MATHEMATICS FOR INTELLIGENCE SYSTEMS – 5 (21MAT301)

PROJECT REPORT (PHASE 2)

Employee Retention Analytics using Hidden Markov Models (HMMs).

Group – 13

Sai Varun Avadhuta	AM.EN.U4AIE21120
Sai Nikhil Guptha Mammula	AM.EN.U4AIE21142

Introduction:

Employee retention analytics with Hidden Markov Models (HMMs) is like having a crystal ball to understand why employees might leave a job. Picture it as a smart detective analyzing information about how employees perform, how long they've been part of the company, their feedback, and other important details. HMMs help by breaking down employees' situations into different "states," like content or thinking of moving on. This method allows companies to see how likely it is for someone to switch from one state to another. Using this insight, organizations can step in early, offering extra support or tailored solutions to keep their team happier and motivated.

Ultimately, this approach helps companies make smarter decisions to retain valuable employees for the long haul. Hidden Markov Models (HMMs) are a statistical tool used to understand sequences of observations where an underlying process is not directly observable. In the context of employee retention, HMMs help by modelling different "hidden states" that employees might be in, such as satisfied, dissatisfied, or considering leaving. These states are not directly seen but are inferred from observable features like performance ratings, tenure, absenteeism, or engagement levels.

Problem Formulation:

1. Hidden States (S):

Set of hidden states representing employee intentions:

S = {S1 = Satisfied, S2 = Disengaged, S3 = Planning to Leave}

Observable Actions (O): Set of observable employee actions:

O = {O1 = High Performance, O2 = Low Attendance, O3 = Negative Feedback}

2. Model Parameters:

N: *Number of hidden states* (N = 3 in this scenario)

M: *Number of observable actions* (M = 3 in this scenario)

3. Transition Matrix A:

The Transition Matrix A represents the probabilities of transitioning from one hidden state to another in consecutive time steps. Each element aij of the matrix represents the probability of transitioning from hidden state Si at time t to hidden state Sj at time t+1.

$$aij = P(St + 1 = Sj \mid St = Si)$$

4 . The Emission Matrix B

It represents the probabilities of emitting observable actions given the hidden states. Each element bj(k) of the matrix indicates the probability of observing action Ok at time t given that the system is in hidden state Sj at the same time.

$$bj(k) = P(Ot = Ok \mid St = Sj)$$

$bj(k)$: *Probability of observing action Ok at time t given that the system is in hidden state Sj at the same time.*

$P(Ot = Ok \mid St = Sj)$: *Conditional probability of observing action Ok at time t given the system is in state Sj at the same time.*

5. Initial State Distribution Vector (π):

πi *: Represents the initial probability of being in each hidden state*

$P(S1 = Si)$: *Probability of starting in hidden state Si at the initial time step.*

Interpretation: The vector π=[π 1 ,π 2 ,π 3 ,…] indicates the likelihood or chance of the system's initial state being in each specific hidden state Si at the beginning of the sequence.

6. Forward Algorithm

$$\alpha 1\ (i) = \pi i\ \times bi\ (X1\ )$$

Initializes the forward probabilities for each hidden state at time t=1.

$$\alpha t\ (j) = \left[\sum_{i\,=\,1}^{N} \alpha t - 1\ (i) \times aij\ \right] \times bj\ (Xt\ )$$

Iteratively calculates the forward probabilities for each hidden state at time t>1, considering previous probabilities, state transitions, and emissions.

7. Termination

$$P(\ X\ \mid\ \pi, A, B\ ) = \sum_{i=1}^{N} \alpha T\ \ (i)\text{:}$$

Sums up the forward probabilities (α) of all hidden states at the final time step T to obtain the likelihood of the entire observed sequence X.

8. Backward Algorithm

$$\beta T\ (i) = 1$$

Initializes the backward probabilities for each hidden state at the final time step T.

$$\beta t\ (i) = \sum_{j=1}^{N} aij\ \ \times bj\ (Xt + 1\ ) \times \beta t + 1\ (j)$$

9. Viterbi algorithm

$$v1\ (i) = \pi i\ \times bi\ (X1\ )$$

Initializes the probability of each hidden state Si at the initial time step by considering the initial state distribution (π) and the likelihood of observing the first symbol (X1).

$$vt\ (j) = \max i\ \ [vt - 1\ (i) \times\ aij\ ] \times bj\ (Xt\ )$$

$$\psi t\,(j) = \arg\max i\ [vt - 1\,(i) \times aij\,]$$

Weakness of Existing Methods:

Relying on Simplified Patterns:

HMMs focus on short-term connections between employee behaviour's. This might make them miss important long-term influences or big-picture factors affecting retention, like personal growth or company culture.

Struggling with Changes in Retention Trends:

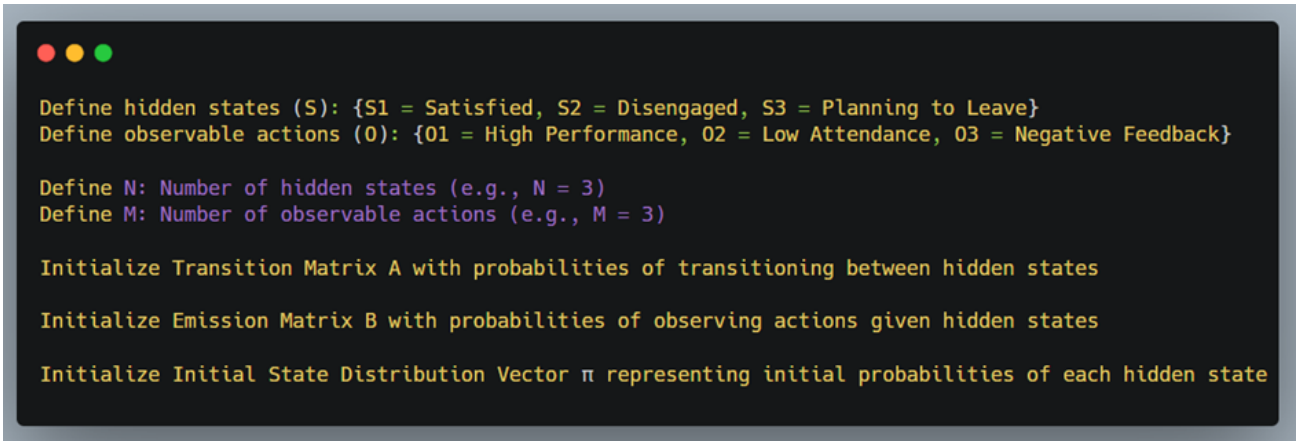
HMMs use fixed rules to understand how employees stay or leave. When things change, like new company policies or market shifts, HMMs find it tricky to adapt quickly and might not predict future trends accurately.

Needing Precise Data for Accurate Predictions:

HMMs need lots of detailed and accurate data to make good predictions about employee retention. If the data isn't complete or doesn't show the whole picture, HMMs might make mistakes in their predictions.

Pseudocode:

- ☐ Initialization



```
Define hidden states (S): {S1 = Satisfied, S2 = Disengaged, S3 = Planning to Leave}
Define observable actions (O): {O1 = High Performance, O2 = Low Attendance, O3 = Negative Feedback}

Define N: Number of hidden states (e.g., N = 3)
Define M: Number of observable actions (e.g., M = 3)

Initialize Transition Matrix A with probabilities of transitioning between hidden states

Initialize Emission Matrix B with probabilities of observing actions given hidden states

Initialize Initial State Distribution Vector π representing initial probabilities of each hidden state
```

- ☐ Forward Algorithm

```
Procedure ForwardAlgorithm(Observations):
    Initialize matrix α with dimensions T x N (T = length of observation sequence)
    // Calculate α1(i)
    for i = 1 to N:
        α[1][i] = π[i] * B[i][Observations[1]]

    // Calculate αt(j) for t > 1
    for t = 2 to T:
        for j = 1 to N:
            α[t][j] = 0
            for i = 1 to N:
                α[t][j] += α[t-1][i] * A[i][j]
            α[t][j] *= B[j][Observations[t]]

    // Termination - Compute likelihood of the entire sequence X
    likelihood = 0
    for i = 1 to N:
        likelihood += α[T][i]
    return likelihood
```

- Backward Algorithm

```
Procedure BackwardAlgorithm(Observations):
    Initialize matrix β with dimensions T x N (T = length of observation sequence)
    // Initialization - Set values for the last row
    for i = 1 to N:
        β[T][i] = 1

    // Calculate βt(i) for t < T
    for t = T-1 to 1:
        for i = 1 to N:
            β[t][i] = 0
            for j = 1 to N:
                β[t][i] += A[i][j] * B[j][Observations[t+1]] * β[t+1][j]

    return β
```

- Viterbi Algorithm

```
Procedure ViterbiAlgorithm(Observations):
    Initialize matrix v with dimensions T x N (T = length of observation sequence)
    Initialize matrix backpointers with dimensions T x N

    // Initialization - Calculate v1(i) and backpointers
    for i = 1 to N:
        v[1][i] = π[i] * B[i][Observations[1]]
        backpointers[1][i] = 0 // Base case

    // Calculate vt(j) and backpointers for t > 1
    for t = 2 to T:
        for j = 1 to N:
            v[t][j] = 0
            max_val = 0
            for i = 1 to N:
                val = v[t-1][i] * A[i][j] * B[j][Observations[t]]
                if val > max_val:
                    max_val = val
                    v[t][j] = val
                    backpointers[t][j] = i // Store the most likely state

    // Backtracking to find the most likely sequence
    sequence = []
    max_prob = 0
    last_state = 0
    for i = 1 to N:
        if v[T][i] > max_prob:
            max_prob = v[T][i]
            last_state = i
    sequence.append(last_state)
    for t = T-1 to 1:
        last_state = backpointers[t+1][last_state]
        sequence.append(last_state)
    return sequence
```

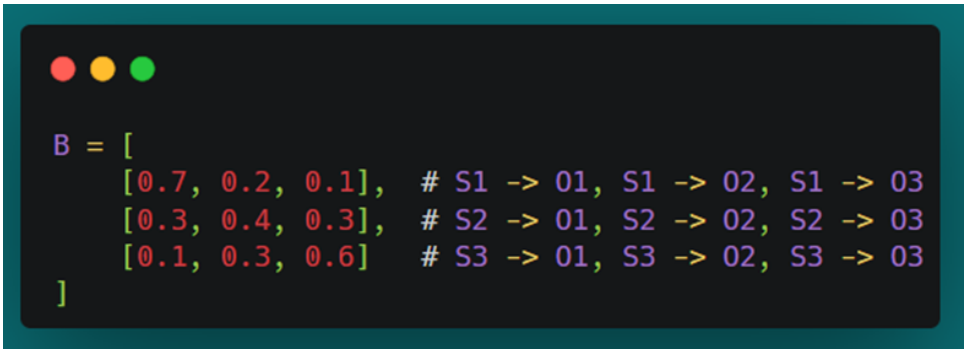
Sample Intermediate Results:

- **Transition Matrix (A):** Display the matrix showing probabilities of transitioning between hidden states. Explain how these probabilities indicate the likelihood of an employee moving from one state to another over consecutive time steps.

```
A = [
    [0.6, 0.2, 0.2],  # S1 -> S1, S1 -> S2, S1 -> S3
    [0.3, 0.5, 0.2],  # S2 -> S1, S2 -> S2, S2 -> S3
    [0.1, 0.3, 0.6]   # S3 -> S1, S3 -> S2, S3 -> S3
]
```

Explanation: The Transition Matrix A indicates that, for instance, there's a 60% chance of an employee in the S1 (Satisfied) state staying satisfied in the next time step, a 20% chance of moving to the S2 (Disengaged) state, and a 20% chance of moving to the S3 (Planning to Leave) state.

- **Emission Matrix (B):** Show the matrix presenting probabilities of observing actions given hidden states. Explain how these probabilities depict the likelihood of observing certain actions based on an employee's hidden state.



```
B = [
    [0.7, 0.2, 0.1],   # S1 -> O1, S1 -> O2, S1 -> O3
    [0.3, 0.4, 0.3],   # S2 -> O1, S2 -> O2, S2 -> O3
    [0.1, 0.3, 0.6]    # S3 -> O1, S3 -> O2, S3 -> O3
]
```

Explanation: The Emission Matrix B illustrates that, for example, an employee in the S1 (Satisfied) state has a 70% chance of exhibiting High Performance (O1), a 20% chance of Low Attendance (O2), and a 10% chance of Negative Feedback (O3)

- **Forward Algorithm Likelihood:**

```
❯ /opt/homebrew/bin/python3.12 /Users/saivarunavadhuta/Desktop/project.py
Forward Algorithm Result:
[[0.21     0.12     0.03    ]
 [0.033    0.0444   0.0252  ]
 [0.003564 0.010908 0.01836 ]]
Likelihood of the Sequence: 0.032832
```

Explanation:

The calculated likelihood of observing a specific sequence of actions given the model's parameters. Higher likelihood values imply the model's better fit to the observed sequence.

- **Backward Algorithm Probabilities:**

```
Backward Algorithm Result:
[[0.0804 0.102  0.1236]
 [0.24   0.3    0.46  ]
 [1.     1.     1.    ]]
```

Explanation:

The Backward Algorithm computes probabilities of future observations given a particular state at a specific time step. These probabilities contribute to understanding the likelihood of subsequent actions.