Solutions

Task 1.1

By looking at the CPD,age(>23) has the smallest representation among those that have delays longer than 2 years (>= 2). This information could also be considered not significant since the model uses the maximum likelihood estimator.Maximum likelihood does not tell us much, besides that our estimate is the best one we can give based on the data. It does not tell us anything about the quality of the estimate, nor about how well we can actually predict anything from the estimates.

delay	delay(0)	delay(1)	delay(>=2)	delay(NA)
age(20-23)	0.18627450980392157	0.14285714285714285	0.35294117647058826	0.4375
age(<=20)	0.7696078431372549	0.7142857142857143	0.6470588235294118	0.5
age(>23)	0.04411764705882353	0.14285714285714285	0.0	0.0625

Figure 1: CPD of the age variable

Task 1.2

We are using a naive Bayes PGM to fit the data i.e there is no interdependence between the variables age,gender,avg_cs and avg_mat. The fit method in this case sets the marginal distribution of the delay variable to the relative frequencies in the data. Consider the CPD of age shown in figure 1. Using the ratio method we get the same answers.

```
print(ratio(RAW_DATA, lambda t: t['age']=='20-23', lambda t: t['delay']=='>=2'))
0.35294117647058826
print(ratio(RAW_DATA, lambda t: t['age']=='<=20', lambda t: t['delay']=='>=2'))
0.6470588235294118
print(ratio(RAW_DATA, lambda t: t['age']=='23', lambda t: t['delay']=='>=2'))
0.0
```

Task 1.3

The values in CPD for variable avg_mat given delay are the same as the relative frequencies.

delay	delay(0)	delay(1)	delay(>=2)	delay(NA)
avg_mat(2<3)	0.39705882352941174	0.5357142857142857	0.47058823529411764	0.0625
avg_mat(3<4)	0.44607843137254904	0.14285714285714285	0.0	0.125
avg_mat(4<5)	0.11274509803921569	0.0	0.0	0.0625
avg_mat(<2)	0.04411764705882353	0.32142857142857145	0.5294117647058824	0.75

Figure 2: CPD of the avg_mat variable

```
Prob of avg_mat=  2<3  given delay is  0  is:  0.39705882352941174
Prob of avg_mat=  2<3  given delay is  1  is:  0.5357142857142857
Prob of avg_mat=  2<3  given delay is  >=2  is:  0.47058823529411764
Prob of avg_mat=  2<3  given delay is  NA  is:  0.0625
Prob of avg_mat=  3<4  given delay is  0  is:  0.44607843137254904
Prob of avg_mat=  3<4  given delay is  1  is:  0.14285714285714285
Prob of avg_mat=  3<4  given delay is  >=2  is:  0.0
Prob of avg_mat=  3<4  given delay is  NA  is:  0.125
Prob of avg_mat=  4<5  given delay is  0  is:  0.11274509803921569
Prob of avg_mat=  4<5  given delay is  1  is:  0.0
Prob of avg_mat=  4<5  given delay is  >=2  is:  0.0
Prob of avg_mat=  4<5  given delay is  NA  is:  0.0625
Prob of avg_mat=  <2  given delay is  0  is:  0.04411764705882353
Prob of avg_mat=  <2  given delay is  1  is:  0.32142857142857145
Prob of avg_mat=  <2  given delay is  >=2  is:  0.5294117647058824
Prob of avg_mat=  <2  given delay is  NA  is:  0.75
```

Figure 3: Relative frequency of avg_mat variable using ratio method

```
print(model.cpds[2]) # print cpds of avg_mat

avgma=['2<3','3<4','4<5','<2']
delay=['0','1','>=2','NA']
for m in avgma:
    for d in delay:
        print('Prob of avg_mat= ',m,' given delay is ',d,' is: ',ratio(data, lambda
            t: t['avg_mat']==m, lambda t: t['delay']==d))
```

Task 2.1

The required variable elimination query and the marginal distribtuion can be seen in figure 4.

```
Finding Elimination Order: : 100%|██████████| 3/3 [00:00<00:00, 1415.72it/s]
Eliminating: avg_cs: 100%|██████████| 3/3 [00:00<00:00, 309.73it/s]
State names:
age ['20-23', '<=20', '>23']
delay ['0', '1', '>=2', 'NA']
The query is : ve.query(variables = ['delay'], evidence = {'age': '<=20'})
+------------+--------------+
| delay      |   phi(delay) |
+============+==============+
| delay(0)   |       0.8010 |
+------------+--------------+
| delay(1)   |       0.1020 |
+------------+--------------+
| delay(>=2) |       0.0561 |
+------------+--------------+
| delay(NA)  |       0.0408 |
+------------+--------------+
--------------------------------------------------------------------------------
```

Figure 4: Elimination query to infer the probability of delay for the age group <= 20

In task 1.1 the evidence was delay>= 2 and we were inferring CPD of age group given the delay(likelihood). , whereas the evidence in this task is age<= 20 and we are inferring the probability of delay given the age group(posterior).

Task 2.2

The age group with the lowest probability of zero delays on the bachelor's degree is age(> 23) with 0.0441 and the one with highest is age(<= 20) with 0.7696.The required variable elimination query and the results are shown in figure 5.

```
Finding Elimination Order: : 100%|██████████| 3/3 [00:00<00:00, 1967.62it/s]
Eliminating: avg_cs: 100%|██████████| 3/3 [00:00<00:00, 594.29it/s]
The query is : ve.query(variables = ['age'], evidence = {'delay': '0')
+------------+------------+
| age        |   phi(age) |
+============+============+
| age(20-23) |     0.1863 |
+------------+------------+
| age(<=20)  |     0.7696 |
+------------+------------+
| age(>23)   |     0.0441 |
+------------+------------+
```

Figure 5: Elimination query to infer the age given delay=0

Task 2.3

The results from the relative frequencies on the data come out to be the same. The results are shown in figure 6. This can be attributed to the structure of the PGM (explained in 1.2) and Naive bayes assumption.

```
Prob of age=  20-23  given delay is  0  is:  0.18627450980392157
Prob of age=  <=20  given delay is  0  is:  0.7696078431372549
Prob of age=  >23  given delay is  0  is:  0.04411764705882353
```

Figure 6: Relative frequency of age variable given delay=0

```
age=['20-23','<=20','>23']
delay=['0']
for m in age:
    for d in delay:
        print('Prob of age= ',m,' given delay is ',d,' is: ',ratio(data, lambda t: t['age']==m,
            lambda t: t['delay']==d))
```

Task 2.4

As expected , we get the same result as in task 2.3 by using the function mapquery.

```
q=ve.map_query(variables=['age'],evidence={'delay':'0'})
print()
print('The required query is :ve.map_query(variables=[\'age\'],evidence={\'delay\':\'0\'})')
print(q)
```

```
Finding Elimination Order: : 100%|██████████| 3/3 [00:00<00:00, 1301.37it/s]
Eliminating: avg_cs: 100%|██████████| 3/3 [00:00<00:00, 283.39it/s]
The required query is :ve.map_query(variables=['age'],evidence={'delay':'0'})
{'age': '<=20'}
```

Figure 7: MAP query for the same result as in task 2.3

Task 3.1

```
data = pd.DataFrame(data=RAW_DATA)
model = BayesianModel([('age', 'delay'),
```

```
                        ('gender', 'delay'),
                        ('avg_mat', 'delay'),
                        ('avg_cs', 'delay')])
model.fit(data) # Uses the default ML-estimation
```

Task 3.2

There are 384 entries in the CPD table for delay. This can be computed as $4 * 3 * 4 * 4 * 2$ i.e each of the possible values that can be taken by the variables age,delay,avgmat,avgcs,gender.

```
model.cpds[3].values.shape

(4, 3, 4, 4, 2)

model.cpds[3].values.size

384
```

Figure 8: Shape and size of CPD for delay

Task 3.3

There are 265 samples each having 5 dimensions. In comparison to number to the number of entries in the CPD of delay , the relative small size of data results in uninformative conditional probability distributions.

```
data.values.shape

(265, 5)

data.values.size

1325
```

Figure 9: Shape and size data

Task 3.4

In absence of samples ,the fit method in PGMPY , assigns equal probability to each of the CPD for delay . This also makes sure that the sum over all cpd (over the column) sums to 1.

Task 3.5

The results of the marginal distribution from the PGM and the relative frequencies can be seen figure 11. The absolute error has been computes as follows : $\frac{|MarginalfromCPD - RelativeFrequency|}{MarginalfromCPD}$

```
ve = VariableElimination(model)
q = ve.query(variables = ['delay'])
print(q)
```

```
Finding Elimination Order: : 100%|██████████| 4/4 [00:00<00:00, 1336.62it/s]
Eliminating: avg_cs: 100%|██████████| 4/4 [00:00<00:00, 158.69it/s]+-------------+---
| delay       |   phi(delay) |
+=============+==============+
| delay(0)    |       0.7611 |
+-------------+--------------+
| delay(1)    |       0.1116 |
+-------------+--------------+
| delay(>=2)  |       0.0754 |
+-------------+--------------+
| delay(NA)   |       0.0520 |
+-------------+--------------+
```

Figure 10: Marginal distribution for delay.

```
age=['<=20']
delay=['0','1','2','NA']
idx0=np.where(data.values[:,3]=='0')
idx1=np.where(data.values[:,3]=='1')
idx2=np.where(data.values[:,3]=='>=2')
idx3=np.where(data.values[:,3]=='NA')
print('Relative Frequency')
print('delay(0):',idx0[0].size/data.shape[0])
print('delay(1):',idx1[0].size/data.shape[0])
print('delay(>=2):',idx2[0].size/data.shape[0])
print('delay(NA):',idx3[0].size/data.shape[0])
```

	Marginal from PGM	Relative Frequency	Relative Error
delay(0)	0.7611	0.769811321	1.144569801
delay(1)	0.1116	0.105660377	5.32224251
delay(>=2)	0.0754	0.064150943	14.91917321
delay(NA)	0.052	0.060377358	16.11030479

Figure 11: Marginal distribution for delay.

For this given structure, the exact inference deviates from the relative frequencies in data because it computes the underlying probabilities using Variable elimination by marginalizing over the variables , gender,avgmat,avgcs,age .On the other hand , the relative frequency , just computes the ratio of observed samples as a fraction of samples size.

Task 4.1

The Kullback–Leibler divergence (also called relative entropy) is a measure of how one probability distribution is different from a second, reference probability distribution. It can also be interpreted as the expected extra message-length per datum due to using a code based on the wrong (target) distribution compared to using a code based on the true distribution.KL divergence is not a metric measure. It may not be symmetric: the KL from p(x) to q(x) is generally not the same as the KL from q(x) to p(x).
$D_{KL}(P \parallel Q) = \sum_{i=0}^{n} p(x_i) log\left(\frac{p(x_i)}{q(x_i)}\right)$

Task 4.2

Consider the situation where q(X) becomes 0. In this case KL divergence
$p(x) \log \frac{p(x)}{0} = \infty$
This implies that KL divergence is infinite if there exists an x where p(x)>0 and q(x)=0. Two such queries are

```
Query: delay given {'gender': '0', 'age': '>23', 'avg_cs': '<2'}
Query: delay given {'avg_cs': '4<5', 'gender': '1', 'avg_mat': '<2'}
```

Task 4.3

```
print([len([r for r in divs2 if len(r[0][1])==n]),
len([r for r in divs2 if len(r[0][1])==n and r[3] < r[5]]),
len([r for r in divs2 if len(r[0][1])==n and r[3] > r[5]]),
len([r for r in divs if len(r[0][1])==n and \
not(math.isfinite(r[3]) and math.isfinite(r[5]))]),
sum(r[3] for r in divs2 if len(r[0][1])==n)])
```

In line 1 we are counting the cases/queries where there are exactly 2 variables in the evidence. In line 2 , we count cases where 2nd model has a greater KL divergence i.e model 1 wins. In line 3 , we count cases where 1st model has a greater KL divergence i.e model 2 wins . We get the counts of cases/queries where kl divergence for both models is not finite. The last line adds the divergences for 1st model.

Task 4.4

The number of randomly generated queries was increased to 1000. Here are the required results. It can be observed that for n=1 i.e fewer evidence, M1 has lower KL divergence as compare to model 2 in all the cases. and for N=3, M2 wins 11 out of 21 times. As the evidences increases , the more complex model 2 generally outperforms model 1. This can also be seen in the last case n=4.

```
################################################################
[N,No. of queries(with N evidence),M1 wins,M2 wins,Number of inf,Sum Div M1,Sum Div M2]
[1, 168, 168, 0, 83, 9.058807993416197e-15, 5.129688000853106]
[2, 85, 58, 27, 185, 2.288365815550963, 3.4941235951365766]
[3, 21, 10, 11, 213, 2.361517518777792, 1.291041925645995]
[4, 10, 0, 5, 235, 0.2949779404266466, 0.0]
################################################################
```

Figure 12: Target:Delay

Task 4.5

By using age as the target variable , we observe similar results as in task 4.4 , for fewer evidence conditions model 1 tends to outperform model 2 and has a higher winning percentage. This percentage however decreases as the evidence condition increases.

```
################################################################
[N,No. of queries(with N evidence),M1 wins,M2 wins,Number of inf,Sum Div M1,Sum Div M2]
[1, 207, 179, 28, 45, 4.29280182148172, 8.113827215296048]
[2, 114, 85, 29, 148, 4.68647878479458, 7.70866098820534]
[3, 42, 28, 14, 187, 1.1964450527274941, 2.2416745624567613]
[4, 27, 14, 13, 230, 0.8161105502763595, 1.104121758108926]
################################################################
```

Figure 13: Target:Age

Task 4.6

A list of performance data (like above) but with both delay and age as targets (for target in ['delay', 'age']), is created for 1000 randomly generated queries .

```
############################################################
[N,No. of queries(with N evidence),M1 wins,M2 wins,Number of inf,Sum Div M1,Sum Div M2]
[1, 185, 171, 14, 70, 2.1672075568650166, 6.530944676379363]
[2, 75, 51, 24, 145, 2.2553076177765377, 4.7900771340209145]
[3, 40, 22, 18, 220, 1.1465720156796417, 2.402674600500891]
[4, 10, 6, 3, 255, 0.22646568458609384, 0.40794730127055745]
############################################################
```

Figure 14: Target: in [Age,Delay]

Task 5.1

In the code, the data is split to training and validation set. The pgm model is fit on the training set, and relative frequency is calculated on the validation set. The divergence /entropy than computed for each model.

Task 5.2

The results from the code can be seen in figure 15. Please note that target variable was set to delay for this computation.

```
############################################################
[N,No. of queries(N evidence),M1 wins(100%Train),M2 wins(100%Train) wins,M1 wins(75%Train25%Val),M2 wins(75%Train25%Val)]
[1, 160, 160, 0, 69, 28]
[2, 60, 45, 15, 3, 7]
[3, 25, 10, 15, 2, 2]
[4, 10, 0, 10, 0, 0]
############################################################
```

Figure 15: Task 5: Checking for noise resistance

Task 5.3

For 100% training set , model 1 and model 2 have a 100% winning rate for n=1 and n=4 , where n denotes the number of variables in the evidence. However ,this winning percentage drops when we use a validation set. In absence of a validation set, the model over fits the data.

Task 6.1

K2 Score algorithm seeks for the Bayesian network structure G^* given a data set D with maximal $p(G^*|D)$, where $p(G|D)$ is the probability of network structure G given the data set D. This algorithm heuristically searches for the most probable Bayesian network structure given a data set of cases. As per the K2 Score , Model 1 outperforms Model 2 since it has greater score.(the less negative the K2 score , the better) since we are using log.

```
Structure scores [Model1 , Model2]: [-1037.7184908064394, -1096.365865151934]
```

Figure 16: K2-Score for each model

Task 6.2

Using 100 % data for training results in over fitting the data and poor results in inference problems on unobserved data points and also poor generality. To compensate for the over fitting problem and to properly tune the hyper parameters involved , we often split the data to training,validation and tests set. This cross validation and scoring can result in models that have a better performance on the test set.

Task 6.3

The best model according to exhaustive search in both the presence as well as absence of gender variable is the same as shown below.It can seen that in either case the variable gender is disconnected from the BN and is therefore independent from the other variables.

```
ar[-1][0]

-925.8214756938006

ar[-1][1].edges()

OutEdgeView([('avg_cs', 'avg_mat'), ('avg_cs', 'age'), ('avg_mat', 'delay')])
```

Figure 17: Best model with K2-Score (without gender)

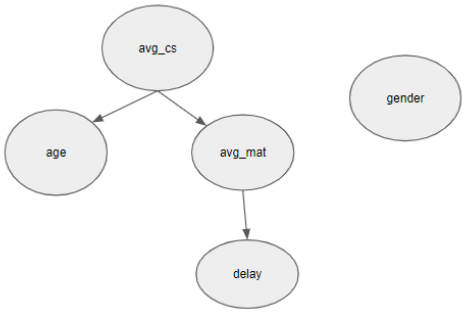
```
arr[-1][0]

-1002.3854437546557

arr[-1][1].edges()

OutEdgeView([('avg_cs', 'avg_mat'), ('avg_cs', 'age'), ('avg_mat', 'delay')])
```

Figure 18: Best model with K2-Score (all variables)



Task 6.5

The results from the "hill climb search" on the data for 4 variables as well as 5 variables using the K2Score can be seen in Figure 19. We observe the same results as in Task 4.3 i.e we get the same BN. The results with BIC score are different. The underlying BN structure is simpler and has fewer number of edges . This can be attributed to the fact that BIC score often tends to under fits the data.

```
Hill Climb Search (4 variables): BicScore
Best Model: [('avg_cs', 'avg_mat'), ('avg_mat', 'delay')]
Hill Climb Search (4 variables): K2Score
Best Model: [('avg_cs', 'avg_mat'), ('avg_cs', 'age'), ('avg_mat', 'delay')]
Hill Climb Search (5 variables): BicScore
Best Model: [('avg_cs', 'avg_mat'), ('avg_mat', 'delay')]
Hill Climb Search (5 variables): K2Score
Best Model: [('avg_cs', 'avg_mat'), ('avg_cs', 'age'), ('avg_mat', 'delay')]
```

Figure 19: Hill Climb Search using K2 Score and Bic Score

Task 6.6

Probabilistic Graphical models can be used for tractable estimation of multivariate probability distributions and are an important ingredient for causal statistical inference. In our case the sample size is small as well as the number of nodes are significantly less(=5) , we did an exhaustive search using different scoring criteria to find the optimal BN structure.The underlying model with the highest score in task 6.5 turned out to be quite simple.The probability distributions for the cases where samples are missing (task 3.4) is uninformative.We can try alternate classifiers on the given dataset and do a detailed model comparison for further.