A primeira atividade proposta no bootcamp foi uma introdução ao conceito de machine learning através de dois vídeos bem didáticos e explicativos.

O primeiro vídeo explica de maneira objetiva, através de comparações e analogias, o que é uma rede neural. No vídeo é utilizado como exemplo uma loja de relíquias, e para representar as camadas da rede neural e utilizado um grupo de amigos (que representam os neurônios dessa rede) e eles tem a função de tomar decisões para se definir se um determinado item deve ou não ser comprado. Também é mencionado um conceito importante, que trata da calibração dos pesos das respostas de cada membro (Isto é, aqueles que quase sempre acertam os itens bons com o tempo recebem mais atenção do que os membros que quase nunca acertam). Após isso, é abordado o teorema da aproximação universal, que descreve uma rede de apenas uma camada capaz de tomar as melhores decisões, porém pode conter muitos neurônios ou exigir muito treinamento. Isso nos leva a entender o conceito de rede profunda, que reorganiza os neurônios em diversas camadas, possibilitando avaliar diversos itens com menos neurônios/pessoas e em menos tempo.

Outros dois conceitos importantes ensinados no vídeo são o de erros, que ajudam a otimizar as decisões das redes (BackPropagation), e a utilização de camadas especializadas que ajudam a identificar padrões importantes (Redes Convolucionais).

O segundo vídeo apresenta inicialmente a diferença entre a programação tradicional (que consiste em fornecer regras e dados ao computador para obter respostas) e o machine learning (onde são fornecidos dados e respostas relacionadas a esses dados, permitindo que o computador descubra/aprenda as regras).

Em seguida, é mostrado um algoritmo que contém uma rede neural treinada com apenas uma camada e um único neurônio, cuja função vai ser adivinhar um determinado Y a partir de um X. O código também inclui uma função de perda e um otimizador, que permite avaliar a precisão da previsão de Y por meio da perda e melhorar a previsão na próxima vez. Além disso, os dados de treinamento são fornecidos no formato de uma matriz (X, Y).

```
import tensorflow as tf
import numpy as np
from tensorflow import keras

model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-2.0, 1.0, 4.0, 7.0, 10.0, 13.0], dtype=float)

model.fit(xs, ys, epochs=500)
print(model.predict([10.0]))
```

No exemplo acima, a função que a rede irá prever é Y = 3X + 1. Foram realizados dois testes para prever o valor de Y quando X é 10, um com 500 iterações e outro com 300 iterações, resultando em:

Para 300 iterações: Y = 31.023977;

Para 500 iterações: Y = 31.00254.

Com isso, podemos observar duas coisas: sabendo que X é 10, o valor esperado seria 31. No entanto, como as redes neurais lidam com probabilidades, elas chegam a valores próximos dos esperados, mas não necessariamente ao valor exato. Outra observação importante é a relação de perda entre 300 e 500 iterações, que não foi grande. Por isso, dependendo do problema (se não for necessária uma solução mais exata), seria mais viável realizar apenas as 300 iterações.