**Minor in AI**

17 Feb 2025

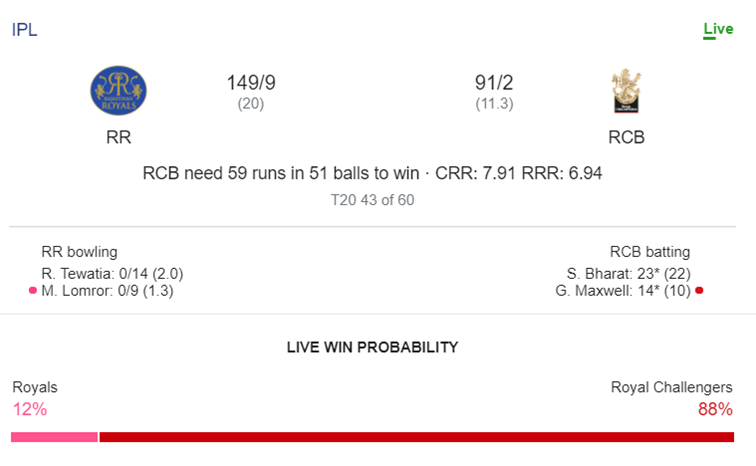
Class Notes

**Topics covered**

Define key probability concepts like sample space, events, and outcomes.  
Explain basic probability rules, including addition and multiplication rules.  
Differentiate between independent and dependent events.

**Activity 01:**

How do you think Google calculates live probability? What are the parameters used? Can the probability be 100%?



**Activity 02:**

Write a program to read the data from 'data.txt' from the mounted drive and count occurrences of each character. Also count the probability of each. Sort and present the results from highest to lowest.

(You can use any file that has text data in it. How can this data be used in compression? Refer to class recorded video)

**Activity 03:**

Prompt 01:

There is a bag with two hats in it - red and blue. You randomly select one hat. If the selected hat is red, you add another red hat back to the bag. If the selected hat is blue, then you add another blue hat to the bag. Repeat this process 50 times. For each iteration print: the bag status, the hat selected and probability that red or blue hat will be selected.

Prompt 02:

There is a bag with two hats in it - red and blue. You randomly select one hat. If the selected hat is red, you add another blue hat back to the bag. If the selected hat is blue, then you add another red hat to the bag. Repeat this process 50 times. For each iteration print: the bag status, the hat selected and probability that red or blue hat will be selected.

In the first case, the values stabilize at 70-30% and in the second case at 50-50%. Why?

Analysis:

Prompt 01:

This is an emergent probability pattern due to preferential attachment. A well-known mathematical result shows that, on average, such reinforcement models tend to stabilize at a 70%-30% split when run for several iteration.

Reasons:

1. Early Bias Effect: If red is picked more often in the first few iterations, it keeps getting reinforced, leading to more red selections.
2. Self-Reinforcing Feedback: The probability keeps increasing for the more frequent color, but it does not completely dominate.
3. Empirical Observations: Similar models in probability theory show stabilization at roughly 70-30 when two categories compete in this way.

Prompt 02:

This process creates equilibrium, where both colours are added in a balanced manner. Unlike the first case (which reinforced the selection and led to a bias), this version keeps pushing the system back toward equal proportions, leading to a stable 50%-50% distribution.

Observations:

* Unlike the reinforcement model in your previous question (which led to 70%-30%), this new rule ensures that the counts of red and blue remain nearly equal over time.
* Every time you pick a red, you add a blue, balancing the ratio.
* Every time you pick a blue, you add a red, preventing any colour from dominating.
* Since each step counteracts the previous selection, the system naturally maintains an equal distribution in the long run.

Based on class discussion refer to:

* Polya process
* Colonel Blotto
* Prisoner’s Dilemma

Dependent and Independent Events:

Refer to video for deck card explanation.

import random

# Independent Events: Drawing Cards WITH Replacement

def draw\_with\_replacement():

    deck = ["Red", "Red", "Black", "Black"]  # 2 Red, 2 Black

    first\_draw = random.choice(deck)  # Draw first card

    second\_draw = random.choice(deck)  # Draw second card (same probabilities)

    # Probabilities remain unchanged

    P\_red = deck.count("Red") / len(deck)  # 2/4 = 0.5

    P\_black = deck.count("Black") / len(deck)  # 2/4 = 0.5

    print("Independent Events: Drawing With Replacement")

    print(f"First draw: {first\_draw}")

    print(f"Second draw: {second\_draw}")

    print(f"P(Red) = {P\_red:.2f}, P(Black) = {P\_black:.2f}")

    print("Probability remains unchanged because the card is put back.\n")

# Dependent Events: Drawing Cards WITHOUT Replacement

def draw\_without\_replacement():

    deck = ["Red", "Red", "Black", "Black"]  # 2 Red, 2 Black

    first\_draw = random.choice(deck)  # Draw first card

    deck.remove(first\_draw)  # Remove it (no replacement)

    # Updated probabilities after first draw

    P\_red\_first = 2/4  # 2 red cards in 4

    P\_black\_first = 2/4  # 2 black cards in 4

    P\_red\_second = deck.count("Red") / len(deck)  # Updated probability

    P\_black\_second = deck.count("Black") / len(deck)  # Updated probability

    second\_draw = random.choice(deck)  # Draw second card

    print("Dependent Events: Drawing Without Replacement")

    print(f"First draw: {first\_draw}")

    print(f"Second draw: {second\_draw}")

    print(f"P(Red first) = {P\_red\_first:.2f}, P(Black first) = {P\_black\_first:.2f}")

    print(f"P(Red second | First draw) = {P\_red\_second:.2f}, P(Black second | First draw) = {P\_black\_second:.2f}")

    print("Probability changes because the first card is not put back.\n")

# Run the functions

draw\_with\_replacement()

draw\_without\_replacement()

Key probability concepts:

1. **Sample Space (S)**: The set of all possible outcomes of a random experiment.
   * Example: If you roll a six-sided die, the sample space is
   * S={1,2,3,4,5,6}
2. **Outcome**: A single possible result of an experiment.
   * Example: Rolling a die and getting a 4 is one outcome.
3. **Event (E)**: A subset of the sample space, representing a collection of outcomes that satisfy a certain condition.
   * Example: The event of rolling an even number is E={2,4,6}

Here are real-world examples of key probability concepts:

1. **Sample Space (S)**:
   * **Weather Forecasting**: The possible weather conditions for tomorrow:  
     S={sunny, rainy, cloudy, snowy}
2. **Outcome**:
   * **Online Order Delivery**: The actual time your package arrives (e.g., 3 PM) is an outcome in the sample space of possible delivery times.
3. **Event (E)**:
   * **Quality Control in Manufacturing**: Suppose a factory produces 100 items, and defective items are a concern. If S={1,2,...,100, the event E of picking a defective item might be E={5,12,27} (i.e., defective items identified).
4. **Probability in AI/ML**:
   * When an AI predicts whether an email is spam or not, the sample space is S={spam, not spam}and an outcome is whether a specific email is classified as spam. The probability of spam is calculated based on prior data.

### ****1. Probability of an Event****

For any event E in a sample space S: 0 ≤ P(E) ≤ 1

where:

* P(E)=0 means the event is impossible.
* P(E)=1 means the event is certain.

### ****2. Addition Rule (for Union of Events)****

#### ****a) Mutually Exclusive Events****

If two events A and B cannot occur together, A∩B=∅ then: P(A∪B)=P(A)+P(B)

#### ****b) Non-Mutually Exclusive Events****

If A and B can occur together: P(A∪B)=P(A)+P(B)−P(A∩B)

### ****3. Multiplication Rule (for Intersection of Events)****

#### ****a) Independent Events****

If two events A and B do not affect each other: P(A∩B)=P(A)×P(B)

#### ****b) Dependent Events****

If A affects B, then: P(A∩B)=P(A)×P(B∣A)

where P(B∣A) is the conditional probability of B given A

**Examples:**

**Independent Events**

1. **Flipping Two Coins**
   * Event A: First coin lands on heads.
   * Event B: Second coin lands on heads.
   * These are independent because the first coin’s result does not affect the second coin’s result.
2. **Rolling Two Dice**
   * Event A: Rolling a 4 on the first die.
   * Event B: Rolling a 6 on the second die.
   * The outcome of the first die does not change the probabilities for the second die.

**Dependent Events**

1. **Drawing Cards Without Replacement**
   * Event A: Drawing a red card from a deck.
   * Event B: Drawing a second red card (without replacing the first).
   * The probability of B changes based on whether A happened because the total number of red cards decreases.
2. **Selecting Students for a Team**
   * Event A: Selecting a girl for a debate team.
   * Event B: Selecting another girl for the same team (without replacement).
   * If the first selection was a girl, the probability of the second selection being a girl changes.

Exercise:

You roll a 6-sided die twice.

* What is the probability that the sum of the two rolls is greater than 8?
* Write a program to simulate rolling two dice 10,000 times and estimate this probability.

To solve:

* The sample space consists of all (die1, die2) pairs, where each die can roll 1 to 6.
* Count how many times die1+die2>8
* Divide by the total number of simulations to get an estimated probability.

Program to Demonstrate the Addition rule:

# Addition Rule

import random

# Simulated students with their preferences

students = [

    {"name": "Amit", "likes\_math": True, "likes\_science": False},

    {"name": "Priya", "likes\_math": True, "likes\_science": True},

    {"name": "Raj", "likes\_math": False, "likes\_science": True},

    {"name": "Sara", "likes\_math": True, "likes\_science": False},

    {"name": "Vikram", "likes\_math": False, "likes\_science": True},

    {"name": "Neha", "likes\_math": True, "likes\_science": True},

    {"name": "Rahul", "likes\_math": False, "likes\_science": False},

    {"name": "Anita", "likes\_math": False, "likes\_science": True},

]

# Count occurrences

total\_students = len(students)

likes\_math = sum(1 for s in students if s["likes\_math"])

likes\_science = sum(1 for s in students if s["likes\_science"])

likes\_both = sum(1 for s in students if s["likes\_math"] and s["likes\_science"])

# Calculate probabilities

P\_math = likes\_math / total\_students

P\_science = likes\_science / total\_students

P\_both = likes\_both / total\_students

P\_either = P\_math + P\_science - P\_both  # Applying the Addition Rule

# Print results

print("Addition Rule in Probability Demonstration")

print("Total students:", total\_students)

print("P(Math)", P\_math)

print("P(Science)", P\_science)

print("P(Both Math & Science)", P\_both)

print("P(Either Math or Science) = P(Math) + P(Science) - P(Both)", P\_either)

Refer to collab notes and vidoe for more programs and analysis.