Test1

March 18, 2019

```
In [1]: import numpy as np
```

0.1 Load the data into a 2-dimensional numpy array

```
In [2]: data1 = np.loadtxt("RPS_transcript1.txt", delimiter =',', usecols=range(0,3), dtype=np
        data2 = np.loadtxt("RPS_transcript2.txt", delimiter =',', usecols=range(0,3), dtype=np
        data1.shape
```

```
Out[2]: (300, 3)
```

0.2 Determine the count of times the agent played rock, paper, and scissor respectively

```
In [3]: agent2Data = data2[:, 1]

        print('\nFor agent2 in RPS_transcript2, counts of rock, paper, and scissor:')
        RPS_counts = [(agent2Data == i).sum() for i in range(3)]
        print( np.array(RPS_counts))
```

```
For agent2 in RPS_transcript2, counts of rock, paper, and scissor:
[ 89 109 102]
```

0.3 Determine the prior probabilities the agent played rock, paper, and scissor respectively

```
In [4]: agent2Data = data2[:, 1]


        print('\nFor agent2 in RPS_transcript2, prior distrubutions of rock, paper, and scisso
        RPS_priorPrbabilities = np.array(  [(agent2Data == i).sum() for i in range(3)]  ) / ag
        print( RPS_priorPrbabilities)
```

```
For agent2 in RPS_transcript2, prior distrubutions of rock, paper, and scissor:
[0.29666667 0.36333333 0.34      ]
```

0.4 Determine the counts of times the agent played rock, paper, and scissor respectively in the previous match when in the current match the agent plays rock

```
In [5]: agent2Data = data2[:, 1]
        isRock = (agent2Data == 0)
        agent2DataPrev1GivenRockNow = agent2Data[:-1][ isRock[1:]]
        RPS_percentagesPrev1GivenRockNow = [(agent2DataPrev1GivenRockNow == i).sum() for i in
        print("""\nFor agent 2 in RPS_transcript2: counts of times the agent played rock, paper
              in the previous match when the agent plays rock in the current match:""")
        print(RPS_percentagesPrev1GivenRockNow)
```

```
For agent 2 in RPS_transcript2: counts of times the agent played rock, paper, and scissor respe
      in the previous match when the agent plays rock in the current match:
[12, 16, 60]
```

0.5 Generalize it to determine the probabilities the agent played rock, paper, and scissor respectively in the previous match when in the current match the agent plays rock, paper, and scissor respectively

```
In [6]: print('\nFor agent 2 in RPS_transcript2:')
        agentData = data2[:, 1]
        actions = ('rock', 'paper', 'scissor')
        counts = []
        for i, action in enumerate(actions):
          isAction = (agentData == i)
          agentDataPrev1GivenActionNow = agentData[:-1][ isAction[1:]]
          RPS_percentagesPrev1GivenActionNow = [(agentDataPrev1GivenActionNow == j).sum() for j
          print('\nCounts of times the agent played rock, paper, and scissor respectively')
          print('in the previous match when the agent plays ', actions[i], ' in the current mat
          print(RPS_percentagesPrev1GivenActionNow)
          counts.append( RPS_percentagesPrev1GivenActionNow )

        counts = np.array(counts)
        print("\nCounts stored in a 2-D numpy array:\n", counts)
        print("\nVerify the total counts in the 2-D numpy array:", counts.sum())

        print('Verify the total counts on each row:', counts.sum(axis = 1) )
        rowCounts = counts.sum(axis = 1)

        prAgentPrev1GivenAgentNow = counts/rowCounts[:, np.newaxis]
        print("\nConvert it to probabilties:", prAgentPrev1GivenAgentNow)
        print("\nVerify each row in the 2-D numpy array sums up to 1:", prAgentPrev1GivenAgentN
```

```
For agent 2 in RPS_transcript2:

Counts of times the agent played rock, paper, and scissor respectively
```

```
in the previous match when the agent plays  rock  in the current match:
[12, 16, 60]

Counts of times the agent played rock, paper, and scissor respectively
in the previous match when the agent plays  paper  in the current match:
[61, 24, 24]

Counts of times the agent played rock, paper, and scissor respectively
in the previous match when the agent plays  scissor  in the current match:
[16, 69, 17]

Counts stored in a 2-D numpy array:
 [[12 16 60]
 [61 24 24]
 [16 69 17]]

Verify the total counts in the 2-D numpy array: 299
Verify the total counts on each row: [ 88 109 102]

Convert it to probabilties: [[0.13636364 0.18181818 0.68181818]
 [0.55963303 0.22018349 0.22018349]
 [0.15686275 0.67647059 0.16666667]]

Verify each row in the 2-D numpy array sums up to 1: [1. 1. 1.]
```

0.6 Similarly, determine the probabilities the user played rock, paper, and scissor respectively two matches ago when in the current match the agent plays rock, paper, and scissor respectively

```
In [7]: print('\nFor agent 2 in RPS_transcript2:')
        agentData = data2[:, 1]
        userData = data2[:, 0]
        actions = ('rock', 'paper', 'scissor')
        counts = []
        for i, action in enumerate(actions):
          isAction = (agentData == i)
          userDataPrev2GivenActionNow = userData[:-2][ isAction[2:]]
          RPS_percentagesPrev2GivenActionNow = [(userDataPrev2GivenActionNow == j).sum() for j
          print('\nCounts of times the user played rock, paper, and scissor respectively')
          print('in 2 matches ago when the agent plays ', actions[i], ' in the current match:')
          print(RPS_percentagesPrev2GivenActionNow)
          counts.append( RPS_percentagesPrev2GivenActionNow )

        counts = np.array(counts)
        print("\nCounts stored in a 2-D numpy array:\n", counts)
        print("\nVerify the total counts in the 2-D numpy array:", counts.sum())
```

```
        print('Verify the total counts on each row:', counts.sum(axis = 1) )
        rowCounts = counts.sum(axis = 1)

        prUserPrev2GivenAgentNow = counts/rowCounts[:, np.newaxis]
        print("\nConvert it to probabilties:", prUserPrev2GivenAgentNow)
        print("\nVerify each row in the 2-D numpy array sums up to 1:", prUserPrev2GivenAgentN
```

```
For agent 2 in RPS_transcript2:

Counts of times the user played rock, paper, and scissor respectively
in 2 matches ago when the agent plays  rock  in the current match:
[29, 35, 24]

Counts of times the user played rock, paper, and scissor respectively
in 2 matches ago when the agent plays  paper  in the current match:
[33, 33, 42]

Counts of times the user played rock, paper, and scissor respectively
in 2 matches ago when the agent plays  scissor  in the current match:
[34, 35, 33]

Counts stored in a 2-D numpy array:
 [[29 35 24]
 [33 33 42]
 [34 35 33]]

Verify the total counts in the 2-D numpy array: 298
Verify the total counts on each row: [ 88 108 102]

Convert it to probabilties: [[0.32954545 0.39772727 0.27272727]
 [0.30555556 0.30555556 0.38888889]
 [0.33333333 0.34313725 0.32352941]]

Verify each row in the 2-D numpy array sums up to 1: [1. 1. 1.]
```

0.7 Show how we may collect the Naive Bayes statistics of derived in the cells above into a dictionary

```
In [8]: print('\nFor agent 2 in RPS_transcript2, the following are part of the Naive Bayes sta
        NaiveBasyesStatistics = { 'prior':RPS_priorPrbabilities, 'userPrev2':prUserPrev2GivenA
        #NaiveBasyesStatistics['userPrev2']
        NaiveBasyesStatistics
```

```
For agent 2 in RPS_transcript2, the following are part of the Naive Bayes statistics Bayes
```

```
Out[8]: {'prior': array([0.29666667, 0.36333333, 0.34      ]),
         'userPrev2': array([[0.32954545, 0.39772727, 0.27272727],
                [0.30555556, 0.30555556, 0.38888889],
                [0.33333333, 0.34313725, 0.32352941]]),
         'agentPrev1': array([[0.13636364, 0.18181818, 0.68181818],
                [0.55963303, 0.22018349, 0.22018349],
                [0.15686275, 0.67647059, 0.16666667]])}
```

1 Test1 function 1: Define getNaiveBayesStatistics(TrainingData) to automate similar calculations to gather all the Naive Bayes statistics

```
In [9]: def getNaiveBayesStatistics(TrainingData):
            #Code removed
            #Write your own code to define the function


        ##############################################################################
        #Test the function defined above, and print out the Naive Bayes statistics collected
        print("\nFor agent 1 in RPS_transcript1: ")
        NaiveBayesStatistics1 = getNaiveBayesStatistics(data1)
        print( NaiveBayesStatistics1 )

        print("\nFor agent 2 in RPS_transcript2: ")
        NaiveBayesStatistics2 = getNaiveBayesStatistics(data2)
        print( NaiveBayesStatistics2 )
```

```
For agent 1 in RPS_transcript1:
{'prior': array([0.29, 0.4 , 0.31]), 'agentPrev1': array([[0.24418605, 0.44186047, 0.31395349]
       [0.30833333, 0.38333333, 0.30833333],
       [0.31182796, 0.37634409, 0.31182796]]), 'agentPrev2': array([[0.31395349, 0.45348837, 0
       [0.29166667, 0.36666667, 0.34166667],
       [0.26086957, 0.39130435, 0.34782609]]), 'userPrev1': array([[0.38372093, 0.27906977, 0.3
       [0.3       , 0.375     , 0.325     ],
       [0.29032258, 0.3655914 , 0.34408602]]), 'userPrev2': array([[0.3372093 , 0.3255814 , 0.3
       [0.275     , 0.39166667, 0.33333333],
       [0.36956522, 0.29347826, 0.33695652]])}

For agent 2 in RPS_transcript2:
{'prior': array([0.29666667, 0.36333333, 0.34      ]), 'agentPrev1': array([[0.13636364, 0.1818
       [0.55963303, 0.22018349, 0.22018349],
       [0.15686275, 0.67647059, 0.16666667]]), 'agentPrev2': array([[0.21590909, 0.55681818, 0.
       [0.23148148, 0.2962963 , 0.47222222],
       [0.44117647, 0.26470588, 0.29411765]]), 'userPrev1': array([[0.34090909, 0.35227273, 0.3
       [0.30275229, 0.33944954, 0.35779817],
       [0.33333333, 0.34313725, 0.32352941]]), 'userPrev2': array([[0.32954545, 0.39772727, 0.2
       [0.30555556, 0.30555556, 0.38888889],
```

```
        [0.33333333, 0.34313725, 0.32352941]])}
```

1.1 Test1 function 2: Define getProbabilityNaiveBayesOneCase(NiaiveStatisticsDictionary, newCase), which returns a 1-D numpy array storing the predicted probabilities of seeing the agent playing rock or paper or scissor given a newCase

```
In [10]: def getProbabilityNaiveBayesOneCase(NiaiveStatisticsDictionary, newCase):
             #Code removed
             #Write your own code to define the function


         ##########################################################################################
         #Test the function defined above, and print out the Naive Bayes statistics collected

         #Test case: the agent played Rock and Paper in the previous two moves
         #  while the user played Scissor and Rock in the previous two movesmy

         newCase = np.array([0,1,2,0])

         print("\n*****************************************************")
         print("For agent 1 in RPS_transcript1: ")
         print( getProbabilityNaiveBayesOneCase(NaiveBayesStatistics1, newCase)  )

         print("\n*****************************************************")
         print("For agent 2 in RPS_transcript2: ")
         print( getProbabilityNaiveBayesOneCase(NaiveBayesStatistics2, newCase)  )
```

```
*****************************************************
For agent 1 in RPS_transcript1:
History: Agent played  Rock then  Paper
History: User  played  Scissor then  Rock

likelihood of Rock = 0.005205512473525075
prior( Rock ) 0.29 *
prAgentPrev2GivenAgentNow( Rock | Rock ) 0.313953488372093 *
prAgentPrev1GivenAgentNow( Paper | Rock ) 0.4418604651162791 *
prUserPrev2GivenAgentNow( Scissor | Rock ) 0.3372093023255814 *
prUserPrev1GivenAgentNow( Rock | Rock ) 0.38372093023255816

likelihood of Paper = 0.004472222222222223
prior( Paper ) 0.4 *
prAgentPrev2GivenAgentNow( Rock | Paper ) 0.2916666666666667 *
prAgentPrev1GivenAgentNow( Paper | Paper ) 0.38333333333333336 *
prUserPrev2GivenAgentNow( Scissor | Paper ) 0.3333333333333333 *
prUserPrev1GivenAgentNow( Rock | Paper ) 0.3
```

```
likelihood of Scissor = 0.0029773156899810965
prior( Scissor ) 0.31 *
prAgentPrev2GivenAgentNow( Rock | Scissor ) 0.2608695652173913 *
prAgentPrev1GivenAgentNow( Paper | Scissor ) 0.3763440860215054 *
prUserPrev2GivenAgentNow( Scissor | Scissor ) 0.33695652173913043 *
prUserPrev1GivenAgentNow( Rock | Scissor ) 0.2903225806451613

Normalize the likelihoods into probabilities: [0.41133874 0.35339426 0.235267  ]
[0.41133874 0.35339426 0.235267  ]

**************************************************
For agent 2 in RPS_transcript2:
History: Agent played  Rock then  Paper
History: User  played  Scissor then  Rock

likelihood of Rock = 0.0010827897684584386
prior( Rock ) 0.2966666666666667 *
prAgentPrev2GivenAgentNow( Rock | Rock ) 0.2159090909090909 *
prAgentPrev1GivenAgentNow( Paper | Rock ) 0.18181818181818182 *
prUserPrev2GivenAgentNow( Scissor | Rock ) 0.2727272727272727 *
prUserPrev1GivenAgentNow( Rock | Rock ) 0.3409090909090909

likelihood of Paper = 0.0021803148714463704
prior( Paper ) 0.36333333333333334 *
prAgentPrev2GivenAgentNow( Rock | Paper ) 0.23148148148148148 *
prAgentPrev1GivenAgentNow( Paper | Paper ) 0.22018348623853212 *
prUserPrev2GivenAgentNow( Scissor | Paper ) 0.3888888888888889 *
prUserPrev1GivenAgentNow( Rock | Paper ) 0.30275229357798167

likelihood of Scissor = 0.010942906574394465
prior( Scissor ) 0.34 *
prAgentPrev2GivenAgentNow( Rock | Scissor ) 0.4411764705882353 *
prAgentPrev1GivenAgentNow( Paper | Scissor ) 0.6764705882352942 *
prUserPrev2GivenAgentNow( Scissor | Scissor ) 0.3235294117647059 *
prUserPrev1GivenAgentNow( Rock | Scissor ) 0.3333333333333333

Normalize the likelihoods into probabilities: [0.07622053 0.15347833 0.77030114]
[0.07622053 0.15347833 0.77030114]
```

1.2 Test1 function 3: Define predictNaiveBayesOneCase(NiaiveStatisticsDictionary, newCase) that can return the most likely action (rock or paper or scissor as a number)

```
In [11]: def predictNaiveBayesOneCase(NiaiveStatisticsDictionary, newCase):
             #Code removed
             #Write your own code to define the function
```

```
        ##########################################################################################
        #Test the function defined above, and print out the Naive Bayes statistics collected

        #Test case: the agent played Rock and Paper in the previous two moves
        #  while the user played Scissor and Rock in the previous two movesmy

        newCase = np.array([0,1,2,0])
        actions = ('Rock', 'Paper', 'Scissor')

        print("\nFor agent 1 in RPS_transcript1: ")
        outcome1 = predictNaiveBayesOneCase(NaiveBayesStatistics1, newCase)
        print( 'Outcome:', outcome1, '==>',  actions[outcome1])

        print("\nFor agent 2 in RPS_transcript2: ")
        outcome2 = predictNaiveBayesOneCase(NaiveBayesStatistics2, newCase)
        print( 'Outcome:', outcome2, '==>',  actions[outcome2])
```

```
For agent 1 in RPS_transcript1:
Outcome: 0 ==> Rock

For agent 2 in RPS_transcript2:
Outcome: 2 ==> Scissor
```

1.3 Test1 function 4: Define the function predictNaiveBayesWholeTranscript(NiaiveStatisticsDictionary, TestingData)

```
In [12]: def predictNaiveBayesWholeTranscript(NiaiveStatisticsDictionary, TestingData):
             #Code removed
             #Write your own code to define the function


         ##########################################################################################
         #Test the function defined above, and print out the Naive Bayes statistics collected

         print("\nFor agent 1 in RPS_transcript1: ")
         outcome = predictNaiveBayesWholeTranscript(NaiveBayesStatistics1, data1)
         print('Predicting Agent 1 based on Naive Bayes model learned from Agent 1 is \n', out

         outcome = predictNaiveBayesWholeTranscript(NaiveBayesStatistics2, data1)
         print('Predicting Agent 1 based on Naive Bayes model learned from Agent 1 is \n', out


         print("\nFor agent 2 in RPS_transcript2: ")
         outcome = predictNaiveBayesWholeTranscript(NaiveBayesStatistics1, data2)
         print('Predicting Agent 2 based on Naive Bayes model learned from Agent 1 is \n', out
```

```
        outcome = predictNaiveBayesWholeTranscript(NaiveBayesStatistics2, data2)
        print('Predicting Agent 2 based on Naive Bayes model learned from Agent 2 is \n', out
```

```
For agent 1 in RPS_transcript1:
Predicting Agent 1 based on Naive Bayes model learned from Agent 1 is
 [1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 2. 1. 1. 1. 1. 1. 0. 2. 1. 1. 0. 0. 1. 1.
 2. 0. 1. 1. 0. 2. 1. 2. 1. 2. 0. 2. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0. 2. 1.
 0. 1. 1. 0. 0. 2. 1. 0. 0. 0. 2. 0. 0. 2. 1. 1. 2. 2. 1. 1. 1. 1. 0. 0.
 0. 2. 1. 2. 0. 0. 0. 2. 1. 0. 2. 1. 2. 1. 2. 1. 1. 0. 0. 1. 1. 2. 1. 1.
 1. 1. 1. 1. 1. 2. 0. 2. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 0. 0. 0. 2. 0.
 2. 2. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 2. 1. 1. 1. 1. 1. 0.
 2. 0. 2. 1. 1. 1. 1. 0. 1. 1. 1. 0. 2. 1. 1. 1. 1. 1. 1. 1. 0. 2. 1. 1.
 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 2. 1. 1. 1. 2. 1. 2. 0. 0. 1. 1. 0.
 1. 1. 1. 0. 2. 1. 1. 1. 1. 1. 2. 0. 0. 1. 2. 1. 0. 2. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 2. 1. 0. 2. 1.
 1. 0. 1. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 0. 2. 1. 0. 2. 1. 1. 0.
 1. 1. 1. 0. 1. 1. 1. 1. 2. 1. 1. 1. 1. 0. 0. 0. 2. 1. 1. 1. 1. 1. 0. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Predicting Agent 1 based on Naive Bayes model learned from Agent 1 is
 [0. 1. 0. 0. 2. 0. 0. 1. 2. 2. 0. 1. 2. 2. 0. 2. 0. 1. 0. 1. 2. 1. 1. 0.
 1. 2. 1. 1. 0. 0. 2. 1. 0. 2. 2. 2. 1. 2. 2. 1. 2. 0. 1. 2. 2. 0. 1. 2.
 2. 0. 1. 2. 2. 1. 1. 2. 1. 0. 1. 2. 0. 2. 2. 0. 0. 0. 1. 1. 1. 2. 2. 2.
 0. 2. 0. 1. 2. 2. 2. 1. 2. 2. 1. 2. 0. 0. 2. 0. 2. 2. 1. 1. 0. 0. 0. 0.
 1. 0. 2. 1. 2. 1. 2. 1. 0. 1. 0. 0. 2. 1. 1. 2. 2. 1. 2. 2. 2. 0. 1. 0.
 2. 1. 0. 1. 2. 2. 0. 2. 2. 2. 1. 1. 0. 0. 2. 0. 1. 2. 0. 1. 2. 0. 1. 0.
 2. 0. 2. 1. 2. 2. 1. 2. 1. 2. 2. 2. 0. 0. 1. 2. 1. 0. 2. 2. 2. 1. 2. 2.
 1. 0. 2. 2. 1. 2. 1. 0. 2. 1. 2. 0. 2. 0. 0. 2. 2. 0. 2. 2. 2. 1. 2. 1.
 1. 1. 1. 0. 0. 0. 0. 2. 0. 1. 2. 1. 0. 2. 0. 2. 0. 1. 1. 1. 0. 1. 2. 1.
 0. 0. 0. 1. 2. 0. 2. 2. 0. 1. 0. 1. 0. 0. 2. 2. 0. 2. 0. 0. 2. 0. 1. 2.
 2. 2. 2. 2. 2. 1. 2. 2. 0. 2. 2. 2. 1. 0. 1. 1. 0. 0. 1. 0. 2. 0. 2. 2.
 1. 0. 1. 0. 2. 1. 1. 2. 1. 2. 2. 1. 1. 2. 2. 2. 1. 1. 1. 2. 2. 1. 2. 0.
 1. 2. 0. 1. 1. 1. 1. 0. 1. 1.]

For agent 2 in RPS_transcript2:
Predicting Agent 2 based on Naive Bayes model learned from Agent 1 is
 [1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 0. 2. 1. 0. 1. 1. 1. 0. 2. 1. 1. 1. 1. 2.
 2. 1. 1. 2. 1. 1. 2. 1. 0. 2. 1. 1. 1. 2. 1. 0. 2. 1. 2. 0. 2. 1. 0. 2.
 1. 1. 2. 1. 1. 1. 0. 1. 1. 1. 1. 2. 0. 0. 1. 1. 0. 0. 2. 2. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 2. 1. 1. 1. 1. 0. 2. 1. 1. 1. 2. 1. 2.
 1. 1. 2. 1. 1. 0. 2. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 2. 0.
 0. 2. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 2. 1. 2. 1. 1. 1. 1. 0. 2. 1.
 1. 1. 0. 2. 0. 2. 1. 1. 1. 1. 0. 2. 1. 1. 1. 0. 2. 1. 0. 2. 1. 1. 1. 1.
 1. 2. 0. 2. 1. 1. 1. 1. 1. 2. 0. 1. 1. 1. 2. 1. 1. 2. 1. 1. 1. 1. 1. 2.
 1. 1. 2. 1. 0. 1. 0. 0. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 2. 1. 1. 0. 2.
 1. 0. 0. 1. 0. 2. 1. 1. 1. 1. 1. 0. 2. 1. 0. 1. 0. 2. 1. 1. 1. 0. 1. 1.
 2. 1. 0. 0. 2. 1. 1. 2. 1. 1. 2. 1. 1. 1. 1. 1. 0. 2. 1. 1. 1. 1. 2. 1.
```

```
 0. 1. 1. 1. 1. 2. 1. 1. 0. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 2. 0.
 0. 0. 2. 1. 1. 1. 1. 0. 1. 1.]
Predicting Agent 2 based on Naive Bayes model learned from Agent 2 is
 [2. 0. 0. 1. 1. 2. 0. 1. 0. 1. 2. 0. 2. 2. 0. 1. 2. 2. 1. 2. 2. 0. 1. 2.
 0. 2. 0. 1. 2. 0. 1. 2. 2. 0. 1. 2. 0. 2. 2. 0. 1. 0. 1. 2. 1. 2. 2. 0.
 1. 2. 1. 2. 0. 1. 2. 0. 2. 2. 0. 1. 2. 2. 0. 1. 2. 0. 0. 1. 1. 2. 0. 1.
 2. 0. 1. 2. 0. 1. 1. 2. 0. 1. 2. 0. 0. 1. 1. 2. 2. 2. 0. 1. 2. 0. 2. 0.
 1. 1. 2. 0. 1. 2. 1. 0. 1. 1. 1. 2. 0. 1. 1. 2. 0. 2. 0. 1. 1. 0. 1. 2.
 2. 0. 2. 0. 2. 0. 1. 2. 0. 1. 2. 0. 1. 2. 0. 2. 1. 1. 0. 1. 2. 2. 1. 2.
 0. 2. 0. 1. 0. 0. 1. 2. 0. 1. 0. 0. 2. 0. 1. 2. 0. 1. 2. 0. 1. 1. 2. 1.
 0. 1. 2. 2. 0. 1. 0. 1. 0. 1. 2. 1. 2. 0. 1. 0. 2. 0. 1. 2. 0. 1. 2. 0.
 0. 1. 1. 1. 0. 1. 2. 2. 0. 1. 2. 0. 1. 2. 0. 1. 2. 1. 0. 2. 0. 2. 0. 0.
 2. 2. 2. 1. 2. 0. 1. 1. 2. 1. 2. 2. 0. 2. 2. 2. 2. 0. 1. 0. 1. 2. 1. 2.
 0. 1. 2. 2. 0. 1. 2. 0. 1. 2. 0. 2. 1. 2. 1. 1. 0. 1. 0. 1. 2. 0. 1. 1.
 2. 1. 2. 0. 2. 0. 1. 2. 2. 1. 2. 0. 1. 2. 0. 0. 1. 1. 2. 0. 2. 0. 1. 2.
 2. 0. 1. 0. 2. 0. 2. 2. 0. 2.]
```

1.4 Test1 function 5: Define the function getPrecisionNaiveBayesWholeTranscript(NiaiveStatisticsDictionary, TestingData)

```
In [13]: def getPrecisionNaiveBayesWholeTranscript(NiaiveStatisticsDictionary, TestingData):
             #Code removed
             #Write your own code to define the function


         ##############################################################################
         #Test the function defined above, and print out the Naive Bayes statistics collected

         print("\nFor agent 1 in RPS_transcript1: ")
         outcome = getPrecisionNaiveBayesWholeTranscript(NaiveBayesStatistics1, data1)
         print('Precision of predicting Agent 1 based on Naive Bayes model learned from Agent

         outcome = getPrecisionNaiveBayesWholeTranscript(NaiveBayesStatistics2, data1)
         print('Precision of predicting Agent 1 based on Naive Bayes model learned from Agent


         print("\nFor agent 2 in RPS_transcript2: ")
         outcome = getPrecisionNaiveBayesWholeTranscript(NaiveBayesStatistics1, data2)
         print('Precision of predicting Agent 2 based on Naive Bayes model learned from Agent

         outcome = getPrecisionNaiveBayesWholeTranscript(NaiveBayesStatistics2, data2)
         print('Precision of predicting Agent 2 based on Naive Bayes model learned from Agent 2


For agent 1 in RPS_transcript1:
Precision of predicting Agent 1 based on Naive Bayes model learned from Agent 1 is
 0.4261744966442953
```

```
Precision of predicting Agent 1 based on Naive Bayes model learned from Agent 1 is
 0.34563758389261745

For agent 2 in RPS_transcript2:
Precision of predicting Agent 2 based on Naive Bayes model learned from Agent 1 is
 0.348993288590604
Precision of predicting Agent 2 based on Naive Bayes model learned from Agent 2 is
 0.6308724832214765
```