The Condorcet Jury Theorem

The Condorcet Jury Theorem is a political science and probability theory concept named after the French mathematician and philosopher Marquis de Condorcet. The theorem explores the relationship between the accuracy of individual decision-makers (voters or jurors) and the accuracy of the collective decision made by the group.

The theorem assumes that:

- Each individual in the group has a probability p of making the correct decision (independently of the others), where $0.5 < p < 1$. In other words, each individual is more likely than not to make the correct decision.
- The individuals make their decisions independently of each other.

Under these conditions, the Condorcet Jury Theorem states that:

As the group size (number of individuals) increases, the probability that the group makes the correct decision approaches 1 (certainty). If each individual's probability of making the correct decision is above 0.5, then adding more individuals to the group will improve the group's decision-making accuracy. Conversely, if each individual's probability of making the correct decision is below 0.5, then adding more individuals to the group will reduce the group's decision-making accuracy. In simpler terms, the theorem suggests that, under the specified conditions, the "wisdom of the crowd" prevails. A larger group of individuals, each with a better-than-random chance of making the correct decision, will collectively make better decisions than any single individual or a smaller group. However, if the individual's probability of making the correct decision is below 0.5, the group's collective decision-making would worsen with additional members.

The Condorcet Jury Theorem has applications in various fields, such as political science, economics, and decision-making theory. It provides a basis for understanding how collective decision-making can lead to more accurate outcomes, given that certain conditions are met.

In [151]:
```
# Review of for loops
print("Count to 10")
for i in range(10):
    print(i)

print("Iterate over values in a list")
for item in [5, 'hello', 3.13, True, [1,2,3]]:
    print(item)
```

```
Count to 10
0
1
2
3
4
5
6
7
8
9
Iterate over values in a list
5
hello
3.13
True
[1, 2, 3]
```

In [150]:
```
# Review of list comprehension

[x**2 for x in range(10)]

[len(word) for word in "to be or not to be that is the question".
```

Out[150]:
```
[2, 2, 2, 3, 2, 2, 4, 2, 3, 8]
```

In [114]:
```
# Review of random number generation
import random as rnd

rnd.seed(3.14159)
[rnd.randrange(1,5) for _ in range(5)]
[rnd.choice(['a', 'b', 'c']) for _ in range(5)]
[rnd.uniform(.2, .3) for _ in range(5)]
[rnd.gauss(0, 1) for _ in range(5)]
```

Out[114]:
```
[-0.6239279552210337,
 0.255883175412692,
 0.19679498371365348,
 0.012348240098453199,
 0.301546769068397]
```

In [124]:
```
# Review of counter library
dna = "agctagctagcatgcatcgatagtcgatctattctaatatattcggcggcgattatcg
counts = Counter(dna)
print(cnt)

print("Most common: ", counts.most_common(1)[0][0])
```

```
Counter({'t': 18, 'a': 16, 'g': 15, 'c': 13})
Most common:  t
```

In [135]:
```
# Create 11 "jurors" with verdict accuracy in range (0.5, x)
n = 11
max_accuracy = 0.667
jury = [rnd.uniform(.5, max_accuracy) for _ in range(n)]
jury
```

Out[135]:
```
[0.5386514461914312,
 0.6490018875549747,
 0.6509485369268001,
 0.6264403149307383,
 0.6006604526422817,
 0.6456462101085015,
 0.5836102701615514,
 0.5656482996136852,
 0.6288250691054713,
 0.5792069752540076,
 0.5114937016621027]
```

In [136]:
```
# Create a series of outcomes (0=innocent or 1=guilty) to be deci
n_outcomes = 100000
outcomes = [rnd.choice([0,1]) for _ in range(n_outcomes)]
```

In [137]:
```
# for each outcome, and each juror makes a verdict decision
# In this model, we don't require unanimity. Instead, the jurors
# The majority vote is the decision of the jury.
verdicts = []
for out in outcomes:
    verdict = []
    for juror in jury:
        if rnd.random() < juror:
            verdict.append(out)
        else:
            verdict.append(1-out)
    verdicts.append(Counter(verdict).most_common(1)[0][0])
```

In [138]:
```
# Compute the jury verdict accuracy for each

correct = 0
for i in range(len(outcomes)):
    if outcomes[i] == verdicts[i]:
        correct += 1
print(correct / len(outcomes))

```

```
0.7494
```

In [ ]:
```

```