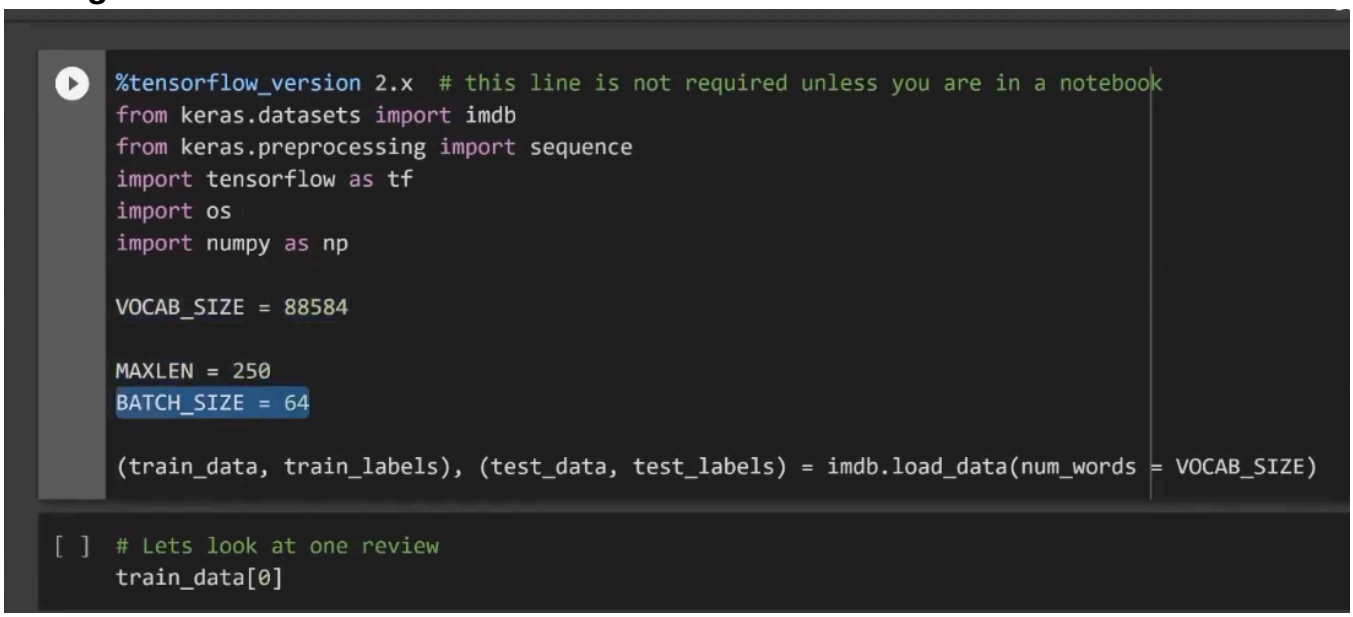
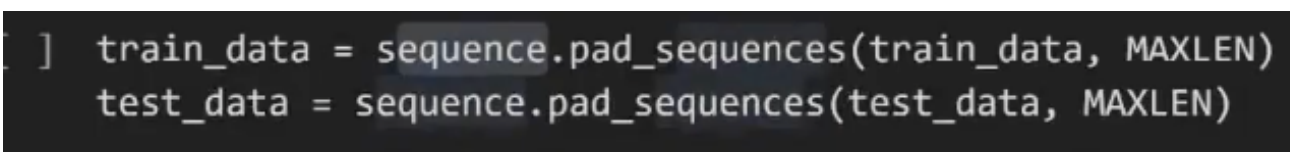
- ***Sentiment analysis of movie reviews to tell if they are positive or negative***
  - ***about the dataset***
    - -> there is a movie review dataset
      - -> the dataset comes from keras and contains 25,000 reviews
    - -> all of the words are encoded by an integer
      - -> so - the more common the word the higher the number
      - -> e.g the 3rd most common word
      - -> there are 80,000 unique words in the dataset - ranked according to which is the most frequent
  - ***importing the dataset***



```
%tensorflow_version 2.x  # this line is not required unless you are in a notebook
from keras.datasets import imdb
from keras.preprocessing import sequence
import tensorflow as tf
import os
import numpy as np

VOCAB_SIZE = 88584

MAXLEN = 250
BATCH_SIZE = 64

(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words = VOCAB_SIZE)
```

```
[ ]  # Lets look at one review
     train_data[0]
```

    - -> defining the vocabulary size, the length of the review, the training data etc
    - -> he's printed out a review example in the second cell -> which is the review in an array with integers
    - -> all of these are unique
      - -> they have different lengths
  - ***more preprocessing <- adding padding to make sure the reviews have the same length***
    - -> all of the reviews have to have the same lengths
    - -> trimming words off of the longer reviews and adding them onto the shorter ones
    - -> this is called adding padding to the reviews
    - -> these extra words are added into the left side of the review

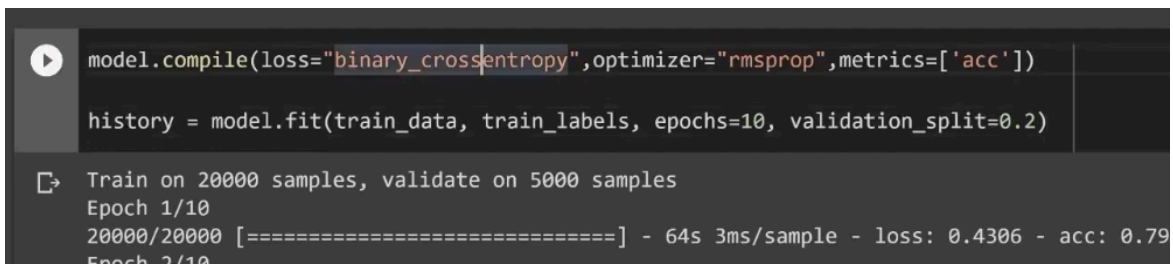


```
[ ]  train_data = sequence.pad_sequences(train_data, MAXLEN)
     test_data = sequence.pad_sequences(test_data, MAXLEN)
```

    - -> assigning test data and train data
    - -> the second argument is the length we want to pad it to
      - -> it's making sure the length of each of those elements is the same
  - ***creating the model***
    - -> a sigmoid function is used to place the reviews between 0 and 1
    - -> if it's above or below 0.5 determines whether it's a good review or not
    - -> 32 because the output is 32 dimensions
      - to long term short term memory layer will have 32 elements
    - -> we have a sequential model

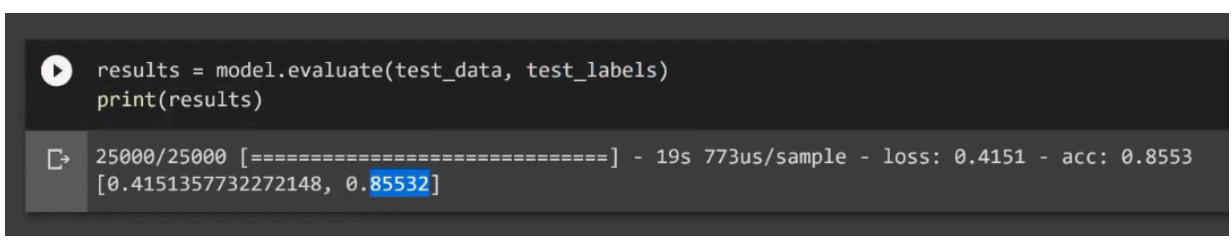
```
[7]  model = tf.keras.Sequential([
         tf.keras.layers.Embedding(VOCAB_SIZE, 32),
         tf.keras.layers.LSTM(32),
         tf.keras.layers.Dense(1, activation="sigmoid")
     ])

     model.summary()
```

- ***training the model***
  - -> changing the runtime type
    - this increases the speed of the training by using a GPU
  - -> picking the model
  - -> a binary cross entropy model because there are two things which we are predicting (the model is good or bad)
  - -> then the optimiser



  - -> rmsprop <- this is the optimiser
    - using the atom optimiser
  - -> then the metrics as acc
  - -> the training data, labels and doing a validation split
    - -> a validation split is the percentage of the data which is used as training data for the model
    - -> in this case the model is overfit -> which means that this number should be increased

- ***results of training the model***



  - -> in other words evaluating the model (returning the accuracy of its predictions)