REPORT

1. Predicting the Stock Market(DJIA) using Hidden Markov Model

2a. Partner 1: Akshit Arora, aeora44

Akshit worked on finding the dataset for the real-time DJIA values, fetching it to get the desired format in python, finding the start, emission, and transition probabilities for the value. He also contributed to the HMM.py file, though the program structure was created by Sheershak. Akshit created a method to print the Trellis Diagram. They both made the report together.

2b. Partner 2: Sheershar Agarwal, sa1998

Sheershak mainly worked on the HMM.py file, designing the algorithm for the forward and Viterbi. Akshit and Sheershak both worked on the Main.py. They both worked on the Report together. Sheershak created the interactive part of the main.py

3. **What the program is supposed to do**

We modeled the data so that it could be used for predicting the Dow Jones Industrial Average behavior for the week (The program should also be compatible with other stock market indexes such as S&P 500, Nasdaq with some minor changes). The observed states are the real-life scenarios (invest, do not invest, your choice or neutral). Our program will output the Prediction by Forward Algorithm and the Prediction by Viterbi Algorithm as well as the real values.

4. **Techniques used and a brief description**

Hidden Markov models are characterized by three fundamental problems:

- **Problem 1 (Likelihood):** Given an HMM λ = (A, B) and an observation sequence O, determine the likelihood P(O|λ).
- **Problem 2 (Decoding):** Given an observation sequence O and an HMM λ = (A, B), discover the best hidden-state sequence Q.
- **Problem 3 (Learning):** Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

The first problem can be solved by using forward algorithms, the second problem was solved by using the Viterbi algorithm and the Baum–Welch algorithm is used to solve the last problem. In the assignment, we only use the algorithms to solve the first and second problems.

We will first implement the **Forward algorithm** which uses a table to store intermediate values as it builds up the probability of the observation sequence. The forward algorithm computes the observation probability by summing over the probabilities of all possible hidden state paths that could generate the observation sequence, but it does so efficiently by implicitly folding each of these paths into a single forward trellis ultimately calculating the probability of observation to predict trending signals for Dow Jones Industrial Average behavior for the week.

After that, we will implement the **Viterbi Algorithm** which does the task of determining which sequence of variables is the underlying source of some sequence of decoding observation. This algorithm makes use of a dynamic programming trellis.

For this project, we decided to use the DJIA values to predict the behavior of the stock market based on real-time data. The three possible operations are Invest, Do Not Invest, and Your Choice. Each state of the DJIA data (high, low, normal, etc) has a probability value that will cause the operation. Our aim is to predict the probability of a scenario using the above-specified information.

5. **Screenshot of the sample session.**

```
Do you want to run the default test?
Y/N: Y

 Trellis diagram for the Dataset is:

             Invest   Do_Not_Invest   Your_Choice
Low           0.0      0.012            0.0
Moderate_Low  0.061      0.01            0.001
Normal        0.073      0.01            0.002
Moderate_High 0.128      0.01            0.001
High          0.089      0.0            0.001


The predicted probability by the forward algorithm is: 0.036339011715395796

The predicted best path by the viterbi algorithm is: ['High', 'Moderate_High', '
Normal']

The predicted best path probability of algorithm is: 0.0015874822190611664
```

6. **Brief demo instructions**

For getting the above output, please run the main.py file. Main.py is an interactive script that allows the user to either run the default test or not. If you press Y, you will get the above output. However, if you press N, the program will ask you to enter your operation. A comma will be the delimiter. An example of entering operation is below:

```
Do you want to run the default test?
Y/N: N
Your input operation (Invest,Do_Not_Invest,Your_Choice(delemiter = ,)): Invest,Your_Choice

 Trellis diagram for the Dataset is:

             Invest   Your_Choice
Low           0.0      0.003
Moderate_Low  0.061      0.007
Normal        0.073      0.021
Moderate_High 0.128      0.01
High          0.089      0.004


The predicted probability by the forward algorithm is: 0.12184817170111288

The predicted best path by the viterbi algorithm is: ['Moderate_High', 'Normal']

The predicted best path probability of algorithm is: 0.020945945945945947
```

7. **Code excerpt showing some interesting part of your Python code and explanation of it**

```
#Calculating the start probability
for vals in Dow_Jones_Dict:
    temp_val = Dow_Jones_Dict[vals]
    temp_val = float(temp_val)
    if (temp_val < -2.0):
        Low += 1
        tranition_transformation.append("Low");
    elif (temp_val >= -2.0 and temp_val <= -0.5):
        Moderate_Low += 1
        tranition_transformation.append("Moderate_Low");
    elif (temp_val > -0.5 and temp_val < 0.5):
        Normal += 1
        tranition_transformation.append("Normal");
    elif (temp_val > 0.5 and temp_val < 2.0):
        Moderate_High += 1
        tranition_transformation.append("Moderate_High");
    else:
        High += 1
        tranition_transformation.append("High");

start_prob["Low"] = Low/len(Dow_Jones_Dict)
start_prob["High"] = High/len(Dow_Jones_Dict)
start_prob["Normal"] = Normal/len(Dow_Jones_Dict)
start_prob["Moderate_Low"] = Moderate_Low/len(Dow_Jones_Dict)
start_prob["Moderate_High"] = Moderate_High/len(Dow_Jones_Dict)
```

The dataset had the dates and DJIA values. To get an interesting result, Akshit first made another column in excel which had a relative percentage increase or decrease of the DJIA as compared to the previous week. Then Akshit tagged all the data into five hidden states:

Low, Moderate_Low, Normal, Moderate_High, High. The if loop in the above snippet shows how he converts the DJIA percentages into the hidden states.

```
def viterbi(obs, states, start_p, trans_p, emit_p):
        V = [{}]
        path = {}
        # Initialize base cases (t == 0)
        for y in states:
                V[0][y] = start_p[y] * emit_p[y][obs[0]]
                path[y] = [y]
        # Run Viterbi for t > 0
        for t in range(1,len(obs)):
                V.append({})
                newpath = {}
                for y in states:
                        (prob, state) = max([(V[t-1][y0] * trans_p[y0][y] * emit_p[y][obs[t]], y0) for y0 in states])
                        V[t][y] = prob
                        newpath[y] = path[state] + [y]
                # Don't need to remember the old paths
                path = newpath
        print_brief_fns(obs, V)
        (prob, state) = max([(V[len(obs) - 1][y], y) for y in states])
        return (prob, path[state])
```

This is the Viterbi algorithm Sheershak implemented. First Sheershak made a pseudo code for this algorithm and then both the partners worked on it, though Sheershak played the main role in this.

If you read the snippet, you can see that the Viterbi method calls the print_brief_fns(obs, V) which prints the Trellis Table. Below is the code for that

```
# Prints sequence of belief functions in a sort of vertical format using print statements
def print_brief_fns(obs, V):
        s = "              "
        space = []
        temp_space = ""
        space.append(temp_space)
        for i in obs:
                s = s + i + "   "
                for j in range(1, len(i) - 6):
                        temp_space += " ";
                space.append(temp_space)
        s = s + '\n'
        for y in V[0]:
                s += y
                for i in range(len(y), 14):
                        s = s + " ";
                for index in range(0,len(obs)):
                        s = s + space[index] + str(round(V[index][y], 3)) + "       "
                s += "\n"
        print("\n Trellis diagram for the Dataset is: \n")
        print(s)
        print("\n")
```

Akshit developed the print_brief_fns(obs, V) method. He uses the string concatenation for getting the desired output.

8. **Brief description of what each team member learned in this project**

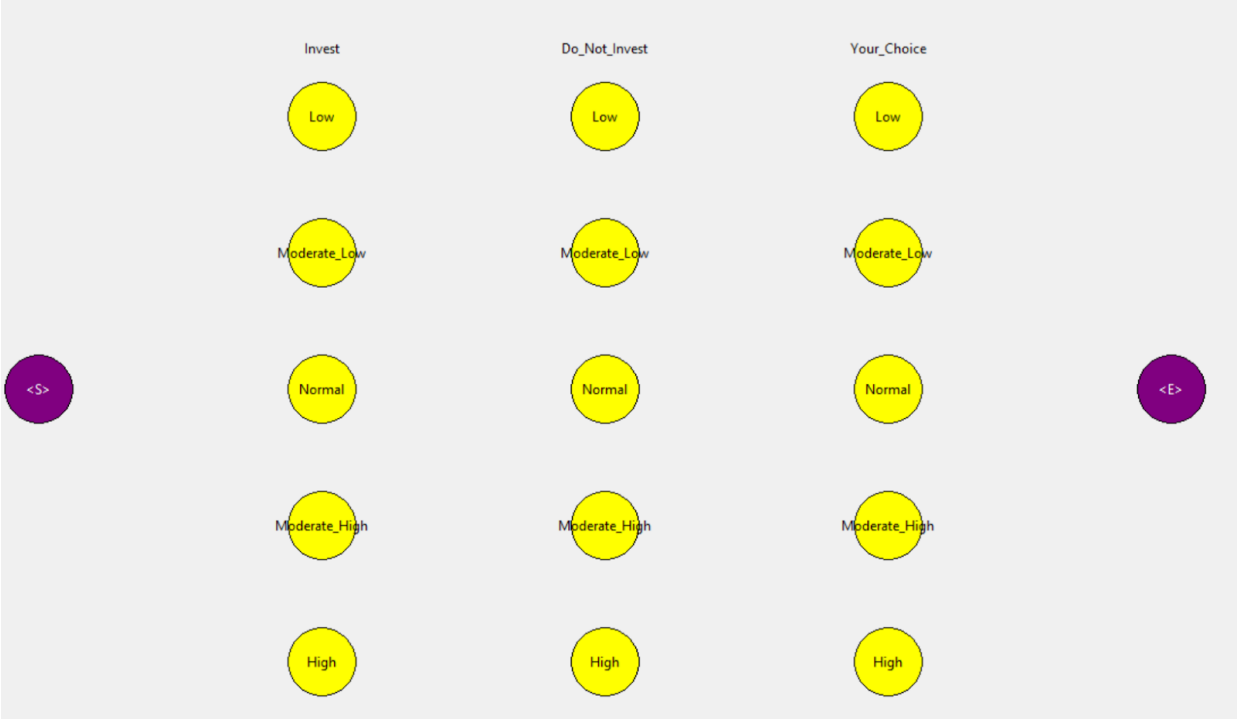
Both Akshit and Sheershak learned about the industrial application of Hidden Markov Models as well as how to implement them. They both learned about the Forward and Viterbi algorithms.

Akshit also learned how to model the raw data to get the states required for predicting the probabilities. He also learned how to create a custom visualization dashboard.

Sheershak learned how to write pseudo-codes and how to create an interactive application that allows the user to enter an input.

9. **What you would like to add to your program if you had more time.**

If we had more time, we would have created a GUI that prints the graphical dashboard on the canvas. We would also have added some animation showing how we got the best path. We tried creating a GUI but weren't able to complete it. Hence, we didn't submit the program for grading. Below's a screenshot of how much we got it working.



10. **Citations for any references you used in the project**

Stanford Notes for HMM: https://web.stanford.edu/~jurafsky/slp3/A.pdf

Stock Market Prediction using HMM: https://www.cs.cmu.edu/~bdhingra/papers/stock_hmm.pdf (We referred to this paper for finding some of the thresholds required for creating the hidden states)

Wikipedia HMM: https://en.wikipedia.org/wiki/Hidden_Markov_model

11. **Partners' Reflections**

Partner 1: Akshit Arora

My partner was Sheershak Agarwal and he contributed equally towards the project. His main role was to research about the algorithms, find a mathematical model, and create the main file for executing the program. Even though we had some conflicts with our ideas, we resolved them in a professional way and came up with an idea to predict the stock market. I think the main benefit of partnering with Sheershak was pair programming and good communication skill. He also helped me some other things such as documenting the code, etc.

Partner 2: Sheershak Agarwal

Akshit played an important role in the HMM project. He spent time finding a real-time dataset for the stock market and modeled it into what we needed for our project. He also worked with me on coding the algorithms and he developed a dashboard for printing the Trellis Table. We had some communication problems due to our different schedules but we fixed them. Benefits of having Akshit as a partner were the share of the workload and his good modeling and visualization skills.