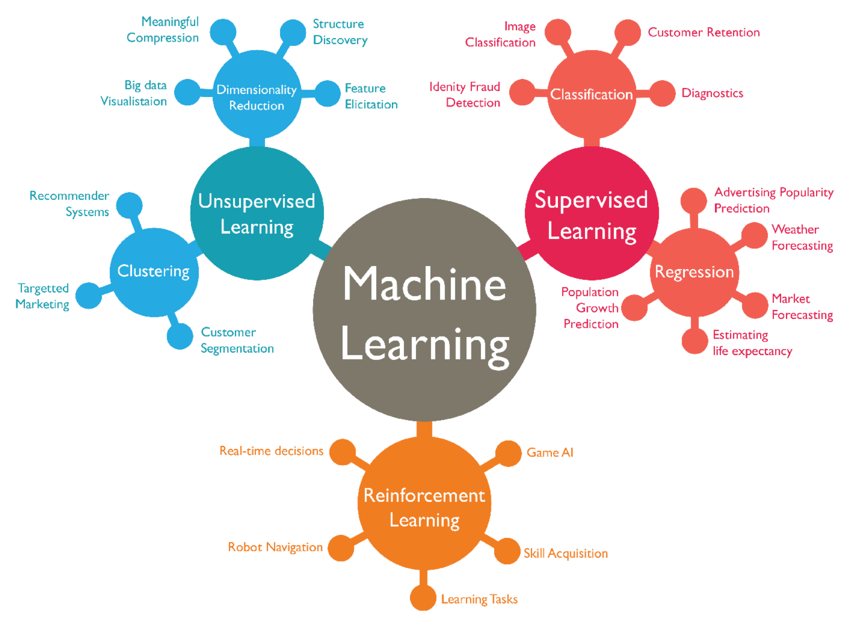
Machine Learning

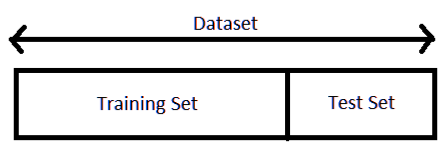
- ML is a technique followed to make a computer learn from the previous experience and make an assumption for the future outcome.
- It can learn and adapt to the new data without any human intervention.
- It needs prior training so that it can be tested to the new data.



Credits - Image from Internet (www.favouriteblog.com)

ML Dataset

In machine learning, we divide the dataset into two.



- **Training Data** - Here we train the machine learning model by showing both inputs and outputs.
- **Testing Data** - Here we test the model where inputs (new data) are not mapped with outputs. We will check the model performance in terms of accuracy.

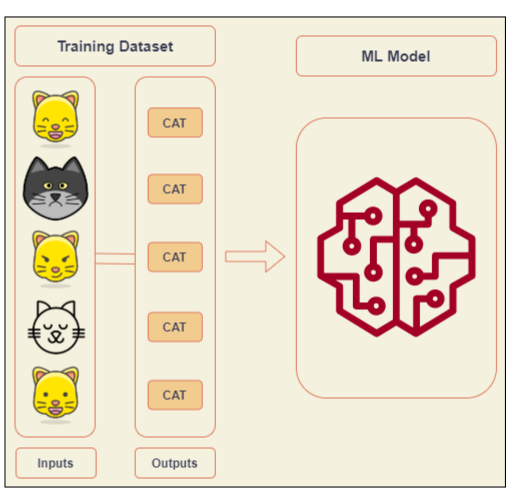
Credits - Image from Internet

Supervised Learning

The computer is presented with both example inputs and their respective outputs. The algorithm learns a general rule to map the inputs to the outputs.

Training stage

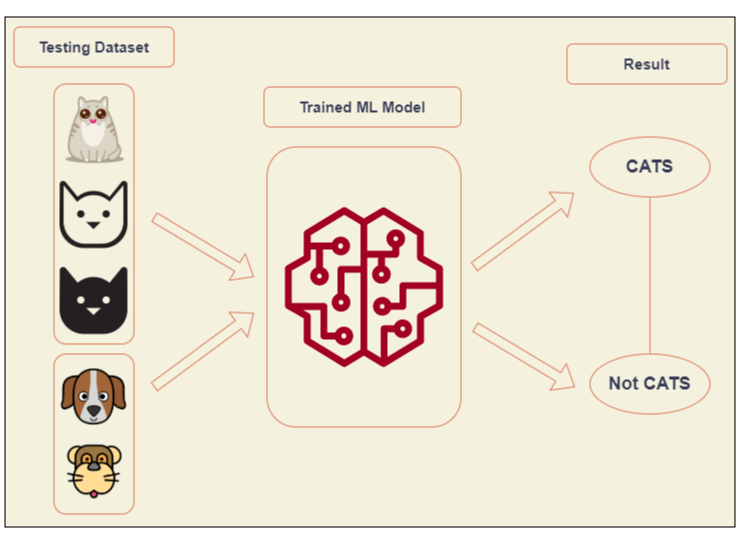
We show the model a set of inputs along with the respective outputs. The task of the model is to learn by mapping the inputs and outputs.



The model is trained from the dataset that is fed. It will completely learn from it.

Testing stage

We show the model a set of new inputs without the respective outputs. The aim of the model is to predict based on the learning it had undergone.



The model predicts the category based on the previous training or learning.

Images by Author

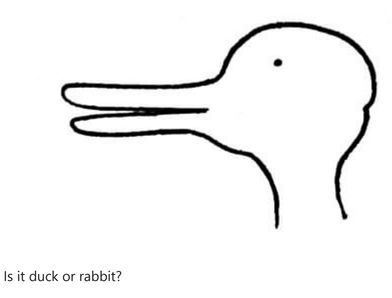
Note

- Algorithms learn from data.
- They find -
  - relationships
  - develop understanding
  - make decisions
  - evaluate their confidence from the training data they are given.
- The better the training data is, the better the model performs.

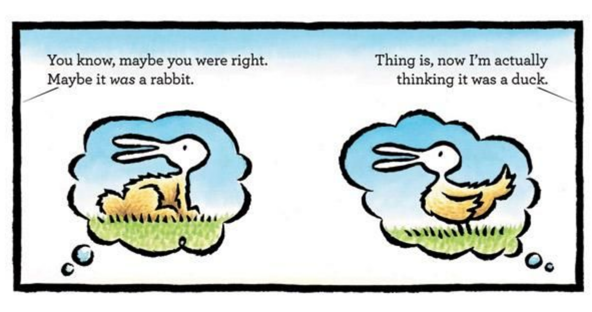
Exception case for the above example

1. Suppose I have trained my model to identify/separate `duck` and `rabbit` images from the large dataset.

1. Now, I need to test my model with the new data.
   - New image



   - Is it duck or rabbit?



2. What can you say about this?

Credits - Images from Internet

Important Terminology

- **Conditional Probability**
  - In probability theory, **conditional probability** is a measure of the probability of an event occurring, given that another event (by assumption, presumption, assertion or evidence) has already occurred.
  - If the event of interest is **A** and the event **B** is known or assumed to have occurred, "the conditional probability of A given B", or "the probability of A under the condition B", is usually written as $P(A|B)$, or sometimes $P_B(A)$ or $P(A/B)$.
  - Example - https://en.wikipedia.org/wiki/Conditional_probability

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- **Bayes Theorem**
  - If we have two events such as **A** and **B** then

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}; \text{ if } P(B) \neq 0$$

  - Here
    - $P(A|B) \implies \text{Posterior}$
    - $P(B|A) \implies \text{Likelihood}$
    - $P(A) \implies \text{Prior}$
    - $P(B) \implies \text{Evidence}$

- **Naive-Bayes Algorithm**
  - Categorical NB → https://bit.ly/3eSCr9J
  - Blog link → https://bit.ly/373Ghs4
  - Formula

$$\hat{y} = \text{argmax}_{k \in \{1,2,\ldots,k\}} P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

  - Here
    - $\hat{y} \implies \text{predicted class label (category)}$
    - $x_i \implies \text{each category in feature x}$
    - $C_k \implies \text{each class label (target)}$

`import` Packages

In [1]:
```python
import pandas as pd
import numpy as np
```

Data Reading

- Data source → http://www.shatterline.com/MachineLearning/data/tennis_anyone.csv

In [2]:
```python
data_source = 'http://www.shatterline.com/MachineLearning/data/tennis_anyone.csv'
df = pd.read_csv(data_source)
df.columns = ['outlook', 'temp', 'humidity', 'wind', 'class']
```

In [3]:
```python
df.head()
```

Out[3]:

	outlook	temp	humidity	wind	class
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes

Features and Target

In [4]:
```python
y = df['class']
X = df.drop(columns=['class'], axis=1)
```

In [5]:
```python
X.head()
```

Out[5]:

	outlook	temp	humidity	wind
0	Sunny	Hot	High	Weak
1	Sunny	Hot	High	Strong
2	Overcast	Hot	High	Weak
3	Rain	Mild	High	Weak
4	Rain	Cool	Normal	Weak

In [6]:
```python
y.head()
```

Out[6]:
```
0     No
1     No
2    Yes
3    Yes
4    Yes
Name: class, dtype: object
```

Likelihood

In [7]:
```python
def feature_probability(X_features, y_target, f, yt_kv):
    ddf = pd.DataFrame({f : X_features[f].values, 'class' : y_target.values})
    xf = X_features[f]
    xf_c = xf.value_counts().to_frame().index.to_list()

    each_x = {}
    for xi in xf_c:
        df_x = ddf[ddf[f] == xi]
        each_xy = {}
        for yi in list(yt_kv.keys()):
            df_xy = df_x[df_x['class'] == yi]
            each_xy[yi] = len(df_xy) / yt_kv[yi]
        each_x[xi] = each_xy

    return each_x
```

In [8]:
```python
yt_kv = {'Yes': 9, 'No': 5}
feature_probability(X_features=X, y_target=y, f='wind', yt_kv=yt_kv)
```

Out[8]:
```
{'Weak': {'Yes': 0.6666666666666666, 'No': 0.4},
 'Strong': {'Yes': 0.3333333333333333, 'No': 0.6}}
```

In [9]:
```python
def compute_likelihood(X_features, y_target):
    yt = y_target.value_counts().to_frame()
    yt_k = yt.index.to_list()
    yt_v = yt.values[:, 0]
    yt_kv = {i : j for (i, j) in zip(yt_k, yt_v)}

    X_likelihood = {}
    for col in X_features:
        X_likelihood[col] = feature_probability(
            X_features=X_features,
            y_target=y_target,
            f=col,
            yt_kv=yt_kv
        )
    y_likelihood = {i : j / np.sum(yt.values[:, 0]) for (i, j) in yt_kv.items()}

    return X_likelihood, y_likelihood
```

In [10]:
```python
X_l, y_l = compute_likelihood(X_features=X, y_target=y)
```

In [11]:
```python
X_l
```

Out[11]:
```
{'outlook': {'Sunny': {'Yes': 0.2222222222222222, 'No': 0.6},
  'Rain': {'Yes': 0.3333333333333333, 'No': 0.4},
  'Overcast': {'Yes': 0.4444444444444444, 'No': 0.0}},
 'temp': {'Mild': {'Yes': 0.4444444444444444, 'No': 0.4},
  'Hot': {'Yes': 0.2222222222222222, 'No': 0.4},
  'Cool': {'Yes': 0.3333333333333333, 'No': 0.2}},
 'humidity': {'High': {'Yes': 0.3333333333333333, 'No': 0.8},
  'Normal': {'Yes': 0.6666666666666666, 'No': 0.2}},
 'wind': {'Weak': {'Yes': 0.6666666666666666, 'No': 0.4},
  'Strong': {'Yes': 0.3333333333333333, 'No': 0.6}}}
```

In [12]:
```python
y_l
```

Out[12]:
```
{'Yes': 0.6428571428571429, 'No': 0.35714285714285715}
```

Prediction

In [13]:
```python
# case - 1
p1 = [['Sunny', 'Cool', 'High', 'Strong']]

# case - 2
p2 = [['Sunny', 'Mild', 'Normal', 'Weak'],
      ['Sunny', 'Mild', 'High', 'Weak'],
      ['Rain', 'Cool', 'Normal', 'Strong']]

print('p1', len(p1))
print('p2', len(p2))
```

```
p1 1
p2 3
```

In [14]:
```python
def predictor(X_new, X_l, y_l):
    cols = list(X_l.keys())
    col_val = {i : j for (i, j) in zip(cols, X_new)}

    lprobs = {}
    for l, v in y_l.items():
        cate_v = [X_l[cn][cl][l] for (cn, cl) in col_val.items()]
        lprobs[l] = round((np.prod(cate_v) * v), 4)

    prob_ks = list(lprobs.keys())
    prob_vs = list(lprobs.values())

    return prob_ks[np.argmax(prob_vs)]

def predict(X_new, X_l, y_l):
    if (len(X_new) == 1):
        return predictor(X_new=X_new[0], X_l=X_l, y_l=y_l)
    preds = [predictor(X_new=i, X_l=X_l, y_l=y_l) for i in X_new]
    return preds
```

In [15]:
```python
prediction = predict(X_new=p1, X_l=X_l, y_l=y_l)
print(prediction)
```

```
No
```

In [16]:
```python
prediction = predict(X_new=p2, X_l=X_l, y_l=y_l)
print(prediction)
```

```
['Yes', 'No', 'Yes']
```