Machine Learning Basics

8 January 2026

Alex Lyman

Machine Learning

- Machine learning uses algorithms and models to make predictions and discover patterns in data.
- A **model** is a mathematical function for describing the relationship between inputs and outputs and making predictions.
- An **algorithm** is a procedure or set of decision rules used to carry out a machine learning task, such as making a prediction.

Traditional Programming Paradigm

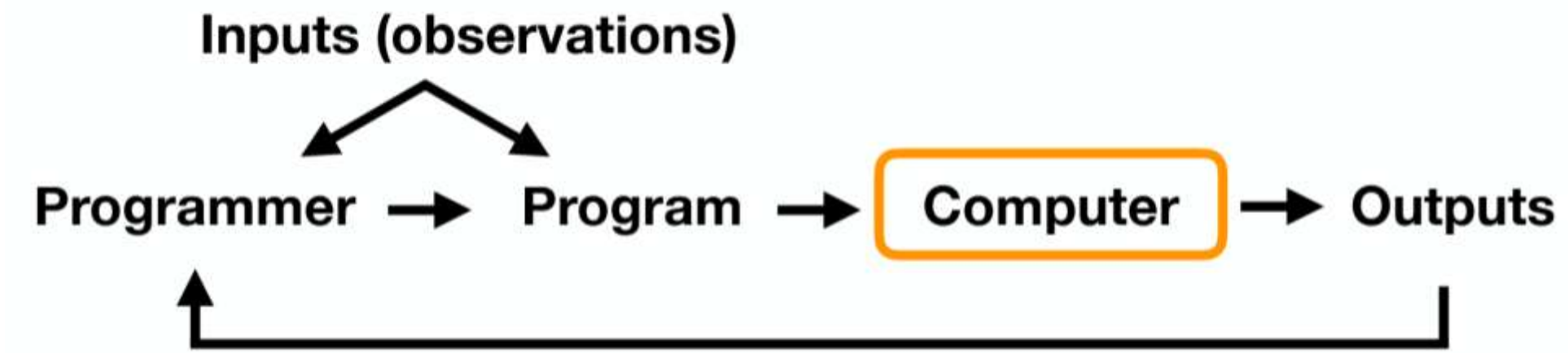


Image Credit: Sebastian Raschka

Machine Learning Paradigm

Image Credit: Sebastian Raschka

Types of Machine Learning - Supervised

- The machine learns from **labeled data** (input-output pairs). The algorithm is provided with the "correct answers" during training.
- Goal - to map input variables to an output variable effectively enough to predict the output for new, unseen data.
- **Examples -** Spam filtering, house price prediction, image classification.

Supervised Learning: Classification

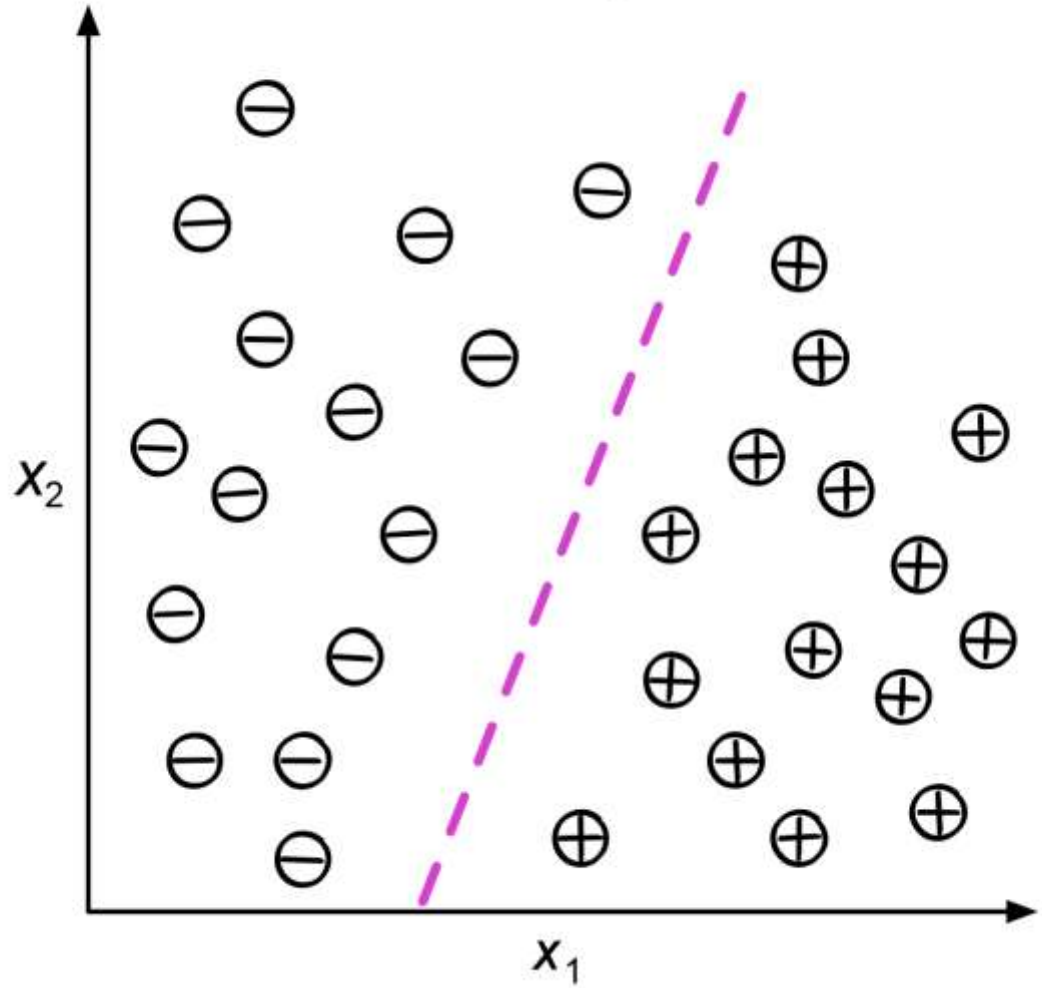


Image Credit: Sebastian Raschka

Supervised Learning: Regression

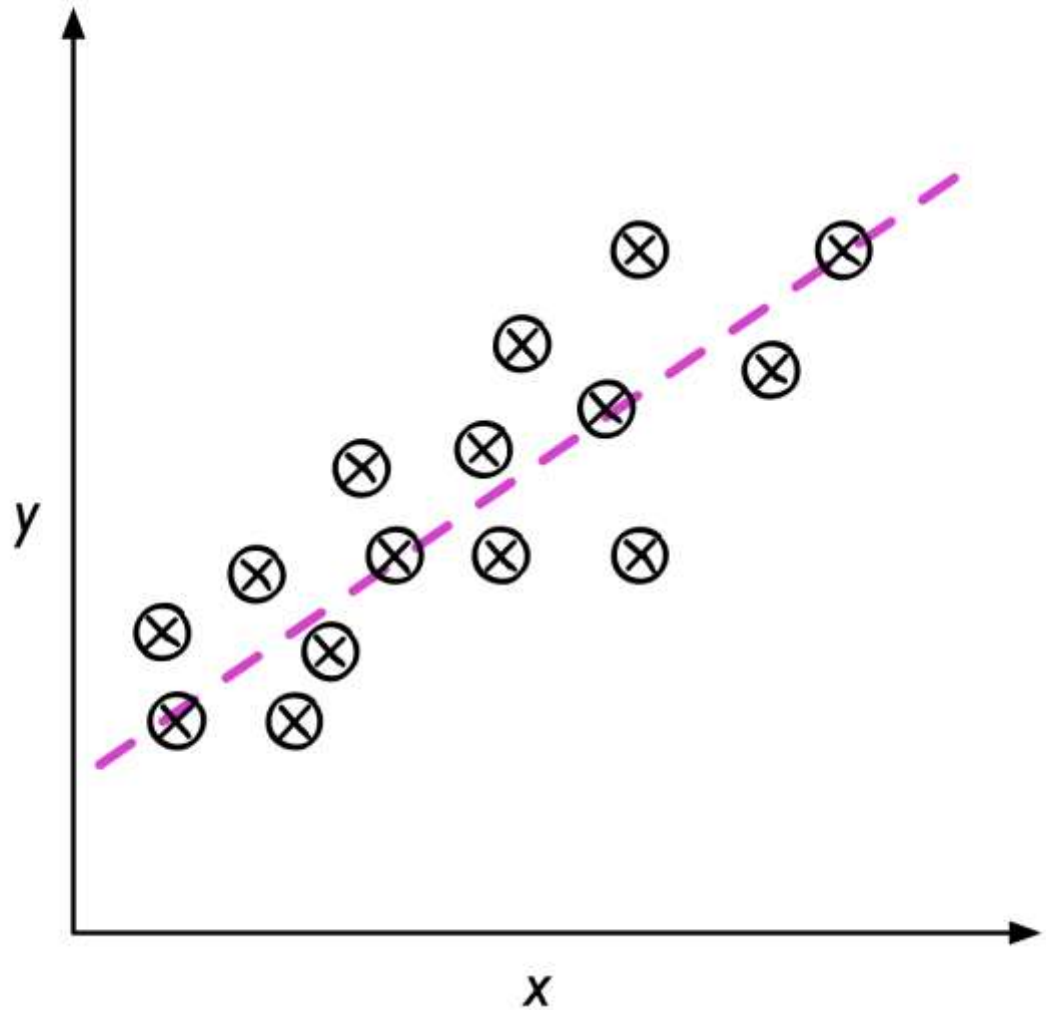


Image Credit: Sebastian Raschka

Types of Machine Learning - Unsupervised

- The machine learns from **unlabeled data**. The algorithm must explore the data to find structure or patterns on its own without guidance.
- Goal -To model the underlying structure or distribution in the data to learn more about it (e.g., grouping similar items).
- Examples – Clustering, segmentation, dimensionality reduction, anomaly detection, recommendation systems.

Unsupervised Learning -- Clustering

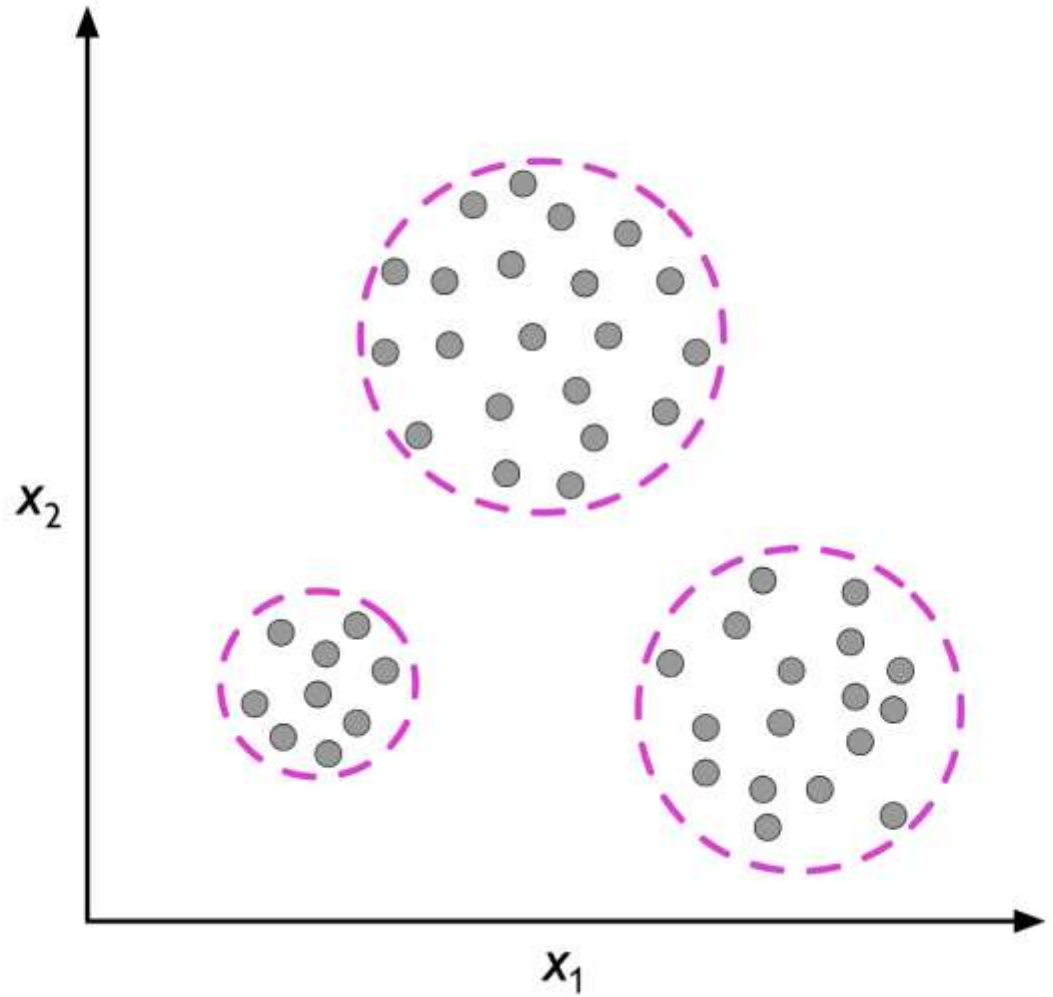


Image Credit: Sebastian Raschka

Unsupervised Learning
-- Dimensionality Reduction

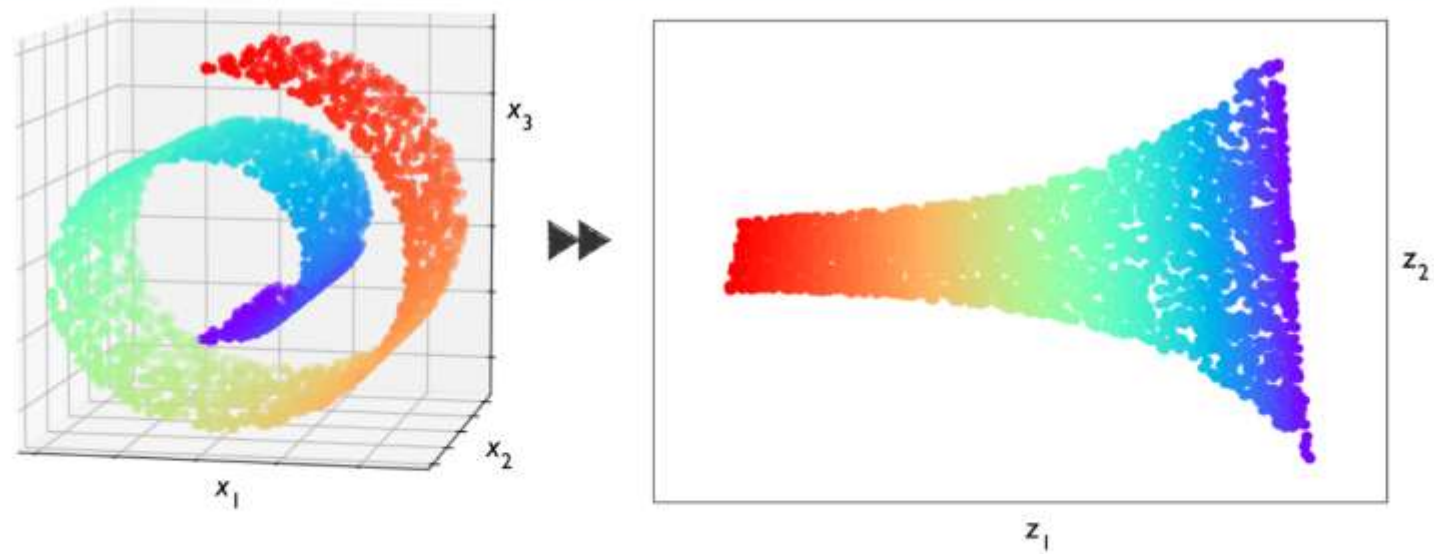


Image Credit: Sebastian Raschka

Types of Machine Learning - Reinforcement

- An agent learns to make decisions by interacting with an environment. It learns through a trial-and-error process based on feedback (rewards or penalties).
- Goal - To maximize the cumulative reward over time to achieve a specific goal.
- Examples - Game playing (e.g., Chess, Go), robotics, self-driving cars.

Train-Test Paradigm

- If the goal is to get our model to generalize to unseen data, how do we do that?
- We split our data into Train, Test, and Validation sets
- Train set - we use this to build our models.
- Test set – We use this to test the model
- Validation set – never seen while making the model, makes sure we generalize

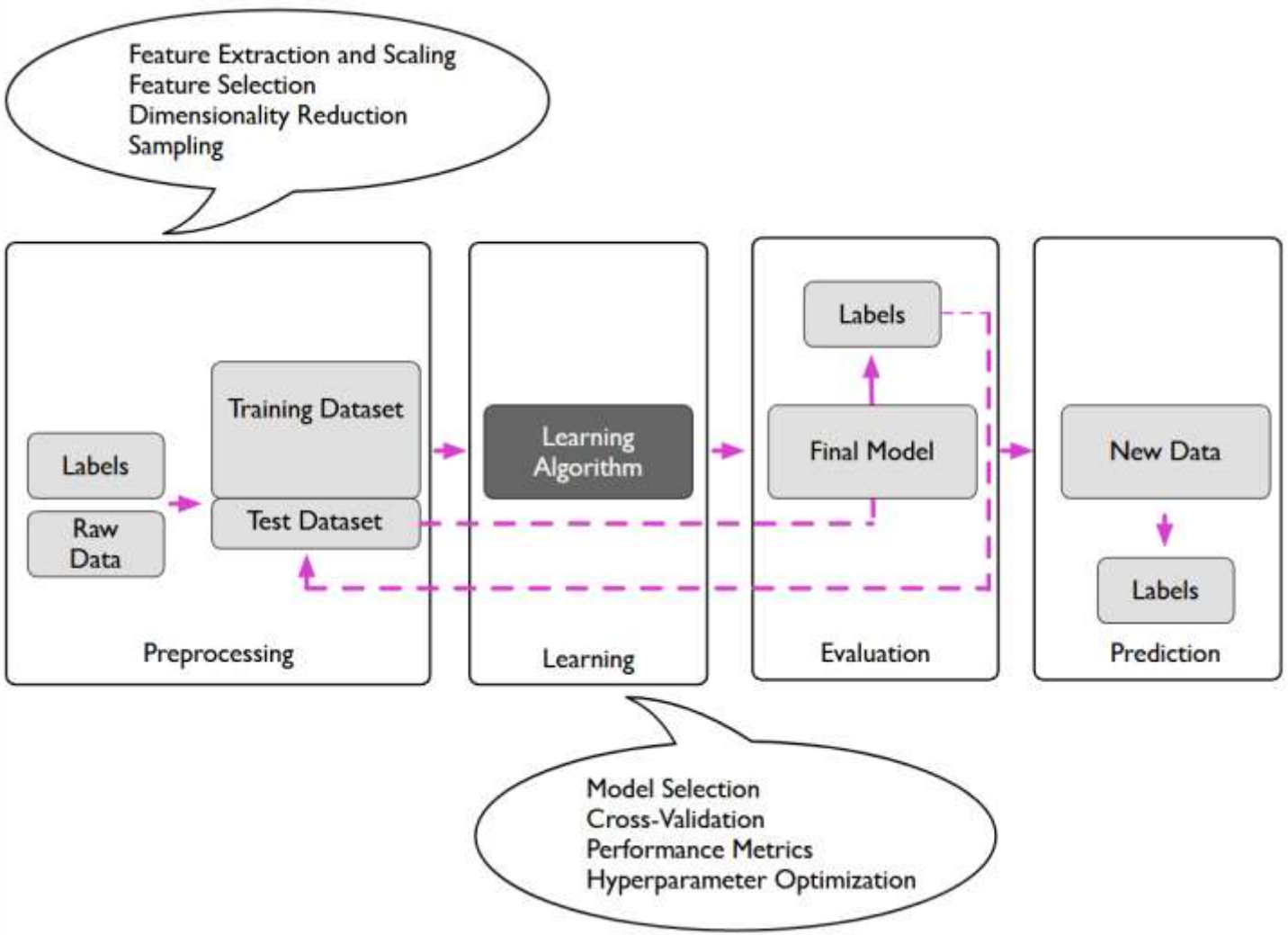


Image Credit: Sebastian Raschka

Machine Learning, AI, and Deep Learning

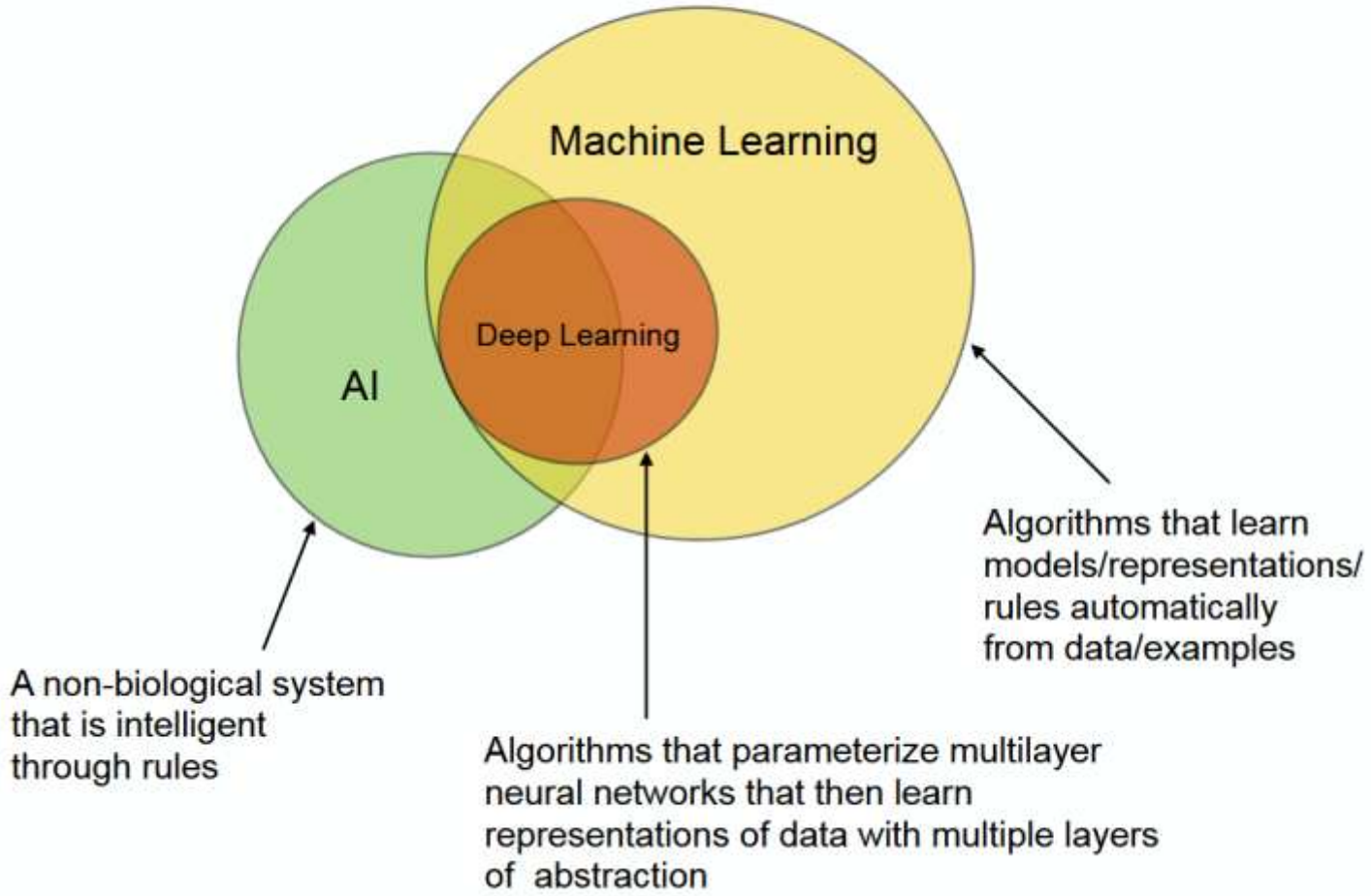


Image Credit: Sebastian Raschka

Data

Getting to Know the Data (I)

- Metadata
  - Attribute types:
    - binary, nominal (categorical), ordinal, numeric, …
  - Attribute roles:
    - Input, target, id, …
  - Attribute descriptions
- Simple visualization tools are very useful
  - Nominal attributes: histograms (Distribution consistent with background knowledge?)
  - Numeric attributes: graphs (Any obvious outliers?)
    - DO NOT UNDERESTIMATE

Sweetviz

```
In [2]: import pandas as pd
        import sweetviz
        train = pd.read_csv(r"C:\Users\lenovo\Desktop\train.csv")
        test = pd.read_csv(r"C:\Users\lenovo\Desktop\test.csv")

In [3]: my_report = sweetviz.analyze([train, "Train"],target_feat='SalePrice')

        Done! Use 'show' commands to display/save. [100%] 00:07 -> (00:00 left)
```

Getting to Know the Data (II)

- Data quantity
  - Number of instances
    - If too few, results less reliable; use special methods (boosting, …)
  - Number of features
    - If too many, use feature reduction/selection
  - Number of targets
    - If very unbalanced, use stratified sampling
- Data relevance
  - What data is available for the task?
  - Is this data relevant?
  - Is additional relevant data available?
  - How much historical data is available?
  - Who are the data experts?

Attribute Types

- Nominal
  - E.g., eye color={brown, blue, …}
  - No relation, ordering, or distance implied
  - Only equality tests
- Ordinal
  - E.g., grade={k, 1, …, 12}, height = {tall > med > short}
  - Order BUT no distance
- Continuous (numeric)
  - Interval quantities – integer (e.g., year)
    - Difference makes sense, not sum/product
  - Ratio quantities – real (e.g., length)
    - Measurement scheme defines 0 point, all operations allowed

Type Conversion

- Some algorithms require/fare better with continuous inputs (neural nets, regression, nearest neighbor)
- Some algorithms require discrete inputs (most versions of naïve Bayes, decision trees)
- Different encodings likely to produce different results
- Only show some here

Sample Conversions

- Ordinal-to-Boolean
  - Temp: cold, medium, hot => Temp>cold: 0/1, Temp>medium: 0/1
  - Thermometer/unary encoding (X => X 1s followed by 0)
- Binary-to-Numerical
  - Gender: M/F => 0/1
- Ordinal-to-Numerical (order must be preserved)
  - Grade: A, A-, … => 4.0, 3.7, …
- Nominal-to-Numerical
  - One-hot encoding
- Numerical-to-Nominal (e.g., discretization)

One-Hot Encoding

- Used for Nominal to Numerical conversions
  - Book genre to number
- Used because some models require numbers
- Can't just convert to a number
  - Number order implies order/level of the nominal value
  - All genres should be equally weighted
- Define a feature for each nominal value
- Feature value is 1 or 0 indicating whether it belongs to that class

Python

```python
import pandas as pd

df = pd.DataFrame({'color': ['red', 'blue', 'green', 'red']})

# Create dummy variables
df = pd.get_dummies(df, columns=['color'])

print(df)
```

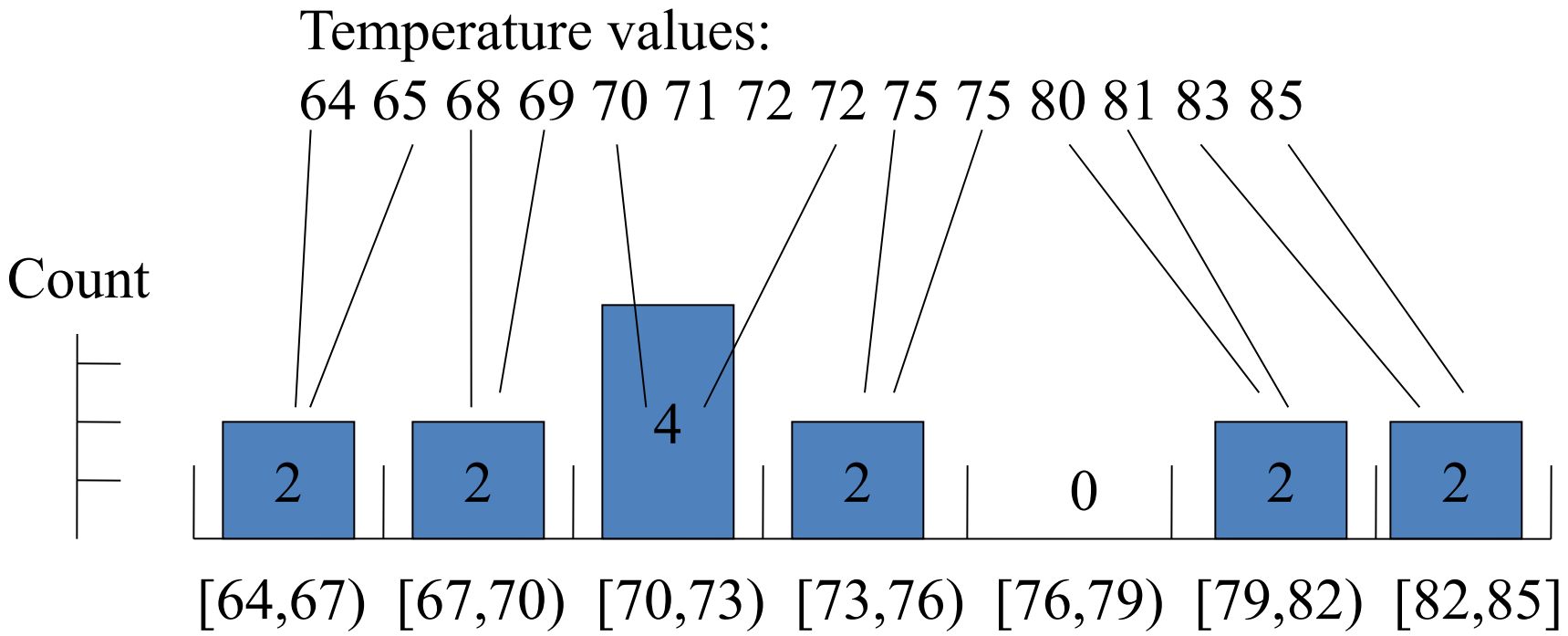
One-Hot Encoding

Fruit	Categorical value of fruit	Price
apple	1	5
mango	2	10
apple	1	15
orange	3	20

The output after applying one-hot encoding on the data is given as follows,

Fruit_apple	Fruit_mango	Fruit_orange	price
1	0	0	5
0	1	0	10
1	0	0	15
0	0	1	20

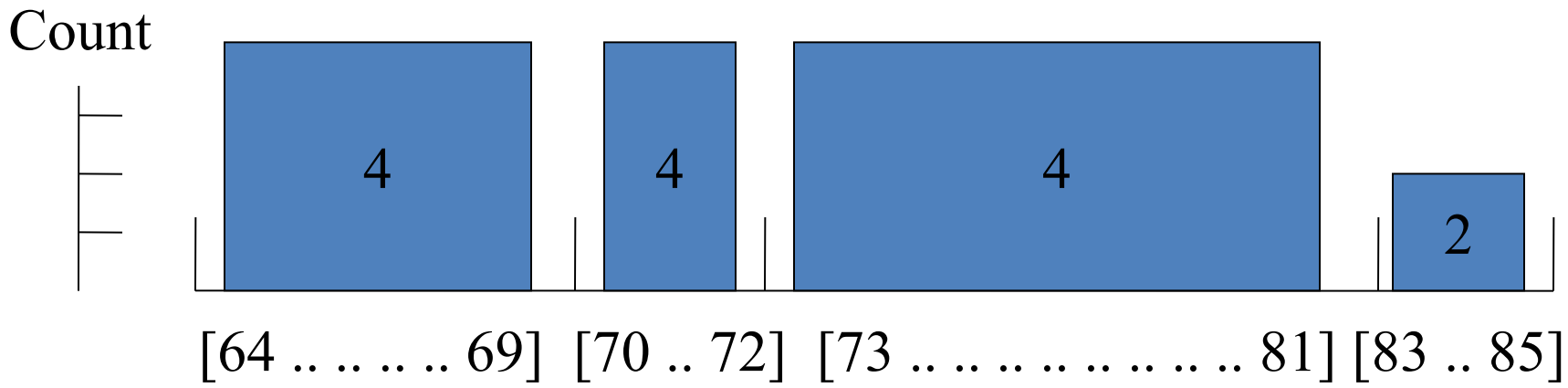
Discretization: Equal-Width



- May produce clumping if data is skewed

Discretization: Equal-Height

Temperature values:
64 65 68 69 70 71 72 72 75 75 80 81 83 85



- Gives more intuitive breakpoints
  - don't split frequent values across bins
  - create separate bins for special values (e.g., 0)

Standardization and Normalization

- Assume two input features in an astronomical task as follows:
  - Weight of the planet in grams, Diameter of the planet in light-years
- Solutions:
  - Standardization
    - –Transform values into number of standard deviations from the mean

$$new\ value = \frac{current\ value\ - average}{standard\ deviation}$$

  - Normalization
    - –Transform values to fall within a specified range

$$new\ value = \frac{current\ value\ - \min value}{range}$$

    - –Often, range = max value – min value => [0, 1]

Precision “Illusion”

- Example: gene expression may be reported as X83 = 193.3742, but measurement error may be +/- 20
- Actual value is in [173, 213] range, so it may be appropriate to round the data to 190
- Do not assume that every reported digit is significant!

Missing Values

Missing Values (I)

- Types: unknown, unrecorded, irrelevant
  - malfunctioning equipment
  - changes in experimental design
  - collation of different datasets
  - measurement not possible

Name	Age	Sex	Pregnant	…
Mary	25	F	N	
Jane	27	F	-	
Joe	30	M	-	
Anna	2	F	-	

In medical data, value for **Pregnant** attribute for **Jane** is missing, while for **Joe** or **Anna** should be probably be considered **Not applicable**

Missing Values (II)

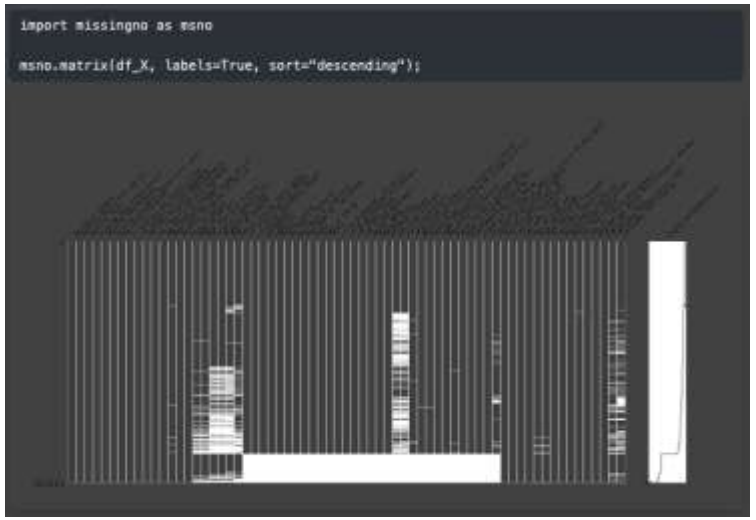
- Handling methods:
  - Remove records with missing values
  - Treat as separate value
  - Treat as don’t know
  - Treat as don’t care
  - Use imputation techniques
    - Mode, Median, Average
    - Regression (i.e., predict based on others)
  - Danger: BIAS
- Python missingno package

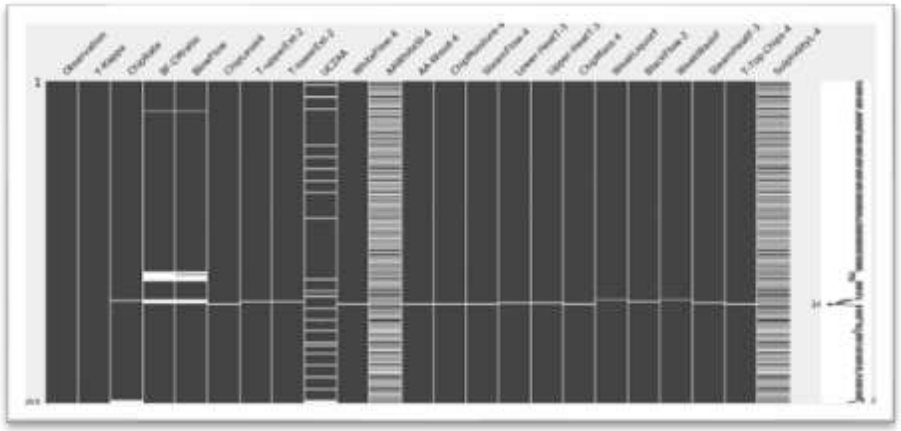
Quality Investigation – Missing Values

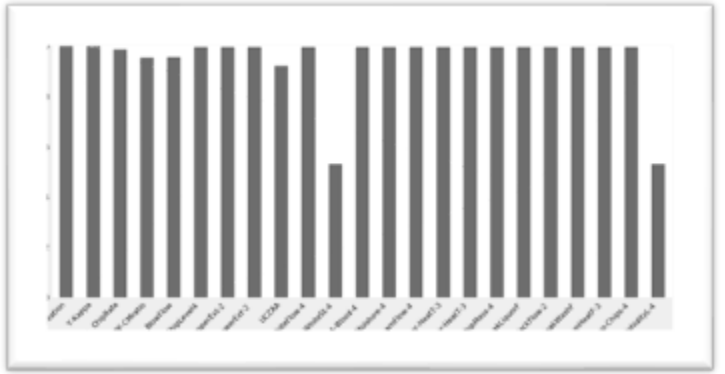
- missingno package

```
import missingno as msno

msno.matrix(df)
msno.bar(df)
```







Missing Data

- Structurally missing
  - Missing because it does not exist (# pregnant men)
- Missing completely at random
  - Whether or not a value is missing is unrelated to its value or the value of any other variable
  - Missing values can be excluded or imputed with mean or median
- Missing at random
  - Whether or not a value is missing does not depend on its value after controlling for another variable
  - Missing values can be predicted from other variables
- Missing not at random
  - Missing values that are not MCAR or MAR
  - Hardest type of missingness

For more information: https://www.uvm.edu/~statdhtx/StatPages/Missing_Data/Missing.html

Outliers and Errors

- Outliers are values thought to be out of range
- Approaches:
  - Do nothing
  - Enforce upper and lower bounds
  - Let binning/discretization handle the problem

Feature Issues

Feature Issues

- Typically do not use features where
  - Almost all instance have the same value (no information)
    - If there is a significant, though small, percentage of other values, then might still be useful
  - Almost all instances have unique values (e.g., SSN, phone-numbers)
    - Might be able to use a variation of the feature (such as area code)
  - The feature is highly correlated with another feature
    - In this case the feature may be redundant and only one is needed
  - Careful if feature is too highly correlated with the target (leaky)
    - Check this case as the feature may just be a synonym with the target and will thus lead to overfitting (e.g., the output target was bundled with another product so they always occurred together)

Improving Performance

- When trying to improve performance, you may need
  - More data
  - Better features
  - Different ML models or hyperparameters
  - Etc.
- One way to decide if you need more/better data
  - Compare your accuracy on training and test set
    - bad training set accuracy, probably need better data/ features. (though might need a different learning model/hyperparameters)
    - test set accuracy much worse than training set accuracy, gathering more data usually a good direction. (though regularization or learning model/hyperparameters could still be significant issue)

Class Imbalance (I)

- Sometimes, class distribution is skewed
  - Monthly attrition: 97% stay, 3% defect
  - Medical diagnosis: 90% healthy, 10% disease
  - eCommerce: 99% don't buy, 1% buy
  - Security: >99.99% of Americans are not terrorists
- Similar situation with multiple classes
- Majority class classifier can be 97% correct, yet completely useless

Class Imbalance (II)

- Two class data
  - Undersample (select desired number of minority class instances, add equal number of randomly selected majority class)
  - Oversample (select desired number of majority class, sample minority class with replacement)
  - Use boosting, cost-sensitive learning, etc.
- Generalize to multiple classes
  - Approximately equal proportions of each class in training and test sets (stratification)

Performance Measures

Performance Measures

- There are a number of ways to measure the performance of a learning algorithm:
  - Predictive accuracy of the induced model (or error)
  - Size of the induced model
  - Time to compute the induced model
  - etc.
- We will focus mostly on
  - accuracy/error
  - Precision and recall
- Fundamental Assumption:

Future novel instances are drawn from the same/similar distribution as the training instances

Confusion Matrix

- Used mostly for binary classification problems
  - Can be extended to multiple classes
- Enumerates the outcomes
- Calculate performance metrics from the values

Actual Class

Predicted Class	Positive	Negative
Positive	***True Positive***	***False Positive***
Negative	***False Negative***	***True Negative***

Evaluation of Classification

predicted outcome \ actual outcome	1	0
1	*a*	*b*
0	*c*	*d*

Accuracy = (a+d) / (a+b+c+d)

- Not always the best choice
  - Assume 1% fraud,
  - model always predicts no fraud
  - What is the accuracy?

Actual Class

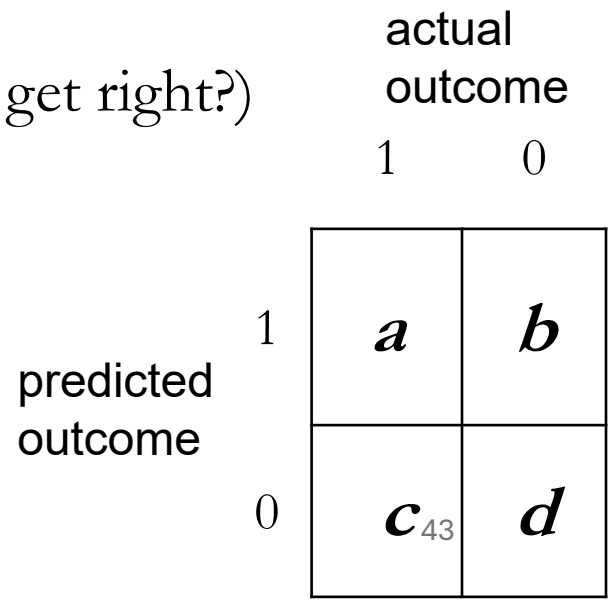
Predicted Class

	Fraud	No Fraud
Fraud	**0**	**0**
No Fraud	**10**	**990**

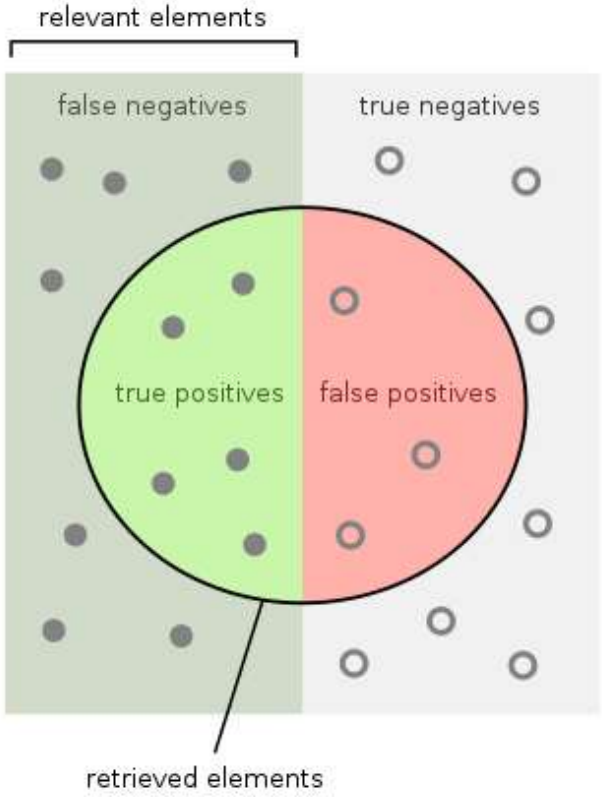
Evaluation of Classification

Other options:

- **recall** or sensitivity (how many of those that are really positive did you predict?):
  - a/(a+c)
- **precision** (how many of those predicted positive really are?)
  - a/(a+b)
- **Specificity** (how many of the negatives did you get right?)
  - d/(b+d)



		Actual Class: Positive	Actual Class: Negative
Predicted Class	Positive	***True Positive***	***False Positive***
	Negative	***False Negative***	***True Negative***

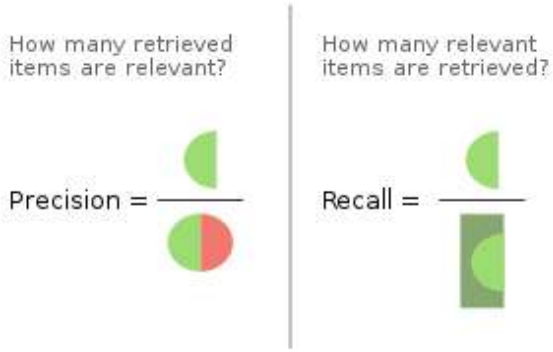


how many of those predicted positive really are?

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

how many of those that are really positive did you predict?



So What?

- Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.
  - fraud detection - If a fraudulent transaction (Actual Positive) is predicted as non-fraudulent (Predicted Negative), the consequence can be very bad for the bank.
  - sick patient detection - If a sick patient (Actual Positive) goes through the test and predicted as not sick (Predicted Negative). The cost associated with False Negative will be extremely high if the sickness is contagious.
- Precision is a good measure to determine, when the costs of False Positive is high.
  - email spam detection - In email spam detection, a false positive means that an email that is non-spam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.

F1

- Together, precision and recall give more information than just accuracy.
- Precision and recall are always in tension
  - Increasing one tends to decrease another
- Can combine for a single measure – F1
  - F1 = 2 * precision * recall / (precision + recall)
- Harmonic mean of precision and recall
  - Use harmonic mean when units are different
  - Eg. - Average MPH over different distances