# Question 1

1. False. There still exists a common cause D which makes the trail active.
2. True. There is no active trail between A and E.
3. True. There is no active trail between F and E.
4. True. There is no active trail between B and D.

# Question 2

B

M1

M2

M3

O

|  |  |  |
| --- | --- | --- |
| Variable Name | Domain (set of values) | Interpretation |
| B | {M1, M2, M3} | Refers to the box containing the 3 machines |
| M1 | {1, 2, 3, 4} | Refers to Machine 1 with the specific probability for each of the 4 values |
| M2 | {1, 2, 3, 4} | Refers to Machine 2 with the specific probability for each of the 4 values |
| M3 | {1, 2, 3, 4} | Refers to Machine 3 with the specific probability for each of the 4 values |
| O | {111, 112, 113, 114, ..., 444} | Refers to the 3 outcomes that depends on the machine chosen |



CPT for variable 1

P(1 | machine)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | M1 | M2 | M3 |
| 1 | 0.05 | 0.01 | 0.1 |

CPT for variable 2

P(2 | machine)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | M1 | M2 | M3 |
| 2 | 0.075 | 0.02 | 0.1 |

CPT for variable 3

P(3 | machine)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | M1 | M2 | M3 |
| 3 | 0.075 | 0.07 | 0.1 |

CPT for variable 0

P(0 | machine)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | M1 | M2 | M3 |
| 0 | 0.8 | 0.9 | 0.7 |

It is most likely to be from machine M3.

# Question 3

1. Diagram

   Description automatically generated
2. Radiation = 0: 0.3757

Radiation = 1: 0.6243

Text

Description automatically generated

1. Cancer = 0: 0.5178

Cancer = 1: 0.4822

Text

Description automatically generated

1. Solar Flares is currently not independent of Cancer as there exists an active path between them.
2. The probability is 0.3754.

Graphical user interface, text

Description automatically generated

# Question 4

a)

## download and load data

from keras.datasets import mnist

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

b)

from sklearn.metrics import f1\_score, roc\_auc\_score

from keras.models import Sequential

from keras.layers import Dense

model = Sequential()

layers = [

Dense(

input\_dim=X\_train.shape[1],

units=256,

kernel\_initializer="uniform",

activation="relu",

),

Dense(units=10, kernel\_initializer="uniform", activation="softmax"),

]

for layer in layers:

model.add(layer)

model.compile(

optimizer="adam", loss="sparse\_categorical\_crossentropy", metrics=["accuracy"]

)

model.summary()

model\_history = model.fit(X\_train, y\_train, epochs=25, verbose=0)

test\_loss = model.evaluate(X\_test, y\_test, verbose=0)

y\_pred = np.argmax(model.predict(X\_test), axis=-1)

f1 = f1\_score(y\_test, y\_pred, average="weighted")

y\_pred\_prob = model.predict(X\_test)

roc = roc\_auc\_score(y\_test, y\_pred\_prob, multi\_class="ovo")

print(

f"""

Accuracy: {round(test\_loss[1] \* 100, 5)}%

F1 Score: {round(f1, 5)}

ROC AUC Score: {round(roc, 5)}

Total Loss: {round(test\_loss[0], 5)}

"""

)

c) The output will be 10 probabilities that add up to 1 for the 10 classes. The activation function is softmax which normalises the final output.

d) The loss function is a type of cross entropy that measures the average information gain needed to identify the class. Because I am using integers for classes i.e., 1,2,3, I use the sparse categorical version.

e)

Shape

Description automatically generated

f)

Accuracy for 0: 97.245%

Accuracy for 1: 98.15%

Accuracy for 2: 94.864%

Accuracy for 3: 94.653%

Accuracy for 4: 94.705%

Accuracy for 5: 92.152%

Accuracy for 6: 96.555%

Accuracy for 7: 96.012%

Accuracy for 8: 96.509%

Accuracy for 9: 93.954%

Total accuracy: 95.54%

# Question 5

a)

base\_model = MobileNetV2(

include\_top=False,

weights="imagenet",

input\_shape=(MIN\_SIZE, MIN\_SIZE, 3),

classes=y\_train.shape[1],

)

b) the include\_top flag in part a)

c)

model = Sequential()

model.add(base\_model)

model.add(Flatten())

model.add(Dropout(0.5))

model.add(Dense(1024, activation='relu'))

model.add(BatchNormalizationV2())

model.add(Dropout(0.3))

model.add(Dense(256, activation='relu'))

model.add(BatchNormalizationV2())

model.add(Dropout(0.3))

model.add(Dense(64, activation='relu'))

model.add(BatchNormalizationV2())

model.add(Dropout(0.3))

model.add(Dense(10, activation=("softmax")))

d)

model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=["accuracy"])

checkpoint\_path = f"{ROOT\_FOLDER}cp.ckpt"

checkpoint = ModelCheckpoint(

"best\_model.hdf5",

monitor='val\_accuracy',

verbose=1,

save\_best\_only=True,

mode='max',

)

datagen = ImageDataGenerator(

width\_shift\_range=0.1,

height\_shift\_range=0.1,

horizontal\_flip=True,

rotation\_range=20,

validation\_split=0.1,

)

datagen.fit(X\_train)

hist = model.fit(

datagen.flow(X\_train, y\_train, batch\_size=100),

epochs=20,

callbacks=[checkpoint],

validation\_data=(X\_test, y\_test),

)

1) I just added simple dense and batch normalisation layers. I also added Dropout layers to reduce overfitting.

The dense layers start with 1024 neurons to match the output from MobileNetV2 and divides by 4 each time a new dense layer is applied.

The batch normalisation layer just normalises the input from the dense layer.

2)

Chart, line chart

Description automatically generated

I used a callback function to save the best model based on validation accuracy.  
  
For the batch size I just tried a increasing it by 50 each time and ended up with 100.

I actually trained with 50 epochs but google colab crashed but I managed to save the model due to a callback function.

3) 91.54%