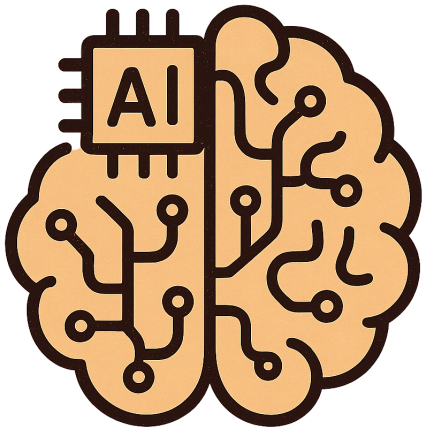
How Probability & Statistics Help AI to Learn



Course Name:

IE102 – Probability & Statistics

Group Members

Prashant Gautam, 24B4526

Harshit Singh Bhandari, 24B4506

Vaibhav Negi, 24B4503

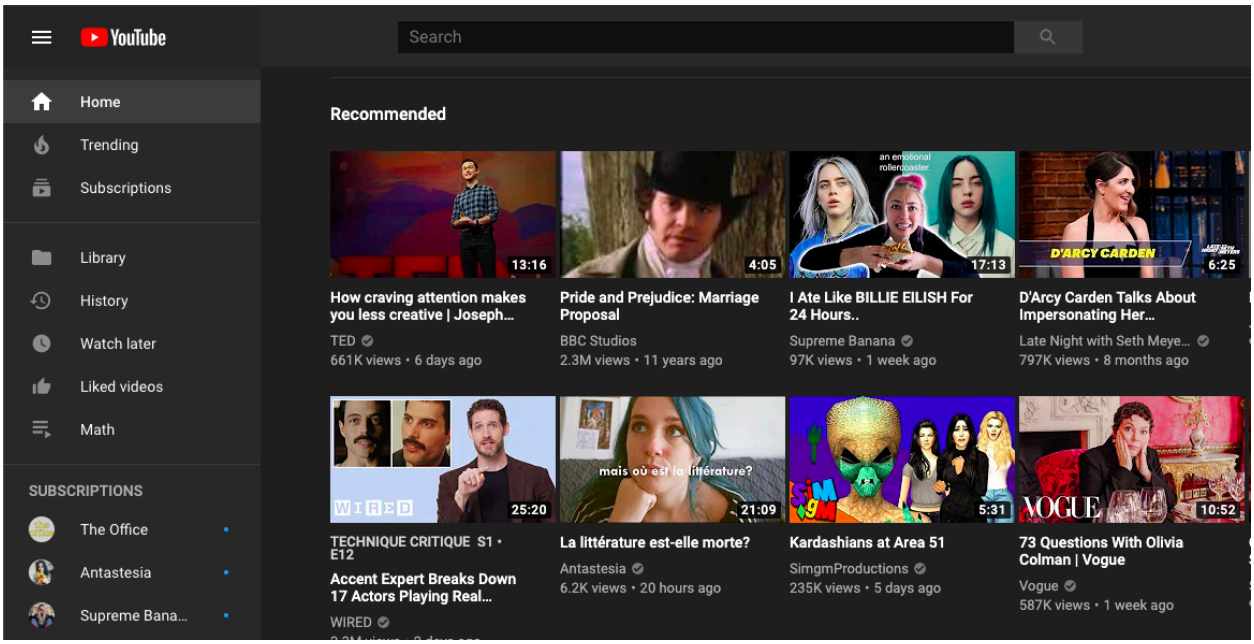
Anant, 24B4530

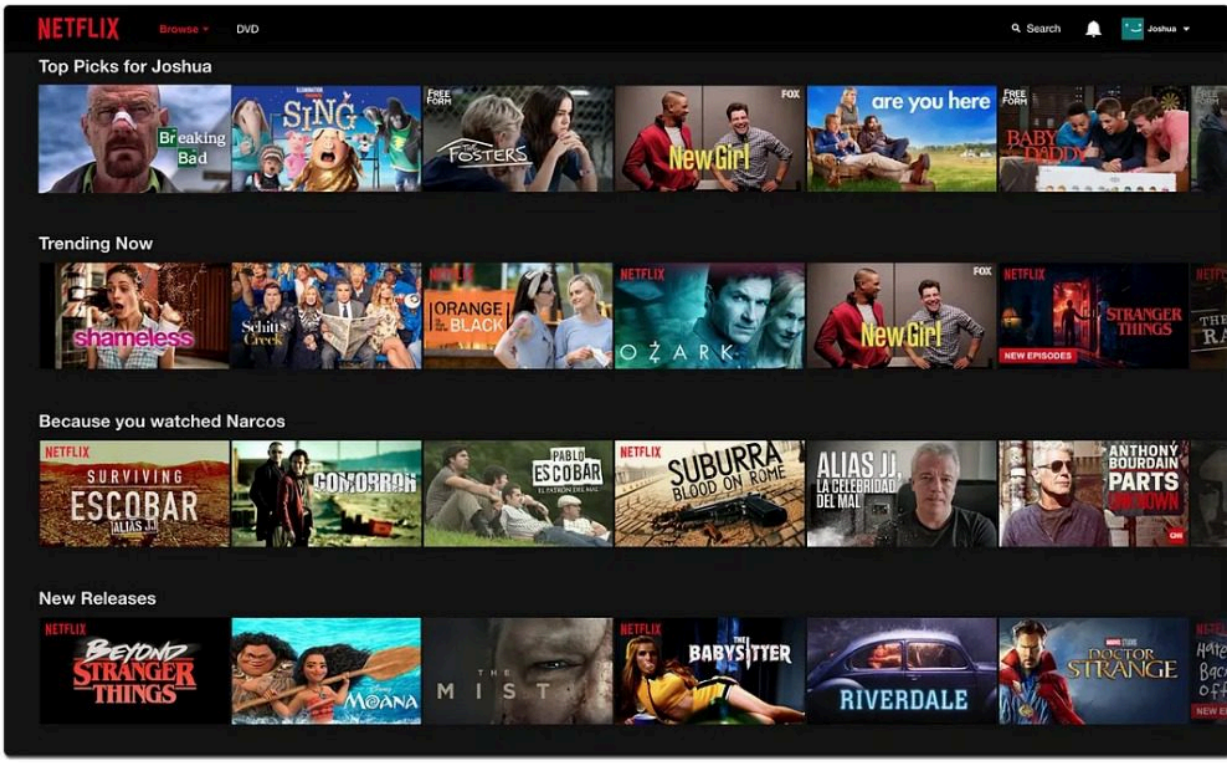
1.Introduction

In this project, we explored how mathematics fundamental tools “**Probability & Statistics**” (taught in IE102), help **artificial intelligence (AI) systems to learn from experience** (by Reward). We used a Q-learning based agent to develop a personalized news recommendation system that learns user preferences through simulated feedback.

Modern AI systems often need to make decisions in uncertain environments. Whether it's suggesting a movie, article, or product, these systems rely on **probabilistic feedback and statistical learning** to adapt over time. Our project aims to demonstrate these concepts using a simple but powerful model: a news recommender system based on **reinforcement learning.**

Two Examples of YouTube & Netflix Recommendation





Real-world examples of AI-based recommendations.

2. Objective:

⊳ **Build a news recommendation system**

⊳ **Simulate user clicks using probability**

⊳ **Use Q-learning to improve performance over time**

⊳ **Understand how statistics and feedback guide learning**

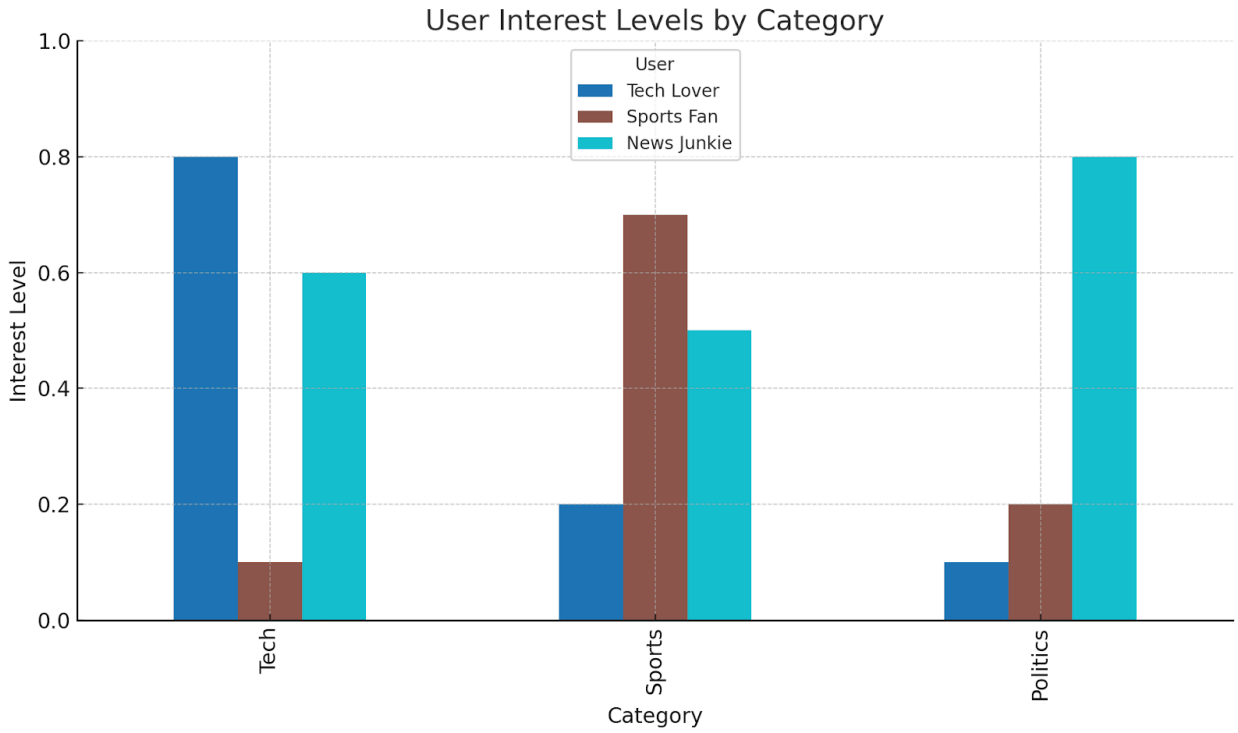
3. The Setup

We designed three types of users with different preferences:

- **Tech Lover**: High chance of clicking tech articles
- **Sports Fan**: Prefers sports content
- **News Junkie**: Has moderate interest in all categories, especially politics

Each user has a probability distribution over three categories: Tech, Sports, and Politics. Five articles from these categories were included in the simulation. The user clicks were generated using **Bernoulli trials based on these probabilities.**

User	Tech	Sports	Politics
Tech Lover	0.8	0.2	0.1
Sports Fan	0.1	0.7	0.2
News Junkie	0.6	0.5	0.8



4. What is Q-Learning?

Q-learning is a reinforcement learning technique where an agent learns optimal actions by receiving rewards and updating estimates. In our setup:

- **The state is defined by the user profile**
- **The action is the article chosen**
- **The reward is binary (1 for click, 0 for no click)**

The Q-value for each user-article pair is updated using:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max Q(s',a') - Q(s,a))$$

- α is the learning rate
- γ is the discount factor
- r is the reward received

We also implemented an epsilon-greedy strategy to balance exploration and exploitation during learning.

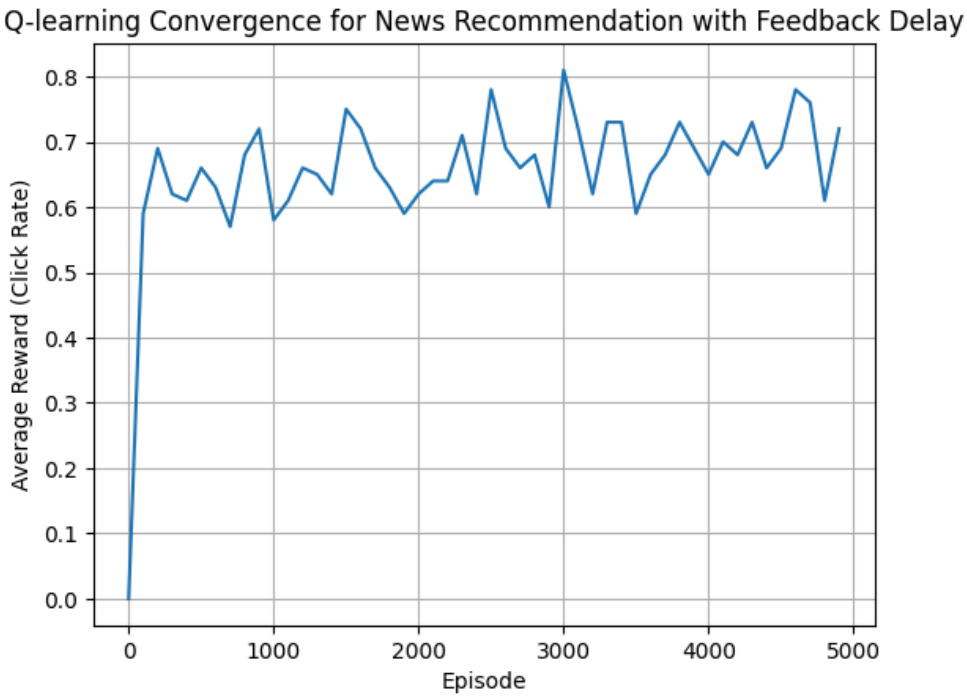
5. Where Statistics Comes In?

- Every trial is like a **Bernoulli trial**
- AI uses **average rewards** (statistics) to update choices
- Bellman equation + Expected values

6. Simulation Setup

- 5000 episodes
- α = 0.1, γ = 0.9, ε = 0.2
- Feedback delay between 0–2ms
- Use of NumPy, pandas, matplotlib, time module

7. Learning Graph



Agent's learning curve: average reward vs. training episodes

Final Q-table & Recommendations

Final recommendations show that the agent learned to recommend articles matching user interests:

- Tech Lover → AI Breakthrough
- Sports Fan → Football Highlights
- News Junkie → Election Update

8. Observations

- The average reward increased over time, confirming that the agent was learning.
- The final Q-table reflected the true preferences of users, showing convergence.
- Simple statistical feedback (click or no click) was enough for the agent to improve.

```
Final Q-table (Estimated Click Probabilities):
                    tech_user  sports_user  news_junkie
AI Breakthrough      8.034986     6.324924     7.505302
Football Highlights  7.566917     7.062802     7.354018
Election Update      7.394386     6.223531     8.067346
New Smartphone       8.263430     6.352018     7.329654
Olympics Recap       7.627095     6.841167     7.775679

Sample Recommendations:
Recommended article for tech_user: 'New Smartphone' [Tech]
Recommended article for sports_user: 'Football Highlights' [Sports]
Recommended article for news_junkie: 'Election Update' [Politics]
```

Real-World Applications This approach mirrors real-world recommendation systems used in:

- News platforms (Google News)
- Streaming services (Netflix, YouTube)
- E-commerce (Amazon)
- Online advertising (personalized ad targeting)

All these systems rely on probability and statistics to estimate preferences and make decisions.

Code Appendix: Q-Learning Implementation

9. Conclusion

This project highlights how core mathematical tools such as **Bernoulli trials, expected values, and statistical averages** enable AI agents to learn effectively. By simulating **click behavior and reward-based learning**, we built a simplified model of real-world AI recommendation systems. This demonstrates the **essential role that probability and statistics play in enabling AI to learn from data**.

9. Code Appendix: Q-Learning Implementation

Below are selected code snippets from our Q-learning-based news recommendation system. These illustrate how probability and statistics were used to simulate user feedback and how the agent learned from it.

User Profiles and Click Probabilities

```
user_profiles = {
    "tech_user":    {"Tech": 0.8, "Sports": 0.2, "Politics": 0.1},
    "sports_user":  {"Tech": 0.1, "Sports": 0.7, "Politics": 0.2},
    "news_junkie":  {"Tech": 0.6, "Sports": 0.5, "Politics": 0.8}
}
```

Each user has probabilistic preferences over three article categories, simulating real-world click behavior using Bernoulli trials.

Article Catalog

```
article_catalog = [
    {"title": "AI Breakthrough", "category": "Tech"},
    {"title": "Football Highlights", "category": "Sports"},
    {"title": "Election Update", "category": "Politics"},
    {"title": "New Smartphone", "category": "Tech"},
    {"title": "Olympics Recap", "category": "Sports"}
]
```

A simple set of five articles was created, each tagged with a category.

Q-Learning Algorithm

```
Q = {user: np.zeros(len(article_catalog)) for user in user_profiles}
epsilon, alpha, gamma = 0.2, 0.1, 0.9  # Parameters

for episode in range(5000):
    user = random.choice(list(user_profiles.keys()))

    # ε-greedy strategy
    if random.random() < epsilon:
        action = random.randint(0, len(article_catalog) - 1)
    else:
        action = np.argmax(Q[user])

    category = article_catalog[action]["category"]
    click_prob = user_profiles[user].get(category, 0.1)

    reward = 1 if random.random() < click_prob else 0

    # Q-value update
    Q[user][action] += alpha * (reward + gamma * np.max(Q[user]) - Q[user][action])
```

This is the core learning loop, where the agent uses past reward data and exploration to refine its recommendations.

Recommendation Function

```python
def recommend(user):
    best_article_idx = np.argmax(Q[user])
    article = article_catalog[best_article_idx]
    print(f"Recommended for {user}: {article['title']} [{article['category']}]")
```

After training, the agent uses the Q-table to recommend the most suitable article to each user.

Note:

All probabilistic interactions (clicks) are modeled using **Bernoulli trials**, and the learning is governed by **statistical averages and expected rewards**, which show how core concepts from **Probability & Statistics** power AI learning.

10. Conclusion

This project highlights how core mathematical tools such as **Bernoulli trials, expected values, and statistical averages** enable AI agents to learn effectively. By simulating **click behavior and reward-based learning**, we built a simplified model of real-world AI recommendation systems. This demonstrates the **essential role that probability and statistics play in enabling AI to learn from data**.

References:

- Probability & Statistics course material (IE102)
- OpenAI & Scikit-learn documentation