

A circular petri dish containing a dense culture of small, translucent, rod-shaped bacteria, likely E. coli, growing in a liquid medium. The bacteria are arranged in a somewhat organized pattern, possibly forming a biofilm or a structured colony.

SEQUENTIAL DATA AND TIME SERIES ANALYSIS

TIM AJAR KECERDASAN ARTIFICIAL

AGENDA

- UNDERSTANDING SEQUENTIAL DATA & TIME SERIES
- HIDDEN MARKOV MODEL
- PANDAS LIBRARY



UNDERSTANDING SEQUENTIAL DATA



- Sequential data refers to data where the ordering is important. Whenever the points in the dataset are dependent on the other points in the dataset the data is said to be Sequential data.
- Example:
 - Genomic Sequence Data
 - Human Languages
 - Computer Languages
 - Stock market prices
 - Applications logs
 - IoT activity

TIME SERIES DATA

- A common example of this is a Timeseries such as a stock price or a sensor data where each point represents an observation at a certain point in time
- Time series data has a lot of important characteristics that need to be modeled in order to be effectively analyzed.
- Measurements for certain parameters in time series data are taken at regular time intervals.
- These measurements are arranged and stored on a timeline, and the order of their appearance is critical. This order is used to extract patterns from the data.
- Time series data analysis is used extensively in :
 - finance, sensor data analysis, speech recognition, economics, weather forecasting, manufacturing, and many more areas

HANDLING TIME SERIES DATA WITH PANDAS

- Pandas comes from the term panel data, which is an econometrics term for datasets that include observations over multiple time periods

```
$ pip3 install pandas
```

```
$ pip3 install hmmlearn
```

```
$ pip3 install pystruct
```

```
$ pip3 install cvxopt
```

```
$ pip3 install timeseries
```

MORE FUNCTIONS CAN BE PERFORMED WITH PANDAS

- DataFrame manipulation with integrated indexing
- Methods to read data from a variety of different file formats and write data into in-memory data structures
- Data sorting
- Data filtering
- Missing value imputation
- Reshaping and pivoting datasets
- Label-based slicing, indexing, and creation of subsets
- Efficient column insertion and deletion
- Group by operations on datasets
- Merging and joining of datasets

EX. CONVERT A SEQUENCE OF NUMBERS INTO TIME SERIES DATA AND VISUALIZE IT

- Load data from an external file
- convert it into time series format
- plot it and visualize it

In [5]:

```
input_file = 'data_2D.txt'

# Specify the columns that need to be converted
# into time-series data
indices = [2, 3]

# Iterate through the columns and plot the data
for index in indices:
    # Convert the column to timeseries format
    timeseries = read_data(input_file, index)

    # Plot the data
    plt.figure()
    timeseries.plot()
    plt.title('Dimension ' + str(index - 1))

plt.show()
```

In [4]:

```
def read_data(input_file, index):
    # Read the data from the input file
    input_data = np.loadtxt(input_file, delimiter=',')

    # Lambda function to convert strings to Pandas date format
    to_date = lambda x, y: str(int(x)) + '-' + str(int(y))

    # Extract the start date
    start = to_date(input_data[0, 0], input_data[0, 1])

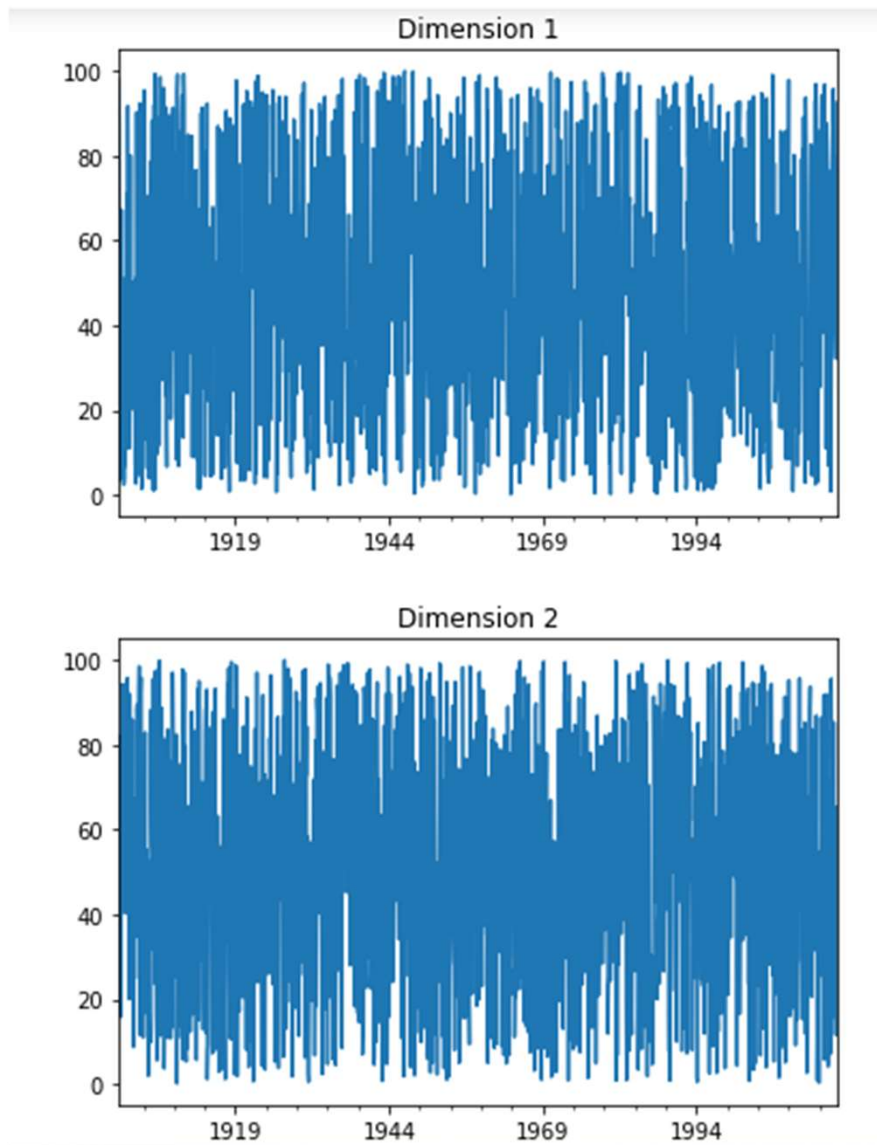
    # Extract the end date
    if input_data[-1, 1] == 12:
        year = input_data[-1, 0] + 1
        month = 1
    else:
        year = input_data[-1, 0]
        month = input_data[-1, 1] + 1

    end = to_date(year, month)

    # Create a date list with a monthly frequency
    date_indices = pd.date_range(start, end, freq='M')

    # Add timestamps to the input data to create time-series data
    output = pd.Series(input_data[:, index], index=date_indices)

    return output
```



SLICING TIME SERIES DATA

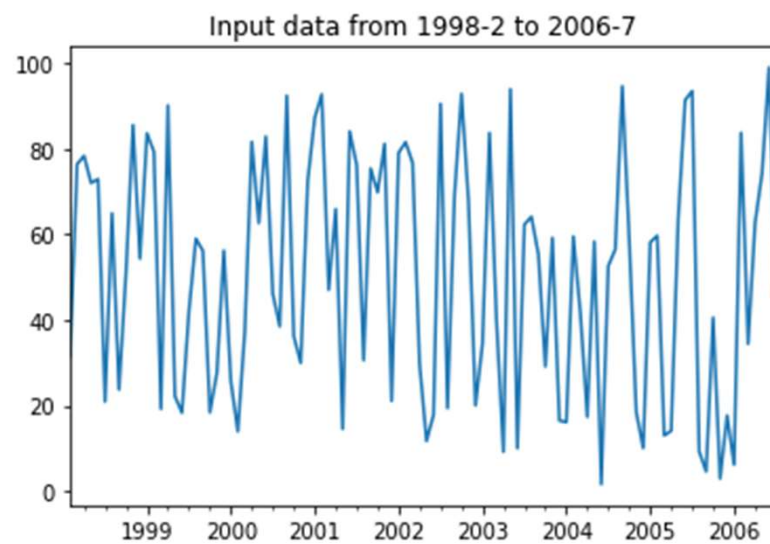
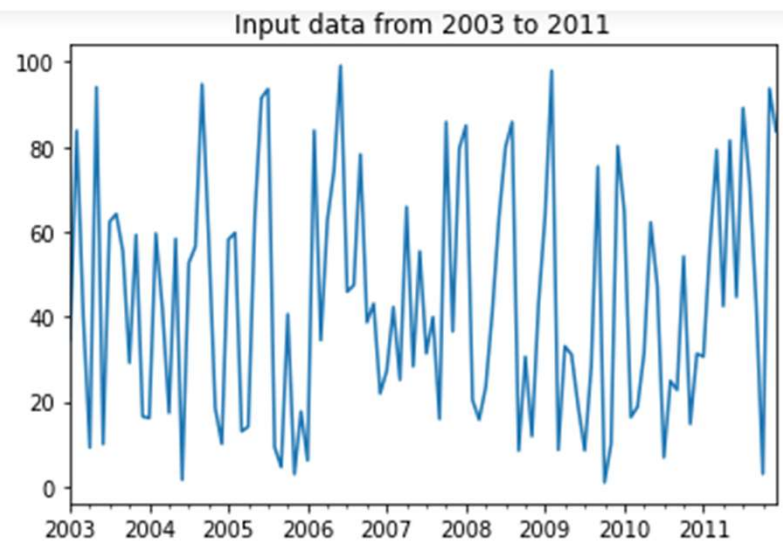
- dividing the data into various sub-intervals
- extracting relevant information
- Instead of using indices, we will use timestamps to slice our data

```
In [6]: # Load input data
index = 2
data = read_data('data_2D.txt', index)

# Plot data with year-level granularity
start = '2003'
end = '2011'
plt.figure()
data[start:end].plot()
plt.title('Input data from ' + start + ' to ' + end)

# Plot data with month-level granularity
start = '1998-2'
end = '2006-7'
plt.figure()
data[start:end].plot()
plt.title('Input data from ' + start + ' to ' + end)

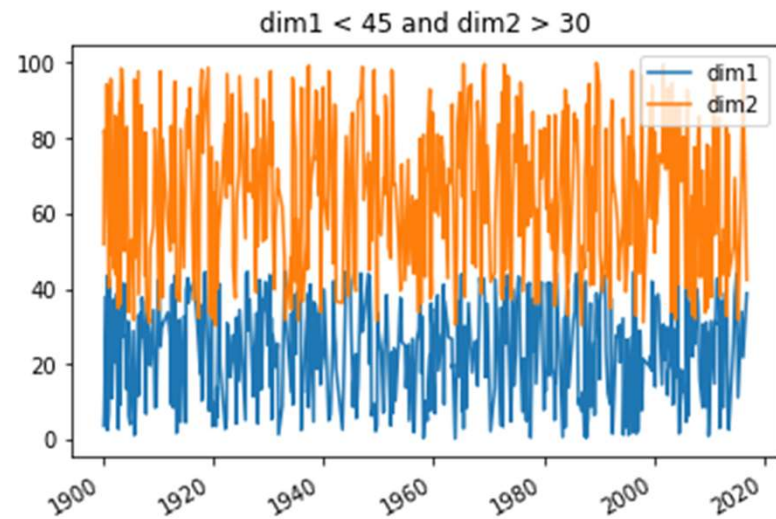
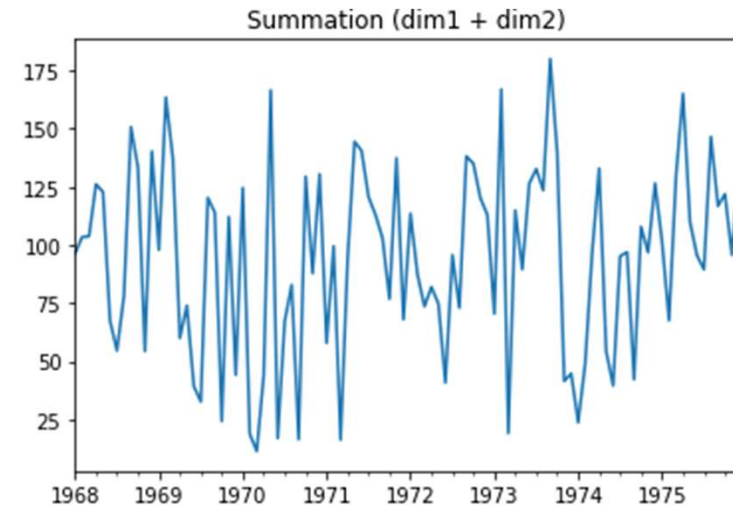
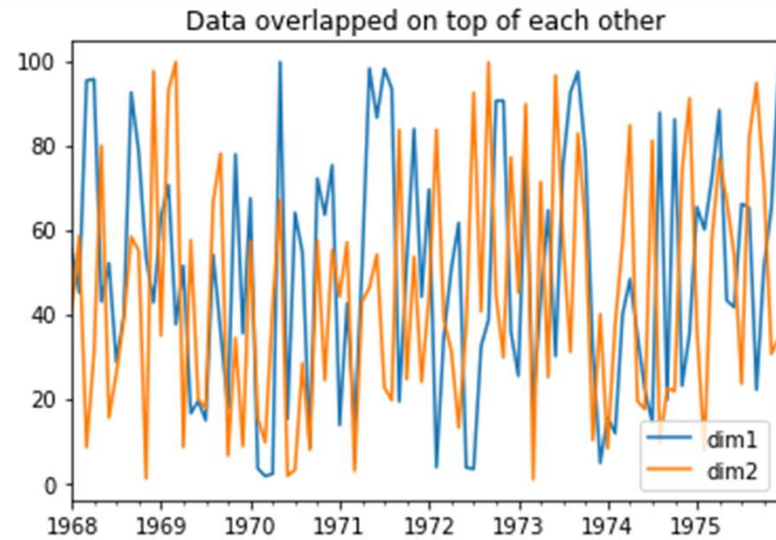
plt.show()
```



OPERATING ON TIME SERIES DATA

- The Pandas library can operate on time series data efficiently and perform various operations like filtering and addition.
- Conditions can be set, and Pandas will filter the dataset and return the right subset based on the condition.
- Time series data can be loaded and filtered as well.

```
In [7]:  
  
# Input filename  
input_file = 'data_2D.txt'  
  
# Load data  
x1 = read_data(input_file, 2)  
x2 = read_data(input_file, 3)  
  
# Create pandas dataframe for slicing  
data = pd.DataFrame({'dim1': x1, 'dim2': x2})  
  
# Plot data  
start = '1968'  
end = '1975'  
data[start:end].plot()  
plt.title('Data overlapped on top of each other')  
  
# Filtering using conditions  
# - 'dim1' is smaller than a certain threshold  
# - 'dim2' is greater than a certain threshold  
data[(data['dim1'] < 45) & (data['dim2'] > 30)].plot()  
plt.title('dim1 < 45 and dim2 > 30')  
  
# Adding two dataframes  
plt.figure()  
diff = data[start:end]['dim1'] + data[start:end]['dim2']  
diff.plot()  
plt.title('Summation (dim1 + dim2)')  
  
plt.show()
```



- it is important before selecting and training models to understand the datasets that are being analyzed.
- Pandas is a useful tool to accomplish this

STOCK MARKET ANALYSIS

- We will analyze stock market data in this section using HMMs. This is an example where the data is already organized and timestamped.
- We will use the dataset available in the matplotlib package. The dataset contains the stock values of various companies over the years.
- HMMs are generative models that can analyze such time series data and extract the underlying structure. We will use this model to analyze stock price variations and generate the outputs

```
In [7]: import datetime
import warnings

import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
import finplot as fplt
from hmmlearn.hmm import GaussianHMM
```

```
In [*]: # Load historical stock quotes from matplotlib package
start = datetime.date(2021, 11, 1)
end = datetime.date(2023, 11, 1)
stock_quotes = yf.download('PTBA.JK', start=start, end=end)

# Extract the closing quotes everyday
closing_quotes = np.array([quote[2] for quote in stock_quotes])

# Extract the volume of shares traded everyday
volumes = np.array([quote[5] for quote in stock_quotes])[1:]

# Take the percentage difference of closing stock prices
diff_percentages = 100.0 * np.diff(closing_quotes) / closing_quotes[:-1]

# Take the list of dates starting from the second value
dates = np.array([quote[0] for quote in stock_quotes], dtype=np.int)[1:]

# Stack the differences and volume values column-wise for training
training_data = np.column_stack([diff_percentages, volumes])
```

```
In [ ]: # Create and train Gaussian HMM
hmm = GaussianHMM(n_components=7, covariance_type='diag', n_iter=1000)
with warnings.catch_warnings():
    warnings.simplefilter('ignore')
    hmm.fit(training_data)

# Generate data using the HMM model
num_samples = 300
samples, _ = hmm.sample(num_samples)

# Plot the difference percentages
plt.figure()
plt.title('Difference percentages')
plt.plot(np.arange(num_samples), samples[:, 0], c='black')

# Plot the volume of shares traded
plt.figure()
plt.title('Volume of shares')
plt.plot(np.arange(num_samples), samples[:, 1], c='black')
plt.ylim(ymin=0)

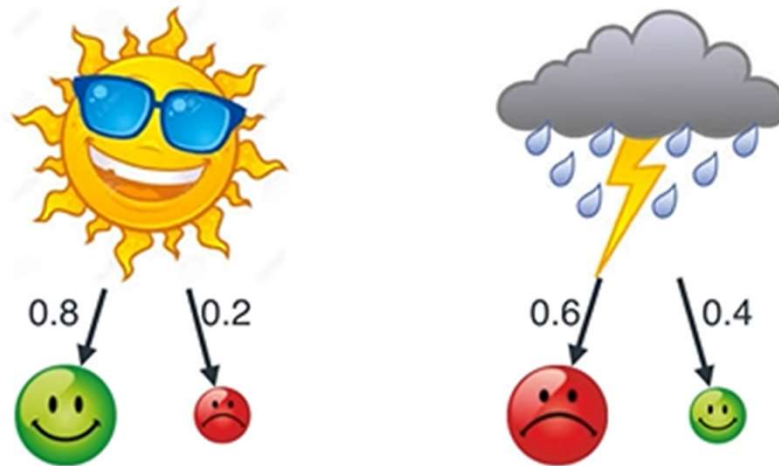
plt.show()
```


HIDDEN MARKOV MODEL













- Markov models are named after Andrey Markov, who first developed them in the early 1900s. Markov models are a type of probabilistic model that is used to predict the future state of a system, based on its current state. In other words, Markov models are used to predict the future state based on the current hidden or observed states.
- let's look at an example. Say you have a bag of marbles that contains four marbles: two red marbles and two blue marbles. You randomly select a marble from the bag, note its color, and then put it back in the bag. After repeating this process several times, you begin to notice a pattern: The probability of selecting a red marble is always two out of four, or 50%
- The concept of a Markov model: the future state of a system is determined by its current state and past history. In the case of the bag of marbles, the current state is determined by the number of each color of marble in the bag. The past history is represented by the contents of the bag, which determine the probabilities of selecting each color of marble.

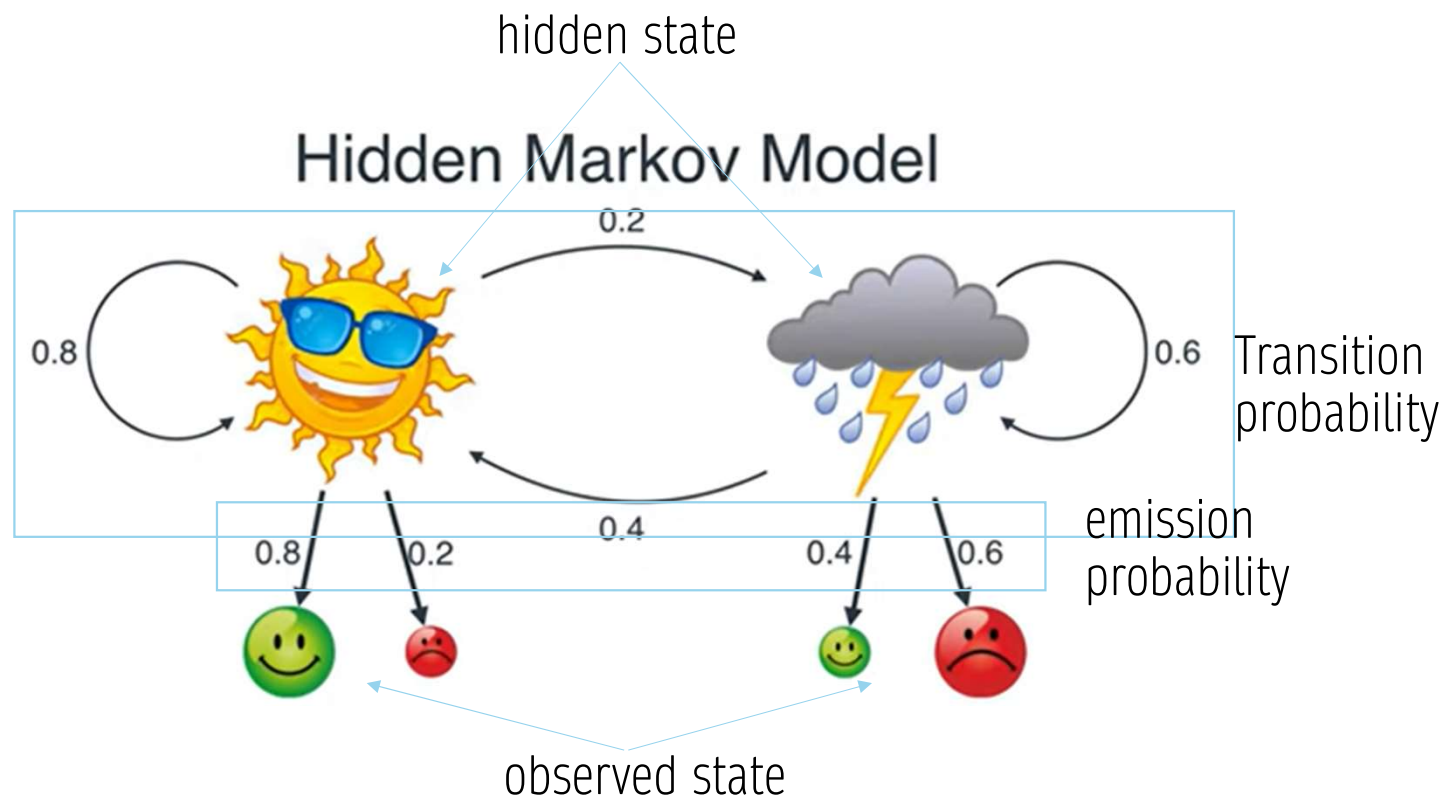
HIDDEN MARKOV MODEL (1)

Weather

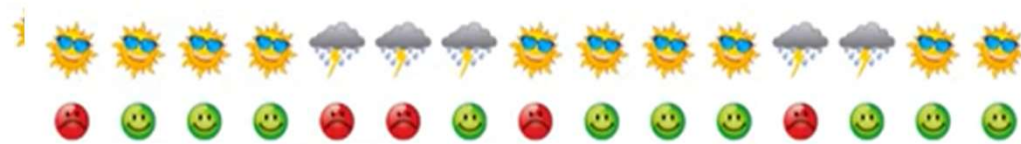












Weather

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	
 H	 G	 H	 G	 H	 G	
 S	 R	 S	 R	 S	 R	?

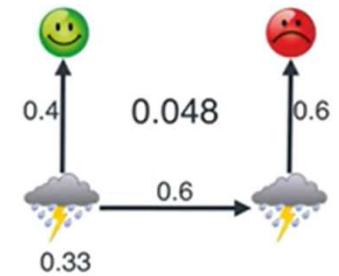
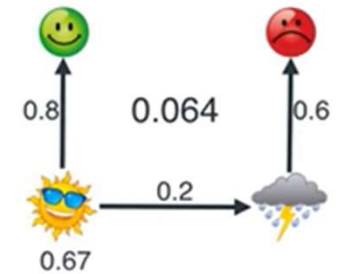
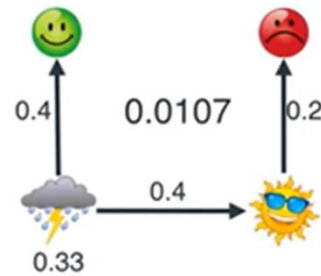
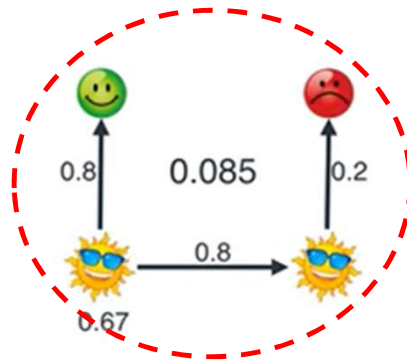
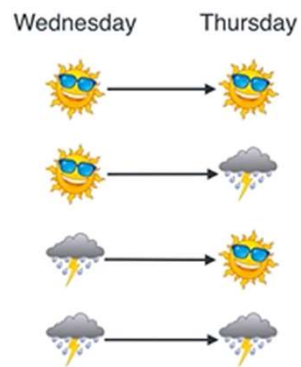
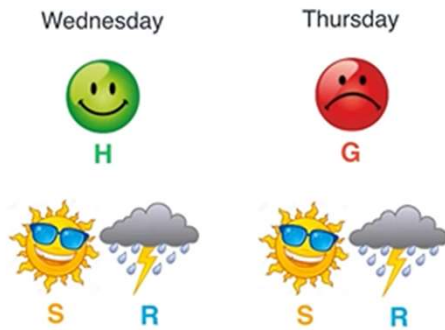


How did we find the probabilities?

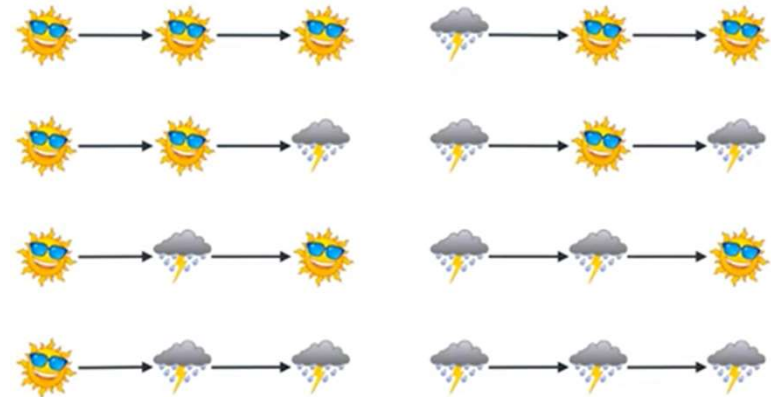
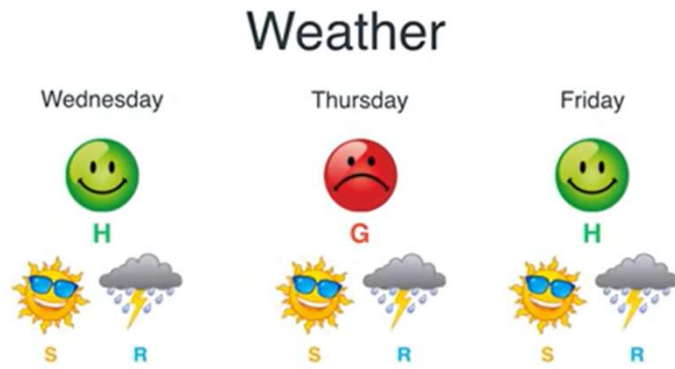


	→		8	0.8		→		2	0.4
	→		2	0.2		→		3	0.6
				10	$\frac{2}{3}$				
				5	$\frac{1}{3}$				

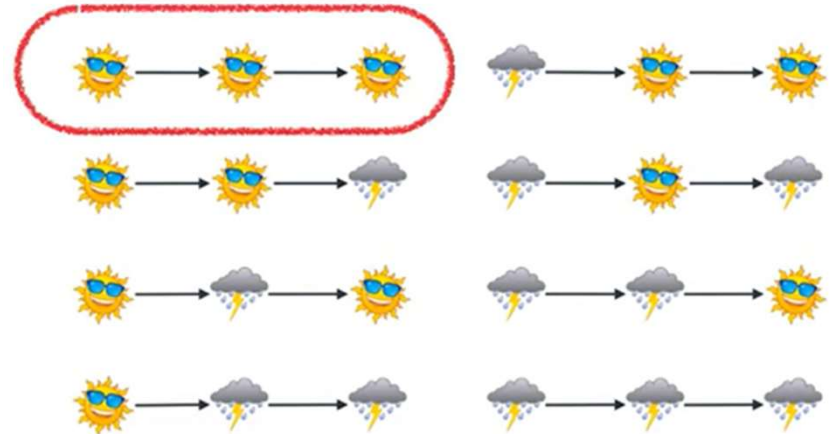
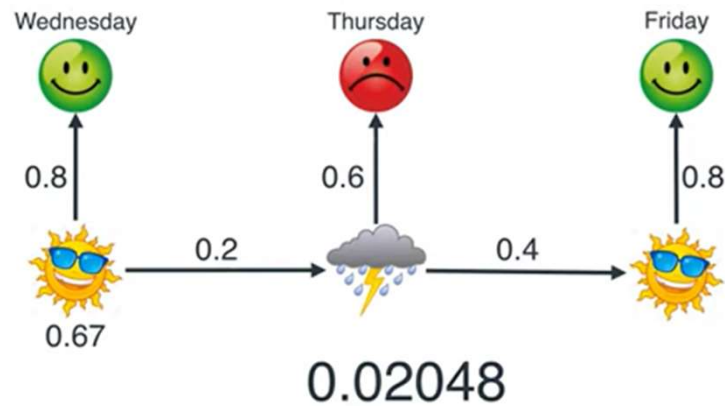
IF HAPPY-GRUMPY, WEATHER?



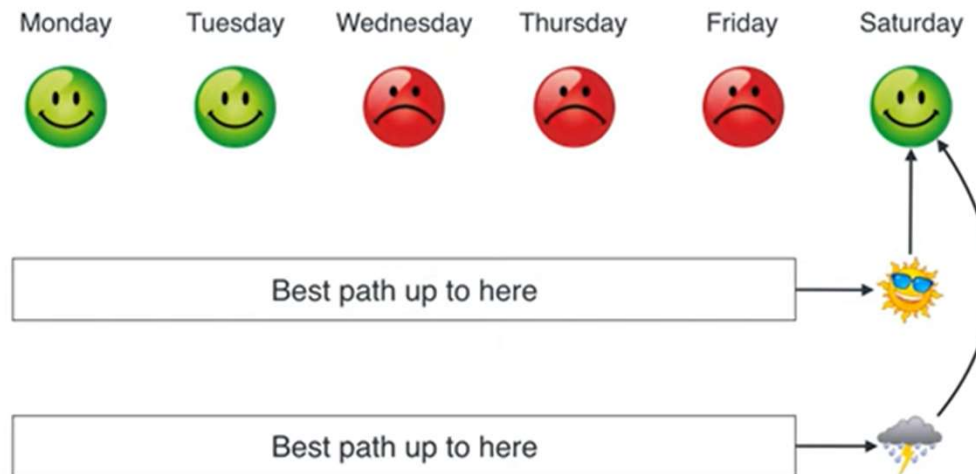
IF HAPPY-GRUMPY-HAPPY, WEATHER?

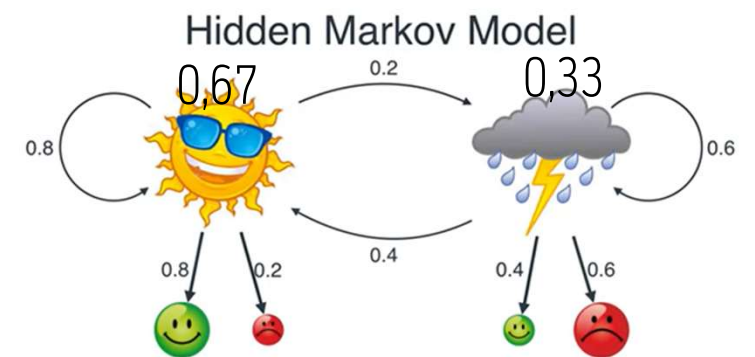
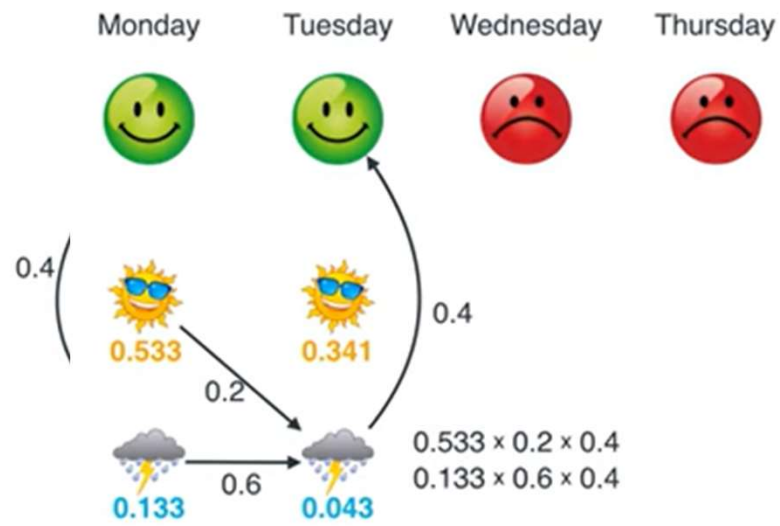


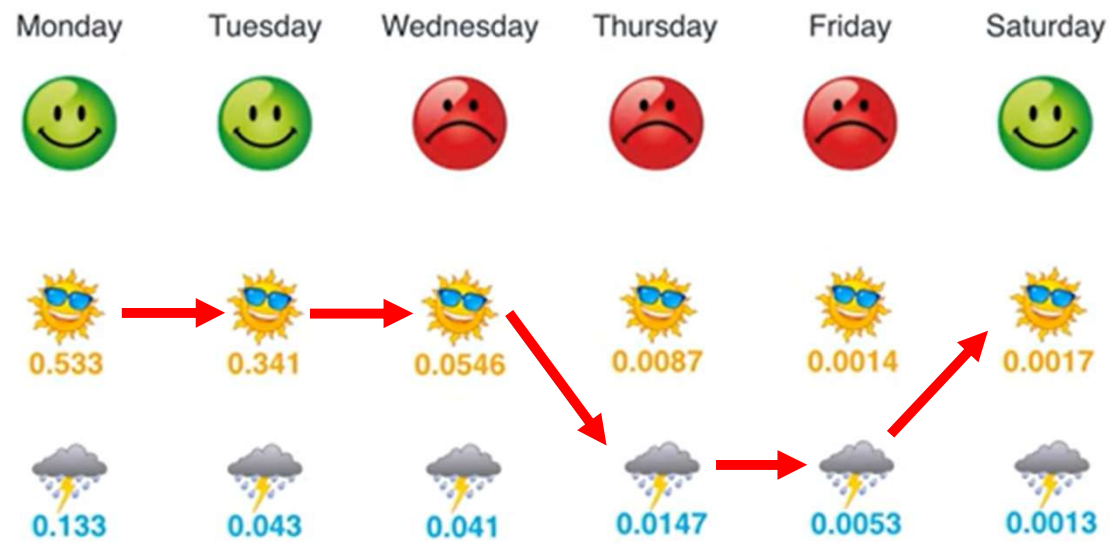
IF HAPPY-GRUMPY-HAPPY, WEATHER? (1)



IF HAPPY-HAPPY-GRUMPY-GRUMPY-GRUMPY-HAPPY, WEATHER?









THANK YOU

