Q1. Implement the given three questions of hidden markov model, on the basis of given information in the table and figure.

Today's weather	Tomorrow's weather		
	0.8	0.05	0.15
	0.2	0.6	0.2
	0.2	0.3	0.5

Table 1: Probabilities $p(q_{n+1}|q_n)$ of tomorrow's weather based on today's weather

1. Given that today the weather is , what's the probability that tomorrow is and the day after is ?

2. Assume, the weather yesterday was $q_1 =$, and today it is $q_2 =$, what is the probability that tomorrow it will be $q_3 =$?

3. Given that the weather today is $q_1 =$, what is the probability that it will be two days from now: $q_3 =$. (Hint: There are several ways to get from today to two days from now. You have to sum over these paths.)

In [ ]:

```
import numpy as np
import pandas as pd
hidden_states = ['Sunny', 'Rainy', 'Foggy']

a_df = pd.DataFrame(columns=hidden_states, index=hidden_states)
a_df.loc[hidden_states[0]] = [0.8, 0.05, 0.15]
a_df.loc[hidden_states[1]] = [0.2, 0.6, 0.2]
a_df.loc[hidden_states[2]] = [0.2, 0.3, 0.5]
print("\n HMM matrix:\n", a_df)
a_df_val = a_df.values
a_df_index=a_df.index;
index=[0,1,2]
index_dict={}

for key in a_df_index:
    for value in index:
        index_dict[key]=value
        index.remove(value)
        break

# print(a_df_val)
# print(a_df_ind)
# print(index_dict)

seq = input('Enter the sequence: ').split()
prob_till_now = 1
for i in range(1,len(seq)):
    prob_till_now *= a_df_val[index_dict[seq[i-1]]][index_dict[seq[i]]]
    if i%10==1:
        print("prob of having ",seq[i]," on ", i , "st day is ",prob_till_now)
    elif i%10==2:
        print("prob of having ",seq[i]," on ", i , "nd day is ",prob_till_now)
```

```
    elif i%10==3:
        print("prob of having ",seq[i]," on ", i , "rd day is ",prob_till_now)
    else:
        print("prob of having ",seq[i]," on ", i , "th day is ",prob_till_now)
print("final prob :",prob_till_now)

print("Yesterday was cloudy, today is foggy, the probability that tomorrow is sunny = {0}
".format(a_df_val[index_dict['Foggy']][index_dict['Sunny']]))

p1 = a_df_val[index_dict['Rainy']][index_dict['Sunny']]*a_df_val[index_dict['Sunny']][in
dex_dict['Foggy']]
p2 = a_df_val[index_dict['Rainy']][index_dict['Rainy']]*a_df_val[index_dict['Rainy']][in
dex_dict['Foggy']]
p3 = a_df_val[index_dict['Rainy']][index_dict['Foggy']]*a_df_val[index_dict['Foggy']][in
dex_dict['Foggy']]

print("Today = Rainy, Probabilty that day after tomorrow it rains = {0}".format(p1+p2+p3
))
```

```
 HMM matrix:
       Sunny Rainy Foggy
Sunny   0.8  0.05  0.15
Rainy   0.2   0.6   0.2
Foggy   0.2   0.3   0.5
Enter the sequence: Sunny Sunny Rainy
prob of having  Sunny  on  1 st day is  0.8
prob of having  Rainy  on  2 nd day is  0.04000000000000001
final prob : 0.04000000000000001
Yesterday was cloudy, today is foggy, the probability that tomorrow is sunny = 0.2
Today = Rainy, Probabilty that day after tomorrow it rains = 0.25
```

Q3. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.

In [ ]:

```
pip install pgmpy
```

```
Collecting pgmpy
  Downloading https://files.pythonhosted.org/packages/06/19/d508949e8ac7b32e639f15e854a5f
5ed710a4118e4f6692bddaccc390d88/pgmpy-0.1.13-py3-none-any.whl (324kB)
     |[redacted]| 327kB 6.8MB/s
Requirement already satisfied: pyparsing in /usr/local/lib/python3.7/dist-packages (from
pgmpy) (2.4.7)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (fr
om pgmpy) (0.22.2.post1)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from pgmp
y) (1.4.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from pgm
py) (1.1.5)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages (fro
m pgmpy) (0.10.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from pgm
py) (1.0.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.7/dist-packages (from p
gmpy) (2.5)
Requirement already satisfied: torch in /usr/local/lib/python3.7/dist-packages (from pgmp
y) (1.7.1+cu101)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from pgmp
y) (1.19.5)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from pgmpy
) (4.41.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-pa
ckages (from pandas->pgmpy) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (fr
om pandas->pgmpy) (2018.9)
Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.7/dist-packages (fr
```

```
om statsmodels->pgmpy) (0.5.1)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.7/dist-packages
(from networkx->pgmpy) (4.4.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-package
s (from torch->pgmpy) (3.7.4.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from p
ython-dateutil>=2.7.3->pandas->pgmpy) (1.15.0)
Installing collected packages: pgmpy
Successfully installed pgmpy-0.1.13
```

In [ ]:

```
import numpy as np
import csv
import pandas as pd
from pgmpy.models import BayesianModel
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination
heartDisease = pd.read_csv('heart.csv')
heartDisease = heartDisease.replace('?',np.nan)

model = BayesianModel([('age','trestbps'),
                       ('age','fbs'),
                       ('sex','trestbps'),
                       ('exang','trestbps'),
                       ('trestbps','target'),
                       ('fbs','target'),
                       ('target','restecg'),
                       ('target','thalach'),
                       ('target','chol')])

print('Learning CPD using Maximum Likelihood Estimators....')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

print('\n Inferencing with Bayesian Network....')
HeartDisease_infer = VariableElimination(model)
print("\nModel is ready...\n")
model.local_independencies(['age','sex','trestbps','chol','fbs','restecg','thalach','exan
g','target'])
```

```
Learning CPD using Maximum Likelihood Estimators....

 Inferencing with Bayesian Network....

Model is ready...
```

Out[ ]:

```
(age _|_ exang, sex)
(sex _|_ exang, fbs, age)
(trestbps _|_ fbs | exang, age, sex)
(chol _|_ exang, age, trestbps, fbs, sex, restecg, thalach | target)
(fbs _|_ exang, sex, trestbps | age)
(restecg _|_ exang, age, chol, trestbps, fbs, sex, thalach | target)
(thalach _|_ exang, age, chol, trestbps, fbs, sex, restecg | target)
(exang _|_ fbs, age, sex)
(target _|_ exang, age, sex | fbs, trestbps)
```

In [ ]:

```
print('\n 1. Probability of Heart Disease given Age=28')
q=HeartDisease_infer.query(variables=['target'],evidence={'age':28})
print(q)

print('\n 2. Probability of Heart Disease given cholesterol = 350 and trestbps = 120')
q=HeartDisease_infer.query(variables=['target'],evidence={'chol':300,'trestbps':120})
print(q)

print('\n 3. Probability of Heart Disease given age = 35 and sex = male and restecg = 1')
q=HeartDisease_infer.query(variables=['target'],evidence={'age':35,'sex':0,'restecg':1})
print(q)
```

```
print('\n 3. Probabilities of Heart Disease and restecg, given age = 35 and sex = female'
)
q=HeartDisease_infer.query(variables=['target','restecg'],evidence={'age':35,'sex':1})
print(q)
```

```
/usr/local/lib/python3.7/dist-packages/pgmpy/factors/discrete/DiscreteFactor.py:519: User
Warning: Found unknown state name. Trying to switch to using all state names as state num
bers
  "Found unknown state name. Trying to switch to using all state names as state numbers"
Finding Elimination Order: : 100%|██████████| 7/7 [00:00<00:00, 1518.81it/s]
Eliminating: thalach: 100%|██████████| 7/7 [00:00<00:00, 214.44it/s]
Finding Elimination Order: : 100%|██████████| 6/6 [00:00<00:00, 1438.29it/s]
Eliminating: age:   0%|          | 0/6 [00:00<?, ?it/s]
```

```
 1. Probability of Heart Disease given Age=28
+-----------+---------------+
| target    |   phi(target) |
+===========+===============+
| target(0) |        0.4058 |
+-----------+---------------+
| target(1) |        0.5942 |
+-----------+---------------+

 2. Probability of Heart Disease given cholesterol = 350 and trestbps = 120
```

```
Eliminating: thalach: 100%|██████████| 6/6 [00:00<00:00, 205.80it/s]
Finding Elimination Order: : 100%|██████████| 5/5 [00:00<00:00, 1620.55it/s]
Eliminating: thalach: 100%|██████████| 5/5 [00:00<00:00, 225.12it/s]
Finding Elimination Order: : 100%|██████████| 5/5 [00:00<00:00, 1218.85it/s]
Eliminating: thalach: 100%|██████████| 5/5 [00:00<00:00, 252.39it/s]
```

```
+-----------+---------------+
| target    |   phi(target) |
+===========+===============+
| target(0) |        1.0000 |
+-----------+---------------+
| target(1) |        0.0000 |
+-----------+---------------+

 3. Probability of Heart Disease given age = 35 and sex = male and restecg = 1
+-----------+---------------+
| target    |   phi(target) |
+===========+===============+
| target(0) |        0.1580 |
+-----------+---------------+
| target(1) |        0.8420 |
+-----------+---------------+

 3. Probabilities of Heart Disease and restecg, given age = 35 and sex = female
+-----------+------------+-----------------------+
| target    | restecg    |   phi(target,restecg) |
+===========+============+=======================+
| target(0) | restecg(0) |                0.1805 |
+-----------+------------+-----------------------+
| target(0) | restecg(1) |                0.1280 |
+-----------+------------+-----------------------+
| target(0) | restecg(2) |                0.0069 |
+-----------+------------+-----------------------+
| target(1) | restecg(0) |                0.2822 |
+-----------+------------+-----------------------+
| target(1) | restecg(1) |                0.3983 |
+-----------+------------+-----------------------+
| target(1) | restecg(2) |                0.0041 |
+-----------+------------+-----------------------+
```

Q2. Write a program to construct a Bayesian network for Tumor Type Classification for the attached leukemia samples dataset, and You need to classify leukemia samples into two classes based on gene expression patterns using Bayesian networks.

In [ ]:

```
import numpy
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

In [ ]:

```
train_df=pd.read_csv('/content/data_set_ALL_AML_train.csv')
test_df=pd.read_csv('/content/data_set_ALL_AML_independent.csv')
y=pd.read_csv('/content/actual.csv')
```

In [ ]:

```
print(train_df.shape)
print(test_df.shape)
print(y.shape)
```

```
(7129, 78)
(7129, 70)
(72, 2)
```

In [ ]:

```
test_df.head(3)
```

Out[ ]:

	Gene Description	Gene Accession Number	39	call	40	call.1	42	call.2	47	call.3	48	call.4	49	call.5	41	call.6	43	call.7	44	c
0	AFFX-BioB-5_at (endogenous control)	AFFX-BioB-5_at	-342	A	-87	A	22	A	-243	A	-130	A	-256	A	-62	A	86	A	-146	
1	AFFX-BioB-M_at (endogenous control)	AFFX-BioB-M_at	-200	A	-248	A	-153	A	-218	A	-177	A	-249	A	-23	A	-36	A	-74	
2	AFFX-BioB-3_at (endogenous control)	AFFX-BioB-3_at	41	A	262	A	17	A	-163	A	-28	A	-410	A	-7	A	-141	A	170	

In [ ]:

```
train_df.head(3)
```

Out[ ]:

	Gene

	Gene Description	Gene Accession Number	1	call	2	call.1	3	call.2	4	call.3	5	call.4	6	call.5	7	call.6	8	call.7	9
0	AFFX-BioB-5_at (endogenous control)	AFFX-BioB-5_at	-214	A	-139	A	-76	A	-135	A	-106	A	-138	A	-72	A	-413	A	5
1	AFFX-BioB-M_at (endogenous control)	AFFX-BioB-M_at	-153	A	-73	A	-49	A	-114	A	-125	A	-85	A	-144	A	-260	A	-127
2	AFFX-BioB-3_at (endogenous control)	AFFX-BioB-3_at	-58	A	-1	A	-307	A	265	A	-76	A	215	A	238	A	7	A	106

In [ ]:

```
y.head(3)
```

Out[ ]:

	patient	cancer
0	1	ALL
1	2	ALL
2	3	ALL

In [ ]:

```
y['cancer'].value_counts()
```

Out[ ]:

```
ALL    47
AML    25
Name: cancer, dtype: int64
```

remove cols with 'call'

In [ ]:

```
cols = [col for col in test_df.columns if 'call' in col]
test_df = test_df.drop(cols, 1)
cols = [col for col in train_df.columns if 'call' in col]
train_df = train_df.drop(cols, 1)
```

In [ ]:

```
train_df.head(3)
```

Out[ ]:

	Gene Description	Gene Accession Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	AFFX-BioB-5_at (endogenous control)	AFFX-BioB-5_at	-214	-139	-76	-135	-106	-138	-72	-413	5	-88	-165	-67	-92	-113	-107	-117	-476	-81	-44	17
1	AFFX-BioB-M_at (endogenous control)	AFFX-BioB-M_at	-153	-73	-49	-114	-125	-85	-144	-260	-127	-105	-155	-93	-119	-147	-72	-219	-213	-150	-51	229
2	AFFX-BioB-3_at (endogenous	AFFX-BioB-3_at	-58	-1	-307	265	-76	215	238	7	106	42	-71	84	-31	-118	-126	-50	-18	-119	100	79

control) Gene	Gene

merge train test data into a single df for easier preprocessing

In [ ]:

```
patients = [str(i) for i in range(1, 73, 1)]
X = pd.concat([train_df, test_df], axis = 1)[patients]
```

In [ ]:

```
X=X.T
```

convert 'ALL' to 0 and 'AML' to 1 and merge entire dataset into 1 df 'data'

In [ ]:

```
X["patient"] = pd.to_numeric(patients)
y["cancer"]= pd.get_dummies(y.cancer, drop_first=True)
# add the cancer column to train data

data = pd.merge(X, y, on="patient")
```

In [ ]:

```
data.head()
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	-214	-153	-58	88	-295	-558	199	-176	252	206	-41	-831	-653	-462	75	381	-118	-565	15091	7	311	-231	21	-107
1	-139	-73	-1	283	-264	-400	-330	-168	101	74	19	-743	-239	-83	182	164	-141	-423	11038	37	134	-161	-21	-180
2	-76	-49	-307	309	-376	-650	33	-367	206	-215	19	-1135	-962	-232	208	432	84	-501	16692	183	378	-221	67	-203
3	-135	-114	265	12	-419	-585	158	-253	49	31	363	-934	-577	-214	142	271	-107	-101	15763	45	268	-27	43	-52
4	-106	-125	-76	168	-230	-284	4	-122	70	252	155	-471	-490	-184	32	213	1	-260	18128	-28	118	-153	-8	-111

5 rows × 7131 columns

In [ ]:

```
X, y = data.drop(columns=["cancer"]), data["cancer"]
```

In [ ]:

```
X.shape,y.shape
```

Out[ ]:

```
((72, 7130), (72,))
```

apply standard scaler preprocessing to X cols

In [ ]:

```
ss=StandardScaler()
X=ss.fit_transform(X)
```

split data into train test data

In [ ]:

```
X_train, X_test, y_train, y_test =  train_test_split(X,y,test_size = 0.40, random_state=
0)
```

Apply Gaussian Naive Bayes

In [ ]:

```
gnb=GaussianNB()
gnb.fit(X_train,y_train)
```

Out[ ]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [ ]:

```
y_pred=gnb.predict(X_test)
```

In [ ]:

```
accuracy_score(y_test,y_pred)
```

Out[ ]:

```
0.9655172413793104
```