Machine Learning

Asieh Daneshi
2024

What is Machine Learning?

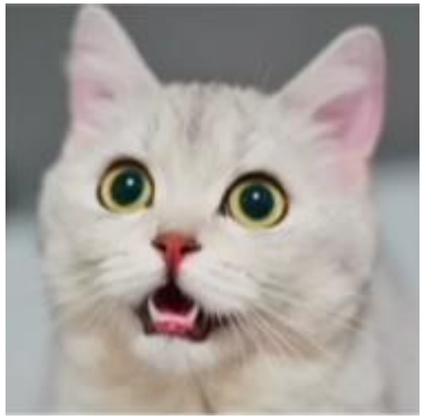
Machine Learning is a subdomain of computer science that focuses on algorithms which help a computer learn from data without explicit programming

What is the difference between AI and ML and DS?

- Artificial intelligence is an area of computer science, where the goal is to enable computers and machines to perform human-like tasks and simulate human behavior
- Machine learning is subset of AI that tries to solve a specific problem and make predictions using data
- Data science is a field that attempts to find patterns and draw insights from data (might use ML!)

Types of Machine Learning

Supervised learning - uses labeled inputs (meaning the input has a corresponding output label) to train models and learn outputs



Cat

Dog

Frog

Types of Machine Learning

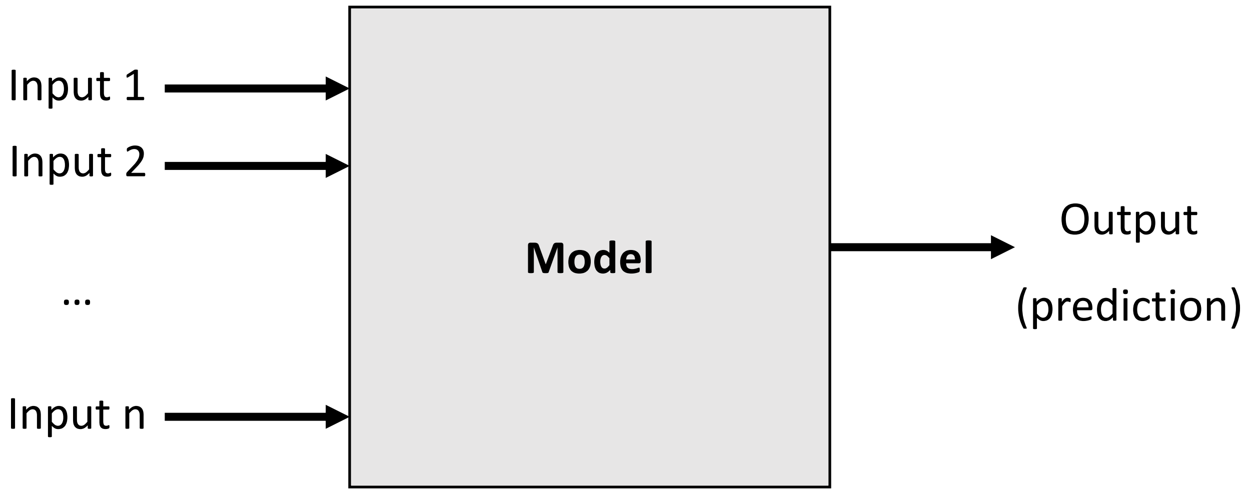
Supervised learning - uses labeled inputs (meaning the input has a corresponding output label) to train models and learn outputs

Unsupervised learning - uses unlabeled data to learn about patterns in data

Types of Machine Learning

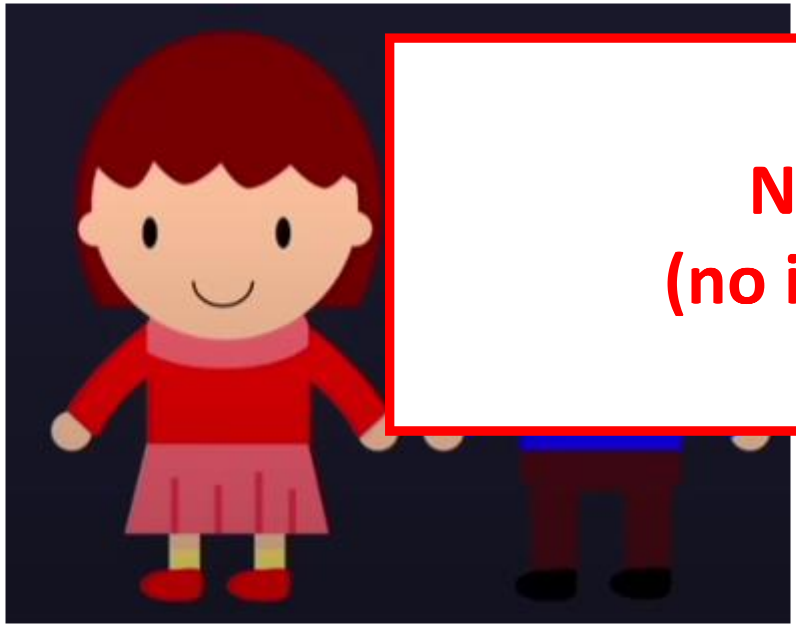
Supervised learning - uses labeled inputs (meaning the input has a corresponding output label) to train models and learn outputs
Unsupervised learning - uses unlabeled data to learn about patterns in data
Reinforcement learning – agent learning in interactive environment based on reward and penalties

Machine Learning



Features

Qualitative – categorical data (finite number of categories or groups)



Nominal Data
(no inherent order)

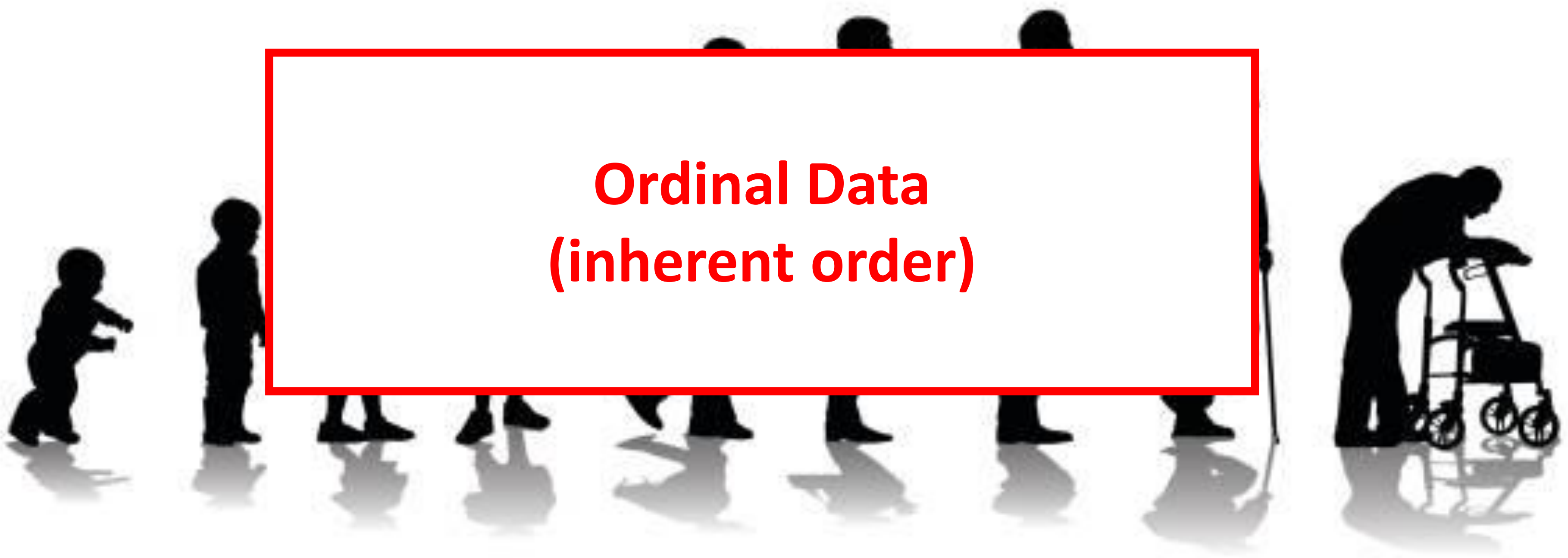
One-Hot Encoding

Every time we just focus on a specific category and label data from that category 1, and all the others are 0

Country	One-Hot Encoding
Japan	1,0,0,0
China	0,1,0,0
Germany	0,0,1,0
Canada	0,0,0,1

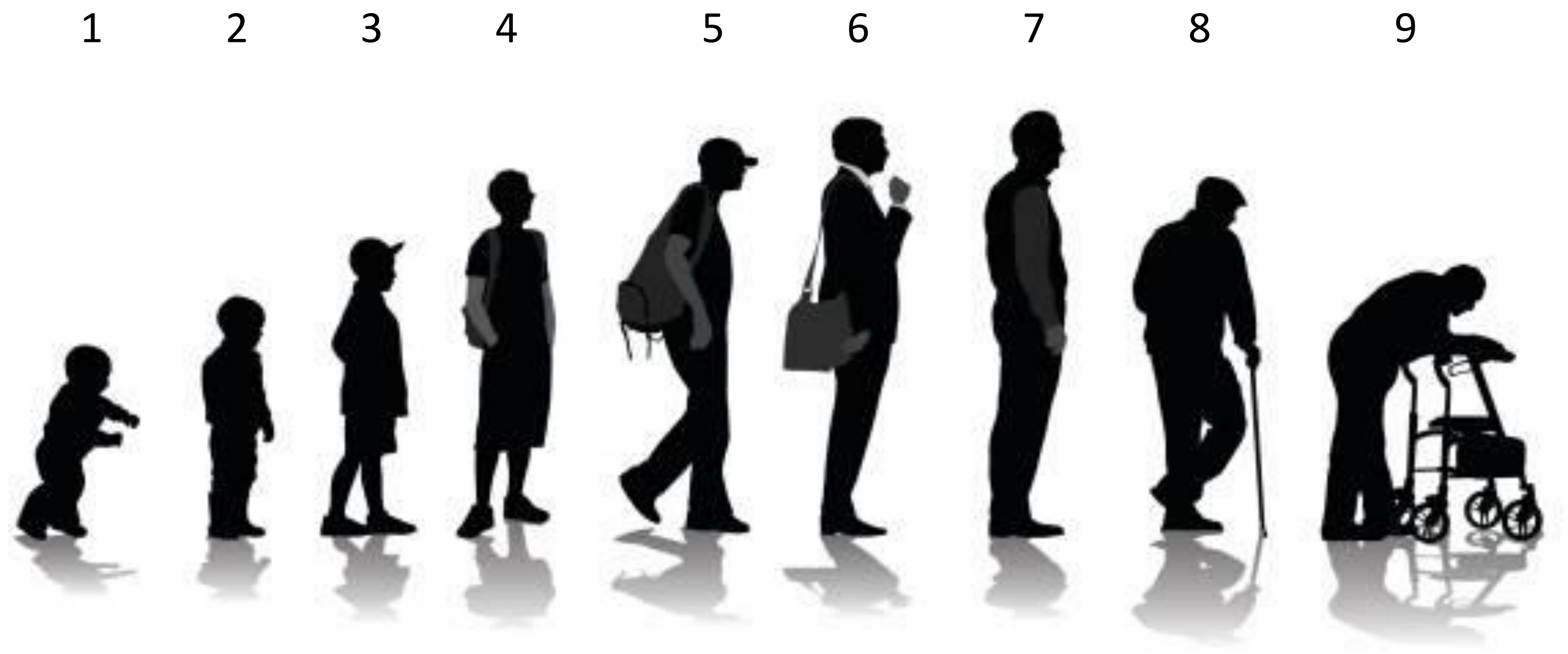
Features

Qualitative – categorical data (finite number of categories or groups)



Ordinal Data
(inherent order)

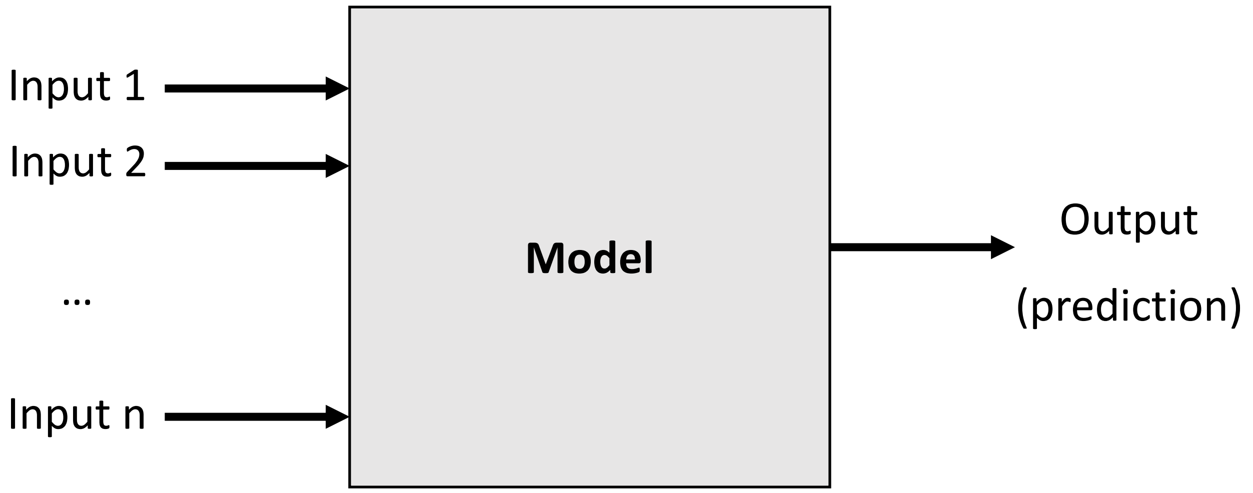
We can assign a number to any of these categories



Features

Quantitative – numerical valued data (discrete or continuous)

Machine Learning



Supervised Learning

Classification – predict discrete classes

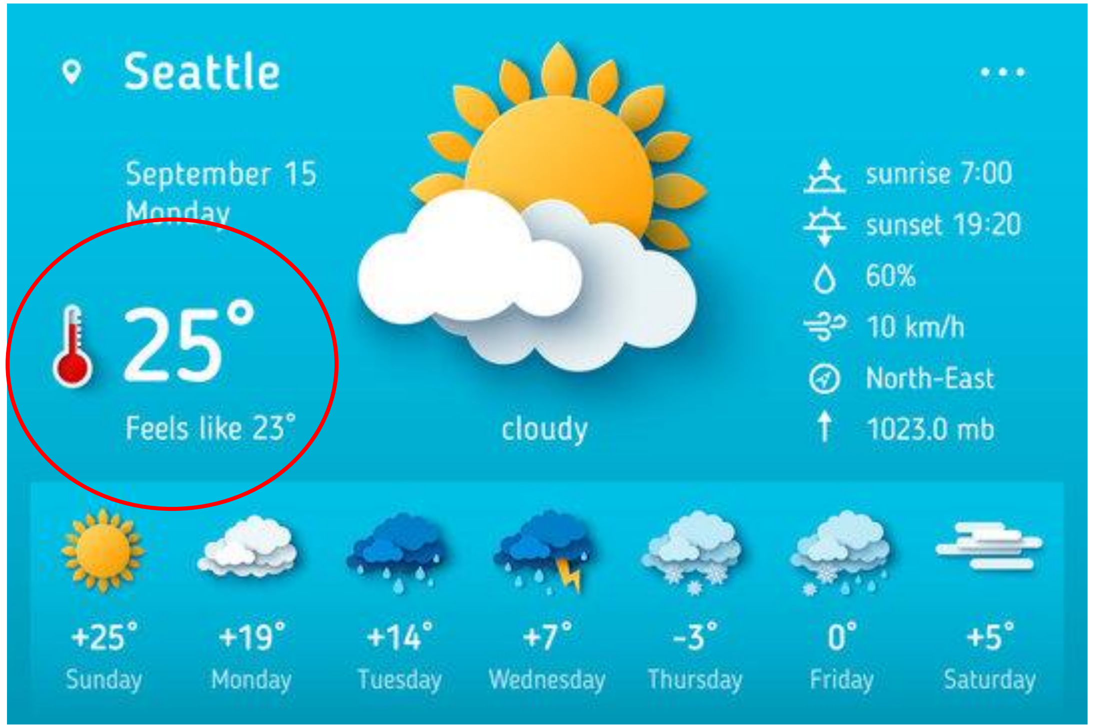
Multiclass classification

Classification – predict discrete classes

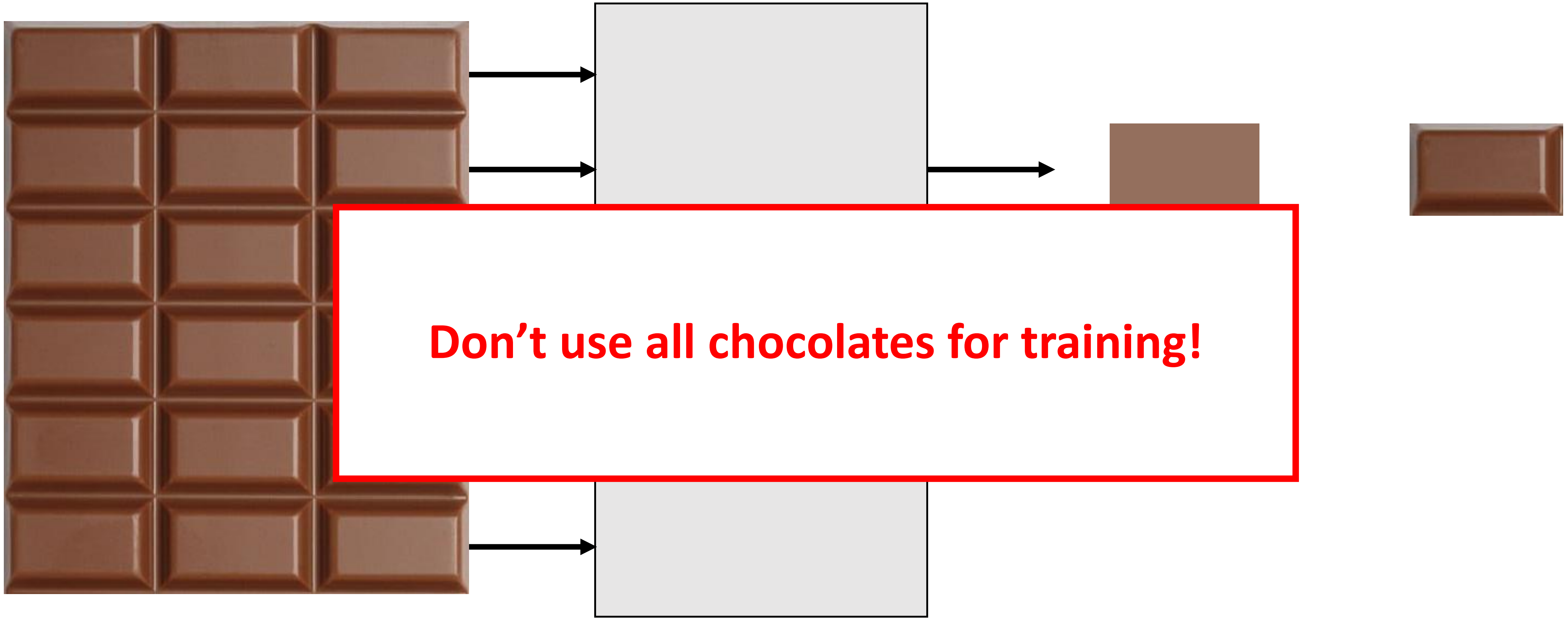
Binary classification

Supervised Learning

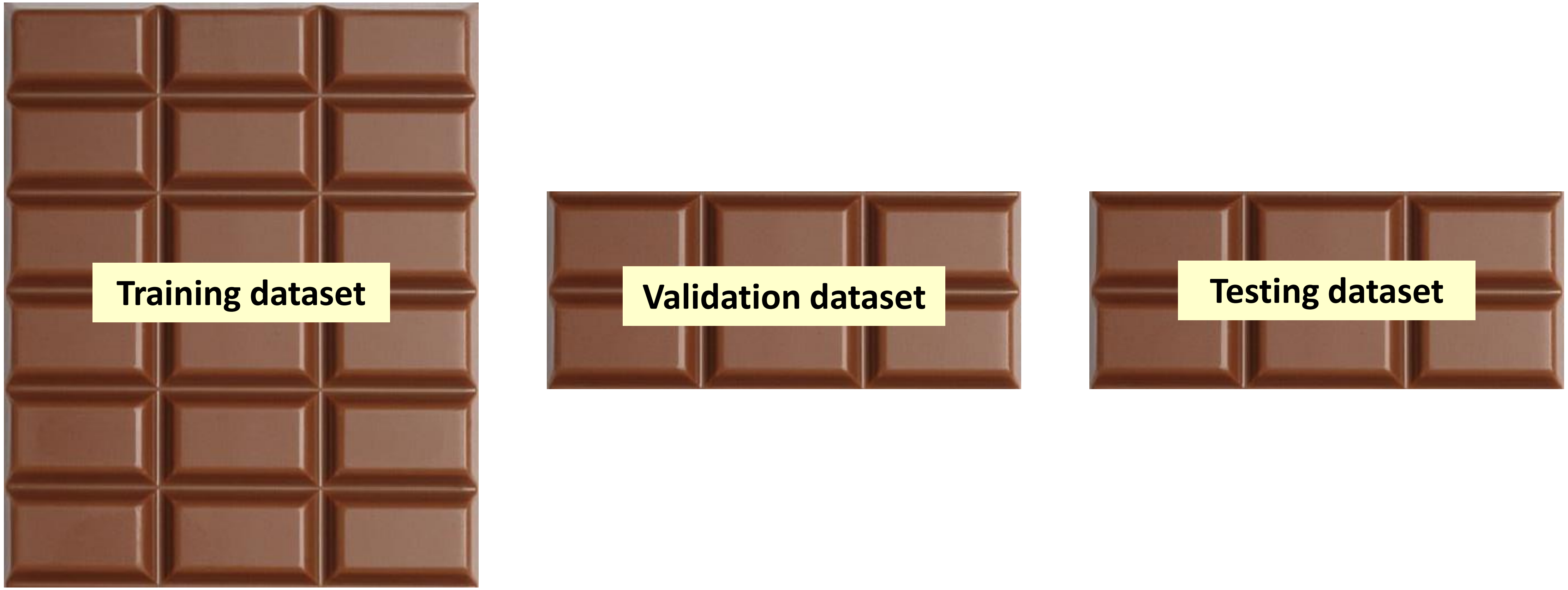
Regression – predict continuous values



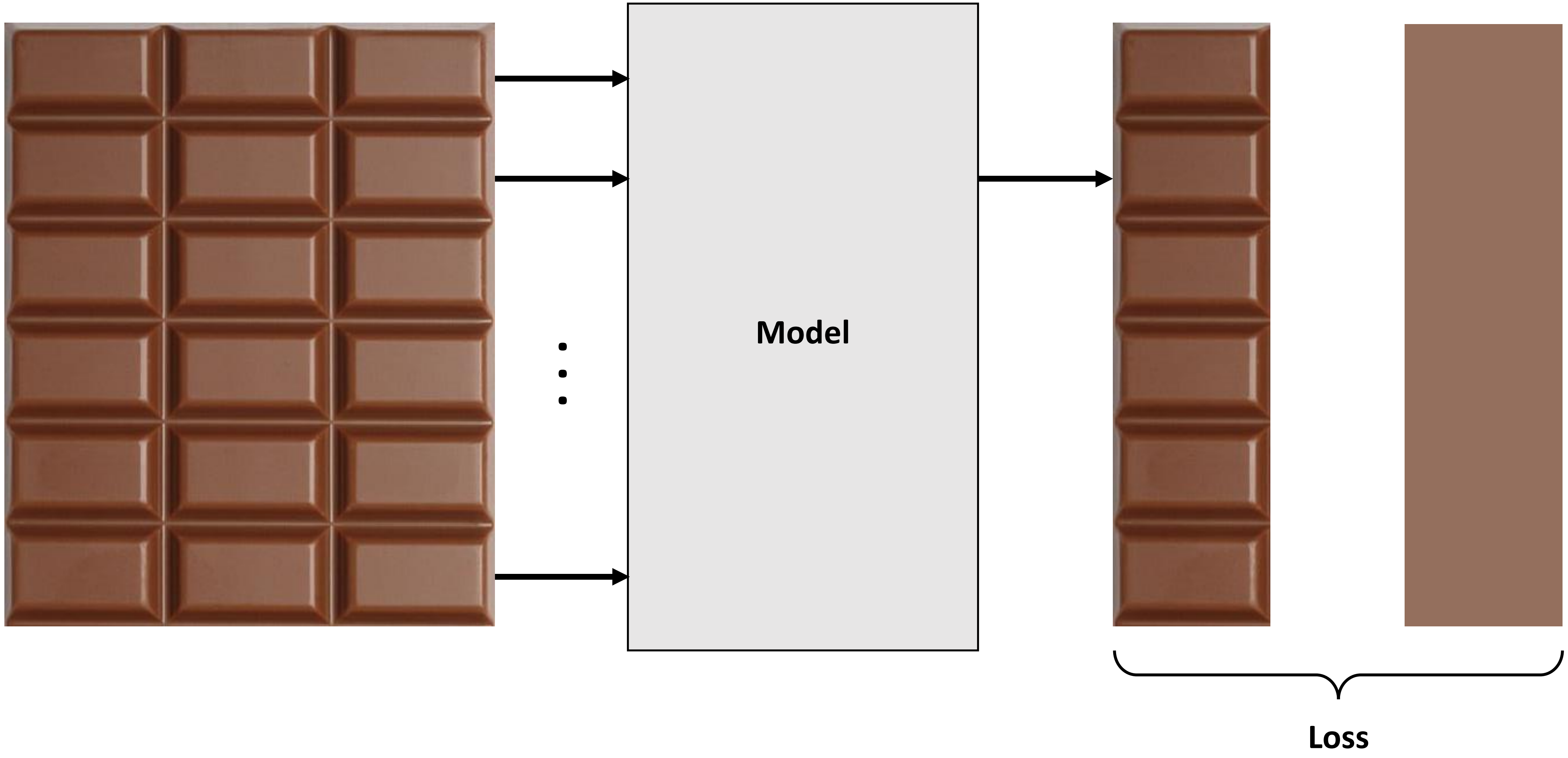
Training



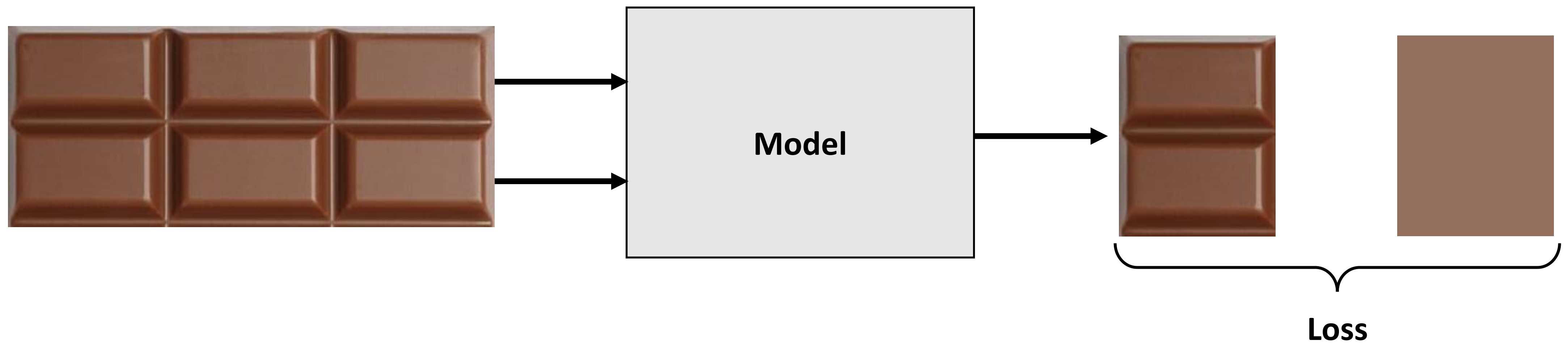
Don’t use all chocolates for training!


Training dataset
Validation dataset
Testing dataset

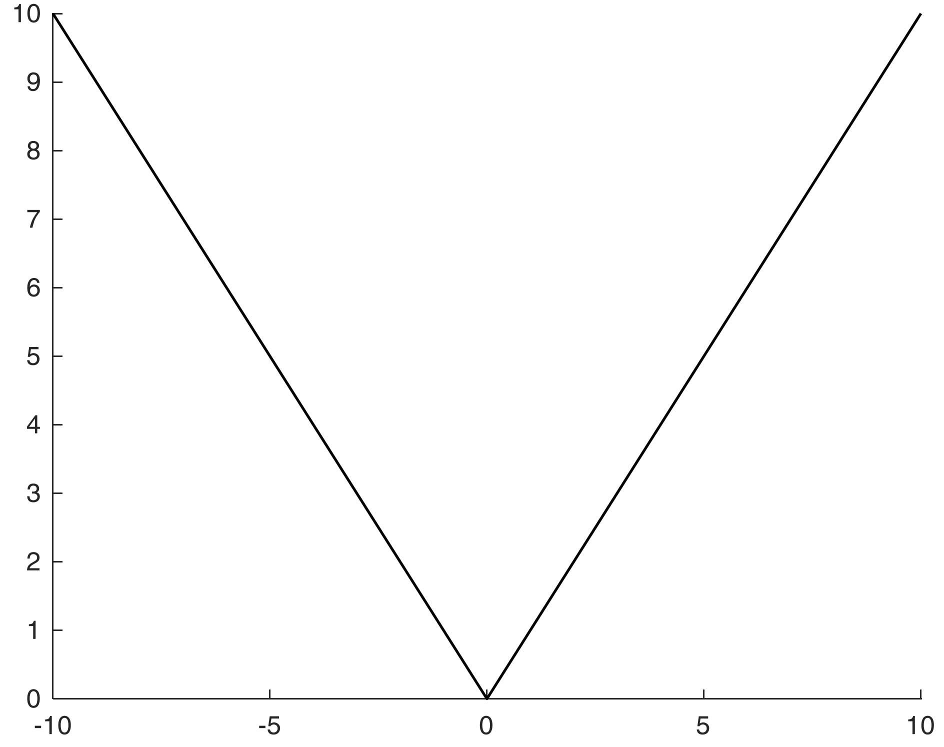
Training



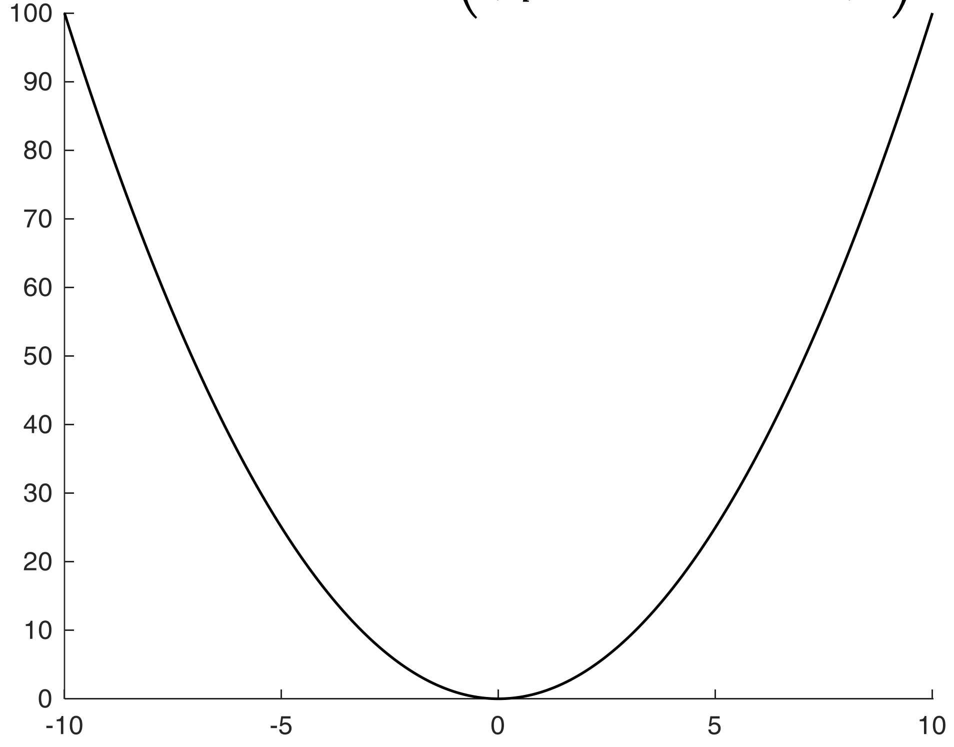
Validation



$$L_1\ Loss = sum\left(\left|Y_{predicted} - Y_{real}\right|\right)$$

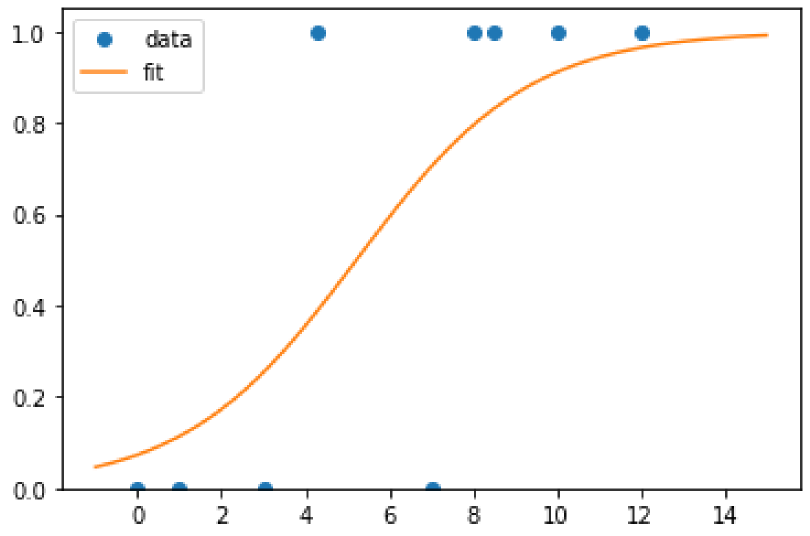


$$L_2\ Loss = sum\left(\left(Y_{predicted} - Y_{real}\right)^2\right)$$



Binary Cross-Entropy $Loss = -\frac{1}{N} * sum(y_{real} * \log\left(y_{predicted}\right) + (1 - y_{real}) * \log(1 - y_{predicted}))$

This loss function gives you the probability that each data point belongs to one of our two classes!



Types of Machine Learning Performance Metrics

Accuracy: is the ratio of correctly predicted instances (both true positives and true negatives) to the total number of instances. It measures the overall correctness of the model's predictions.

$$\frac{\text{true positives} + \text{true negatives}}{\text{total number of instances}}$$

Precision: measures the ratio of true positive predictions to the total number of positive predictions made by the model. It focuses on the accuracy of the positive predictions.

$$\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recall (Sensitivity or true positive rate): measures the ratio of true positive predictions to the total number of actual positives (both true positives and false negatives). It focuses on the model's ability to identify all positive instances.

$$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

F1 score: is the harmonic mean of precision and recall. It combines both metrics into a single measure that balances the trade-off between precision and recall, especially in cases where the dataset is imbalanced.

$$\frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

	positive	negative
true	True positive	True negative
false	False positive	False negative

Hands-on coding

Let’s start with looking into our data:

```
df = pd.read_csv("Session1Data.data")
```

The data comes from a telescope. There are some high-energy particles that the telescope is observing their radiation. The dataset includes some features recorded from these particles and their radiations that help us to understand if the particles were gamma or hadron.

```
cols = ["fLength", "fWidth", "fSize", "fConc", "fConc1", "fAsym", "fM3Long", "fM3Trans", "fAlpha", "fDist", "class"]
df = pd.read_csv("Session1Data.data", names=cols)
```

Variable Name	Role	Type	Description	Units	Missing Values
fLength	Feature	Continuous	major axis of ellipse	mm	no
fWidth	Feature	Continuous	minor axis of ellipse	mm	no
fSize	Feature	Continuous	10-log of sum of content of all pixels	#phot	no
fConc	Feature	Continuous	ratio of sum of two highest pixels over fSize		no
fConc1	Feature	Continuous	ratio of highest pixel over fSize		no
fAsym	Feature	Continuous	distance from highest pixel to center, projected onto major axis		no
fM3Long	Feature	Continuous	3rd root of third moment along major axis	mm	no
fM3Trans	Feature	Continuous	3rd root of third moment along minor axis	mm	no
fAlpha	Feature	Continuous	angle of major axis with vector to origin	deg	no
fDist	Feature	Continuous	distance from origin to center of ellipse	mm	no

To see the first 5 rows of our data:

```
df.head()
```

To see how the different elements in a data column use:

```
.unique()
```

```
df["class"].unique()
```

To convert qualitative (categorical) to quantitative (nominal), if we have only two categories:

```
df["class"] = (df["class"] == "g").astype(int)
```

If we have more than two categories:

```
df["class"] = df['class'].astype('category').cat.codes
```

Or, do the mapping manually:

```
labels = df["class"]
label_mapping = {uniqueClasses[0]:0, uniqueClasses[1]:1}
numerical_labels = [label_mapping[label] for label in labels]
df["class"]=numerical_labels
```

- First, convert the labels from letters to numbers.

1. `df["class"] = (df["class"] == "g").astype(int)`

2. `df["class"] = df['class'].astype('category').cat.codes`

3.
```
labels = df["class"]
label_mapping = {uniqueClasses[0]:0, uniqueClasses[1]:1}
numerical_labels = [label_mapping[label] for label in labels]
df["class"]=numerical_labels
```

We have collected 10 parameters (independent variables). Which one can represent and distinguish the two classes of our data the best?

- Plot histograms!

Python reminder:

df["A"][b]: element of df in column with label "A" and in row [b]

df[df[cond]]["B"]: elements of df in column with label "B", satisfying condition <u>cond</u>

df[df[cond]].iloc[b]: the bth row of subset of df satisfying condition <u>cond</u>

```
*plt.hist(df[df["class"]==1][fSize])
```

Which feature (parameter) do you think is the best for our classification task?

Now, divide data into train, validation and test sets

We can use numpy.split for this purpose

- np.split(data, [end of first portion, end of second portion, …])

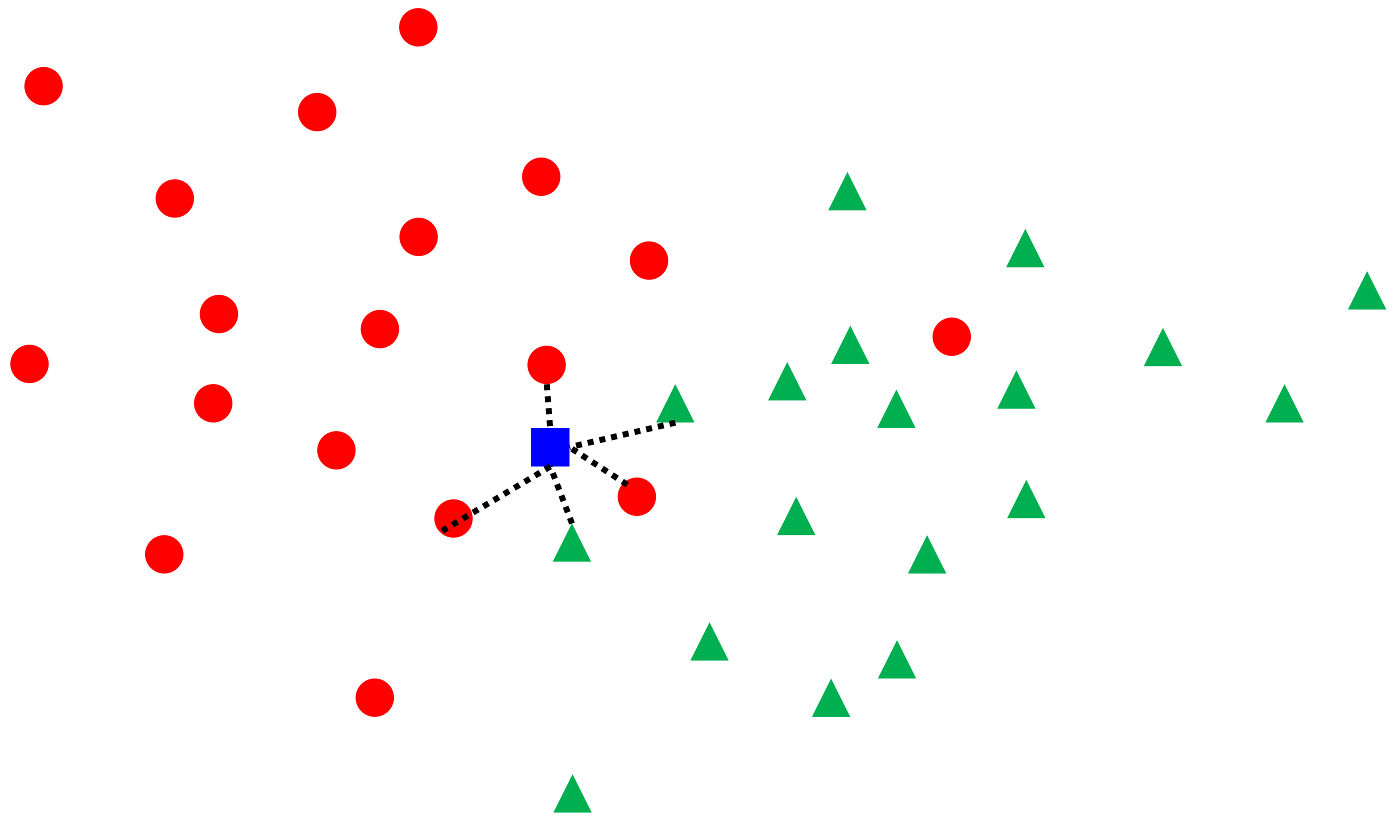
```
train, valid, test = np.split(df.sample(frac=1, replace=False), [int(0.6*len(df)), int(0.8*len(df))])
```

Scale, and oversample or undersample training data!

K nearest neighbors (KNN)

Basic Idea: The fundamental idea behind KNN is to predict the class of a data point by looking at the 'k' closest data points (neighbors) to that point. It assumes that similar data points are close to each other in the feature space.

Distance Metric: KNN relies on a distance metric to measure the similarity between data points. Common distance metrics include Euclidean distance, Manhattan distance, Minkowski distance, etc. Euclidean distance is the most commonly used metric in KNN.



Training: KNN doesn't explicitly train a model during the training phase. Instead, it memorizes the entire training dataset. This makes the training phase fast but the prediction phase potentially slow, especially for large datasets.

Choosing 'k': The choice of 'k' is crucial in KNN. A small 'k' value can make the model sensitive to noise, while a large 'k' value can lead to the inclusion of irrelevant data points. Typically, 'k' is chosen using cross-validation techniques to find the value that yields the best performance on unseen data.

Weighted KNN: In some cases, you may want to assign weights to the neighbors based on their distance to the query point. Closer neighbors are given higher weights, while farther ones are given lower weights. This helps in improving the accuracy of the prediction.

Regression with KNN: KNN can also be used for regression tasks. In regression, instead of predicting a class label, it predicts a continuous value by averaging the target values of the 'k' nearest neighbors.

Pros:

- Simple to implement.
- No assumption about the underlying data distribution.
- Can handle multi-class classification problems.
- It performs well with a small number of input variables.

Cons:

- Computationally expensive during the prediction phase, especially with large datasets.
- Sensitive to irrelevant features and the choice of distance metric.
- Requires a significant amount of memory to store the entire training dataset.
- Not suitable for high-dimensional data due to the curse of dimensionality.