MARINOS MARIOS (Student number : 5353106)

Exercise 1

Source	DF	SS	MS (SS/DF)	F_computed	F-Table(a=0.05)	P-Value
A	1	1.367	1.367	0.043190153	4.00119137675498	0.836071444
B	1	7.825	7.825	0.247229659	4.00119137675498	0.620849767
C	2	1.235	0.6175	0.019509817	3.15041131058273	0.980685487
AB	1	0.012	0.012	0.000379138	4.00119137675498	0.984529604
AC	2	0.565	0.2825	0.008925544	3.15041131058273	0.991115487
BC	2	0.12	0.06	0.001895691	3.15041131058273	0.998106165
ABC	2	1.25	0.625	0.019746778	3.15041131058273	0.980453282
Error	60	1899.044	31.65073333			
Total	71	1911.418				

Figure 1: ANOVA table of Exercise 1.

1. The replications can be found by the error $Df_{error} = abc(r-1)$, and $Df_{error}=60$ so $2*2*3(r-1) = 60 \rightarrow 12r - 12 = 60 \rightarrow r = 4$, so 4 replicates were performed. Alpha chosen for F-Distribution = 0.05.
2. 2 Levels of B were used. ($Df_B + 1$)
3. Levels used for each factor is 2, 2, 3 respectively for A, B, C factors and the replications are 4. So the experiments need to be done is 2*2*3*4 = 48
4. For alpha = 0.05, all factors and their interactions are not significant because P-value > 0.05, thus the model is bad.

Exercise 2.

Question 1,2:

The train-test split I used is 80-20, 75-25 and 70-30. Also, didn't change the parameters in any algorithm and it is as on the provided code for exerc2 and used OneHotEncoder from sklearn to all datasets. Below is the table for question 1 and 2.

	SVM	KNN	MLP
Mushroom(80-20)	100.00%	100.00%	100.00%
Mushroom(75-25)	100.00%	100.00%	100.00%
Mushroom(70-30)	100.00%	100.00%	100.00%
Bank(80-20)	87.90%	87.20%	86.50%
Bank(75-25)	87.50%	87.30%	86.70%
Bank(70-30)	87.80%	87.50%	87.10%
Car(80-20)	87.80%	83.20%	97.90%
Car(75-25)	89.80%	85.60%	97.90%
Car(70-30)	89.00%	84.90%	97.40%
Zoo(80-20)	95.20%	95.20%	95.20%
Zoo(75-25)	96.15%	96.15%	96.15%
Zoo(70-30)	96.77%	93.54%	93.54%

Figure 2: SVM, KNN and MLP on 4 differnt datasets.

Question 3.

From below code :

```
Accuracy = [100, 100, 100, 87.90, 87.50, 87.80, 87.80, 89.80, 89, 95.20, 96.15, 96.77,
            100, 100, 100, 87.20, 87.30, 87.50, 83.20, 85.60, 84.90, 95.20, 96.15, 93.54,
            100, 100, 100, 86.5, 86.7, 87.1, 97.9, 97.9, 97.4, 95.2, 96.15, 93.54]

df = pd.DataFrame({'Accuracy': Accuracy,
                   'MLAlgorithms': np.repeat(['SVM', 'KNN', 'MLP'], 12),
                    'Datasets':np.r_[np.repeat(['80-20split', '75-25split', '70-30split'],3),
                                      np.repeat(['80-20split', '75-25split', '70-30split'],3),
                                      np.repeat(['80-20split', '75-25split', '70-30split'],3),
                                      np.repeat(['80-20split', '75-25split', '70-30split'],3)]})

rp.summary_cont(df.groupby(['MLAlgorithms']), conf = 0.9)['Accuracy']
```

you get the following results.

	N	Mean	SD	SE	90% Conf.	Interval
MLAlgorithms						
KNN	12	91.7158	6.4318	1.8567	88.3814	95.0503
MLP	12	94.8658	5.2606	1.5186	92.1386	97.5930
SVM	12	93.1600	5.3279	1.5380	90.3979	95.9221

So we have :
CI for KNN is[88.3814, 95.0503]
CI for MLP is [92.1386, 97.5930]
CI for SVM is [90.3979, 95.9221]

From the above results we can conclude that the 3 algorithms aren't different with confidence 90% because the above confidence intervals are overlapping.

Question 4 : In order to find the percentage of what variation is explained by the learning model-dataset interaction we need to calculate the SS_AB (Sum Squares of AB) / SST (Sums Squares Total) * 100.

	sum_sq	df	F	PR(>F)
C(MLAlgorithms)	90.338408	2.0	10.752945	3.674392e-04
C(Datasets)	98.344408	2.0	11.705896	2.183872e-04
C(MLAlgorithms):C(Datasets)	859.949809	4.0	51.179741	3.220523e-12
Residual	113.417167	27.0	NaN	NaN

We have to calculate the sum square total = 90.338408 + 98.344408 + 859.949809 + 113.417167 = 1162.049792.

So we have $Percentage\ variation\ AB = \frac{859.949809}{1162.049792} * 100 = 0.74002836618 * 100 = 74.0028\,\%$

Exercise 3.

the generator used for the following sign table is the I = ACD as we found by the combinations of factors A, B, C, D. ACD gives the I.

	I	A	B	C	D	AB	BC	BD	y
	1	1	-1	-1	-1	-1	1	1	100
	1	1	1	-1	-1	1	-1	-1	120
	1	-1	-1	-1	1	1	1	-1	40
	1	-1	1	-1	1	-1	-1	1	20
	1	-1	-1	1	-1	1	-1	1	15
	1	-1	1	1	-1	-1	1	-1	10
	1	1	-1	1	1	-1	-1	-1	30
	1	1	1	1	1	1	1	1	50
SUM	385	215	15	-175	-105	65	15	-15	Total
q's	48.125	26.875	1.875	-21.875	-13.125	8.125	1.875	-1.875	Total/8
	q0	qA	qB	qC	qD	qAB	qBC	qBD	

1. The value of each effect is shown on the above table on the q's row.
2. On the following image we can see on the row VARIANCES EXPLAINED the percentage for A, B, C, D, AB, BC, BD respectively.

q's^2	Formula = 2^(k-p)*qS^2	5778.125	28.125	3828.125	1378.125	528.125	28.125	28.125	
SST									11596.875
VARIANCE EXPLAINED		49.825%	0.243%	33.010%	11.884%	4.554%	0.243%	0.243%	100.00%

3. the list of confoudings with the generator of I = ACD is the following :

- IA = A^2CD → A = CD confouding with 10.
- IB = ABCD → B = ABCD confouding with 15.
- IC = AC^2D → C = AD confouding with 7.
- ID = ACD^2 → D = AC confouding with 6.
- IAB = A^2CD → AB = BCD confouding with 13.
- IAC = A^2C^2D → AC =D √
- IAD = A^2CD^2 → AD = C √
- IBC = AC^2D → BC = ABD confouding with 12.
- IBD = ABCD^2 → BD = ABC confouding □□□□ 11.
- ICD = AC^2D^2 → CD = A √
- IABC = A^2BC^2D → ABC = BD √
- IABD = A^2BCD^2 → ABD = BC √
- IBCD = ABC^2D^2 → BCD = AB √
- IACD = A^2C^2D^2 → ACD = I, Generator.
- IABCD = A^2BC^2D^2 → ABCD = B √

4. I = ABCD might be better generator, since the resolution of this generator is 4.
5. The resolution of the design above with generator I = ACD is equal to 3.

Exercise 4.

We have N = 19 jobs and as David says the 90% of the jobs have their data stored in cache and their expected response time is 1 sec whereas the rest 10% of the jobs don't have their data stored in cache, thus they need to go to look up in the database instead. If we calculate how many jobs on average should go look for the data on database it is equal to 19 * 0.1 = 1.9 jobs. So when David says he sees 5 jobs on average at the database, his advisor notice there is definitely something wrong because the jobs on database should be 2 on average.

Exercise 5.

Considering all the giving values we have :

$E[Z] = 12$
$C = C_{CPU} = 1.600$ jobs
$C_{slow\_Disk} = 12.000$ jobs
$C_{fast\_Disk} = 32.000$ jobs
$B_{CPU} = 1.080$ sec
$B_{slow\_Disk} = 600$ sec
$B_{fast\_Disk} = 400$ sec

Question 1.

$D_{CPU} = 1080 / 1600 = 0.675$ sec/job $\rightarrow$ D_{MAX} Cpu causes the bottleneck.
$D_{slow\_Disk} = 600 / 1600 = 0.375$ sec/job
$D_{fast\_Disk\ B} = 400 / 1600 = 0.25$ sec/job

So for throughput bound, following the theory we have:

$$X \leq min\left(\frac{N}{1.3+12}, \frac{1}{0.675}\right) \Rightarrow min\left(\frac{N}{13.3}, 1.48148\right)$$

for response time bound we have $E[R] \leq max(1.3, N*0.675-12)$

We can calculate the N* that is equal to = $\frac{1.3+12}{0.675} = \frac{13.3}{0.675} = 19.70 < 20$

So according to theory, if we have N>N*, 20>19.70, then the system will certainly have queue waiting time.

Question 2.

i. To find out how the D_{fast_Disk} will change we have to calculate the $E[V_{fast}]$ and $E[V_{slow}]$ and also the $E[S_{fast}]$ and $E[S_{slow}]$ according to the theory. So we have $E[V_{fast}] = 32000 / 1600 = 20$ visit/job and the $E[V_{slow}] = 12000/ 1600 = 7.5$ visit/job. $E[S_{fast}] = 400 / 32000 = 0.0125$ sec/visit and $E[S_{slow}] = 600 / 12000 = 0.05$ sec/visit.

With these we can caclulate the new $D_{fast\_new} = E[S_{fast}] * E[V_{fast\_new}]$, where $E[V_{fast\_new}] = E[V_{fast}] + E[V_{slow}]$.

So we have $E[V_{fast\_new}] = 20 + 7.5 = 27.5$ visit/job and the $D_{fast\_new} = 0.0125 * 27.5 = 0.34375$ sec/job.

So, we can see that the D_{fast_new} is slightly increasing as it takes the "effort" of slow disk also, but it doesn't matter as the bottleneck, D_{max} remains the same and it comes from the cpu which is equal to 0.675 sec/job, therefore to make our system quicker we have to make a change to cpu.

ii. If we make our CPU faster by 50% and keep the rest as it was on Question 1, then we have $E[D_{CPU\_new}] = E[S_{CPU\_new}] * E[V_{cpu}]$, and $E[S_{CPU\_new}] = E[S_{CPU}] / 1.5$. So, firstly we have to calculate the $E[V_{cpu}]$ and $E[S_{CPU}]$. By theory we know that $E[V_{cpu}] = C_{CPU} / C_{total} \rightarrow 1600 / 1600 = 1$ visit/job. For $E[S_{CPU}]$ we have $E[S_{CPU}] = B_{CPU} / C_{CPU} = 1080 / 1600 = 0.675$ sec/visit. Now, having that we can calculate the 2 first equations. By using $E[S_{CPU\_new}] = E[S_{CPU}] / 1.5$ we have $\rightarrow E[S_{CPU\_new}] = 0.675 / 1.5 = 0.45$ sec/visit and eventually the $E[D_{CPU\_new}] = 0.45 * 1 = 0.45$ sec/job.

That means we improved our system because we dropped our D_{max} from 0.675 to 045. If we consider N is small that doesn't affect our system, but when N is bigger our N* is equal to 29.05, thus our N job limit is 29 instead of 19 jobs without having any queue in our system.

Exercise 6.

Question 1 : In below figure we can see the DTMC for the exercise 6 which consists of 3 different states. Working, Down due to backhoe and Down due to software bug.

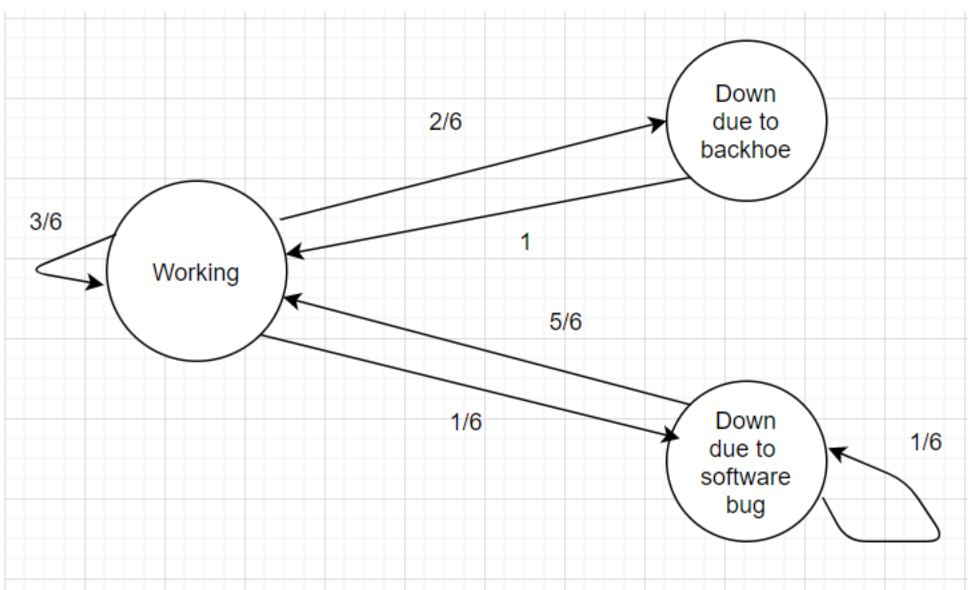

Figure 3: DTCM with 3 states for exercise 6.

And the transition probability matrix is equal to:

	Working	Down Backhoe	Down Software bug
Working	0.5	0.333333333333333	0.166666666666667
Down Backhoe	1	0	0
Down Software bug	0.833333333333333	0	0.166666666666667

Question 2 : Firstly, a DTMC to consider ergodic should have all three properties of Recurrent, Aperiodic and Irreducible for **all states.** Even if it hasn't one of the above it is not ergodic.

- A state **S1** is a transient state if there exists a state **S2** that is reachable from **S1**, but the state **S1** is not reachable from state **S2**. Otherwise the state is recurrent state. (**Recurrent)**
  Note: *A state **S2** is reachable from state **S1** if there is path leading from **S1** to **S2.***
- A state **S1** is periodic with period k>1 if k is the smallest number such that all paths leading from **S1** back to state **S1** have length that is a multiple of k. (**Periodic**)
- Two states **S1, S2** can communicate if **S2** is reachable from **S1**, and **S1** is reachable from **S2**. (**Irreducible**)

So, if we observe Figure 3 (DTMC) we can see that :

- The DTMC is Irreducible as we can get to any state from any state.
- The DTMC is Aperiodic.
- All states are recurrent, so the DTMC is also recurrent.

Taking all into account the DTMC is ergodic.

Question 3 :

We have to use the Stationary Distribution here. Firstly, we need to compute [$\pi_{Working}$, $\pi_{DBackhoe}$, $\pi_{DSoftware}$] and then multiply it with the Transition probability matrix above.

$$equation\,1: \pi(Working) + \pi(DBackhoe) + \pi(DSoftware) = 1$$

$$equation\,2: \pi(Working) = \frac{3\pi(Working)}{6} + \pi(DBackhoe) + \frac{5\pi(DSoftware)}{6}$$

$$equation\,3: \pi(DBackhoe) = \frac{2\pi(Working)}{6}$$

$$equation\,4: \pi(DSoftware) = \frac{\pi(Working)}{6} + \frac{\pi(DSoftware)}{6}$$

set π(Working) = x, π(DBackhoe) = y, π(DSoftware) = z.

then we have :

$$equation\,1: x + y + z = 1$$

$$equation\,2: x = \frac{3x}{6} + y + \frac{5z}{6} \Rightarrow -3x + 6y + 5z = 0$$

$$equation\,3: y = \frac{(2x)}{6} \Rightarrow -2x + 6y = 0 \Rightarrow x = 3y$$

$$equation\,4: z = \frac{x}{6} + \frac{z}{6} \Rightarrow x - 5z = 0 \Rightarrow x = 5z$$

if we now isolate x = 3y we get 3 equations and by using an online calculator we get the following values :

$$x=\frac{15}{23}, y=\frac{5}{23}, z=\frac{3}{23} \qquad where\ x=\pi(Working), y=\pi(DBackhoe), z=\pi(DSoftware)$$

By the Theorem (Theorem 8.6 (Stationary distribution = Limiting distribution)) we know that stationary distribution also represents the limiting probability distribution.

<u>Thus, the data center is working 15 out of 23 days on average.</u>

Question 4 :

From question 3 we already have the probability pf π(DBackhoe) = 5/23.
Now, because we know that our DTMC is ergodic, according to theory we can use the below formula (Figure 4) to find out the expected number of days between backhoe failures.

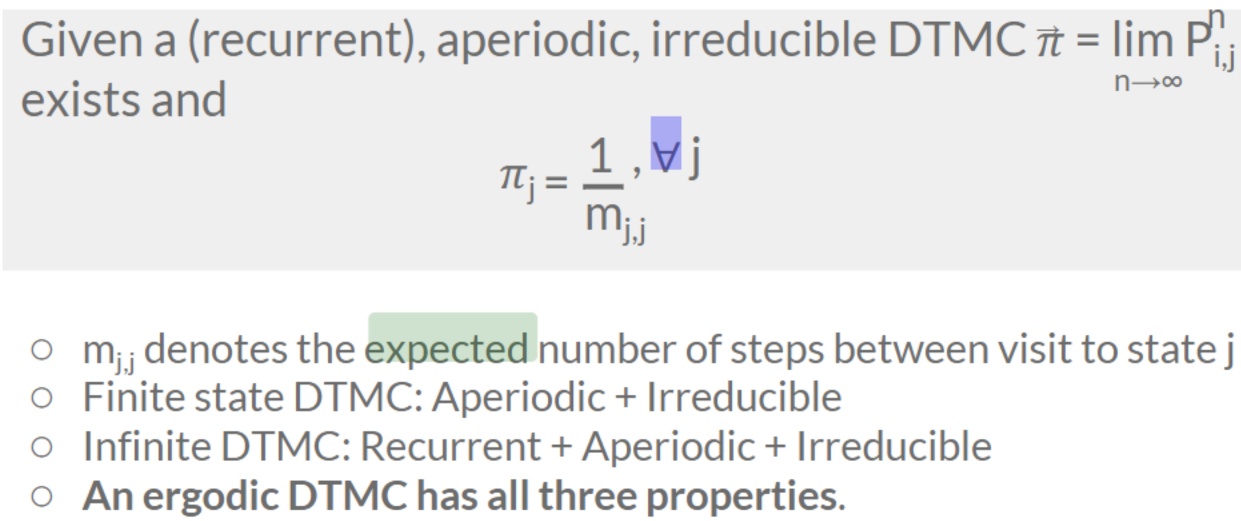

Figure 4: Ergodic Theorem

$$\pi(DBackhoe)=\frac{1}{m(DBackhoe, DBackhoe)}, \pi(DBackhoe)=\frac{5}{23}$$

$$So\ we\ have\ \frac{5}{23}=\frac{1}{m(DBackhoe, DBackhoe)} \Rightarrow 5\,m(DBackhoe, DBackhoe)=23$$

$$\Rightarrow m(DBackhoe, DBackhoe)=4.6\ days$$

So, the expected number of days between backhoe failures is equal to 4.6 days.

Exercise 8

The paper I will review is the paper called AlloX: Compute Allocation in Hybrid Clusters.

1. Question 1. Name: Marinos Marios, Student number: 5353106.

2. Question 2. The problem that the paper addresses, is that the modern deep learning frameworks work with different hardware (e.g. CPU, GPU, other accelerators) to do the computations for training the NN, and that causes a significant problem, the configuration of each job and the order of jobs in order to optimize the performance objective such as the average job completion time and in the meantime providing fairness among multiple users. Now when it comes to the contributions this paper make on the field, the AlloX system attach importance to a very crucial factor when it comes to deep learning, that is the system performance in hybrid systems CPU-GPU. The system AlloX can be applied in multiple domains of Computer Science such as to different network interfaces and storage devices as the authors mention but also can be used in any problem that requires scheduling jobs like scheduling the jobs on operating systems.

3. Question 3,4. I strongly believe that the paper is very clear and well written as it constructs very smoothly and that is the biggest strength in this paper. Firstly, the authors provide the existed algorithms developed for this kind of problems and where they lack. After that, they present the algorithm that the authors designed explained step-by-step and finally they provide the experiment results on cluster which line up with the explanation of algorithm and their assumptions. This algorithm could have a huge impact on training big DNN’s on clusters with different kind of hardware (CPU,GPU) in order to reduce the training time. Finally, I don’t find this paper really creative, despite the fact that is really interesting to read and insightful as machine learning and AI is a really hot field nowadays.

4. Question 5. To conclude, there is multiple ways to further extend this paper on in my opinion. Firstly, we could try to implement this algorithm in different domains of computer science where job scheduling is necessary and we need to speed up the process of scheduling and demonstrate if it has positive or negative effect on it. Also, we could try different system configurations and not only CPU-GPU as the authors did, e.g. more complex configurations as CPU-GPU-TPU for Tensorflow framework as it uses all 3 hardware, or simpler ones with 2 different like CPU-TPU or CPU-FPGA. In addition, we could try to get in play the system parameters from the different frameworks in combination with the system hardware in order to minimize the energy consumption of the system and in the meanwhile minimizing the average job completion time as the AlloX does.