MATERATE

YOUR ONLY COMPETITION IS YOU

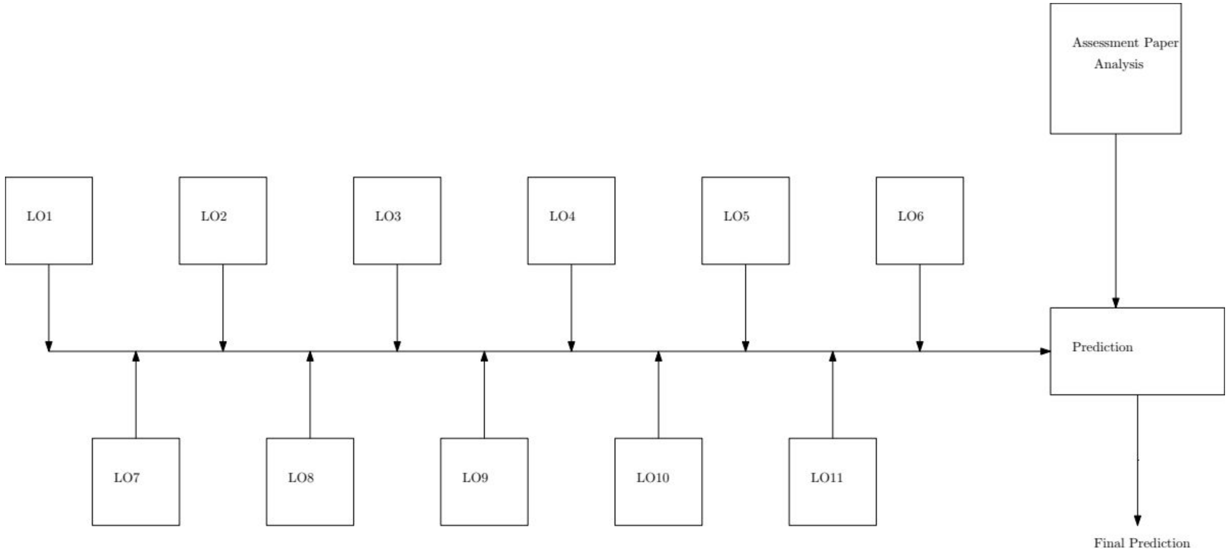
Inference from the dataset and sample model

- Task 1: We have to tell the model for the given dataset and also the list of things we can infer from the dataset.
- Inferences from the dataset
  - We can make a list of Learning Outcomes that is learned by most of the students compared to other which are learned by handful of students. That is it can gives us the idea of relative difficulty level of different Learning Outcomes.
  - We can divided the final score of a student on the basis of different Learn-ing Outcomes so as to get an idea which Learning outcomes is student well-versed.
  - By looking at specific group of student(say a class or a school) we can see which Learning Outcomes does a particular teacher/set of teachers was able to effectively communicate to the student. This can help us understand the teacher where he/she need to work.

Inference from the dataset and sample model

- Proposed Model
  - I propose different datasets to be formed using the given datasets. Each data point corresponds to the (student id, assessment no.). The features of the dataset will be Learning outcomes strength(number of question answered correctly in a LO/number of total questions in a LO) and the final score received by the student. My model will try to predict how well the student performs given a new assessment. We can have 11(number of Learning Outcomes), Linear regression model, to predict the Learning Outcomes strength. Now knowing the LO values for all the LO and knowing the breakup of the assessment, we can predict how the student will perform in upcoming assessments.

Inference from the dataset and sample model



Suggestion in explained structure

- Task 2: Propose another method for Learning Outcome Skill matrix.
  Why the method given in the slide provided to us will give error ?

- Sub Learning Outcomes have more correlation among themselves and with the parent LO compared to the correlation between two different LO.
- Skill level cannot be binary it has to be continuous because the learning curve of a student is a continuous process.
- Skill level should not be the sum of different LO and SLO of the given question.
- It should be the maximum of different LO and SLO of the given question. As we can see L1 has 8 subparts and we know that the correlation between these subparts will be higher as compared to the correlation between different LO. Hence, there is a large probability if a given question has L1S1 then it must have L1S3(say). If this is the case then question that has L1 as its components will seldom occupy the higher ranking than the question which doesn't have L1 as its component. To reduce this discrepancy we have to take 2 steps.

Suggestion in explained structure

- step 1: Take the max instead of the sum.
- step 2: Map the difficulty of a question as = alpha*Question-based difficulty(range [0,1];can be provided with the question or from the feedback of student who attempted the question) + (1-alpha)*LO/SLO specific difficulty(base value; calculated from the matrix; range [0,1]).

MATERATE
YOUR ONLY COMPETITION IS YOU

Implementation of Computer Aided Testing (Rasch Model)

Explaining the Algorithm

- Request next candidate. Set D=0, L=0, H=0, and R=0.
- Find next item near difficulty, D.
- Set D at the actual calibration of that item.
- Administer that item.
- Obtain a response.
- Score that response.
- Count the items taken: L = L + 1
- Add the difficulties used: H = H + D
- If response incorrect, update item difficulty: D = D - 2/L
- If response correct, update item difficulty: D = D + 2/L
- If response correct, count right answers: R = R + 1

Implementation of Computer Aided Testing (Rasch Model)

Explaining the Algorithm

- If not ready to decide to pass/fail, Go to step 2.
- If ready to decide pass/fail, calculate wrong answers: $W = L - R$
- Estimate measure: $B = H/L + \log(R/W)$
- Estimate standard error of the measure: $S = [L/(R*W)]$
- Compare B with pass/fail standard T.
- If $(T - S) < B < (T + S)$, go to step 2.
- If $(B - S) > T$, then pass.
- If $(B + S) < T$, then fail.

Implementation of Computer Aided Testing (Rasch Model)

- Disadvantage of the Rasch Model

- We are trying to fit the data to a model rather than finding any new insights from the data.
- Both the student and question difficulty level are measured on same scale. This leads to uni-skill based difficulty of the question.
- Due to uni-skill based difficulty it is not possible to find gaps in skills of the student.
- CATs are not applicable for all subjects and skills. Most CATs are based on an item-response theory model, yet item response theory is not applicable to all skills and item types.
- Hardware limitations may restrict the types of items that can be administered by computer. Items involving detailed artwork and graphs or extensive reading passages, for example, may be hard to present.
- Examinees can not change the answers once the answers of the question is submitted.

Implementation of Computer Aided Testing (Rasch Model)

- Explaining the code

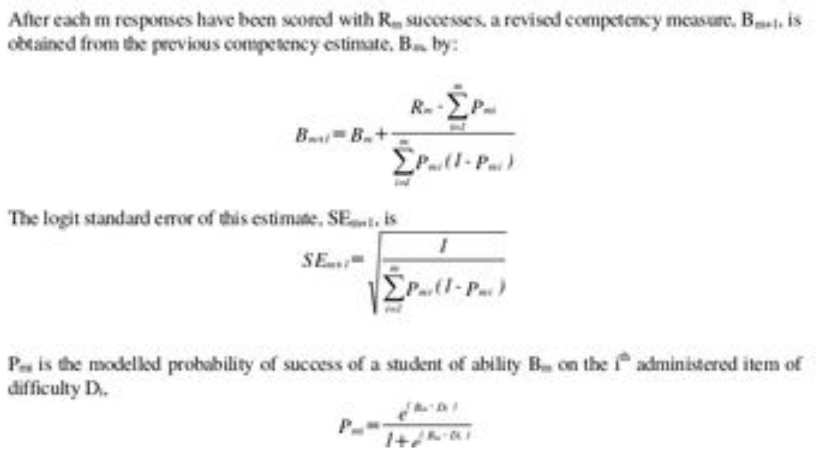
  All the codes written is well commented can be understood directly by reading the code. Here I will the different functions and their use work.

  All the symbols have the meaning as given in the algorithm.
- **Model_Prob(B, D):** Determining the model probability to solve a question based on the current estimate of the student ability
- **Question_Selection(D, L, A, R):** Determining the difficulty of the next question
- **Estimate_Measure1(H, L, R, W):** It measures the ability of the student based on the questions attempted so far. It returns the value of B(student's ability measure).
- **Estimate_Measure2(B, R, Plist):** It measures the ability of the student based on the questions attempted so far. It returns the value of B(student's ability measure).
- **Estimate_Error1(L, R, W):** It gives us the Estimated error of the student.
- **Estimate_Error2(Plist):** It gives us the Estimated error of the student.
- **Prob_ans(D, d):** This function emulates a test giver(child in our case) D = is the Ability of the child vs d = difficulty of the question.

MATERATE
YOUR ONLY COMPETITION IS YOU

Implementation of Computer Aided Testing (Rasch Model)

- I implemented 2 different functions for the student ability measurement first one is given in the algorithm and the other is given below.



After each m responses have been scored with R_m successes, a revised competency measure, B_{m+1}, is obtained from the previous competency estimate, B_m, by:

$$B_{m+1} = B_m + \frac{R_m - \sum_{i=1}^{m} P_{mi}}{\sum_{i=1}^{m} P_{mi}(1 - P_{mi})}$$

The logit standard error of this estimate, SE_{m+1}, is

$$SE_{m+1} = \sqrt{\frac{1}{\sum_{i=1}^{m} P_{mi}(1 - P_{mi})}}$$

P_{mi} is the modelled probability of success of a student of ability B_m on the i th administered item of difficulty D_i,

$$P_{mi} = \frac{e^{(B_m - D_i)}}{1 + e^{(B_m - D_i)}}$$

Code can be found on git repo Group1/Mayank/Rasch Model

MATERATE
YOUR ONLY COMPETITION IS YOU

Implementation of Computer Aided Testing (Rasch Model)

- In attempt to vectorize the Rasch Model so that we can include multiple skills in this model.

  I propose these changes in the model.
  - Instead of returning single value measure B we can return the S_skill(student skill value)

  One Obvious downfall of this number of question will increase many fold to almost (previous question)*(number of skill).
  This will result in long test which will be not effective in measuring the student's ability due to lack of interest.
  <u>**Vector_Estimate_Measure(Q_skill, S_skill, No_of_right_answers, Probability_Estimate_of_question):**</u> Q_skill is an boolean true/1 for skills that will be present in the question. We are going to update those Measure of the student only. All the rest of the parameters are self explanatory. This function returns the S_skill vector.
  Code can be found on git repo Group1/Mayank/Rasch Model/Part1.py

MATERATE
YOUR ONLY COMPETITION IS YOU

Implementing Knowledge Graph

- Requirement is to implement Knowledge Graph that takes input from a file and return us the Knowledge Graph.
- I used Networkx library for implementation of the graph.
- Networkx library has many predefined functions that can be implemented on the graph. Also, it can give us the visual representation of the graph.
- Instead of storing the full information related to concept I suggest it will be much more convenient if we store the node information in an seperate dictionary and form the graph using only concept name. It will be easy for updating the values inside the graph nodes. Internode and Intranode update will be independent from each other.

Implementing Knowledge Graph

- Format of the file which will be used for forming the graph is as follows:
  - Line starting from '#' will be considered as comment and will be ignored. Important to note that any '#' in between the line will not be considered as the comment
  - First n valid line(Lines that are not comment) will be information about nodes and will give information like name of the concept(This has to unique I am using this as the key for dictionary)
  - Rest of the valid lines will contain information about the edges in the graph.
  - using " " to separate the data in a single line. Cannot use simply space or comma because the concept name may contain space or comma or any other commonly used symbols.

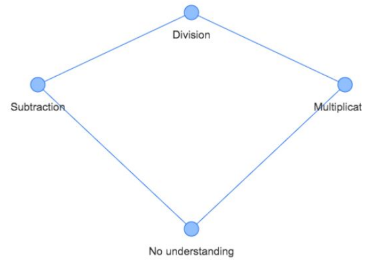
Code can be found in the git-repo Group1/Mayank/Knowledge Graph.

Understanding GNN and Deep-IRT model

- The GNN/Deep IRT model was implemented on given set of questions and it does not help us in predict the next question.
- Due to its inability to predict next question it is not useful for us.
- GNN/Deep IRT also needs extensive data/information about the student's answer on particular set of concept to predict the probability of answering next question. Due to this these model cannot be used for Adaptive testing but it can be used for deducing the concept mastery level. If the probability of answering question is above certain threshold say 0.9 then we can predict that the student has certain level of understanding of the concept.
- More detail about the GNN model can be found on Subhalingam and Vishal report on github and detail about the Deep IRT can be found on git repo in Group 1/Mayank/Deep IRT.

Implementing Jungroo Model

- What jungroo Model wish to achieve?
  - Predict the student’s knowledge state with minimal interactions with the student
  - Predict his/her shortest learning path to mastery.
- How does Jungroo model work?
  - Jungroo uses Markov Decision Process to predict the knowledge state of the student with minimal interactions. Explained in next slide.
  - Here is an sample Knowledge Graph



MATERATE
YOUR ONLY COMPETITION IS YOU

Implementing Jungroo Model

- Markov Decision Process:
  - State space S: {None, {Multiplication}, {Subtraction}, {Subtraction, Multiplication, Division}}
  - Action Space A:{QxS}Set of questions across all concepts.
  - Time step t — Sequence index of every interaction.
  - Reward function Ra(s, s’) is the expected immediate reward after transitioning from state s to s’.
  - Discount factor **γ** — It represents the difference in importance between future rewards and present rewards.
  - The goal is to choose a policy ***Π*** such that it maximizes the expected discounted sum over a time t.

Implementing Jungroo Model

- Transition function P — $P_a(s, s') = P(S_{t+1} = s' \mid S_t = s, \alpha, \beta)$

- Disadvantages of MDP:
  - Time consuming as the state space can grow exponential which in turn can make the calculation time to grow exponential.
  - How to assign values of $P_a(s, s')$?

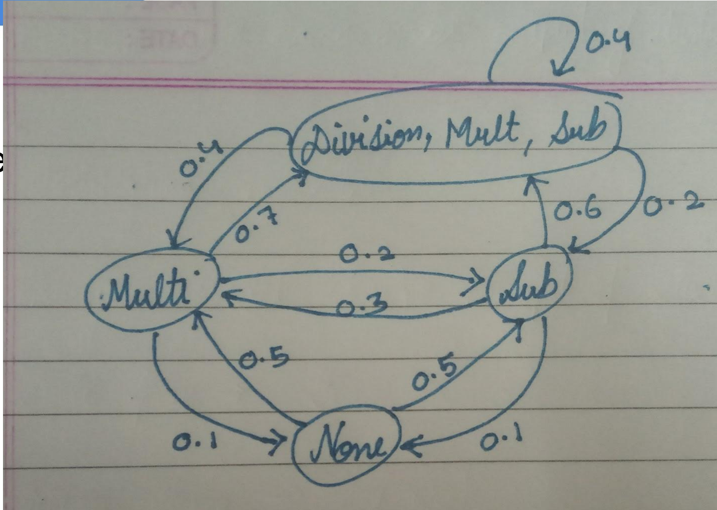
Implementing Jungroo Model

- How I implemented the Jungroo's Model?
  - Input : There are two files one for Rewards associated with each state called "rewards.txt" and the other is probability associated with each pair of state and action(question) called "transition.txt".
  - Output : My code return the appropriate action(question) for each state and the reward value of each state.
  - One thing to note is that the action(question) returned denotes the set of question rather than depicting a single question.

Implementing Jungroo Model

How to calculate the Transition probability ?

- By making such a graph and defining the transition probability and knowing the number of question associated with each state we can calculate the probability associated with each question and state.
- We have to make the graph manually and then calculate the Transition probability.



Question distribution.

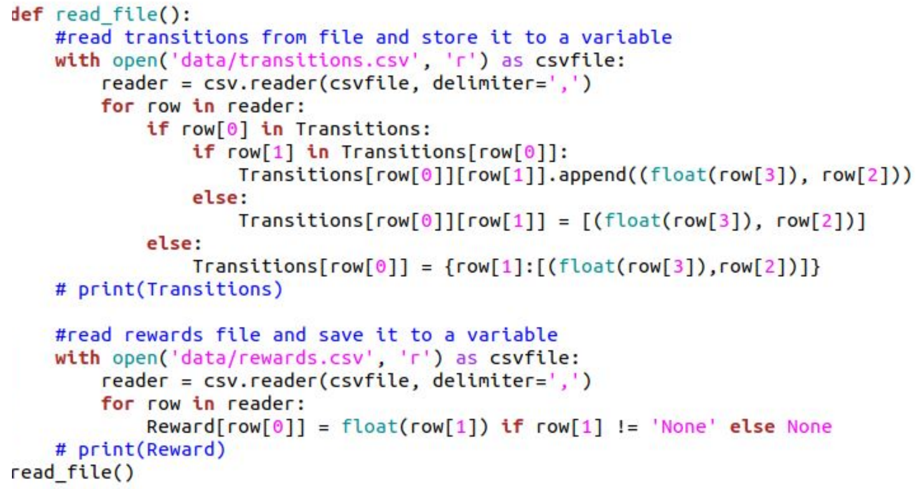
1 - 5 $\rightarrow$ Multiplication
6 - 9 $\rightarrow$ Subtraction
10 - 12 $\rightarrow$ Divi, Mult, Sub
13 - 15 $\rightarrow$ None

MATERATE
YOUR ONLY COMPETITION IS YOU

Implementing Jungroo Model

- Explanation of code:

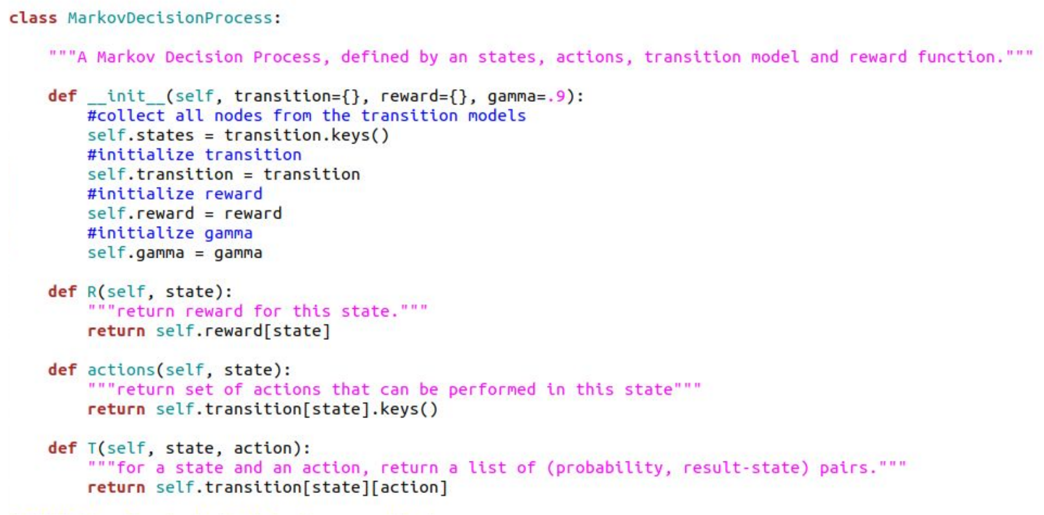
  Given Function reads the rewards value associated with each state and also reads different values associated with Pa(s, s’).



```
def read_file():
    #read transitions from file and store it to a variable
    with open('data/transitions.csv', 'r') as csvfile:
        reader = csv.reader(csvfile, delimiter=',')
        for row in reader:
            if row[0] in Transitions:
                if row[1] in Transitions[row[0]]:
                    Transitions[row[0]][row[1]].append((float(row[3]), row[2]))
                else:
                    Transitions[row[0]][row[1]] = [(float(row[3]), row[2])]
            else:
                Transitions[row[0]] = {row[1]:[(float(row[3]),row[2])]}
    # print(Transitions)

    #read rewards file and save it to a variable
    with open('data/rewards.csv', 'r') as csvfile:
        reader = csv.reader(csvfile, delimiter=',')
        for row in reader:
            Reward[row[0]] = float(row[1]) if row[1] != 'None' else None
    # print(Reward)
read_file()
```

Implementing Jungroo Model

Class that stores the values like rewards associated with the state and transitions values T(s,a).

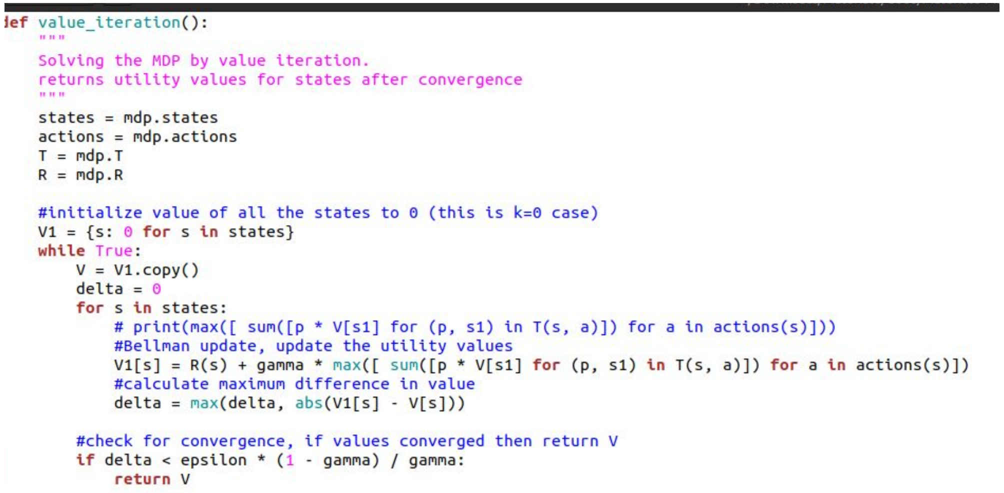


```
class MarkovDecisionProcess:

    """A Markov Decision Process, defined by an states, actions, transition model and reward function."""

    def __init__(self, transition={}, reward={}, gamma=.9):
        #collect all nodes from the transition models
        self.states = transition.keys()
        #initialize transition
        self.transition = transition
        #initialize reward
        self.reward = reward
        #initialize gamma
        self.gamma = gamma

    def R(self, state):
        """return reward for this state."""
        return self.reward[state]

    def actions(self, state):
        """return set of actions that can be performed in this state"""
        return self.transition[state].keys()

    def T(self, state, action):
        """for a state and an action, return a list of (probability, result-state) pairs."""
        return self.transition[state][action]
```

MATERATE
YOUR ONLY COMPETITION IS YOU

Implementing Jungroo Model

This functions calculates the value associated with each state. Exit condition is if the the value converges with difference less than epsilon*(1-gamma)/gamma.

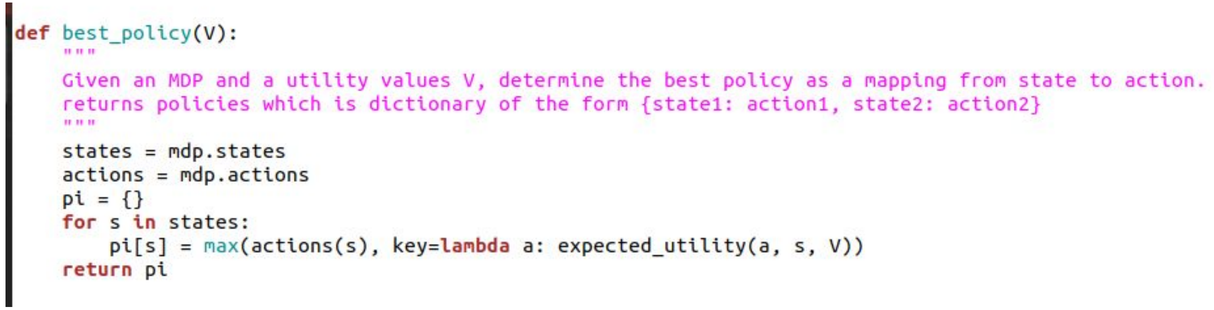


```
ief value_iteration():
    """
    Solving the MDP by value iteration.
    returns utility values for states after convergence
    """
    states = mdp.states
    actions = mdp.actions
    T = mdp.T
    R = mdp.R

    #initialize value of all the states to 0 (this is k=0 case)
    V1 = {s: 0 for s in states}
    while True:
        V = V1.copy()
        delta = 0
        for s in states:
            # print(max([ sum([p * V[s1] for (p, s1) in T(s, a)]) for a in actions(s)]))
            #Bellman update, update the utility values
            V1[s] = R(s) + gamma * max([ sum([p * V[s1] for (p, s1) in T(s, a)]) for a in actions(s)])
            #calculate maximum difference in value
            delta = max(delta, abs(V1[s] - V[s]))

        #check for convergence, if values converged then return V
        if delta < epsilon * (1 - gamma) / gamma:
            return V
```

MATERATE
YOUR ONLY COMPETITION IS YOU

Implementing Jungroo Model

- This function returns best policy for a given state space and transition probability.



```
def best_policy(V):
    """
    Given an MDP and a utility values V, determine the best policy as a mapping from state to action.
    returns policies which is dictionary of the form {state1: action1, state2: action2}
    """
    states = mdp.states
    actions = mdp.actions
    pi = {}
    for s in states:
        pi[s] = max(actions(s), key=lambda a: expected_utility(a, s, V))
    return pi
```

Thank You

By - Mayank Singh Kushwaha

MATERATE
YOUR ONLY COMPETITION IS YOU