Natural Language Processing-Assignment-5

Name: Adithi Shinde

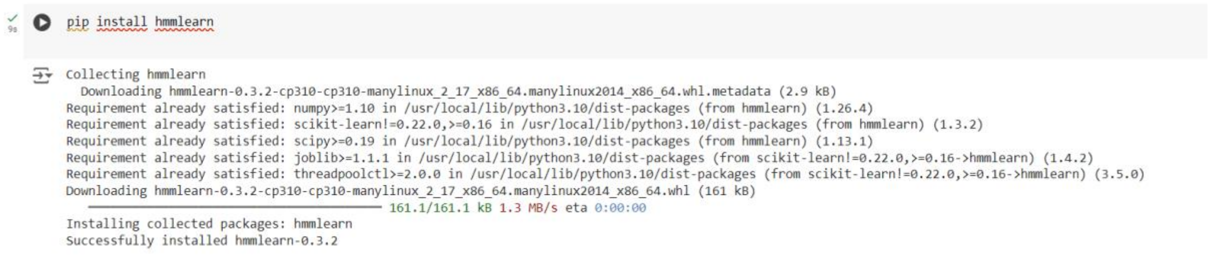
Enrollment no:2203A54040

1.Implement a Hidden Markov Model (HMM) for weather forecasting, taking scenario where we have hidden weather states (e.g., "Sunny" and "Rainy") and observable events (e.g., "Dry", "Damp", "Wet"). The goal is to predict the sequence of weather states based on the observations. [CO2]

2.Problem Setup States (S): Hidden states representing the weather (e.g., "Sunny", "Rainy"). **Observations** (O): Observable events that we can measure (e.g., "Dry", "Damp", "Wet"). Transition **Probabilities** (A): The probability of transitioning from one weather state to another. Emission **Probabilities** (B): The probability of observing a certain event given the weather state. Initial **Probabilities** (π): The probability distribution over the initial states.

2.Take different transition and emission probabilities or observation sequences on above problem and see how the model's predictions change. [CO2] give python codes for the above questions

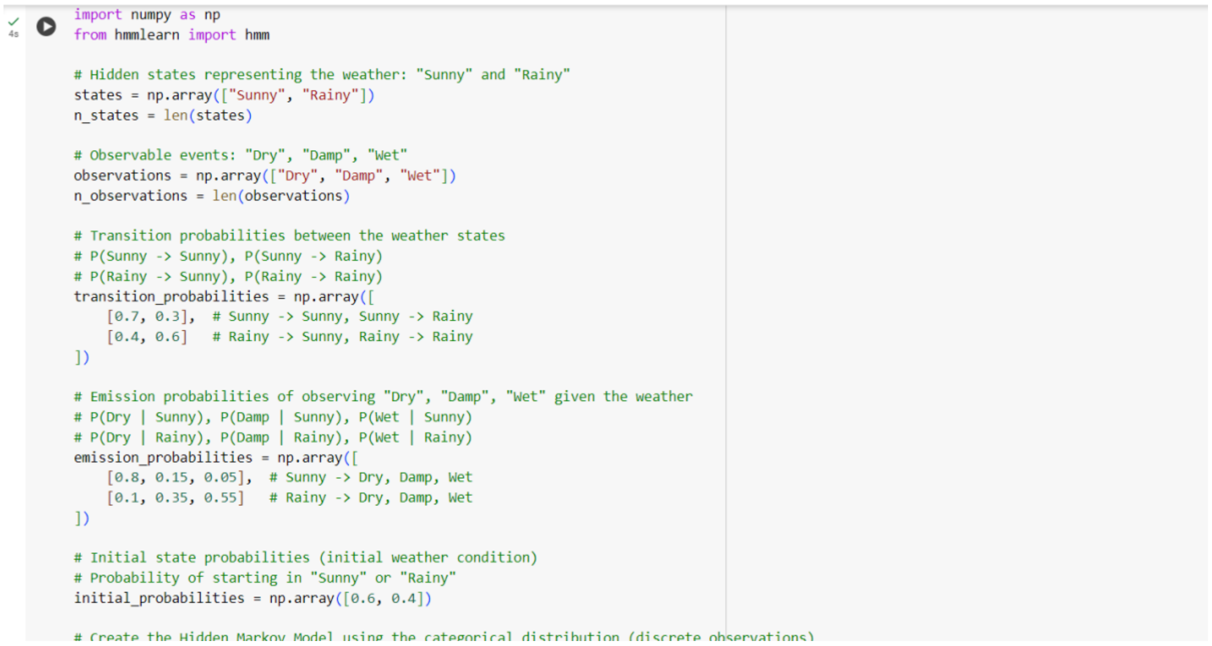
You might need to install the "hmmlearn" package first if it's not installed already:



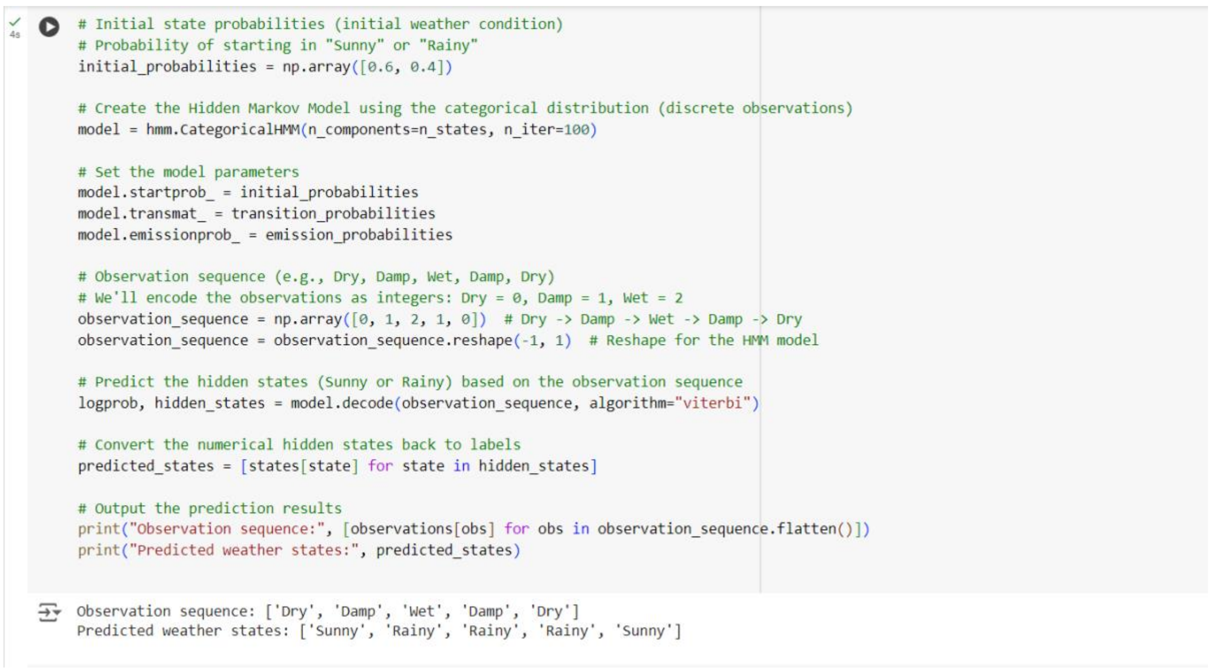
```
pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading hmmlearn-0.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (2.9 kB)
Requirement already satisfied: numpy>=1.10 in /usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.26.4)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in /usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.3.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.13.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (3.5.0)
Downloading hmmlearn-0.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (161 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 161.1/161.1 kB 1.3 MB/s eta 0:00:00
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.2
```

Step 1: Implementing the Hidden Markov Model (HMM) for Weather Forecasting.



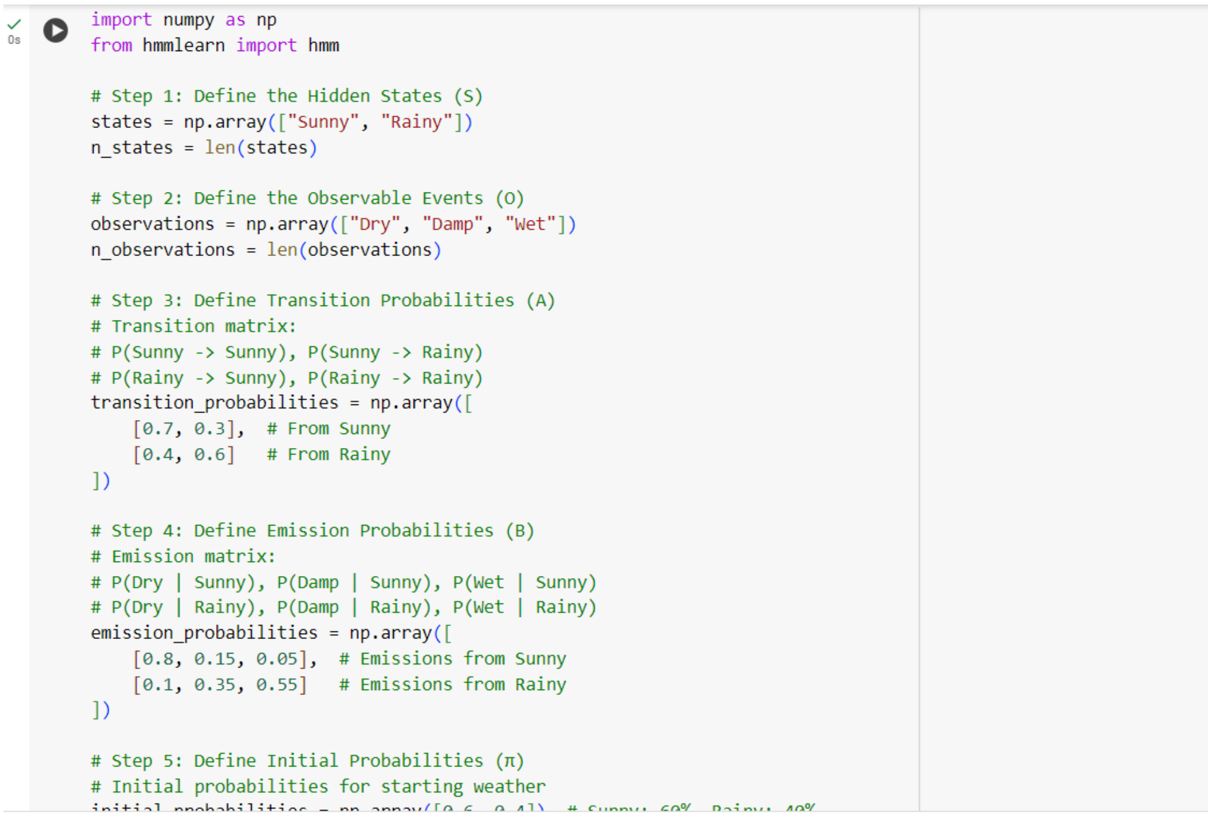
```python
import numpy as np
from hmmlearn import hmm

# Hidden states representing the weather: "Sunny" and "Rainy"
states = np.array(["Sunny", "Rainy"])
n_states = len(states)

# Observable events: "Dry", "Damp", "Wet"
observations = np.array(["Dry", "Damp", "Wet"])
n_observations = len(observations)

# Transition probabilities between the weather states
# P(Sunny -> Sunny), P(Sunny -> Rainy)
# P(Rainy -> Sunny), P(Rainy -> Rainy)
transition_probabilities = np.array([
    [0.7, 0.3],  # Sunny -> Sunny, Sunny -> Rainy
    [0.4, 0.6]   # Rainy -> Sunny, Rainy -> Rainy
])

# Emission probabilities of observing "Dry", "Damp", "Wet" given the weather
# P(Dry | Sunny), P(Damp | Sunny), P(Wet | Sunny)
# P(Dry | Rainy), P(Damp | Rainy), P(Wet | Rainy)
emission_probabilities = np.array([
    [0.8, 0.15, 0.05],  # Sunny -> Dry, Damp, Wet
    [0.1, 0.35, 0.55]   # Rainy -> Dry, Damp, Wet
])

# Initial state probabilities (initial weather condition)
# Probability of starting in "Sunny" or "Rainy"
initial_probabilities = np.array([0.6, 0.4])

# Create the Hidden Markov Model using the categorical distribution (discrete observations)
```



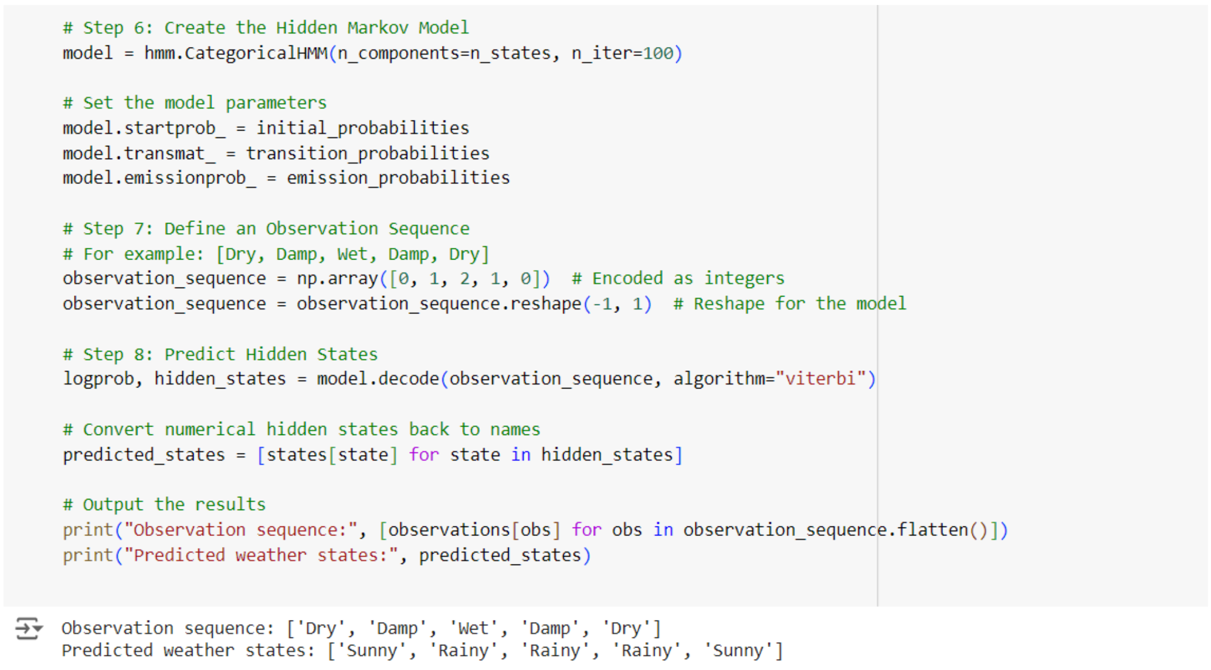
```python
# Initial state probabilities (initial weather condition)
# Probability of starting in "Sunny" or "Rainy"
initial_probabilities = np.array([0.6, 0.4])

# Create the Hidden Markov Model using the categorical distribution (discrete observations)
model = hmm.CategoricalHMM(n_components=n_states, n_iter=100)

# Set the model parameters
model.startprob_ = initial_probabilities
model.transmat_ = transition_probabilities
model.emissionprob_ = emission_probabilities

# Observation sequence (e.g., Dry, Damp, Wet, Damp, Dry)
# We'll encode the observations as integers: Dry = 0, Damp = 1, Wet = 2
observation_sequence = np.array([0, 1, 2, 1, 0])  # Dry -> Damp -> Wet -> Damp -> Dry
observation_sequence = observation_sequence.reshape(-1, 1)  # Reshape for the HMM model

# Predict the hidden states (Sunny or Rainy) based on the observation sequence
logprob, hidden_states = model.decode(observation_sequence, algorithm="viterbi")

# Convert the numerical hidden states back to labels
predicted_states = [states[state] for state in hidden_states]

# Output the prediction results
print("Observation sequence:", [observations[obs] for obs in observation_sequence.flatten()])
print("Predicted weather states:", predicted_states)
```

```
Observation sequence: ['Dry', 'Damp', 'Wet', 'Damp', 'Dry']
Predicted weather states: ['Sunny', 'Rainy', 'Rainy', 'Rainy', 'Sunny']
```

2. Here's a complete Python implementation that sets up the problem for a Hidden Markov Model (HMM) for weather forecasting, covering all specified components: hidden states, observable events, transition probabilities, emission probabilities, and initial probabilities.



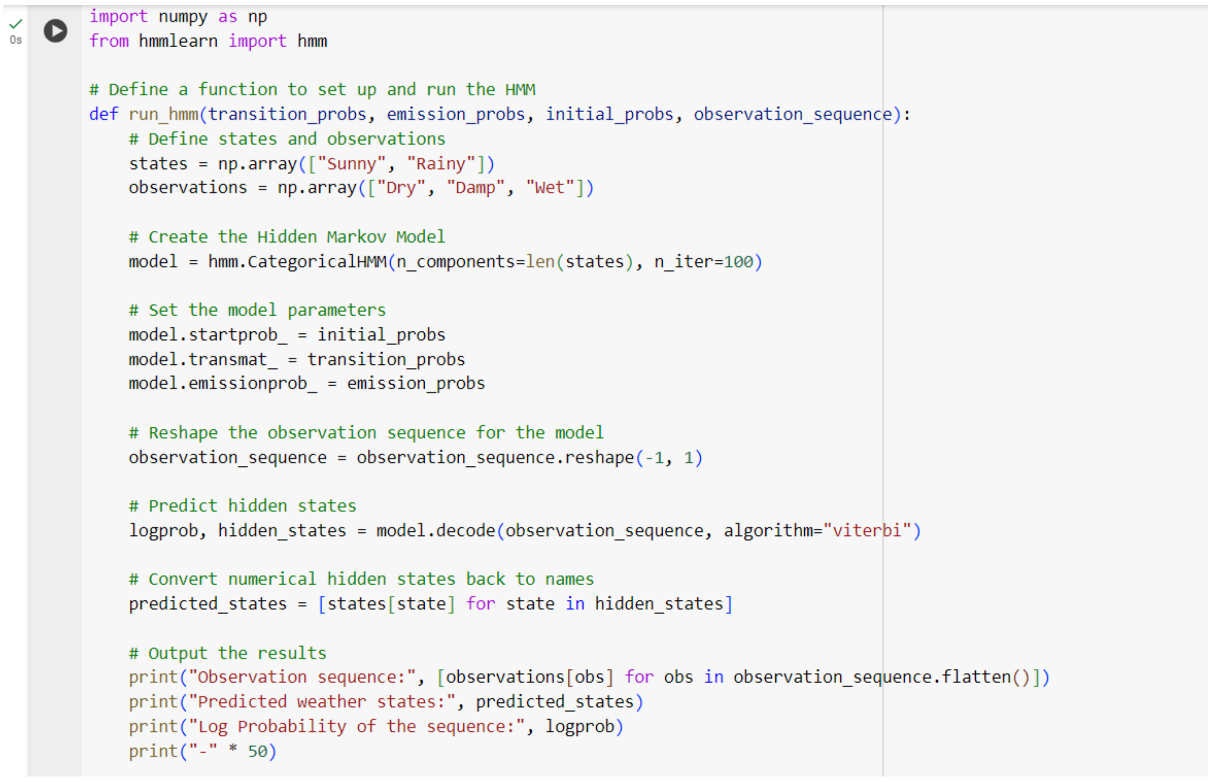
```python
import numpy as np
from hmmlearn import hmm

# Step 1: Define the Hidden States (S)
states = np.array(["Sunny", "Rainy"])
n_states = len(states)

# Step 2: Define the Observable Events (O)
observations = np.array(["Dry", "Damp", "Wet"])
n_observations = len(observations)

# Step 3: Define Transition Probabilities (A)
# Transition matrix:
# P(Sunny -> Sunny), P(Sunny -> Rainy)
# P(Rainy -> Sunny), P(Rainy -> Rainy)
transition_probabilities = np.array([
    [0.7, 0.3],  # From Sunny
    [0.4, 0.6]   # From Rainy
])

# Step 4: Define Emission Probabilities (B)
# Emission matrix:
# P(Dry | Sunny), P(Damp | Sunny), P(Wet | Sunny)
# P(Dry | Rainy), P(Damp | Rainy), P(Wet | Rainy)
emission_probabilities = np.array([
    [0.8, 0.15, 0.05],  # Emissions from Sunny
    [0.1, 0.35, 0.55]   # Emissions from Rainy
])

# Step 5: Define Initial Probabilities (π)
# Initial probabilities for starting weather
initial_probabilities = np.array([0.6, 0.4])  # Sunny: 60%, Rainy: 40%
```



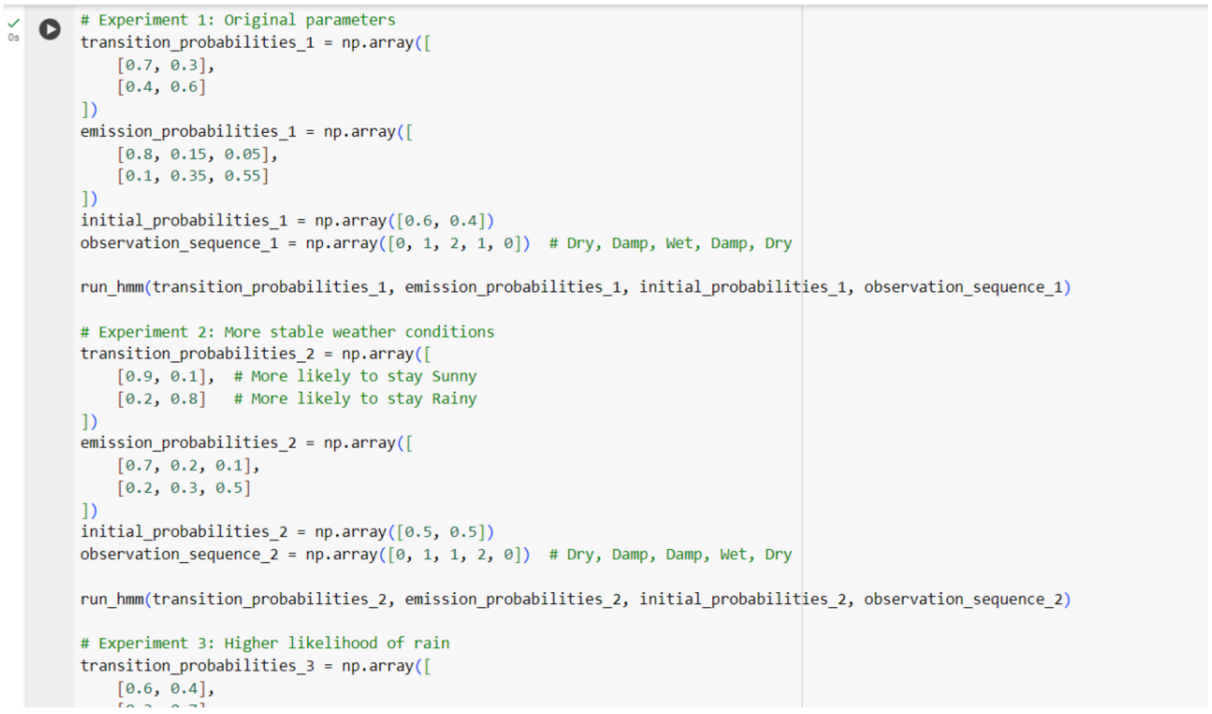
```python
# Step 6: Create the Hidden Markov Model
model = hmm.CategoricalHMM(n_components=n_states, n_iter=100)

# Set the model parameters
model.startprob_ = initial_probabilities
model.transmat_ = transition_probabilities
model.emissionprob_ = emission_probabilities

# Step 7: Define an Observation Sequence
# For example: [Dry, Damp, Wet, Damp, Dry]
observation_sequence = np.array([0, 1, 2, 1, 0])  # Encoded as integers
observation_sequence = observation_sequence.reshape(-1, 1)  # Reshape for the model

# Step 8: Predict Hidden States
logprob, hidden_states = model.decode(observation_sequence, algorithm="viterbi")

# Convert numerical hidden states back to names
predicted_states = [states[state] for state in hidden_states]

# Output the results
print("Observation sequence:", [observations[obs] for obs in observation_sequence.flatten()])
print("Predicted weather states:", predicted_states)
```

```
Observation sequence: ['Dry', 'Damp', 'Wet', 'Damp', 'Dry']
Predicted weather states: ['Sunny', 'Rainy', 'Rainy', 'Rainy', 'Sunny']
```

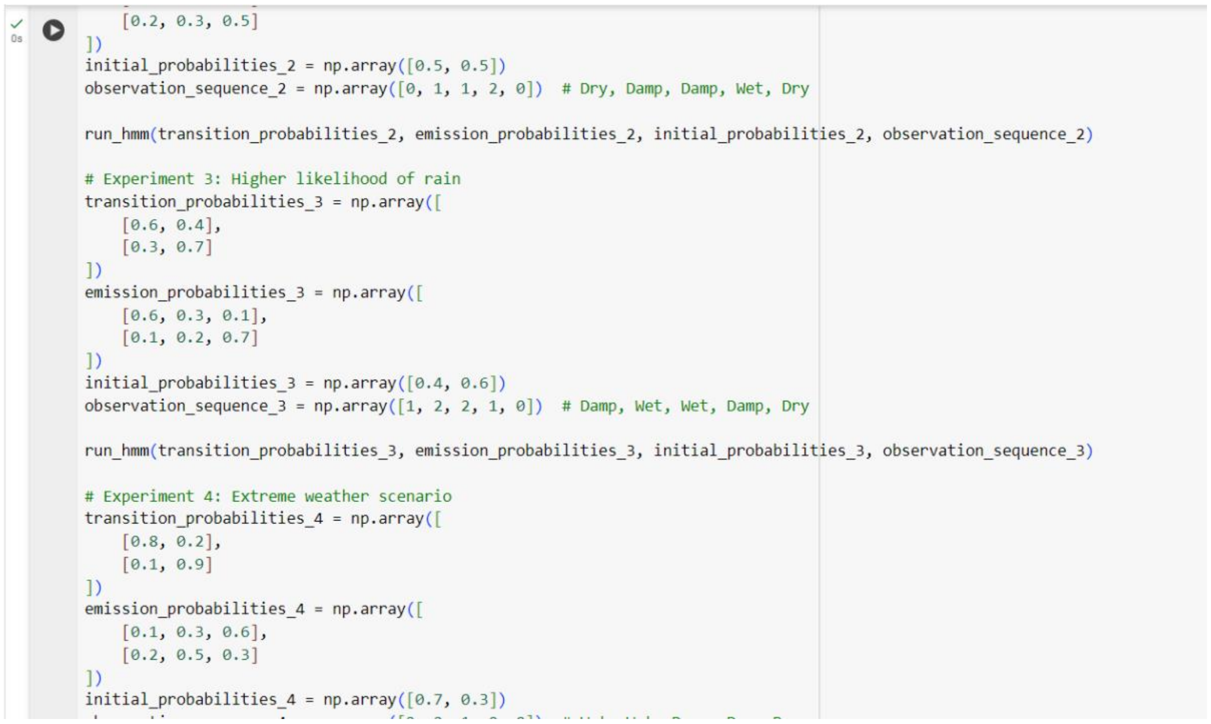
2. Here's a Python script that modifies the transition and emission probabilities, as well as the observation sequences, to see how the model's predictions change. We'll run multiple configurations to illustrate the impact of these changes.



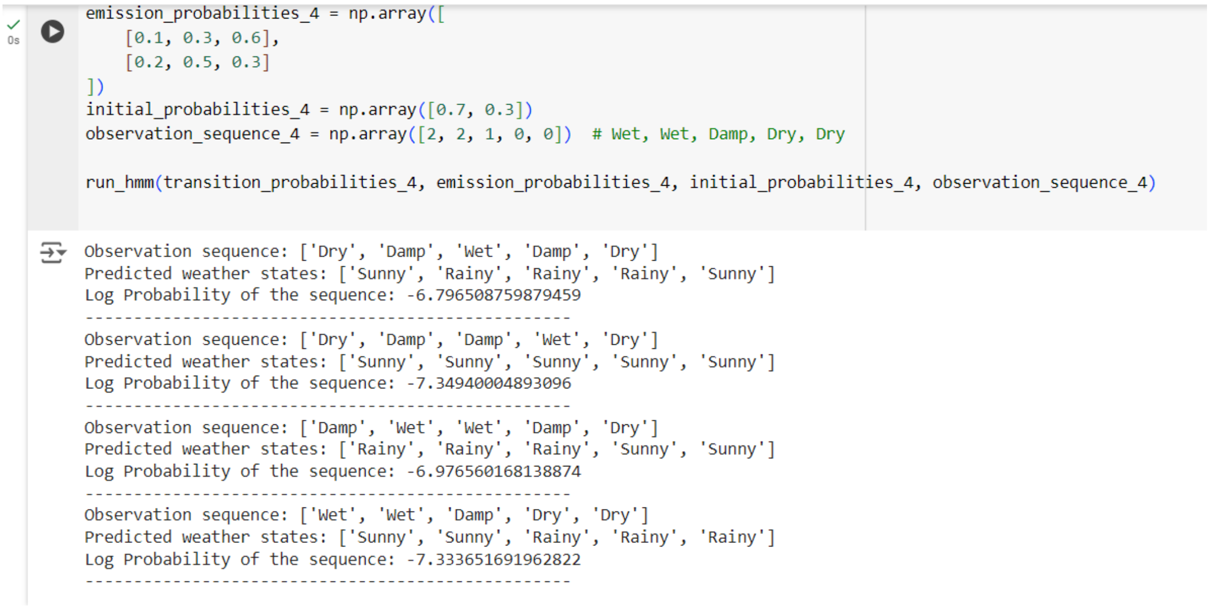
```python
import numpy as np
from hmmlearn import hmm

# Define a function to set up and run the HMM
def run_hmm(transition_probs, emission_probs, initial_probs, observation_sequence):
    # Define states and observations
    states = np.array(["Sunny", "Rainy"])
    observations = np.array(["Dry", "Damp", "Wet"])

    # Create the Hidden Markov Model
    model = hmm.CategoricalHMM(n_components=len(states), n_iter=100)

    # Set the model parameters
    model.startprob_ = initial_probs
    model.transmat_ = transition_probs
    model.emissionprob_ = emission_probs

    # Reshape the observation sequence for the model
    observation_sequence = observation_sequence.reshape(-1, 1)

    # Predict hidden states
    logprob, hidden_states = model.decode(observation_sequence, algorithm="viterbi")

    # Convert numerical hidden states back to names
    predicted_states = [states[state] for state in hidden_states]

    # Output the results
    print("Observation sequence:", [observations[obs] for obs in observation_sequence.flatten()])
    print("Predicted weather states:", predicted_states)
    print("Log Probability of the sequence:", logprob)
    print("-" * 50)
```



```python
# Experiment 1: Original parameters
transition_probabilities_1 = np.array([
    [0.7, 0.3],
    [0.4, 0.6]
])
emission_probabilities_1 = np.array([
    [0.8, 0.15, 0.05],
    [0.1, 0.35, 0.55]
])
initial_probabilities_1 = np.array([0.6, 0.4])
observation_sequence_1 = np.array([0, 1, 2, 1, 0])  # Dry, Damp, Wet, Damp, Dry

run_hmm(transition_probabilities_1, emission_probabilities_1, initial_probabilities_1, observation_sequence_1)

# Experiment 2: More stable weather conditions
transition_probabilities_2 = np.array([
    [0.9, 0.1],  # More likely to stay Sunny
    [0.2, 0.8]   # More likely to stay Rainy
])
emission_probabilities_2 = np.array([
    [0.7, 0.2, 0.1],
    [0.2, 0.3, 0.5]
])
initial_probabilities_2 = np.array([0.5, 0.5])
observation_sequence_2 = np.array([0, 1, 1, 2, 0])  # Dry, Damp, Damp, Wet, Dry

run_hmm(transition_probabilities_2, emission_probabilities_2, initial_probabilities_2, observation_sequence_2)

# Experiment 3: Higher likelihood of rain
transition_probabilities_3 = np.array([
    [0.6, 0.4],
```



```python
    [0.2, 0.3, 0.5]
])
initial_probabilities_2 = np.array([0.5, 0.5])
observation_sequence_2 = np.array([0, 1, 1, 2, 0])  # Dry, Damp, Damp, Wet, Dry

run_hmm(transition_probabilities_2, emission_probabilities_2, initial_probabilities_2, observation_sequence_2)

# Experiment 3: Higher likelihood of rain
transition_probabilities_3 = np.array([
    [0.6, 0.4],
    [0.3, 0.7]
])
emission_probabilities_3 = np.array([
    [0.6, 0.3, 0.1],
    [0.1, 0.2, 0.7]
])
initial_probabilities_3 = np.array([0.4, 0.6])
observation_sequence_3 = np.array([1, 2, 2, 1, 0])  # Damp, Wet, Wet, Damp, Dry

run_hmm(transition_probabilities_3, emission_probabilities_3, initial_probabilities_3, observation_sequence_3)

# Experiment 4: Extreme weather scenario
transition_probabilities_4 = np.array([
    [0.8, 0.2],
    [0.1, 0.9]
])
emission_probabilities_4 = np.array([
    [0.1, 0.3, 0.6],
    [0.2, 0.5, 0.3]
])
initial_probabilities_4 = np.array([0.7, 0.3])
```



```python
emission_probabilities_4 = np.array([
    [0.1, 0.3, 0.6],
    [0.2, 0.5, 0.3]
])
initial_probabilities_4 = np.array([0.7, 0.3])
observation_sequence_4 = np.array([2, 2, 1, 0, 0])  # Wet, Wet, Damp, Dry, Dry

run_hmm(transition_probabilities_4, emission_probabilities_4, initial_probabilities_4, observation_sequence_4)
```

```
Observation sequence: ['Dry', 'Damp', 'Wet', 'Damp', 'Dry']
Predicted weather states: ['Sunny', 'Rainy', 'Rainy', 'Rainy', 'Sunny']
Log Probability of the sequence: -6.796508759879459
--------------------------------------------------
Observation sequence: ['Dry', 'Damp', 'Damp', 'Wet', 'Dry']
Predicted weather states: ['Sunny', 'Sunny', 'Sunny', 'Sunny', 'Sunny']
Log Probability of the sequence: -7.34940004893096
--------------------------------------------------
Observation sequence: ['Damp', 'Wet', 'Wet', 'Damp', 'Dry']
Predicted weather states: ['Rainy', 'Rainy', 'Rainy', 'Sunny', 'Sunny']
Log Probability of the sequence: -6.976560168138874
--------------------------------------------------
Observation sequence: ['Wet', 'Wet', 'Damp', 'Dry', 'Dry']
Predicted weather states: ['Sunny', 'Sunny', 'Rainy', 'Rainy', 'Rainy']
Log Probability of the sequence: -7.333651691962822
--------------------------------------------------
```