo():
    data = pd.read_csv("dating.txt",delimiter="\t" )
    data = data.iloc[:,:3]
    transfer = MinMaxScaler(feature_range=(1,2))
    data_new = transfer.fit_transform(data)
    print ("data:\n", data_new)


minmax_demo()
```

data

	milage	liters	consimtime
0	14488	7.153469	1.673904
1	26050	1.441871	0.805124
2	75136	13.147394	0.428964
3	38344	1.669788	0.134296
4	72993	10.141740	1.032955
5	35948	6.830792	1.213192
6	42666	13.276369	0.543880
7	67497	8.631577	0.749278
8	35483	12.273169	1.503053
9	50242	3.723498	0.831917

```
data:
 [[1.         1.48262275 2.        ]
 [1.19064108 1.         1.43571351]
 [2.         1.98910178 1.19139157]
 [1.3933518  1.0192587  1.        ]
 [1.96466495 1.73512784 1.58369338]
 [1.35384514 1.45535696 1.70076019]
 [1.46461549 2.         1.26603135]
 [1.87404366 1.60752099 1.39944064]
 [1.34617794 1.91523088 1.88902955]
 [1.58953304 1.19279457 1.45311599]]
```



Scaling

Standardization

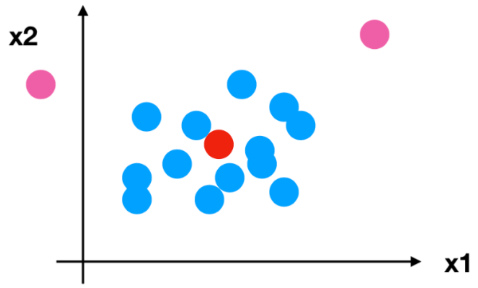
$$X' = \frac{x - \text{mean}}{\sigma}$$

*standard deviation

```
from sklearn.preprocessing import StandardScaler
import pandas as pd

def stand_demo():
    data = pd.read_csv("dating.txt", delimiter="\t")
    data = data.iloc[:,:3]
    transfer = StandardScaler()
    data_new = transfer.fit_transform(data)
    print ("data:\n", data_new)


stand_demo()
```

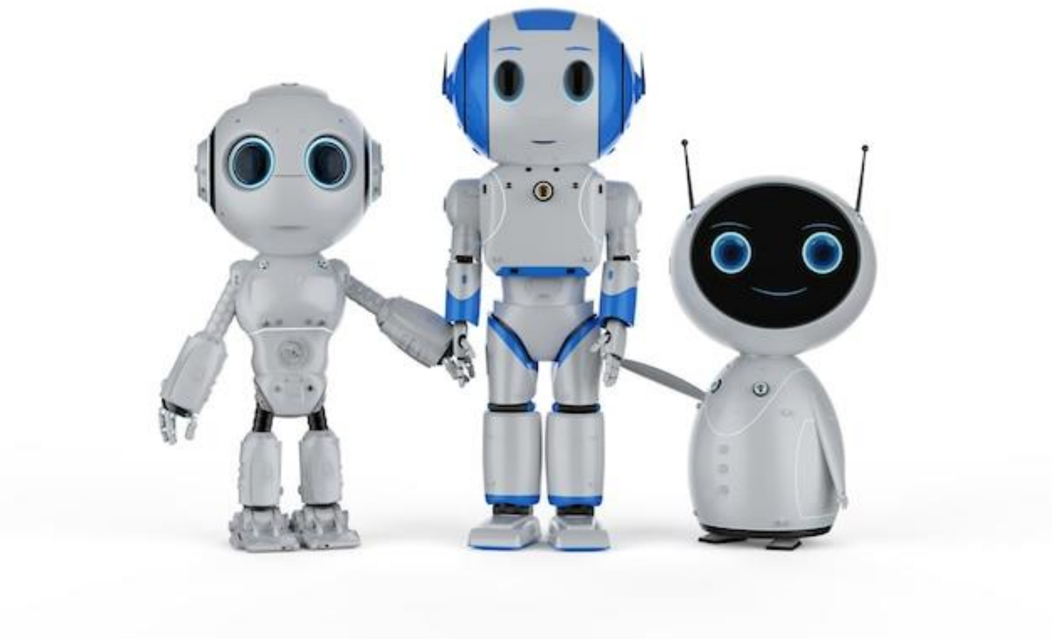
```
data
    milage     liters  consimtime
0   14488   7.153469    1.673904
1   26050   1.441871    0.805124
2   75136  13.147394    0.428964
3   38344   1.669788    0.134296
4   72993  10.141740    1.032955
5   35948   6.830792    1.213192
6   42666  13.276369    0.543880
7   67497   8.631577    0.749278
8   35483  12.273169    1.503053
9   50242   3.723498    0.831917
```

```
data:
 [[-1.62568136 -0.15888923  1.72810788]
 [-1.02701564 -1.50235998 -0.19116342]
 [ 1.51459527  1.25098992 -1.02215987]
 [-0.3904479  -1.4487498  -1.67312771]
 [ 1.40363344  0.54400595  0.31215099]
 [-0.51450974 -0.23478867  0.7103228 ]
 [-0.16666021  1.28132717 -0.7682924 ]
 [ 1.11905752  0.1887884  -0.314536  ]
 [-0.53858685  1.04535645  1.35067122]
 [ 0.22561548 -0.96568021 -0.13197348]]
```



Have a Break!

Group Up!



Work on your assignment!

Guidance

Phase 1

Select Topic: Read papers from Scopus

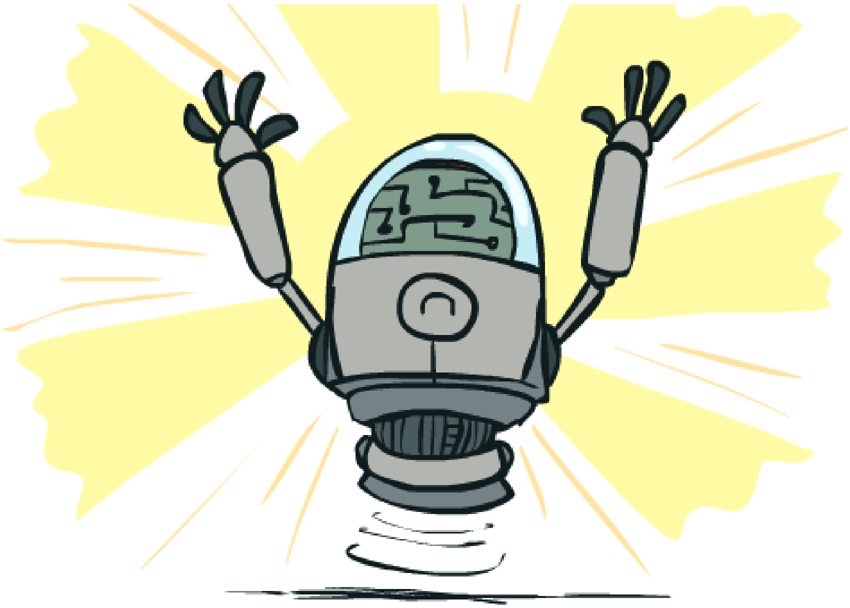
Define Problem

Find data source

University of
South-Eastern Norway
School of Business

Reflection

ASSIGNMENT PROCESS



THANKS