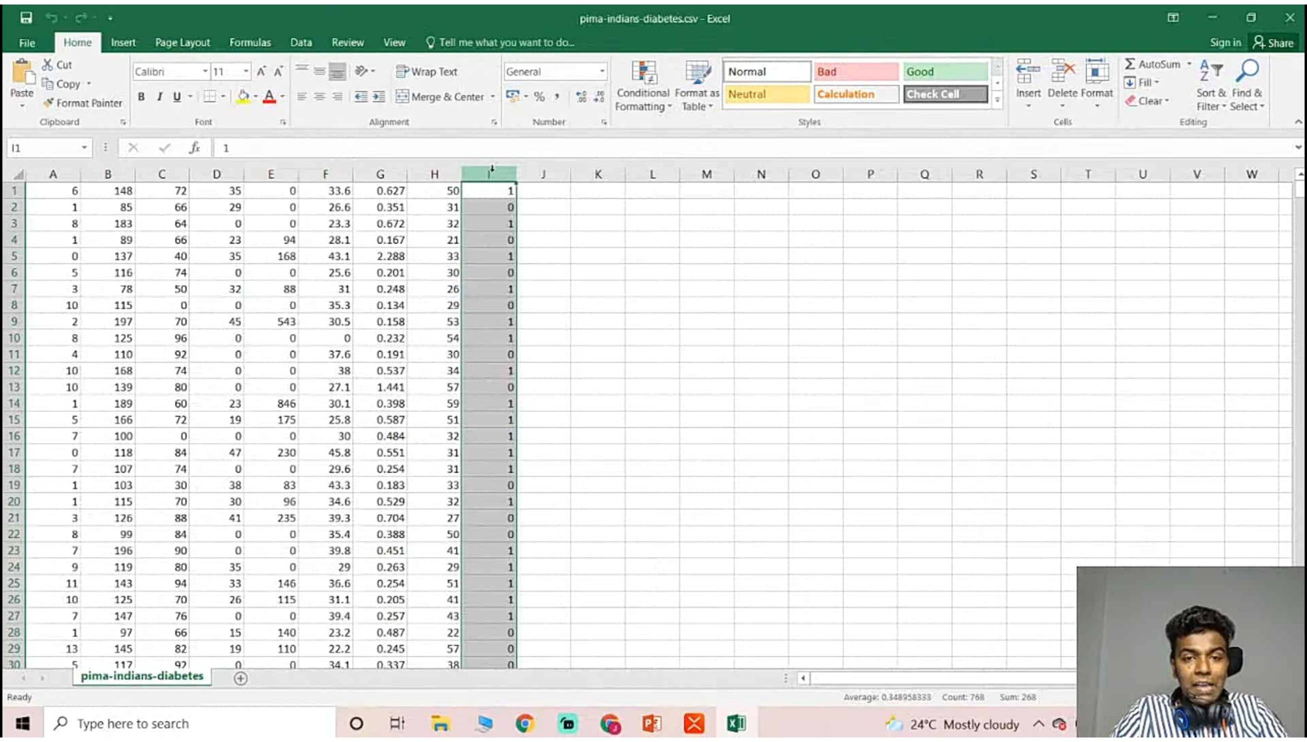
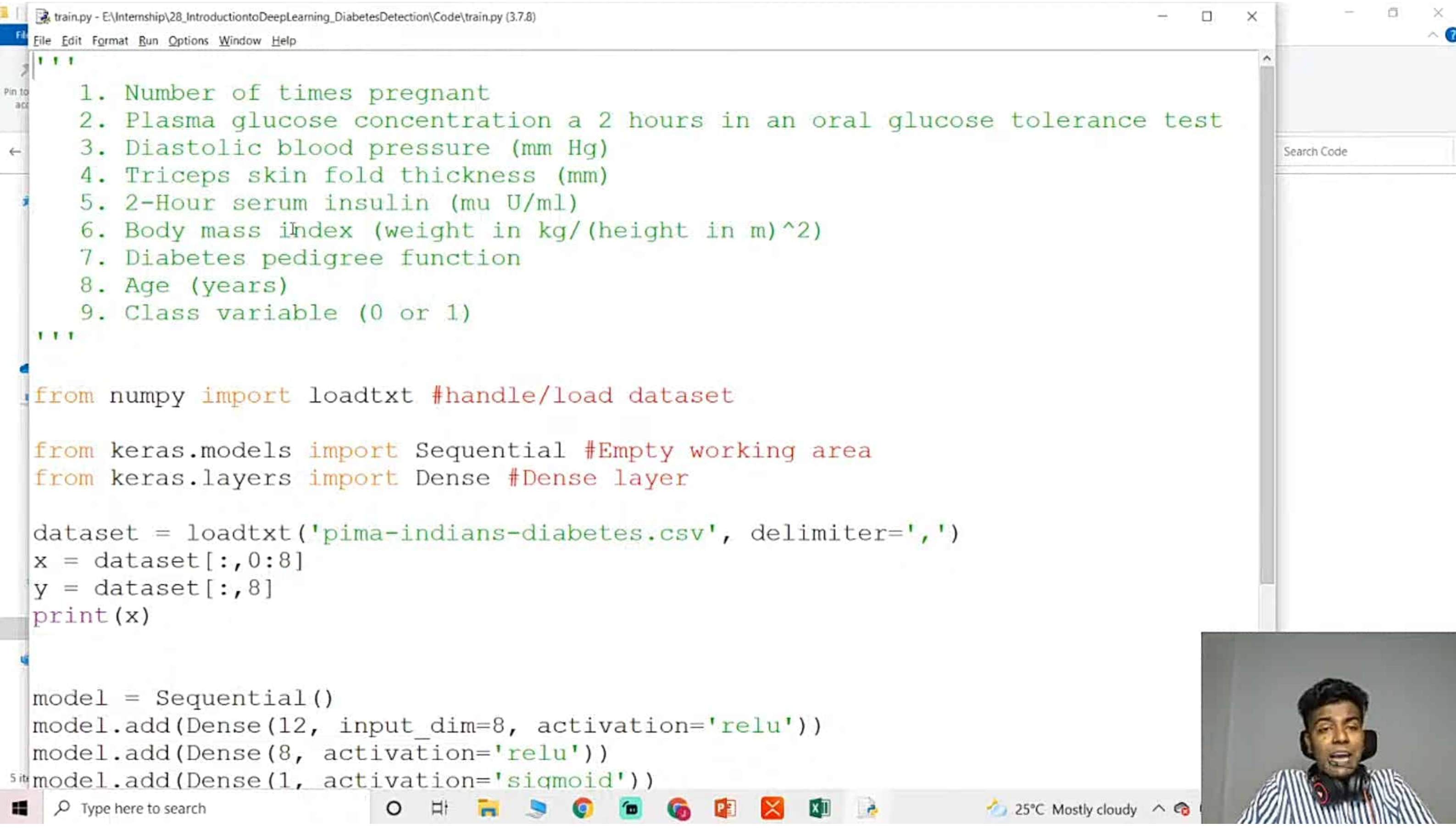
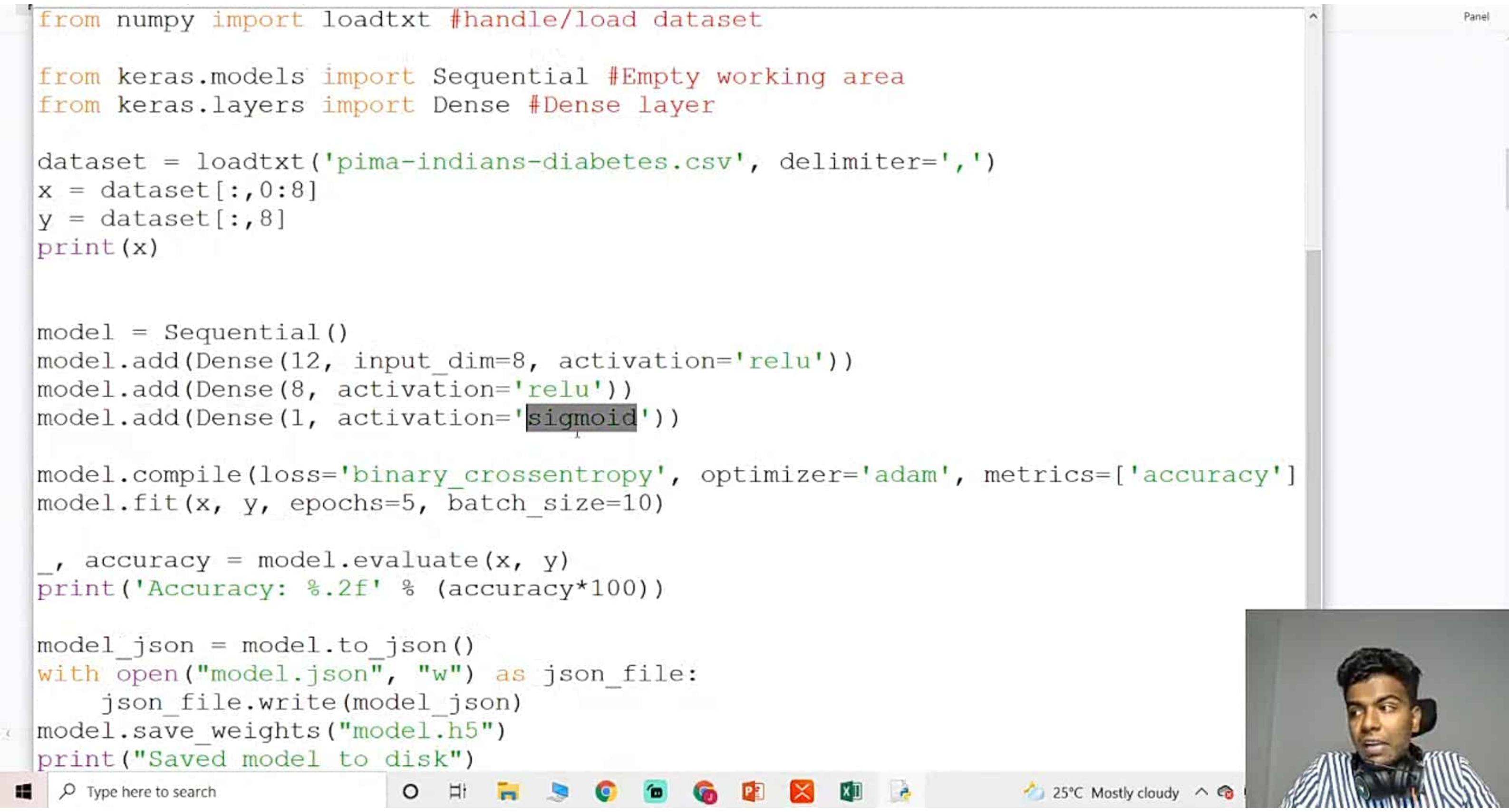


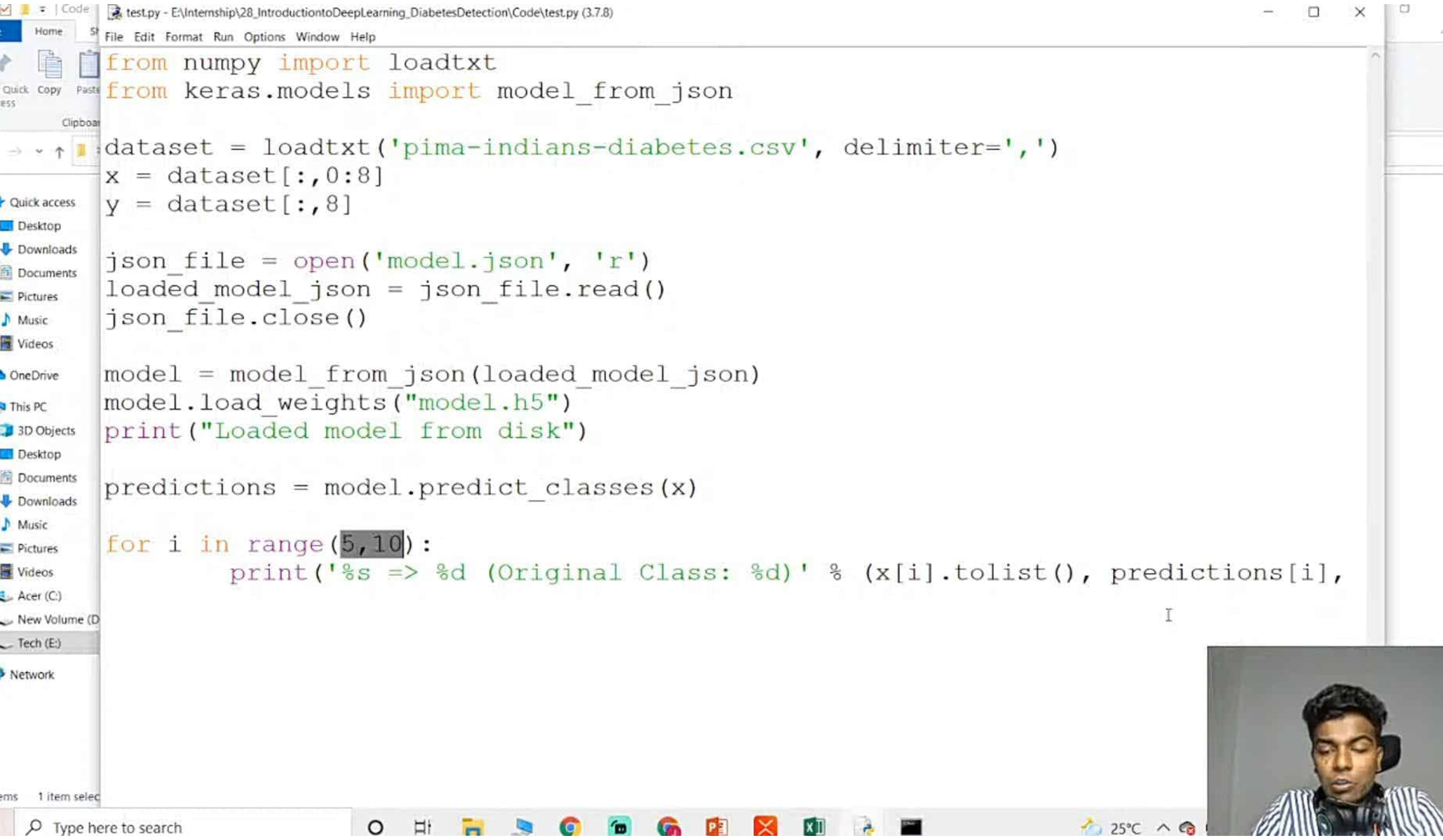
| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 | | | | | | | | | | | | | | |
| 2 | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 | | | | | | | | | | | | | | |
| 3 | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 | | | | | | | | | | | | | | |
| 4 | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 | | | | | | | | | | | | | | |
| 5 | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 | | | | | | | | | | | | | | |
| 6 | 5 | 116 | 74 | 0 | 0 | 25.6 | 0.201 | 30 | 0 | | | | | | | | | | | | | | |
| 7 | 3 | 78 | 50 | 32 | 88 | 31 | 0.248 | 26 | 1 | | | | | | | | | | | | | | |
| 8 | 10 | 115 | 0 | 0 | 0 | 35.3 | 0.134 | 29 | 0 | | | | | | | | | | | | | | |
| 9 | 2 | 197 | 70 | 45 | 543 | 30.5 | 0.158 | 53 | 1 | | | | | | | | | | | | | | |
| 10 | 8 | 125 | 96 | 0 | 0 | 0 | 0.232 | 54 | 1 | | | | | | | | | | | | | | |
| 11 | 4 | 110 | 92 | 0 | 0 | 37.6 | 0.191 | 30 | 0 | | | | | | | | | | | | | | |
| 12 | 10 | 168 | 74 | 0 | 0 | 38 | 0.537 | 34 | 1 | | | | | | | | | | | | | | |
| 13 | 10 | 139 | 80 | 0 | 0 | 27.1 | 1.441 | 57 | 0 | | | | | | | | | | | | | | |
| 14 | 1 | 189 | 60 | 23 | 846 | 30.1 | 0.398 | 59 | 1 | | | | | | | | | | | | | | |
| 15 | 5 | 166 | 72 | 19 | 175 | 25.8 | 0.587 | 51 | 1 | | | | | | | | | | | | | | |
| 16 | 7 | 100 | 0 | 0 | 0 | 30 | 0.484 | 32 | 1 | | | | | | | | | | | | | | |
| 17 | 0 | 118 | 84 | 47 | 230 | 45.8 | 0.551 | 31 | 1 | | | | | | | | | | | | | | |
| 18 | 7 | 107 | 74 | 0 | 0 | 29.6 | 0.254 | 31 | 1 | | | | | | | | | | | | | | |
| 19 | 1 | 103 | 30 | 38 | 83 | 43.3 | 0.183 | 33 | 0 | | | | | | | | | | | | | | |
| 20 | 1 | 115 | 70 | 30 | 96 | 34.6 | 0.529 | 32 | 1 | | | | | | | | | | | | | | |
| 21 | 3 | 126 | 88 | 41 | 235 | 39.3 | 0.704 | 27 | 0 | | | | | | | | | | | | | | |
| 22 | 8 | 99 | 84 | 0 | 0 | 35.4 | 0.388 | 50 | 0 | | | | | | | | | | | | | | |
| 23 | 7 | 196 | 90 | 0 | 0 | 39.8 | 0.451 | 41 | 1 | | | | | | | | | | | | | | |
| 24 | 9 | 119 | 80 | 35 | 0 | 29 | 0.263 | 29 | 1 | | | | | | | | | | | | | | |
| 25 | 11 | 143 | 94 | 33 | 146 | 36.6 | 0.254 | 51 | 1 | | | | | | | | | | | | | | |
| 26 | 10 | 125 | 70 | 26 | 115 | 31.1 | 0.205 | 41 | 1 | | | | | | | | | | | | | | |
| 27 | 7 | 147 | 76 | 0 | 0 | 39.4 | 0.257 | 43 | 1 | | | | | | | | | | | | | | |
| 28 | 1 | 97 | 66 | 15 | 140 | 23.2 | 0.487 | 22 | 0 | | | | | | | | | | | | | | |
| 29 | 13 | 145 | 82 | 19 | 110 | 22.2 | 0.245 | 57 | 0 | | | | | | | | | | | | | | |
| 30 | 5 | 117 | 92 | 0 | 0 | 34.1 | 0.337 | 38 | 0 | | | | | | | | | | | | | | |


train.py - E:\Internship\28_IntroductiontoDeepLearning_DiabetesDetection\Code\train.py (3.7.8)
File Edit Format Run Options Window Help
Search Code
Type here to search
25°C Mostly cloudy

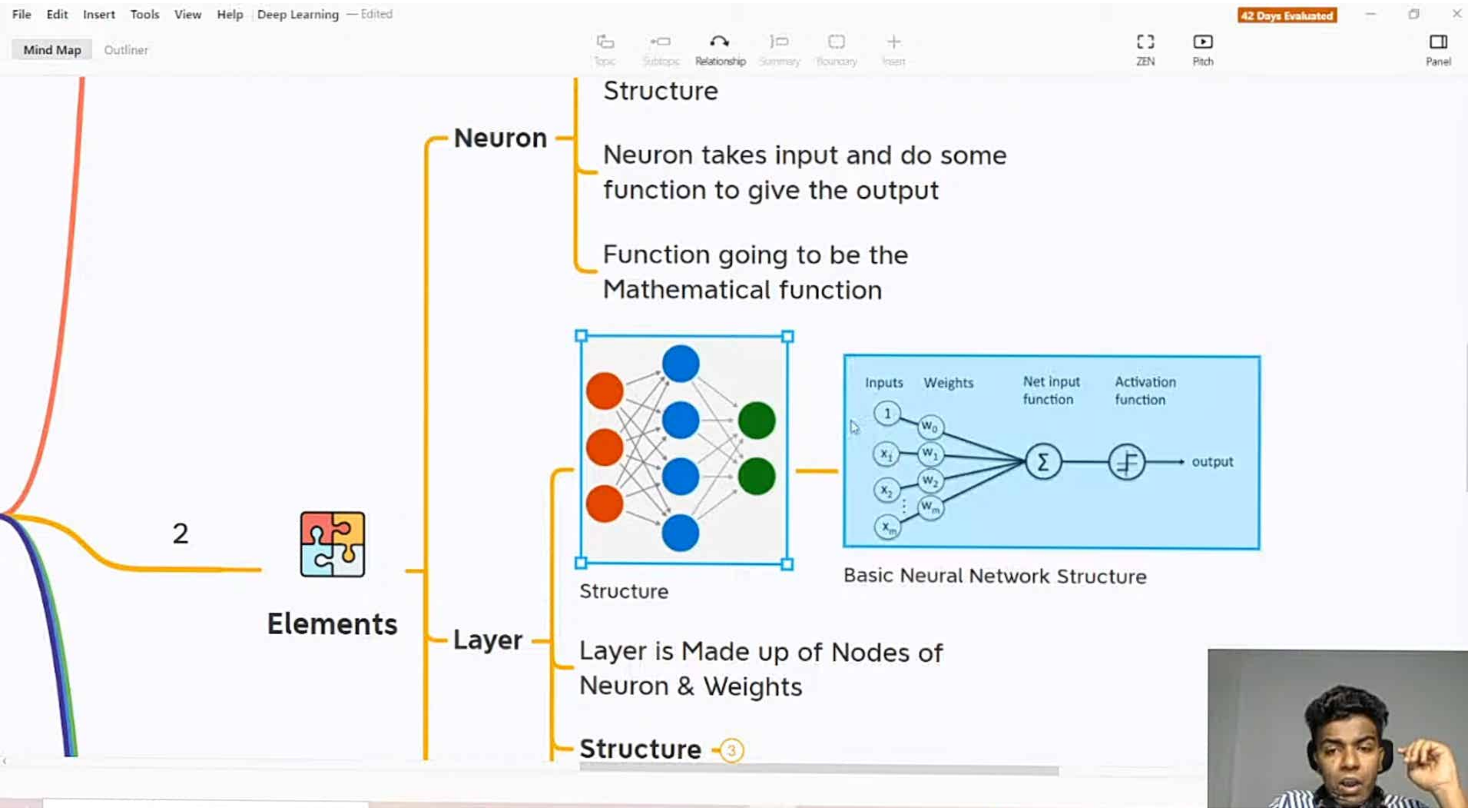
```python
'''
   1. Number of times pregnant
   2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
   3. Diastolic blood pressure (mm Hg)
   4. Triceps skin fold thickness (mm)
   5. 2-Hour serum insulin (mu U/ml)
   6. Body mass index (weight in kg/(height in m)^2)
   7. Diabetes pedigree function
   8. Age (years)
   9. Class variable (0 or 1)
'''

from numpy import loadtxt #handle/load dataset

from keras.models import Sequential #Empty working area
from keras.layers import Dense #Dense layer

dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')
x = dataset[:,0:8]
y = dataset[:,8]
print(x)


model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```



```
from numpy import loadtxt #handle/load dataset

from keras.models import Sequential #Empty working area
from keras.layers import Dense #Dense layer

dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')
x = dataset[:,0:8]
y = dataset[:,8]
print(x)


model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy']
model.fit(x, y, epochs=5, batch_size=10)

_, accuracy = model.evaluate(x, y)
print('Accuracy: %.2f' % (accuracy*100))

model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
model.save_weights("model.h5")
print("Saved model to disk")
```



```
from numpy import loadtxt
from keras.models import model_from_json

dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')
x = dataset[:,0:8]
y = dataset[:,8]

json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()

model = model_from_json(loaded_model_json)
model.load_weights("model.h5")
print("Loaded model from disk")

predictions = model.predict_classes(x)

for i in range(5,10):
        print('%s => %d (Original Class: %d)' % (x[i].tolist(), predictions[i],
```



2

Elements

Neuron

Structure

Neuron takes input and do some function to give the output

Function going to be the Mathematical function

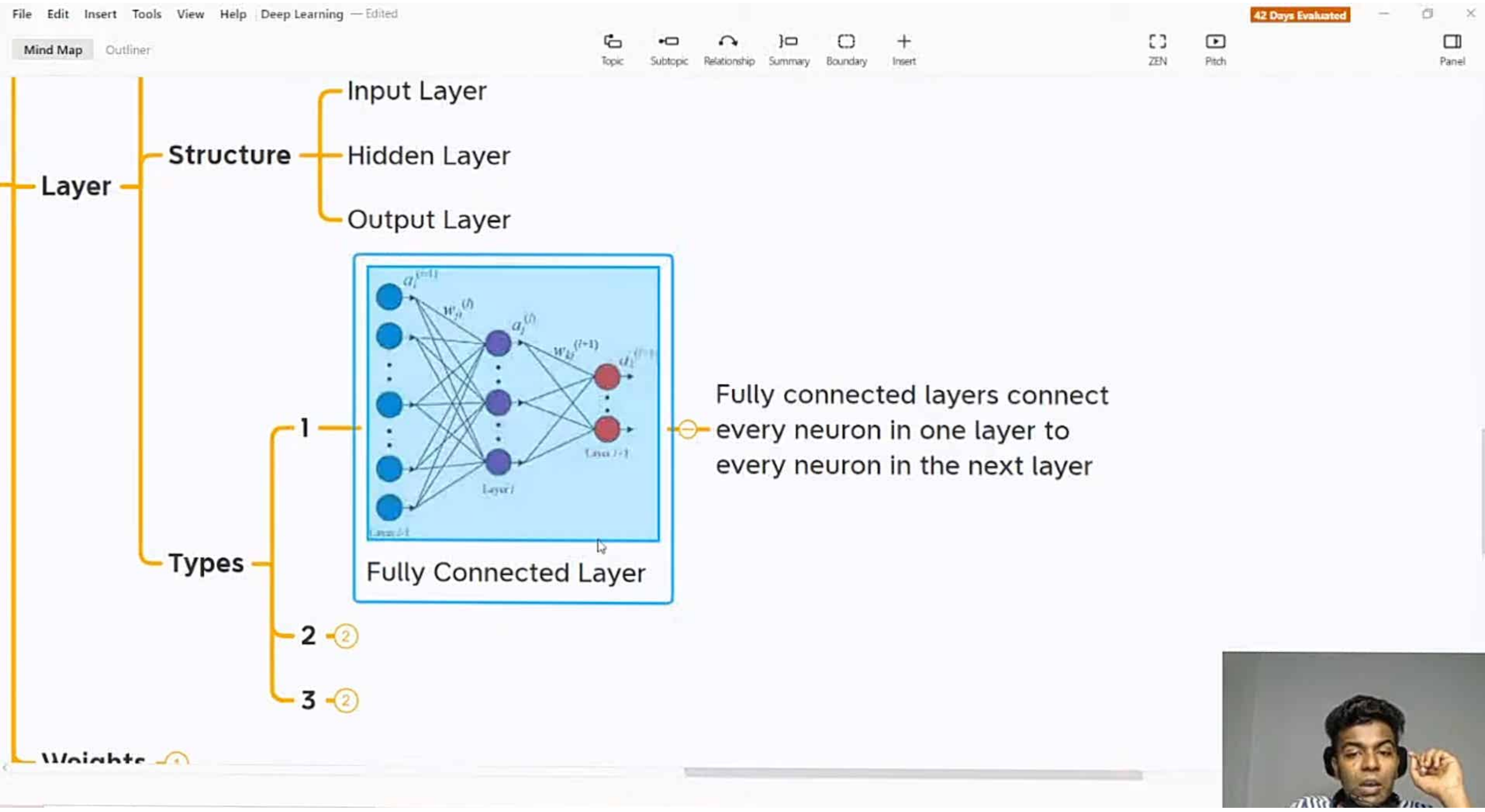
Layer

Structure

Basic Neural Network Structure

Layer is Made up of Nodes of Neuron & Weights

Structure



Layer

Structure

- Input Layer
- Hidden Layer
- Output Layer

Types

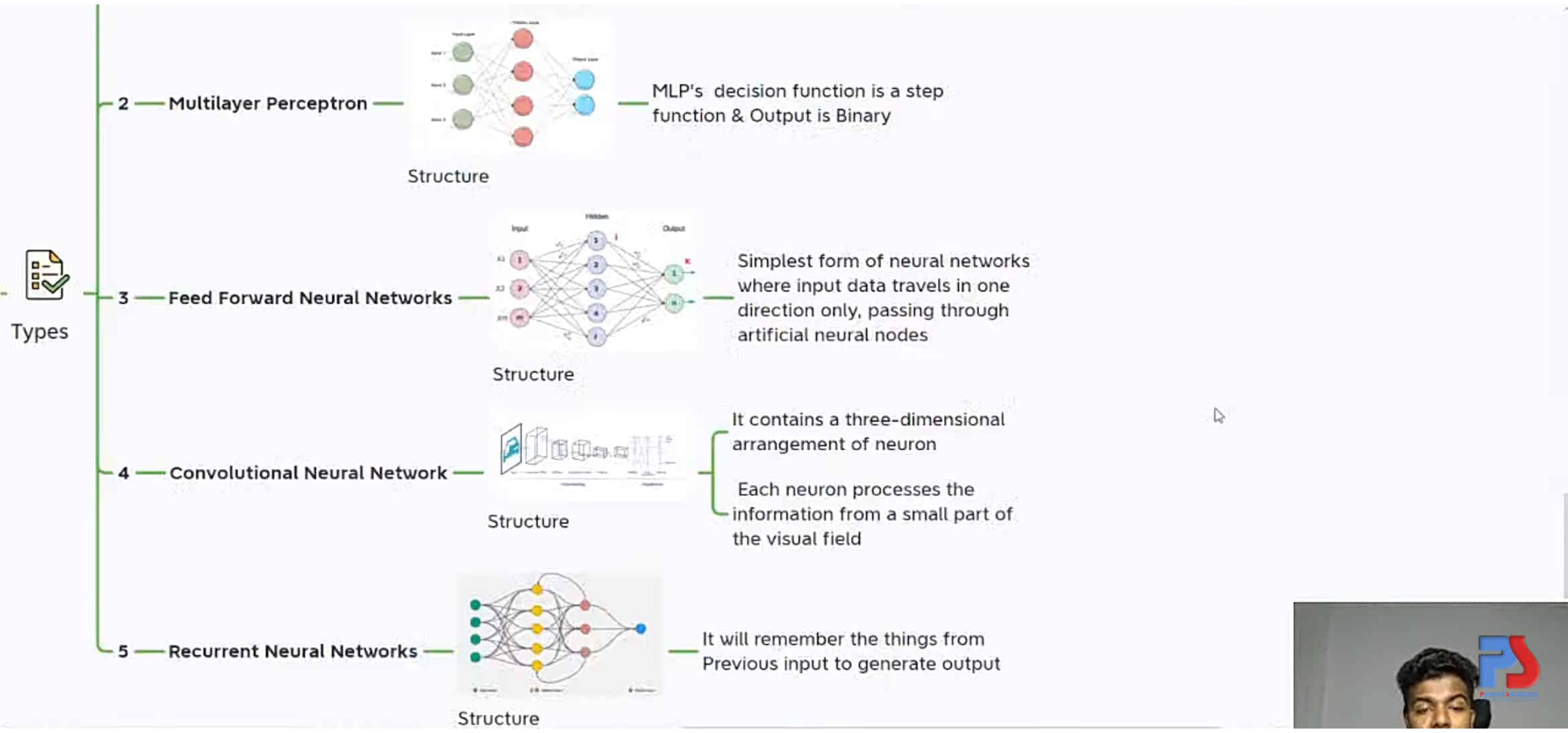
1 Fully Connected Layer

Fully connected layers connect every neuron in one layer to every neuron in the next layer

2 (2)

3 (2)

Weights



Types

2 Multilayer Perceptron

Structure

MLP's  decision function is a step function & Output is Binary

3 Feed Forward Neural Networks

Structure

Simplest form of neural networks where input data travels in one direction only, passing through artificial neural nodes

4 Convolutional Neural Network

Structure

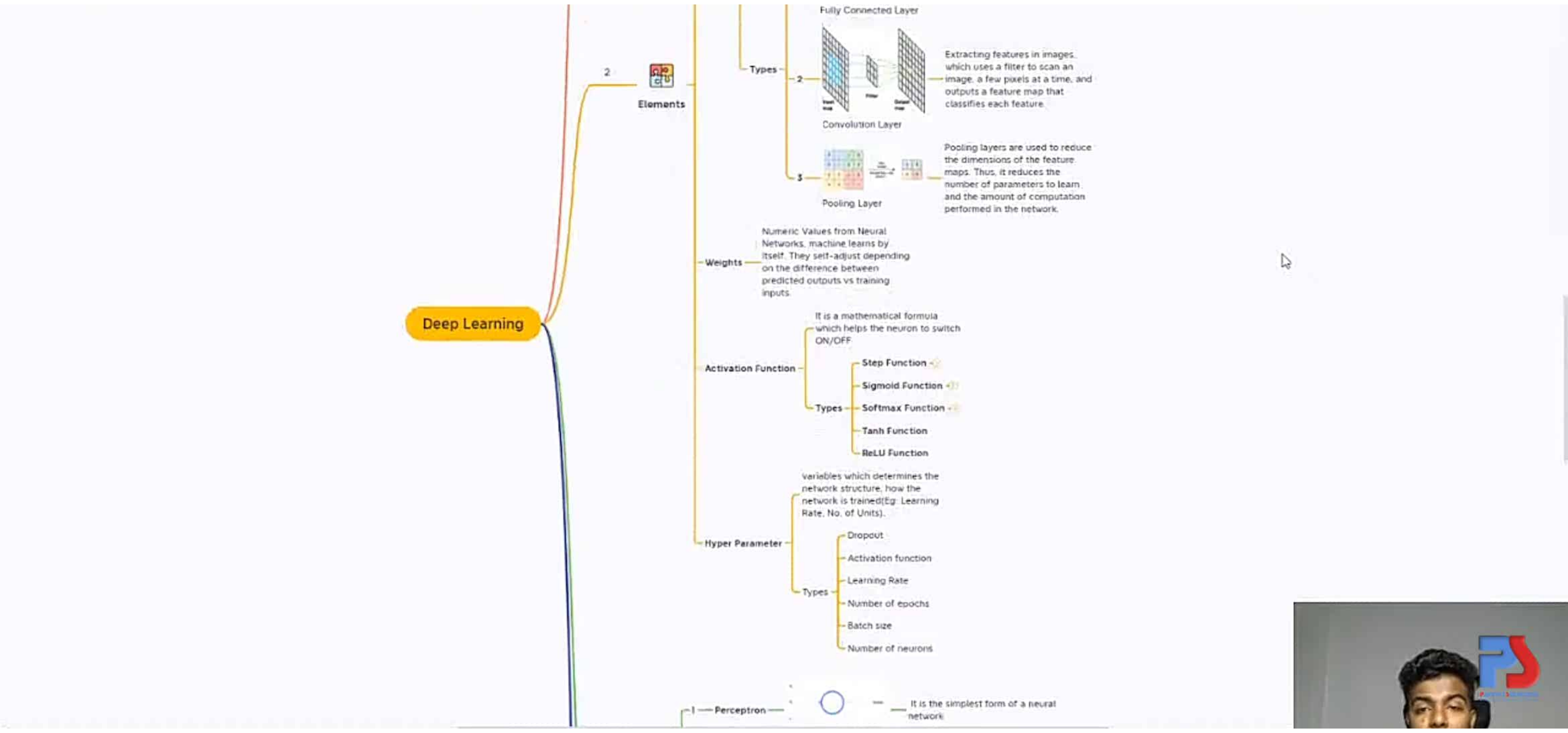
It contains a three-dimensional arrangement of neuron

Each neuron processes the information from a small part of the visual field

5 Recurrent Neural Networks

Structure

It will remember the things from Previous input to generate output



Deep Learning

2 Elements

Types

Fully Connected Layer

2 Convolution Layer

Extracting features in images, which uses a filter to scan an image, a few pixels at a time, and outputs a feature map that classifies each feature.

3 Pooling Layer

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.

Weights

Numeric Values from Neural Networks, machine learns by itself. They self-adjust depending on the difference between predicted outputs vs training inputs.

Activation Function

It is a mathematical formula which helps the neuron to switch ON/OFF

Types

- Step Function
- Sigmoid Function
- Softmax Function
- Tanh Function
- ReLU Function

Hyper Parameter

variables which determines the network structure, how the network is trained(Eg: Learning Rate, No. of Units).

Types

- Dropout
- Activation function
- Learning Rate
- Number of epochs
- Batch size
- Number of neurons

1 Perceptron

It is the simplest form of a neural network