Introduction to Machine Learning

DataHack 2019

Shay Palachy *(and Dana Kaner ❤️)*

We are not going to make it.

Too much material. Too many slides. But...

1. Everything's is in the repo. Dig deeper on your own.

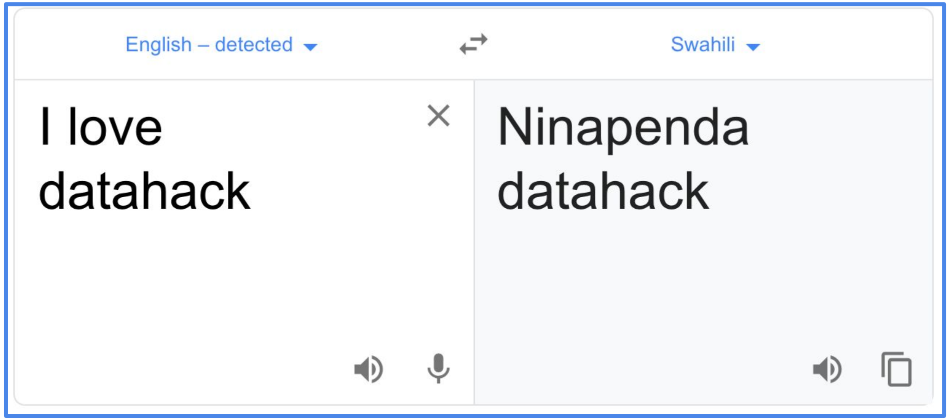
2. Sep. 4th talks will go deeper.

https://github.com/DataHackIL/DataLearn-ML-Intro-2019

What is machine learning?

"The study and construction of algorithms that can learn from past data and make predictions on future data."

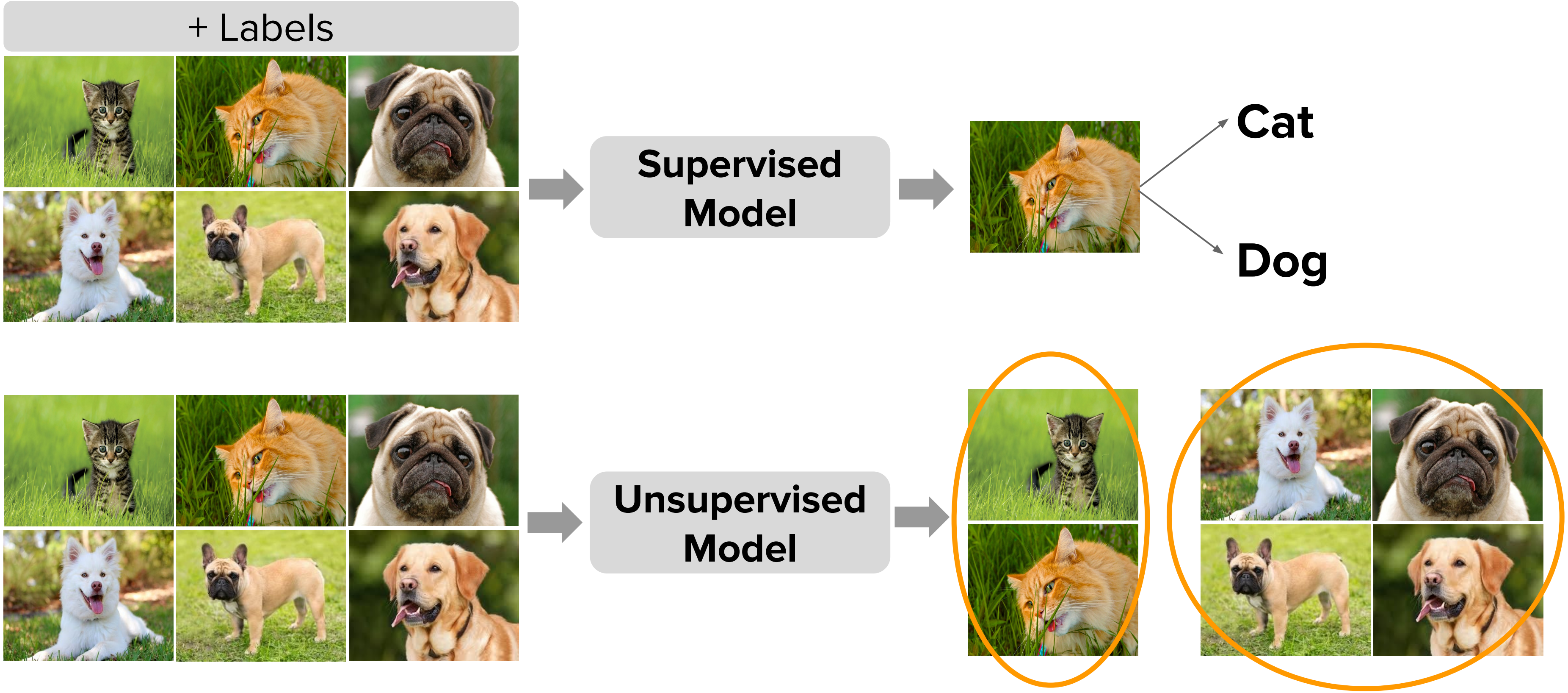
Why do we need it?



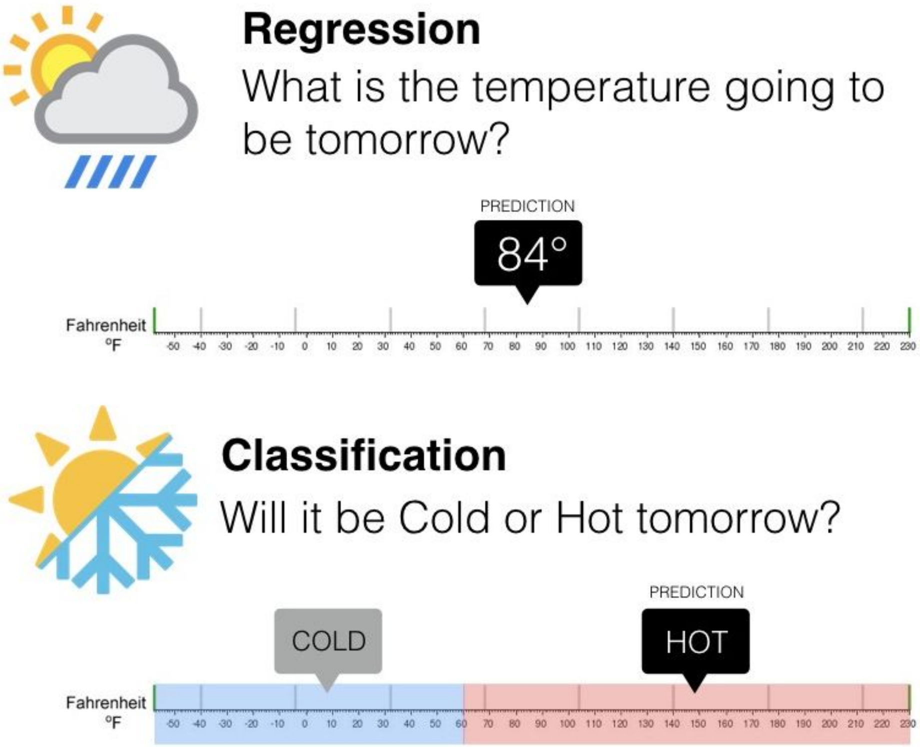
Agenda

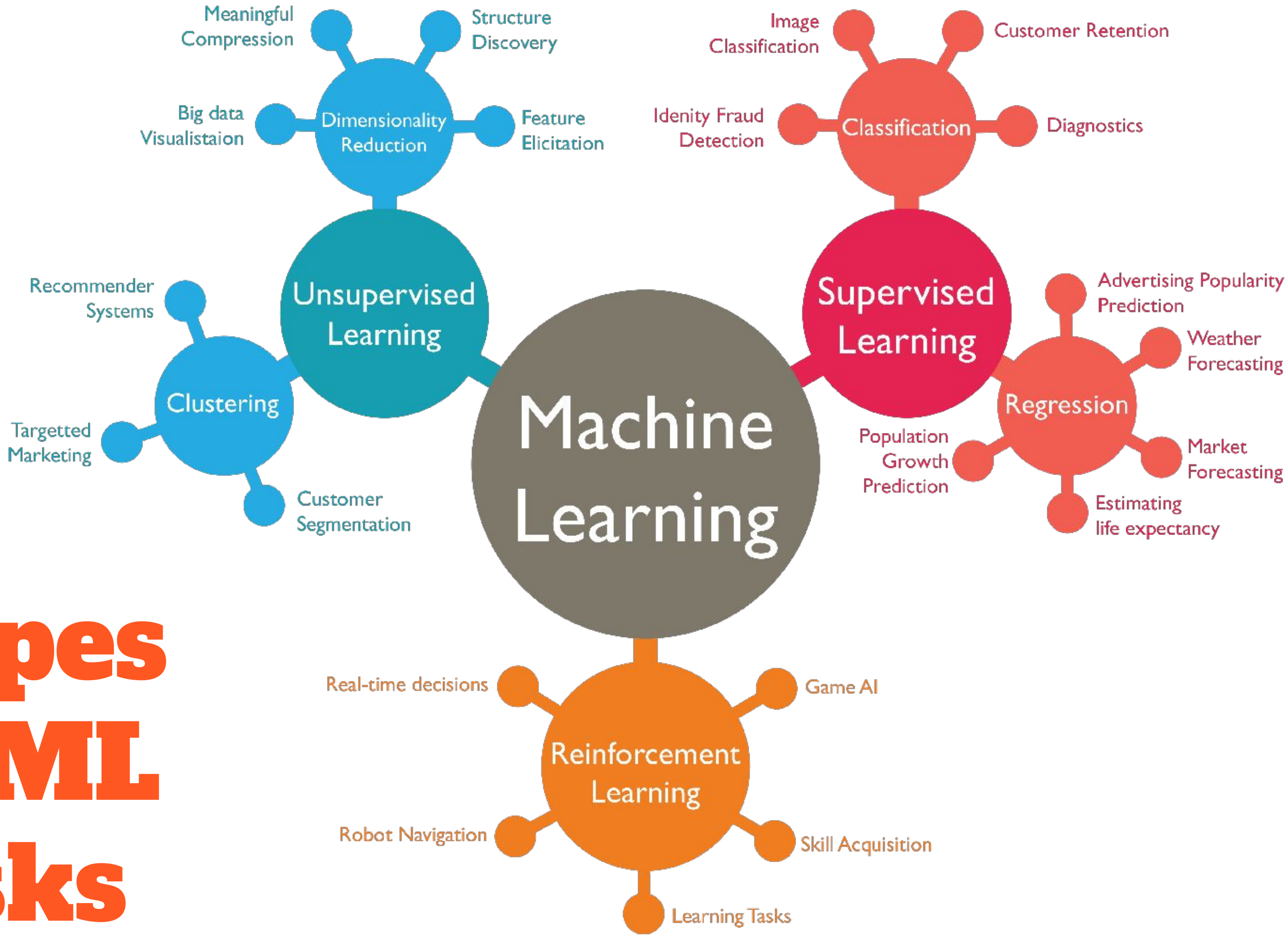
- Tools of the trade
- Data Exploration
- Preprocessing
- Modeling
  - **Supervised** vs. Unsupervised
  - Regression vs. **Classification**
  - Model fit and model selection
- Prediction

Supervised vs. Unsupervised



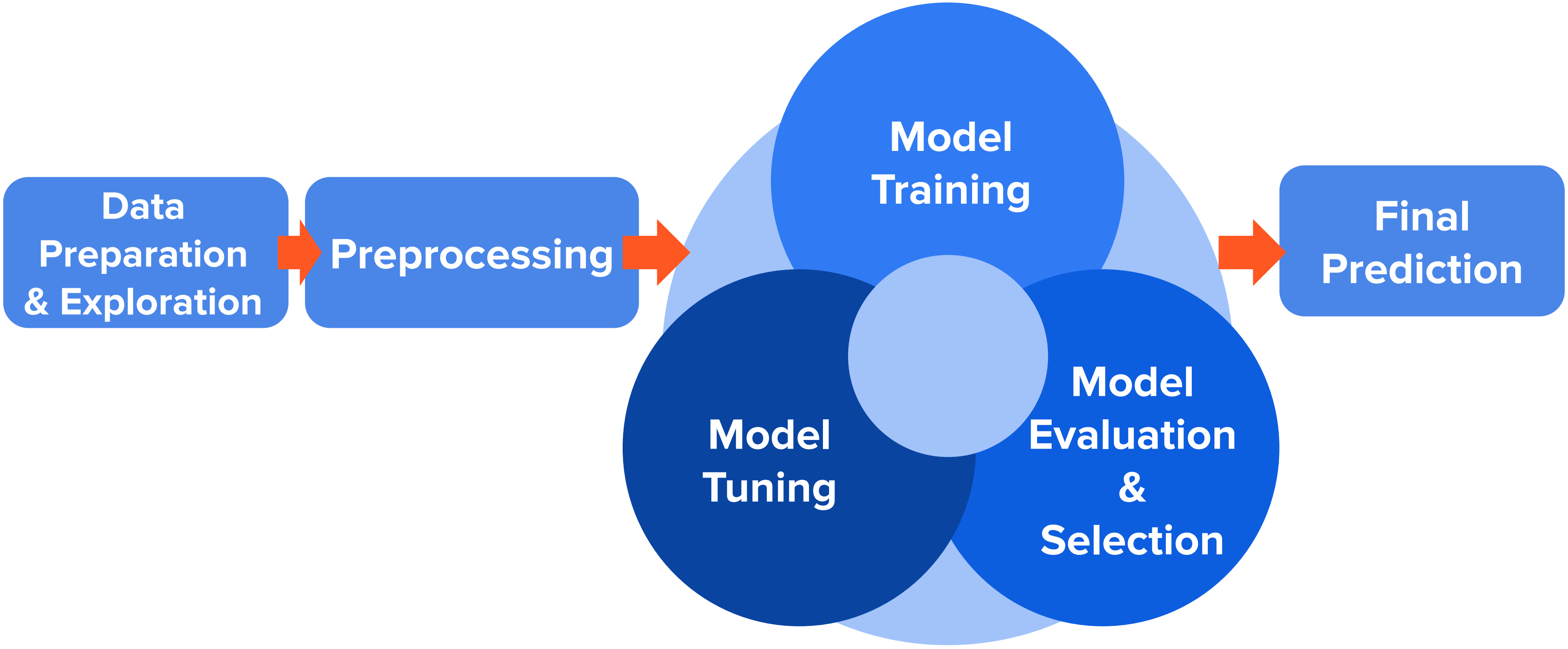
Regression vs. Classification





Types of ML tasks

The ML Pipeline



Tools of the trade

- Common programming language:
    - **Python**
    - R, Matlab
    - Java, C++, ...
- The classic Python stack:
    - Jupyter notebooks *(and their cloud hosted variants)*
    - numpy
    - pandas 🐼🐼
    - scikit-learn

Hands On!

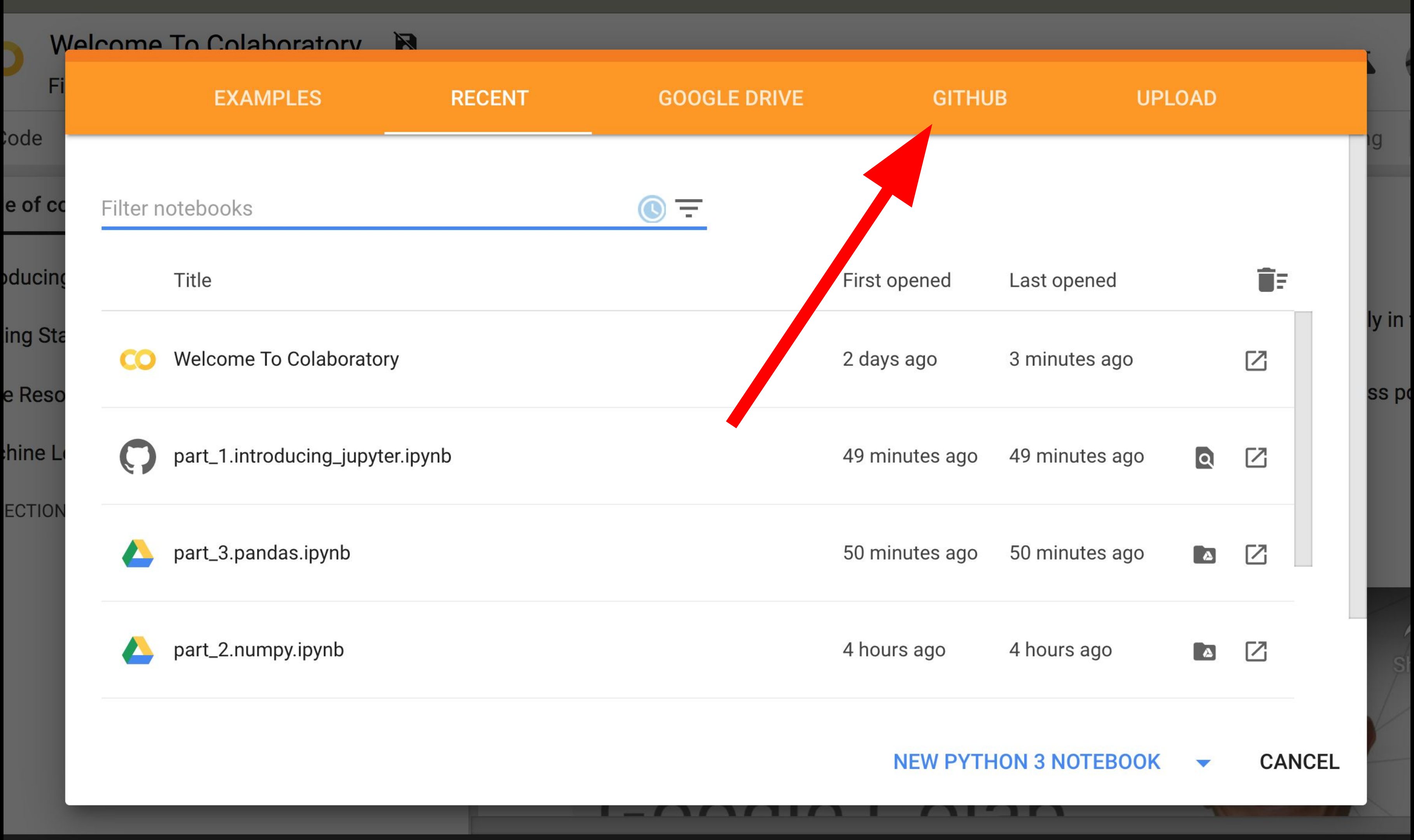
- Jupyter notebooks *(and their cloud hosted variants)*
- numpy
- pandas 🐼🐼

github.com/DataHackIL/DataLearn-ML-Intro-2019

Hands On!

Go to Google Colaboratory:

colab.research.google.com


Welcome To Colaboratory
EXAMPLES
RECENT
GOOGLE DRIVE
GITHUB
UPLOAD
Filter notebooks
Title
First opened
Last opened
Welcome To Colaboratory
2 days ago
3 minutes ago
part_1.introducing_jupyter.ipynb
49 minutes ago
49 minutes ago
part_3.pandas.ipynb
50 minutes ago
50 minutes ago
part_2.numpy.ipynb
4 hours ago
4 hours ago
NEW PYTHON 3 NOTEBOOK
CANCEL

EXAMPLES RECENT GOOGLE DRIVE GITHUB

Enter a GitHub URL or search by organization or user

https://github.com/DataHackIL/DataLearn-ML-Intro-2019

Repository:

DataHackIL/DataLearn-ML-Intro-2019

Branch:

master

Path

part_1.introducing_jupyter.ipynb

part_2.numpy.ipynb

part_3.pandas.ipynb

Data Exploration

Data Exploration

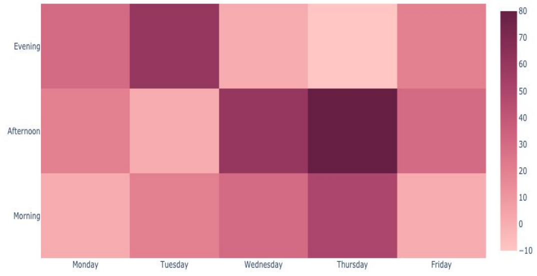
- Aka Exploratory Data Analysis (EDA)
- Discover important characteristic of the data
- In the context of ML:
  - Hints towards what preprocessing to do
  - Hints towards what features to generate and select

Explore important features

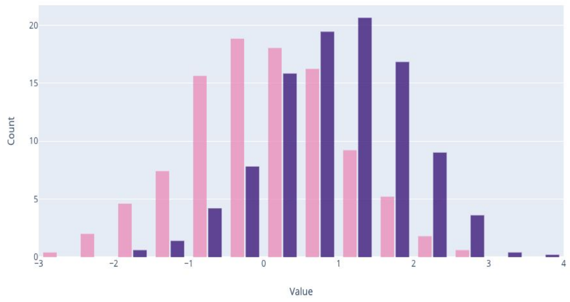
Yes you can!

- By business logic
- By statistical measures and visualization tools

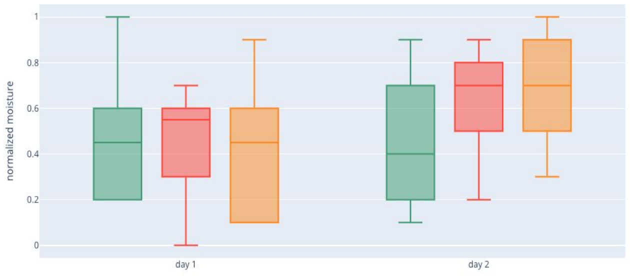
Heatmaps and correlation plots



Histograms



Boxplots and distribution plots



Numerical Data Exploration

- Summary statistics: #uniques, max, min, sum, avg, std, #NA, quantiles, percentiles, etc.
- Histograms
- Cross-feature correlations
  - Bubble plots, correlation matrices, correlation networks and scatter plots
- Fit to specific distribution family, mixtures, etc.

Now it's time to get our hands dirty

https://github.com/DataHackIL/DataLearn-ML-Intro-2019

github.com/DataHackIL/DataLearn-ML-Intro-2019

README.rst

part_1.introducing_jupyter.ipynb

part_2.numpy.ipynb

part_3.pandas.ipynb

part_4.EDA.ipynb

part_5.Preprocessing.ipynb

Preprocessing

Types of data

We'll focus on these

- **Structured:**
  - Numerical ❤️
  - Categorical
  - Ordinal
- **Unstructured:**
  - Text
  - Audio
  - Image

Preprocessing / Feature engineering

- Imputation *(dealing w/ missing data)*
- Scaling and normalization
- Handling outliers
- Feature extraction/generation
  - Categorical and ordinal data
- Feature selection & dimensionality reduction

sometimes this is what people mean by feature engineering

Imputation

Imputation: Dealing with missing data

- Drop columns and/or rows with more the X% missing vals
- Categorical data: Place a dummy value for "missing"
- Replace with the mode (most frequent value)
- Numerical data: Replace w/ mean, median, etc.
- Per feature, learn a model to predict missing value by other features of that entry/row

Note: Some models handle input w/ missing data well, offsetting the need for imputation

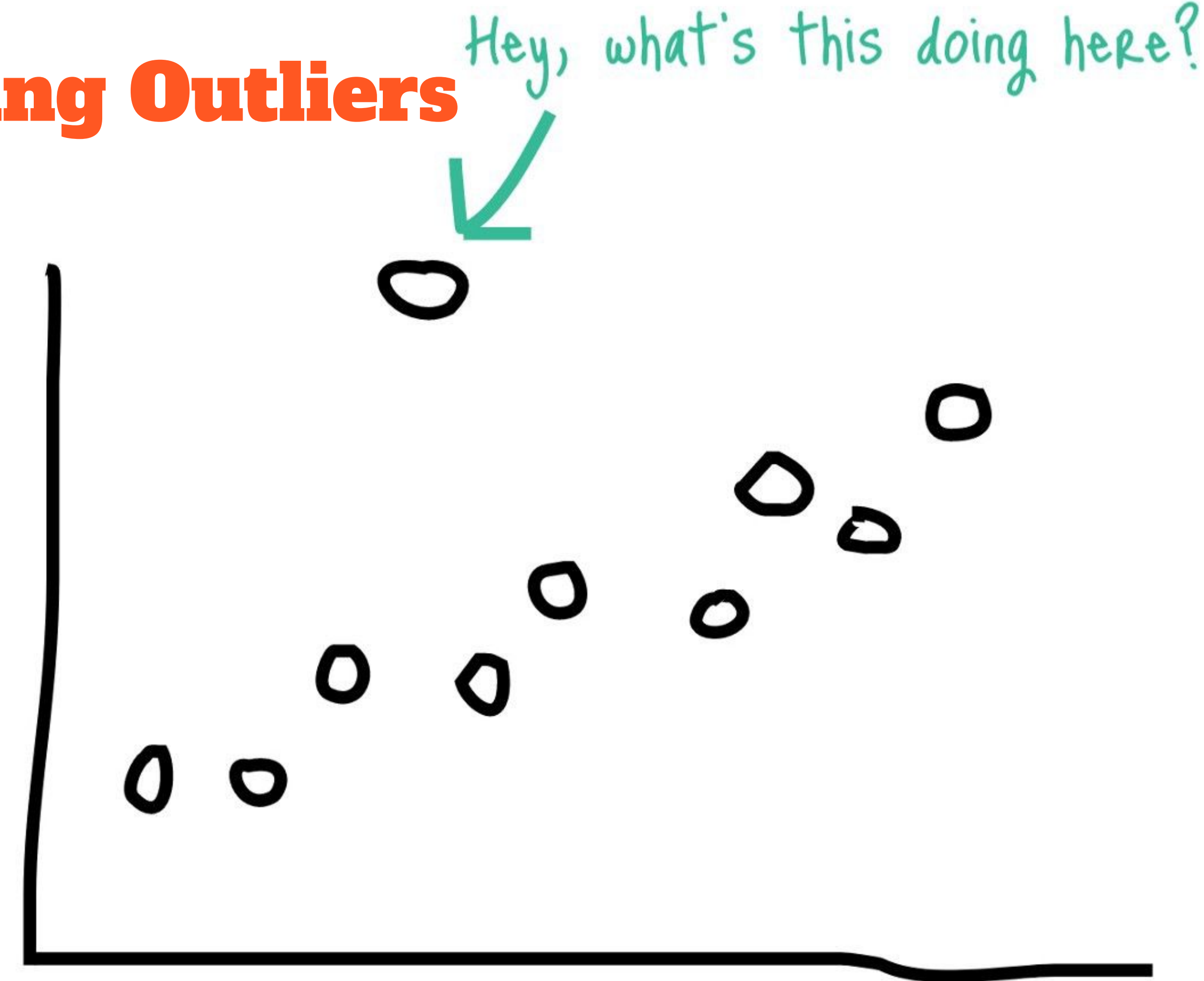
Into the notebook!

part_4.EDA.ipynb

[part_5.Preprocessing.ipynb]()

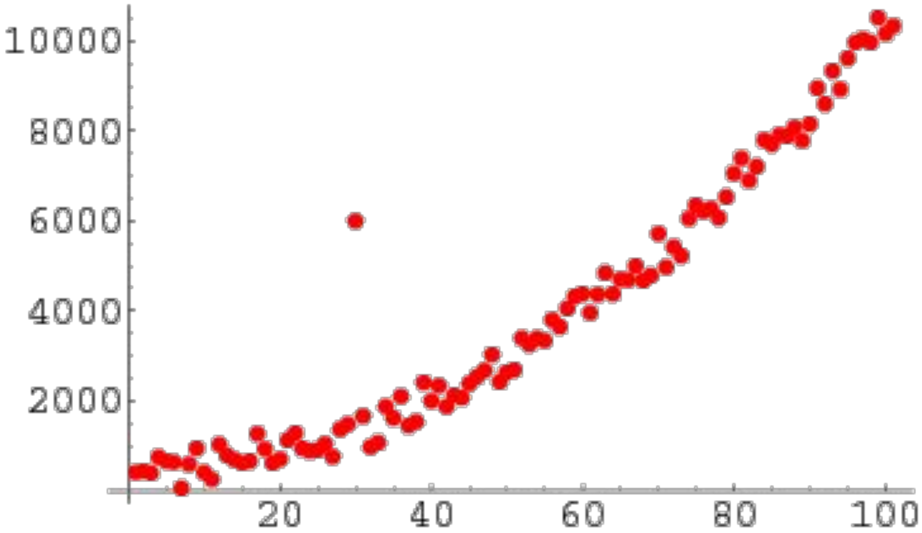
Handling outliers

Handling Outliers

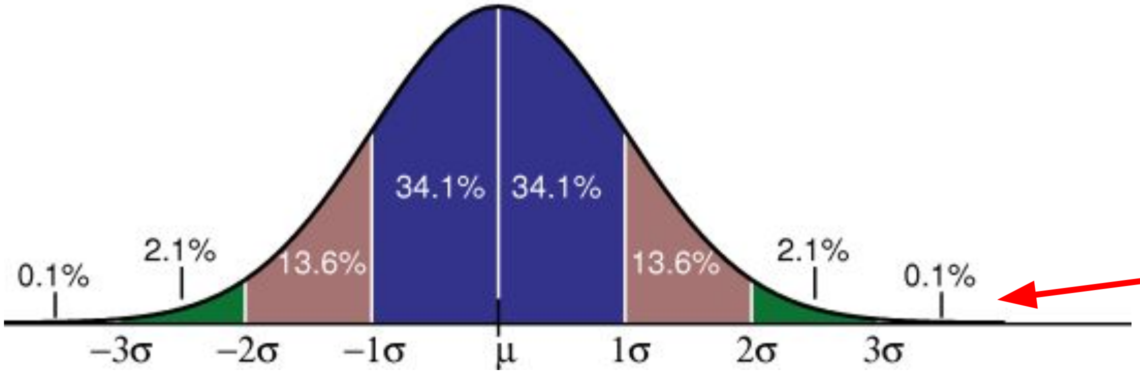


Handling Outliers

Overall



Feature-wise



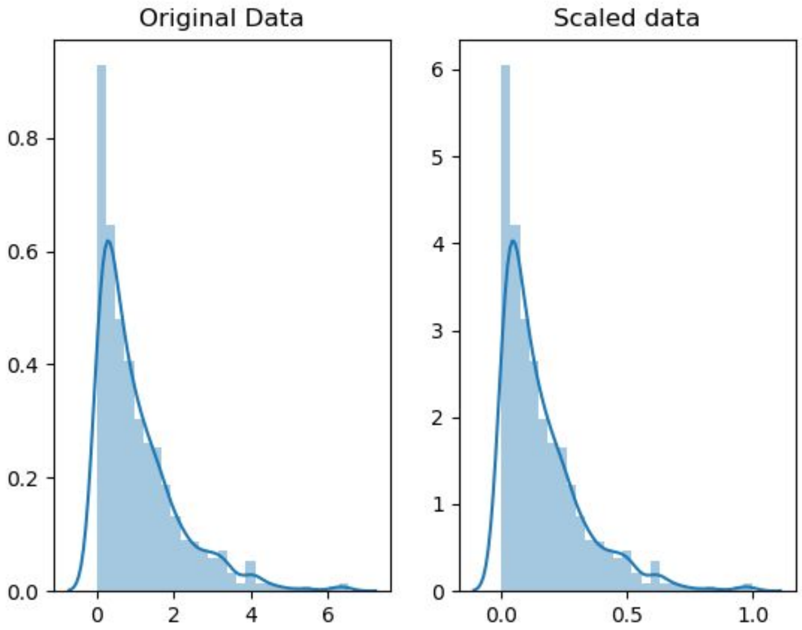
this assumes a normal distribution...

Back to the notebook!

Scaling and normalization

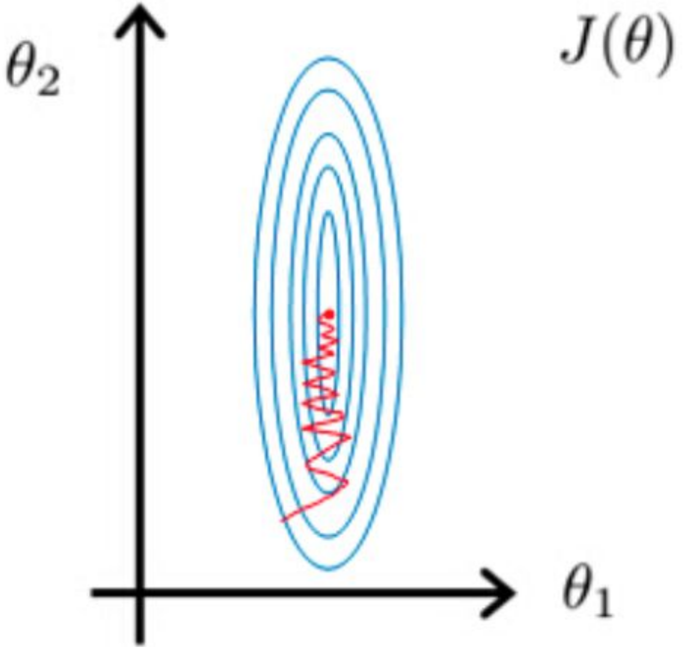
Scaling

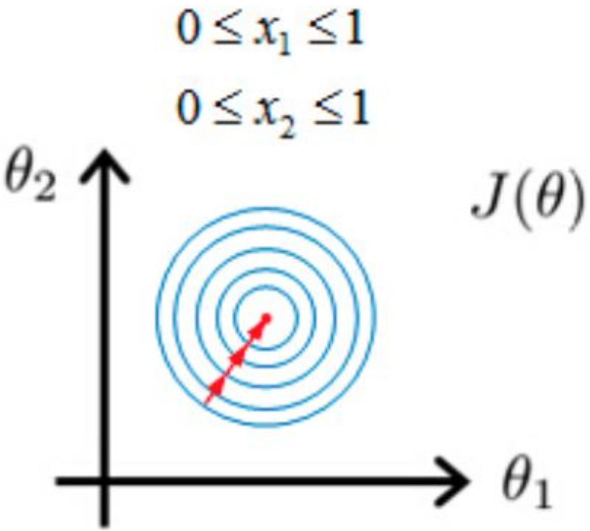
Scale the features so they all fall between some interval; usually [0,1].



Why scale your data?

Scaling ≈ telling the model all features are equally important





MIN MAX SCALING

Rescales feature values to between 0 and 1

Original value

Minimum value in feature

Rescaled value

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

Maximum value in feature

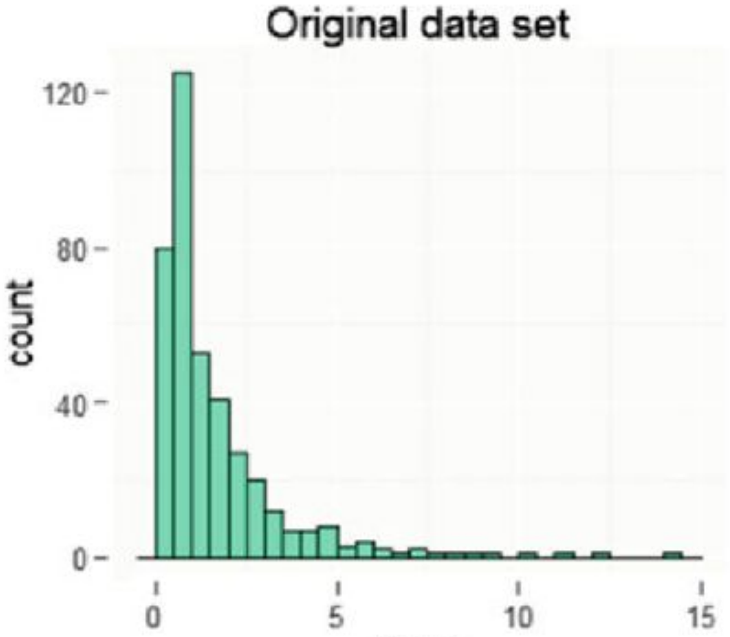
ChrisAlbon

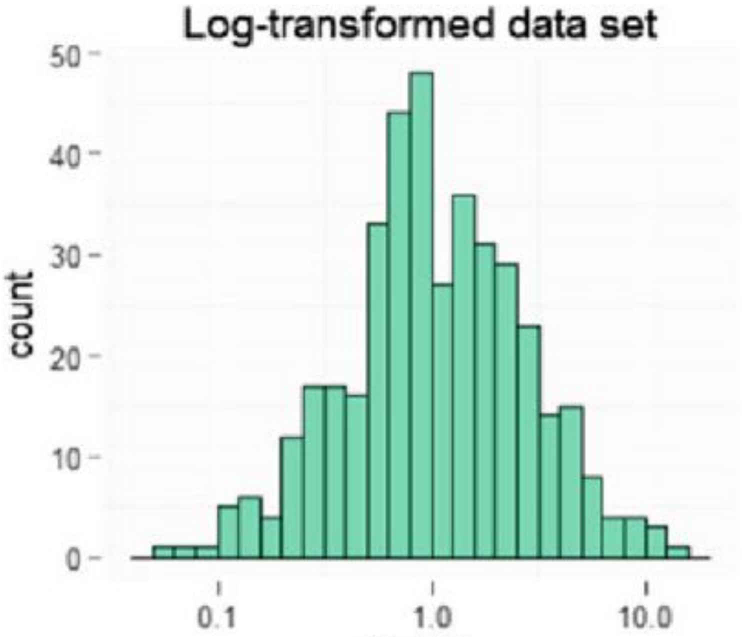
```
>>> from sklearn.preprocessing import MinMaxScaler
```

```
>>> data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
>>> scaler = MinMaxScaler()
>>> print(scaler.fit(data))
MinMaxScaler(copy=True, feature_range=(0, 1))
>>> print(scaler.data_max_)
[ 1. 18.]
>>> print(scaler.transform(data))
[[0.   0.  ]
 [0.25 0.25]
 [0.5  0.5 ]
 [1.   1.  ]]
>>> print(scaler.transform([[2, 2]]))
[[1.5 0. ]]
```

Log-transform

- Shift data to the `(0,m]` range for some `m`.
- Apply the natural logarithmic function to the data.

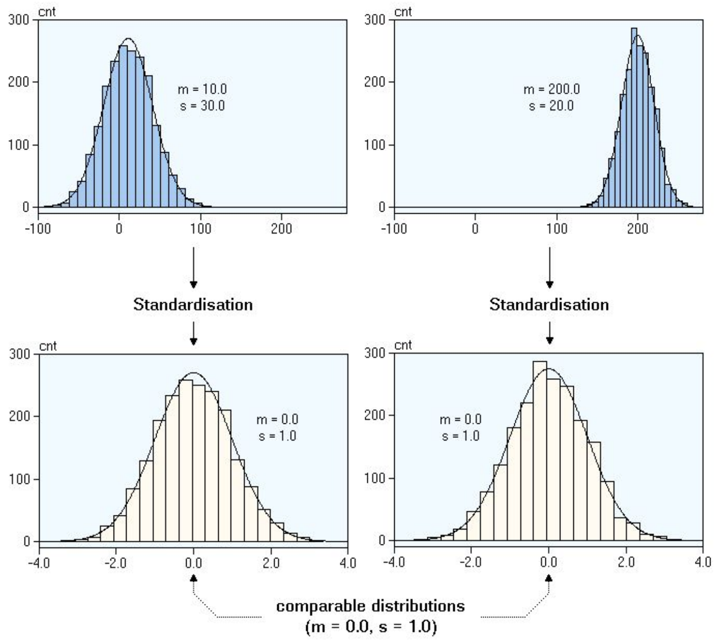




Standardization

Center the features around the origin (zero) and scale so they have standard deviations of one unit.

(this is actually studentizing, as we're scaling the data by an estimate of the StdDev!)



STANDARDIZATION

Value of the ith observation

Standardized feature value

Mean of the feature vector

$$x'_i = \frac{x_i - \bar{x}}{\sigma}$$

Standard deviation of the feature vector

Standardization is a common scaling method. x'_i represents the number of standard deviations each value is from the the mean value. It rescales a feature to have a mean of 0 and unit variance.

Chris Albon

Why standardize your data?

- Many models assume that all features are normal:
    - t-tests and ANOVAs
    - linear regression
- There are still some that don’t require this assumption:
    - Decision trees and tree-based ensembles
    - Naive Bayes

Back to the notebook!

Feature Extraction

Categorical Features

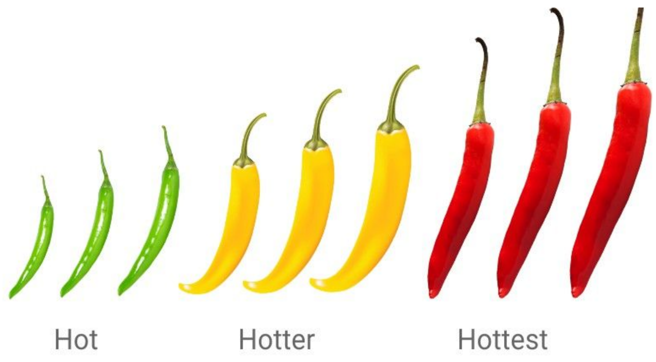
City
Tel Aviv
Jerusalem
Ashdod
Tel Aviv
Tel Aviv
Jerusalem
Eilat

city_Tel_aviv	city_Jerusalem	city_Ashdod	city_Eilat
1	0	0	0
0	1	0	0
0	0	1	0
1	0	0	0
1	0	0	0
0	1	0	0
0	0	0	1

Categorical Features

- Important: drop one of the features, to avoid perfect multicollinearity (aka the dummy variable trap).
  - A feature can be predicted **perfectly** by a linear combination of other features.
- If you have NaNs in your data, an alternative is to not produce a dummy variable for it; missing data is then given by a value of 0 in all dummies.

Ordinal Features



- Handle as Categorical
- Handle as Numeric

Binning

speed_kmh
2
20
80

walking	cycling	driving
1	0	0
0	1	1
0	0	1

Binning

Use knowledge from various domains, to save the model from having to learn it itself.

- Physical and chemical
- Biological, anatomical, medicinal, etc.
- Business and finance
- Common sense!

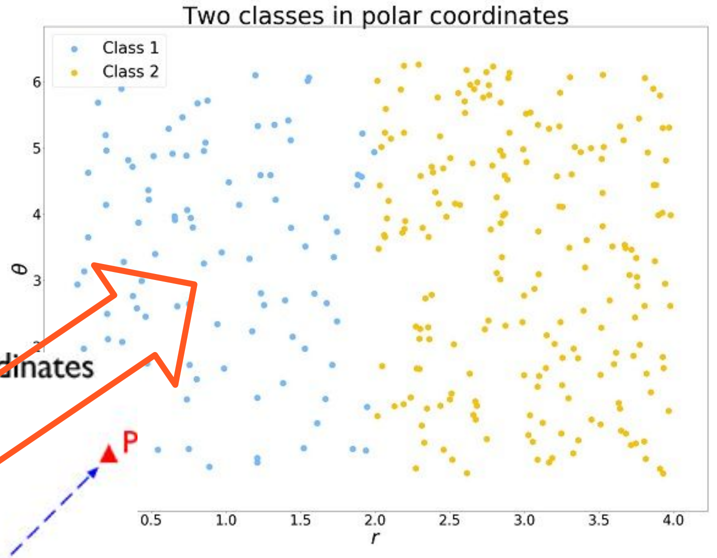
Data & time features

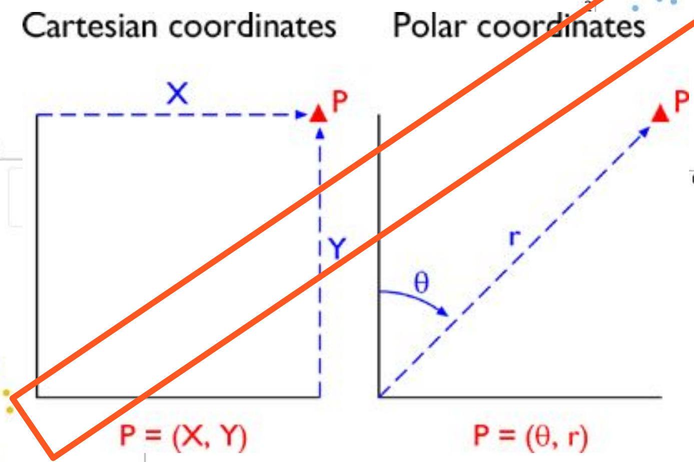
(seconds since the epoch)

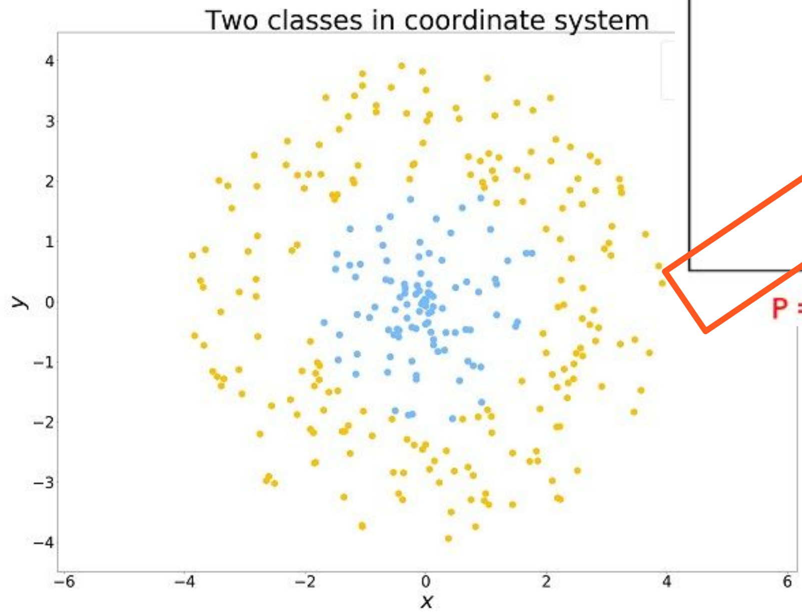
Timestamp
1566296716

dayofweek	dayofmonth	month	hour	season
2	20	8	13	1

Coordinate transform







Grouping/reducing

uid	name
34	peralta

uid	title	num_watched
34	Die hard 1	74
34	Die hard 2	63
34	Die hard 1	67

Grouping/reducing

- When a single entry in our dataset corresponds to a reach document or more than one row in another table/dataset (e.g. all movies likes by a user), we'll have to reduce any such information into summary features, as we (normally) have to work with 1d vectors.
- We'll see very basic examples with pandas: taking the first value, sum, mean, etc.

Back to the notebook!

Feature Selection & Dimensionality Reduction

Dimensionality reduction

- High-dimensional data (hundreds/thousands features +) can cause problems when applying machine learning algorithms to it:
  - Distances behave weird in high dimensions
  - Run time for the optimization/approximation algorithms used can blow up
  - It can be hard to visualize and make sense of

Dimensionality reduction

- Some of these problems can be mitigated by reducing the dimension of our data by:
  - Feature selection
  - Feature projection / synthetic feature extraction

Feature selection

A few basic approaches:

- Get rid of correlated features
- Select the features with the most variance/information
- Select best subsets, taking interaction into accounts

There's a lot more to it, but that's the gist.

Feature projection

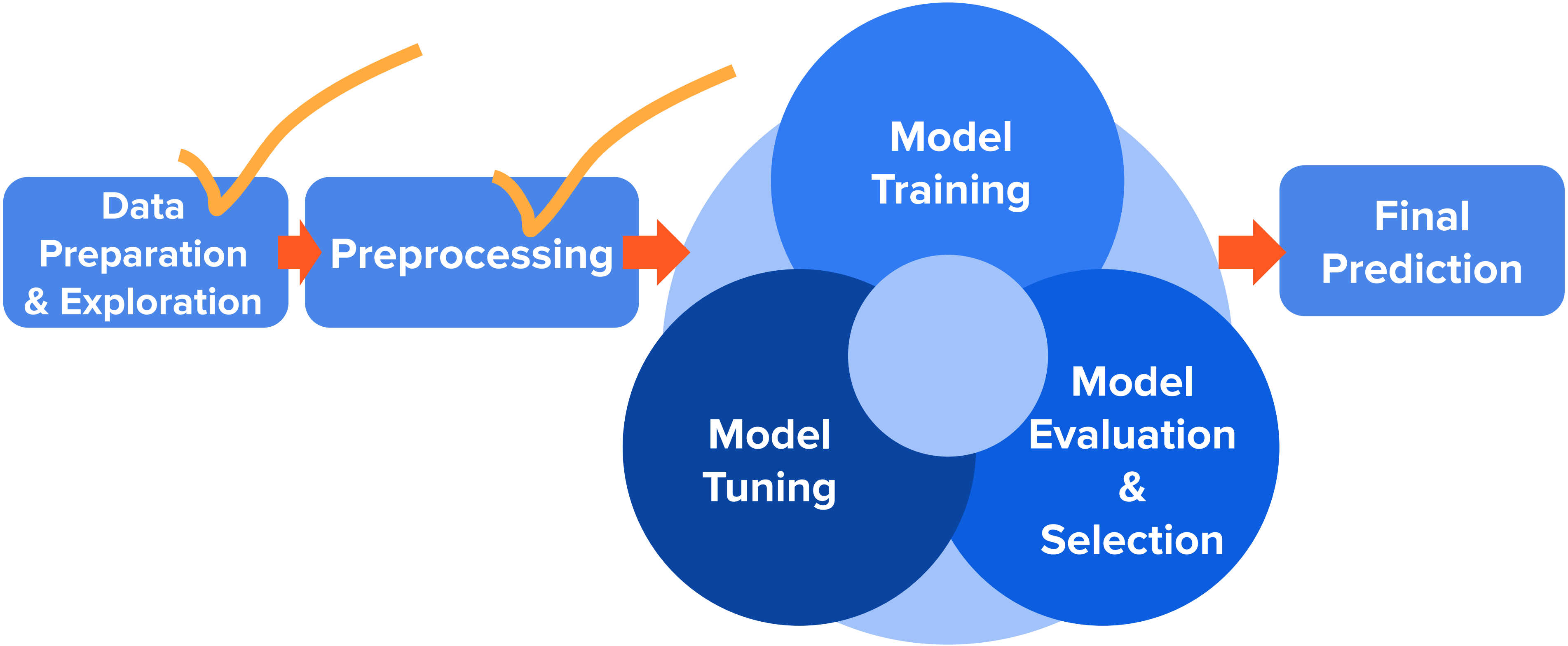
- One common approach: Construct synthetic features that maximize the variance in the data.
  - Linear combination
  - Nonlinear
- Prominent approaches:
  - Principal Component Analysis (PCA)
  - Linear discriminant analysis (LDA)
  - t-SNE

a fascinating topic that's completely out of scope

Back to the notebook!

Modeling

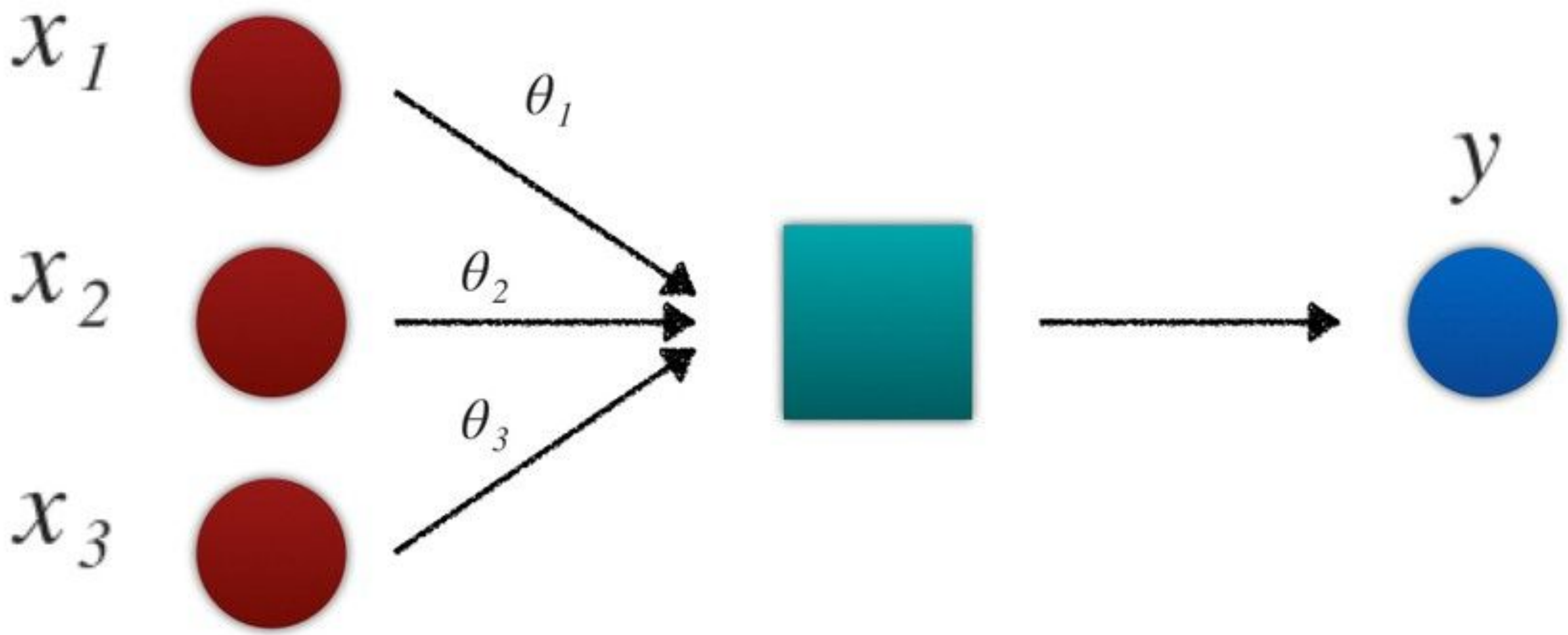
The ML Pipeline



Model Training

- Model: $y \approx f(x) \rightarrow \hat{y} = \hat{f}(x)$
- Goal: minimize the loss function on the training data
- What loss function to use?

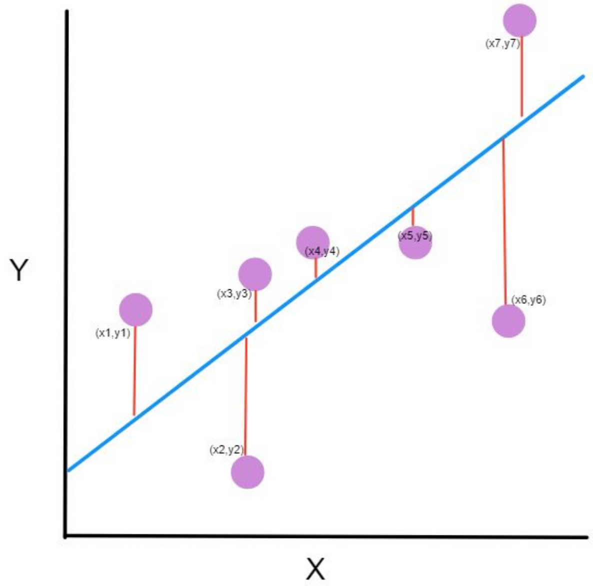
Logistic regression



Mean squared error

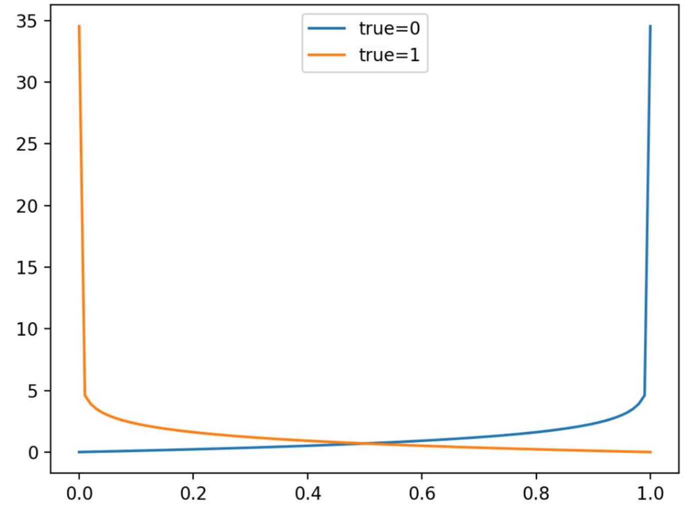
- Literally the average of all squared prediction errors...
- Commonly used in regression

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$



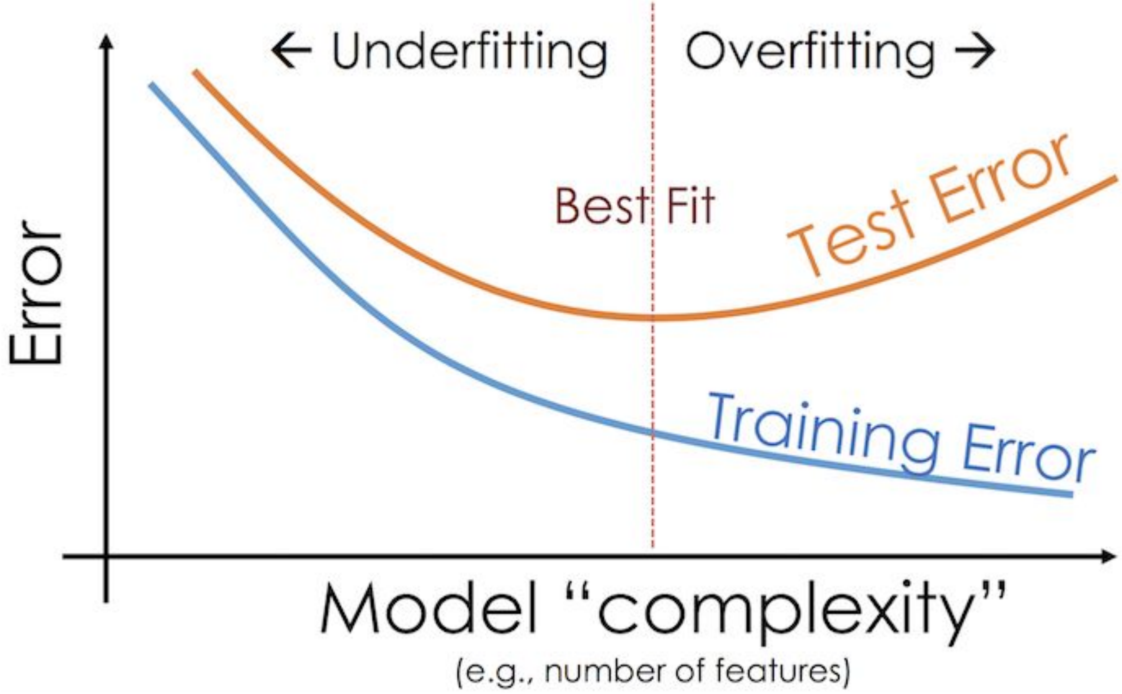
Log loss

- Adequate when prediction output is probability between 0 and 1; easily usable in binary classification.



$$\text{logloss} = -\frac{\sum_{i=1}^{n} y_i \cdot \log(\hat{P}(y_i)) + (1-y_i) \cdot \log(1-\hat{P}(y_i))}{n}$$

Split your data



$\underbrace{\text{Train - test split}}_{}$ -validation-

Train - test split

Split your data

Model Training

Model Tuning

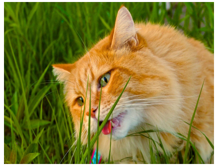
Model Evaluation & Selection

Final Prediction

Train Data

Validation Data

Test Data



new observation

Model Evaluation

- **Predict model predictions on your test data and compute an evaluation metric**
  - Same loss function
  - Other evaluation metric:

		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/

Model Evaluation & Selection

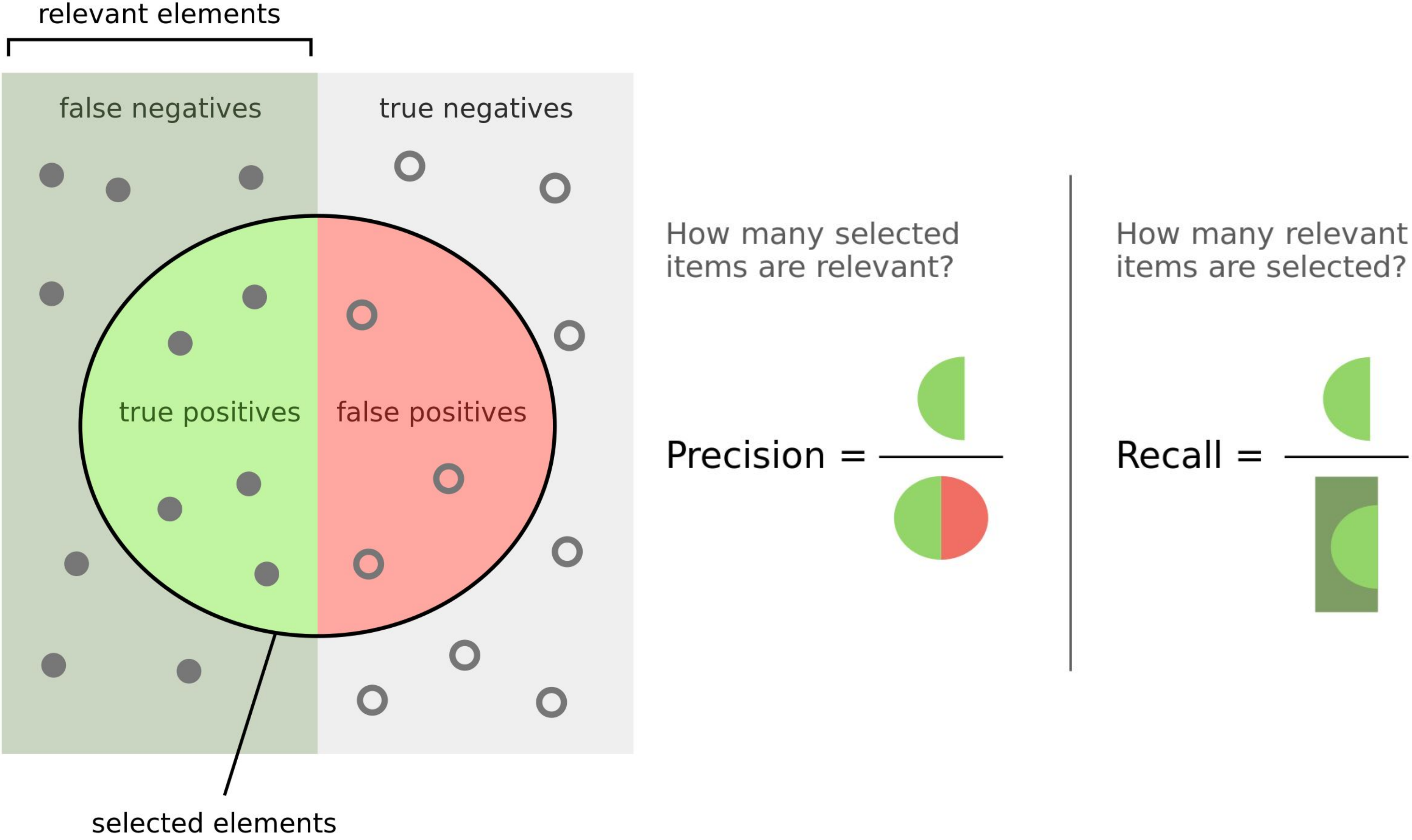
Test Data

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Accuracy} = \frac{TP + TN}{\text{Total}}$$

		Actual	
		Pos	Neg
Predicted	Pos	TP	FP
	Neg	FN	TN


relevant elements
false negatives
true negatives
true positives
false positives
selected elements
How many selected items are relevant?
How many relevant items are selected?
Precision =
Recall =

F1 SCORE

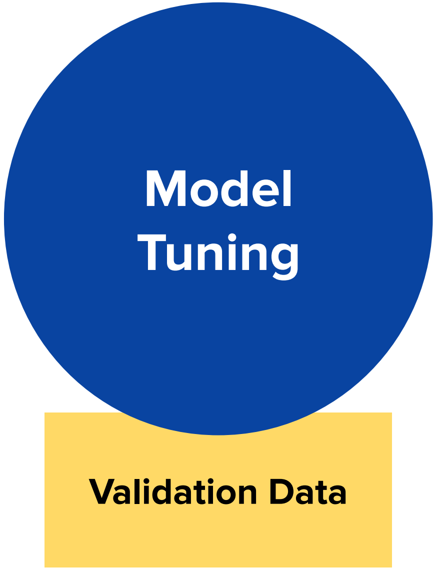
Chris Albon

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1 score is the harmonic mean of precision and recall. Values range from 0 (bad) to 1 (good).

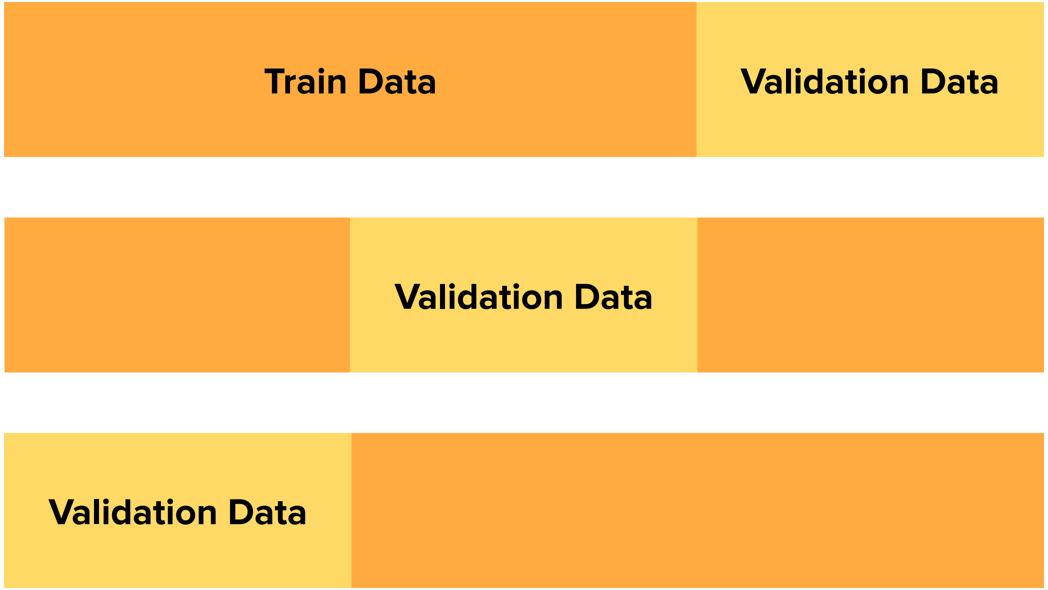
Model Tuning

- **A lot of models have hyper-parameters to tune -**
  - Decision boundary in classification
  - Tree depth in tree-based methods
  - Number of models in ensembles



Model Tuning

- Another option: cross-validation



Prediction

- Now, given a the fixed parameters of the fitted model, we can predict the label of an unseen entry by
    - Injecting its vector representation x_i into the model equation
    - Multiplying each vector entry by the corresponding model parameter

Let’s run some models

https://github.com/DataHackIL/DataLearn-ML-Intro-2019

Done!

Other materials at:
github.com/DataHackIL/DataWorkshops

My materials at:
www.shaypalachy.com

datahack.org.il