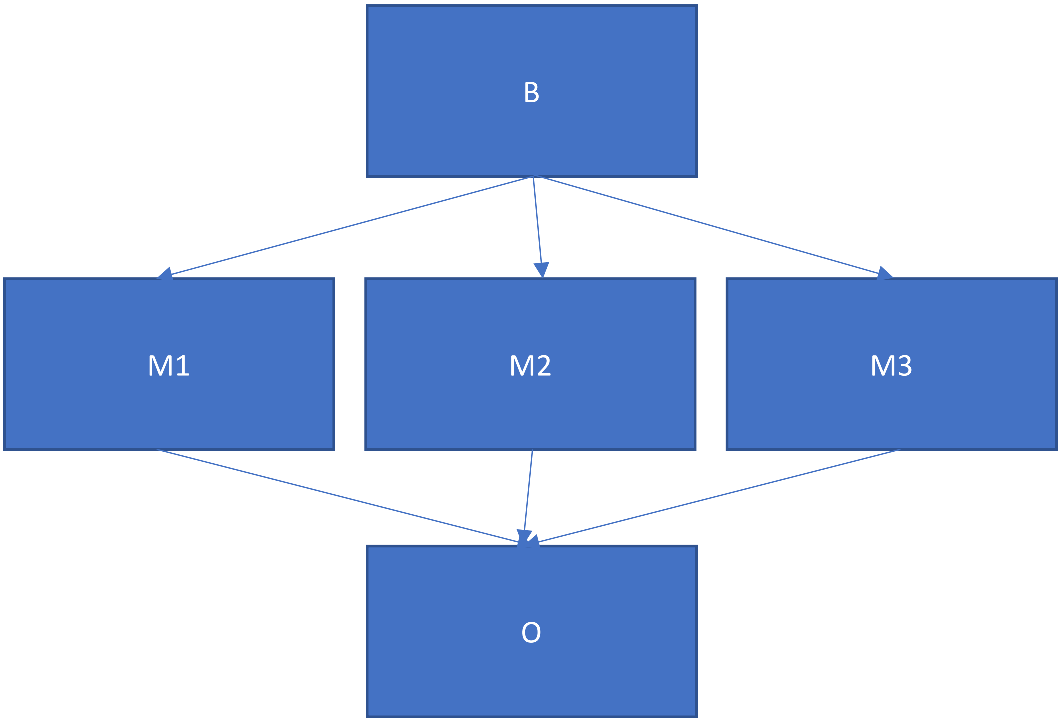
Question 1

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

i)



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

ii)

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)

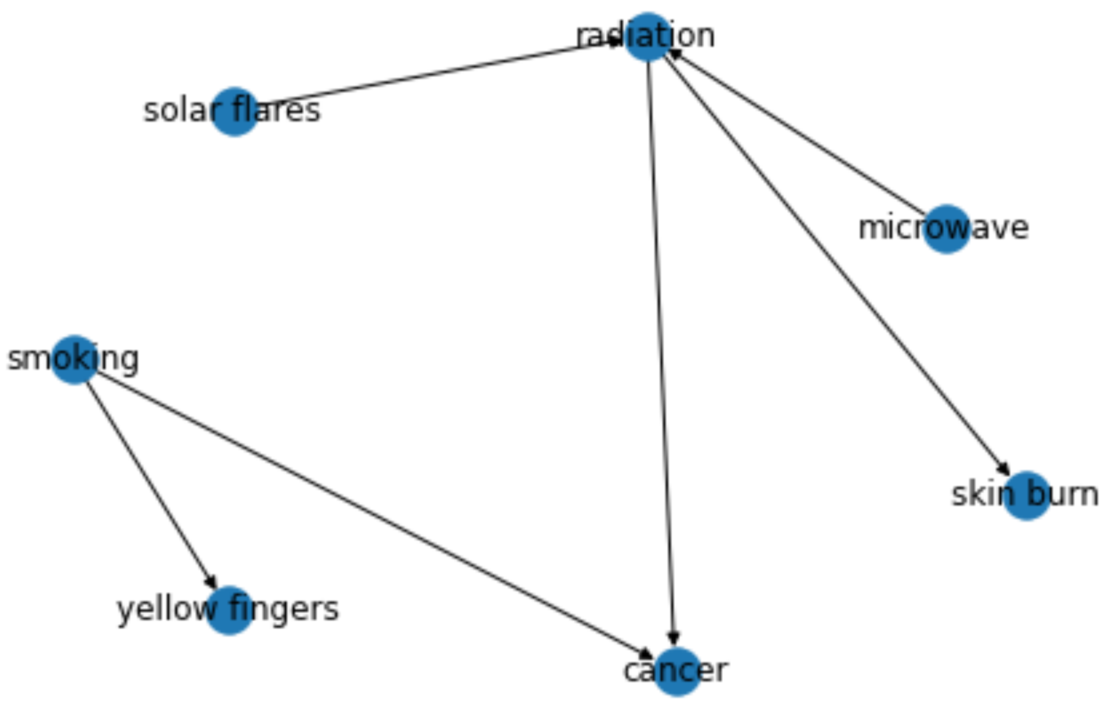
Variable	M1	M2	M3
0	0.8	0.9	0.7

iii)

$$P(O1 = 1|M1) \times P(O2 = 1|M1) \times P(O3 = 3|M1) = 0.05 \times 0.05 \times 0.075 = 1.875 \times 10^{-4}$$
$$P(O1 = 1|M2) \times P(O2 = 1|M2) \times P(O3 = 3|M2) = 0.01 \times 0.01 \times 0.07 = 7 \times 10^{-6}$$
$$P(O1 = 1|M3) \times P(O2 = 1|M3) \times P(O3 = 3|M3) = 0.1 \times 0.1 \times 0.1 = 1 \times 10^{-3}$$

It is most likely to be from machine M3.

Question 3



1.

2. Radiation = 0: 0.3757
   Radiation = 1: 0.6243

```
print(q3_infer.query(variables=[RADIATION], evidence={CANCER: 1}))
```
✓ 0.1s Python

```
Eliminating: microwave: 100%|██████████| 3/3 [00:00<00:00, 458.59it/s]

+--------------+------------------+
| radiation    |   phi(radiation) |
+==============+==================+
| radiation(0) |           0.3757 |
+--------------+------------------+
| radiation(1) |           0.6243 |
+--------------+------------------+
```

3. Cancer = 0: 0.5178
   Cancer = 1: 0.4822

```
print(q3_infer.query(variables=[CANCER], evidence={SKIN_BURN: 1}))
```
✓ 0.1s Python

```
Finding Elimination Order: : 100%|██████████| 3/3 [00:00<00:00, 28.52it/s]
Eliminating: radiation: 100%|██████████| 4/4 [00:00<00:00, 346.29it/s]

+-----------+---------------+
| cancer    |   phi(cancer) |
+===========+===============+
| cancer(0) |        0.5178 |
+-----------+---------------+
| cancer(1) |        0.4822 |
+-----------+---------------+
```

4. Solar Flares is currently not independent of Cancer as there exists an active path between them.

5. The probability is 0.3754.

```
print(q3_infer.query(variables=[CANCER], evidence={MICROWAVE: 0}))
```
✓ 0.1s Python

```
Eliminating: radiation: 100%|██████████| 3/3 [00:00<00:00, 304.32it/s]

+-----------+---------------+
| cancer    |   phi(cancer) |
+===========+===============+
| cancer(0) |        0.6246 |
+-----------+---------------+
| cancer(1) |        0.3754 |
+-----------+---------------+
```

Question 4

a)

```
## download and load data
from keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

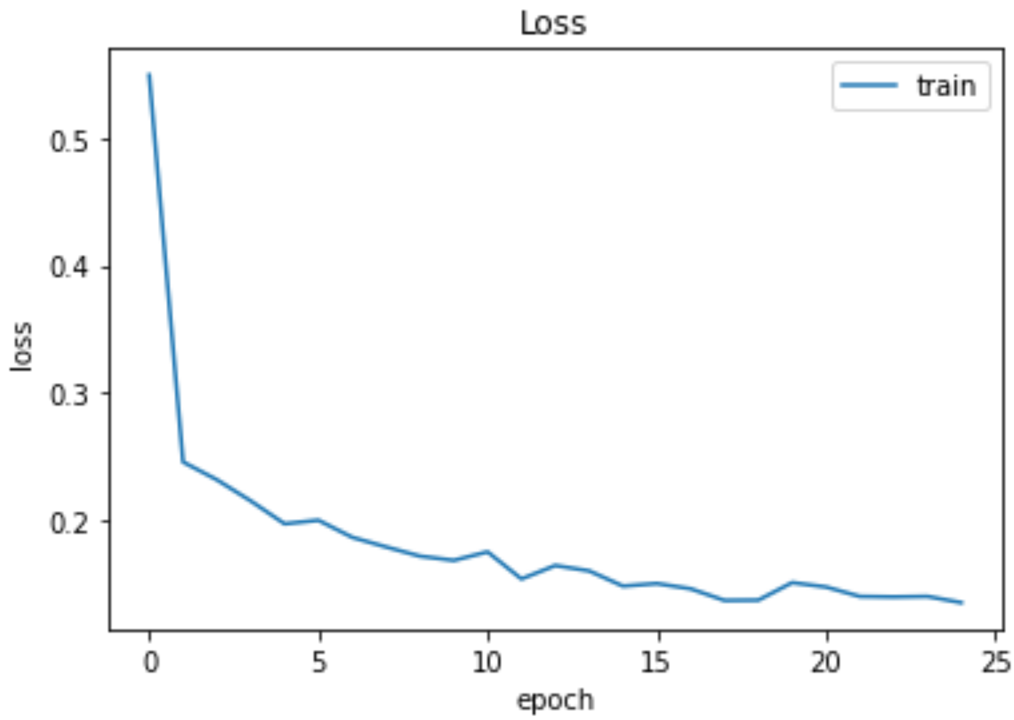
b)

```python
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)



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)

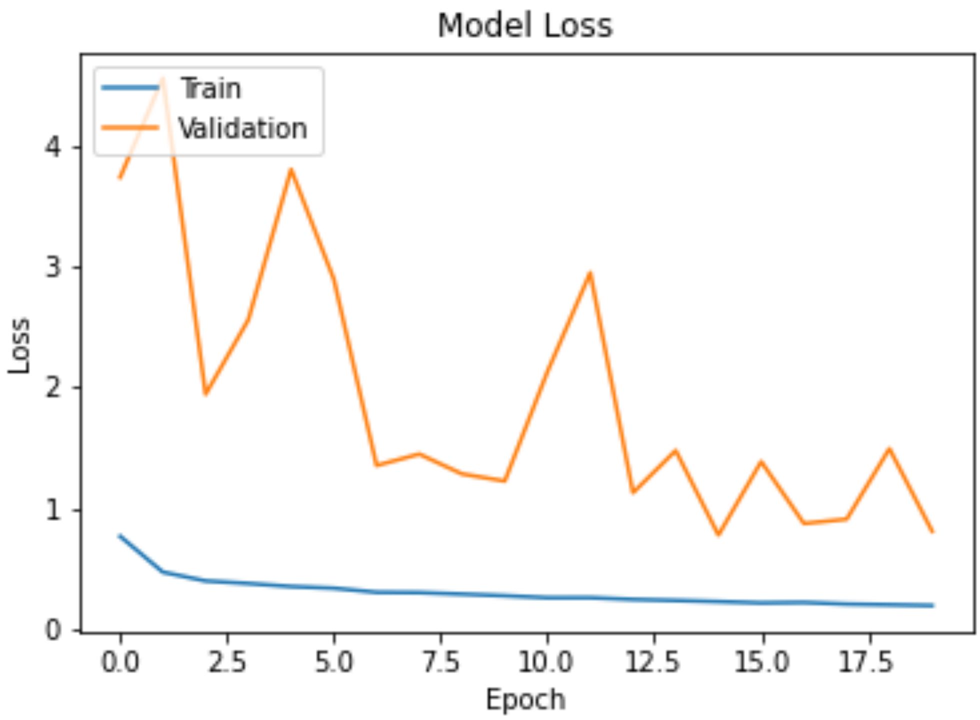
```
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
checkpoint_path = f"{ROOT_FOLDER}cp.ckpt"

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)



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%