MICP_3

1.In the code provided there are three mistake which stops the code from running successfully; find those mistakes and explain why they need to be corrected to be able to get the code run.

Erros that I have faces are

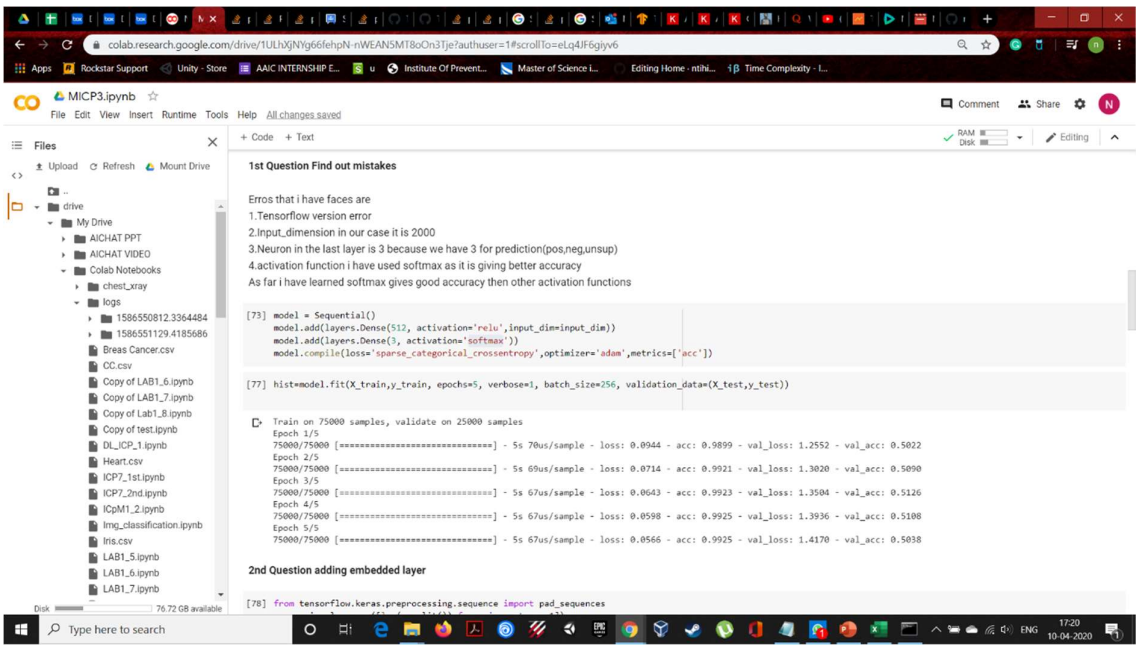
1.Tensorflow version error

2.Input_dimension in our case it is 2000

3.Neuron in the last layer is 3 because we have 3 for prediction (pos, neg, unsup)

4.activation function I have used softmax as it is giving better accuracy

As far I have learned softmax gives good accuracy then other activation functions



2.Add embedding layer to the model, did you experience any improvement?

ANS:

The accuracy got decreased but in the previous model it got overfit but when I have added embedded layer the accuracy and validation accuracy are at same point which means it got underfit.

colab.research.google.com/drive/1ULhXjNYg66fehpN-nWEAN5MT8oOn3Tje?authuser=1#scrollTo=eLq4JF6giyv6

MICP3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

2nd Question adding embedded layer

```
[78] from tensorflow.keras.preprocessing.sequence import pad_sequences
     max_review_len= max([len(s.split()) for s in sentences1])
     vocab_size= len(tokenizer.word_index)+1
     sentences_pre = tokenizer.texts_to_sequences(sentences1)
     padded_docs= pad_sequences(sentences_pre,maxlen=max_review_len)
```

```
[79] #from keras.layers import Embedding
     #model.add(Embedding(vocab_size, 50, input_length=max_review_len)
     X_train, X_test, y_train, y_test = train_test_split(padded_docs, y, test_size=0.25, random_state=1000)
     temp_X=X_test
     temp_y=y_test
```

```
[17] len(padded_docs)
     #max_review_len
     #padded_docs.shape
     #X_train.shape
     #vocab_size
```

```
100000
```

```
[84] #from tensorflow.keras.layers import Embedding
     m = Sequential()
     m.add(Embedding(vocab_size, 50, input_length=max_review_len))
     m.add(Flatten())
     m.add(layers.Dense(300,input_dim=max_review_len, activation='relu'))
     m.add(layers.Dense(3, activation='softmax'))
     #m = Sequential()
     #m.add(layers.Dense(512, activation='relu',input_shape=(max_review_len,)))
     #m.add(Embedding(vocab_size, 50,input_length=len(padded_docs)))
```

colab.research.google.com/drive/1ULhXjNYg66fehpN-nWEAN5MT8oOn3Tje?authuser=1#scrollTo=eLq4JF6giyv6

MICP3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
     #max_review_len
[17] #padded_docs.shape
     #X_train.shape
     #vocab_size
```

```
100000
```

```
[84] #from tensorflow.keras.layers import Embedding
     m = Sequential()
     m.add(Embedding(vocab_size, 50, input_length=max_review_len))
     m.add(Flatten())
     m.add(layers.Dense(300,input_dim=max_review_len, activation='relu'))
     m.add(layers.Dense(3, activation='softmax'))
     #m = Sequential()
     #m.add(layers.Dense(512, activation='relu',input_shape=(max_review_len,)))
     #m.add(Embedding(vocab_size, 50,input_length=len(padded_docs)))
     #m.add(layers.Dense(512, activation='relu'))
     #m.add(layers.Dense(3, activation='softmax'))
     m.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['acc'])
     #h=m.fit(X_train,y_train, epochs=5, verbose=1, batch_size=256, validation_data=(X_test,y_test))
```

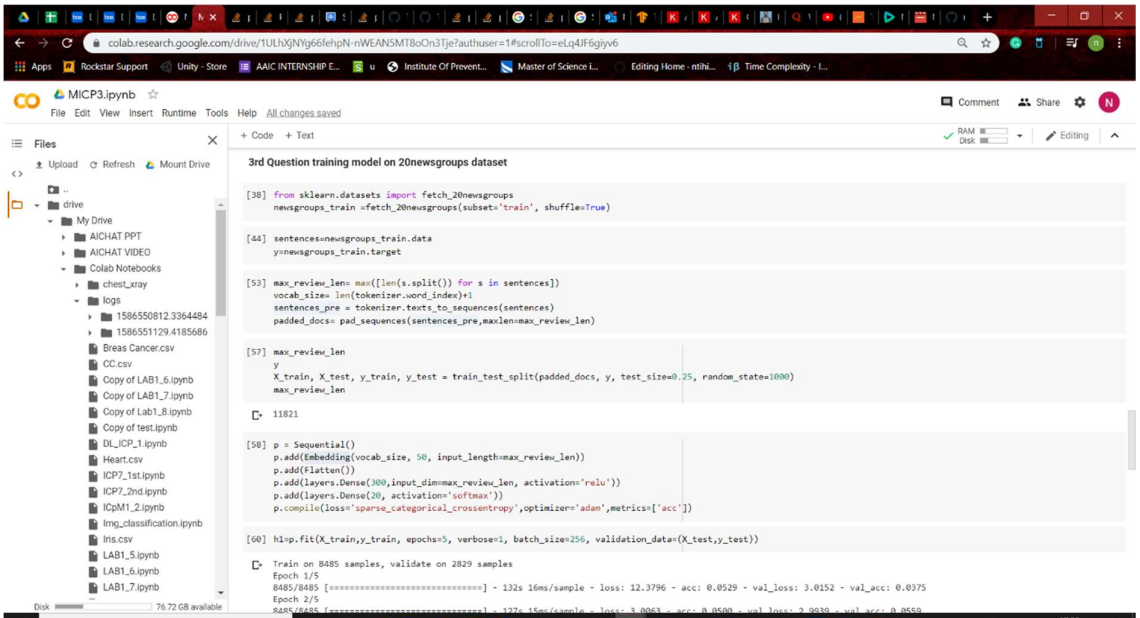
```
[108] m.fit(X_train,y_train,batch_size=256,epochs=5,verbose=1,validation_data=(X_test,y_test))
```

```
Train on 75000 samples, validate on 25000 samples
Epoch 1/5
75000/75000 [==============================] - 233s 3ms/sample - loss: 0.8886 - acc: 0.5116 - val_loss: 0.6730 - val_acc: 0.6254
Epoch 2/5
75000/75000 [==============================] - 228s 3ms/sample - loss: 0.7859 - acc: 0.5497 - val_loss: 0.5682 - val_acc: 0.7039
Epoch 3/5
75000/75000 [==============================] - 228s 3ms/sample - loss: 0.7294 - acc: 0.5977 - val_loss: 0.5154 - val_acc: 0.7777
Epoch 4/5
75000/75000 [==============================] - 229s 3ms/sample - loss: 0.6495 - acc: 0.6623 - val_loss: 0.4957 - val_acc: 0.7855
Epoch 5/5
75000/75000 [==============================] - 221s 3ms/sample - loss: 0.5317 - acc: 0.7464 - val_loss: 0.4182 - val_acc: 0.8173
<tensorflow.python.keras.callbacks.History at 0x7f31dcf52588>
```

3.Apply the code on 20_newsgroup data set we worked in the previous classes

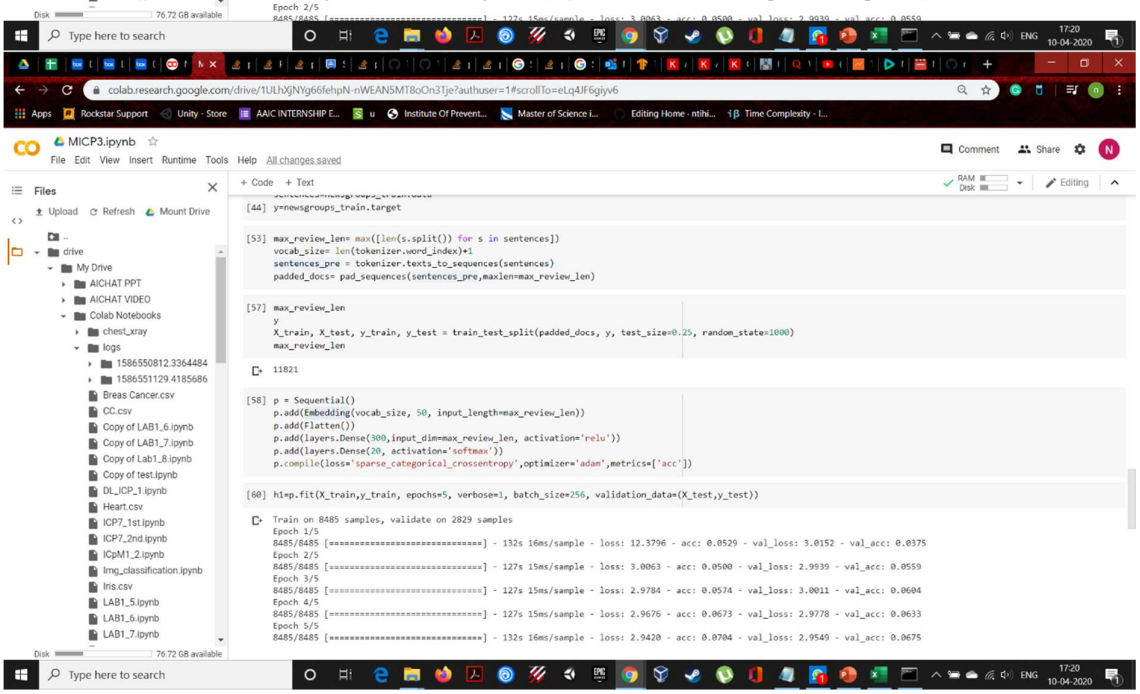
From sklearn. datasets import fetch_20newsgroups

newsgroups_train =fetch_20newsgroups (subset='train', shuffle=True, categories = categories,)



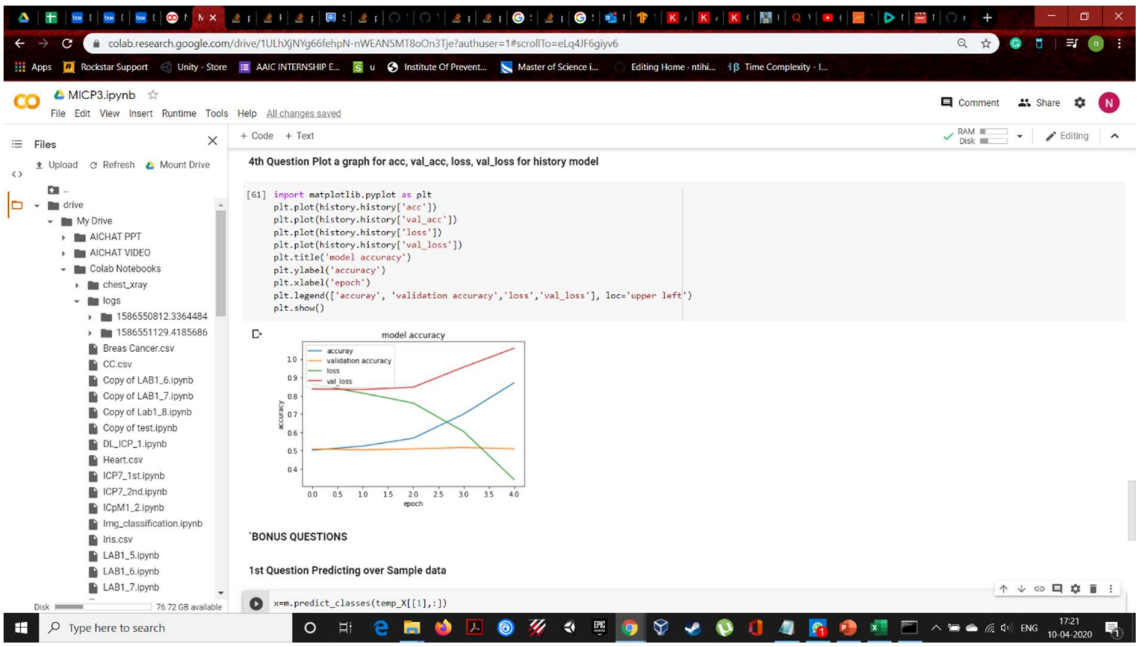
3rd Question training model on 20newsgroups dataset

```
[38] from sklearn.datasets import fetch_20newsgroups
     newsgroups_train =fetch_20newsgroups(subset='train', shuffle=True)

[44] sentences=newsgroups_train.data
     y=newsgroups_train.target

[53] max_review_len= max([len(s.split()) for s in sentences])
     vocab_size= len(tokenizer.word_index)+1
     sentences_pre = tokenizer.texts_to_sequences(sentences)
     padded_docs= pad_sequences(sentences_pre,maxlen=max_review_len)

[57] max_review_len
     y
     X_train, X_test, y_train, y_test = train_test_split(padded_docs, y, test_size=0.25, random_state=1000)
     max_review_len

     11821

[58] p = Sequential()
     p.add(Embedding(vocab_size, 50, input_length=max_review_len))
     p.add(Flatten())
     p.add(layers.Dense(300,input_dim=max_review_len, activation='relu'))
     p.add(layers.Dense(20, activation='softmax'))
     p.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['acc'])

[60] h1=p.fit(X_train,y_train, epochs=5, verbose=1, batch_size=256, validation_data=(X_test,y_test))

     Train on 8485 samples, validate on 2829 samples
     Epoch 1/5
     8485/8485 [==============================] - 132s 16ms/sample - loss: 12.3796 - acc: 0.0529 - val_loss: 3.0152 - val_acc: 0.0375
```



```
[44] y=newsgroups_train.target

[53] max_review_len= max([len(s.split()) for s in sentences])
     vocab_size= len(tokenizer.word_index)+1
     sentences_pre = tokenizer.texts_to_sequences(sentences)
     padded_docs= pad_sequences(sentences_pre,maxlen=max_review_len)

[57] max_review_len
     y
     X_train, X_test, y_train, y_test = train_test_split(padded_docs, y, test_size=0.25, random_state=1000)
     max_review_len

     11821

[58] p = Sequential()
     p.add(Embedding(vocab_size, 50, input_length=max_review_len))
     p.add(Flatten())
     p.add(layers.Dense(300,input_dim=max_review_len, activation='relu'))
     p.add(layers.Dense(20, activation='softmax'))
     p.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['acc'])

[60] h1=p.fit(X_train,y_train, epochs=5, verbose=1, batch_size=256, validation_data=(X_test,y_test))

     Train on 8485 samples, validate on 2829 samples
     Epoch 1/5
     8485/8485 [==============================] - 132s 16ms/sample - loss: 12.3796 - acc: 0.0529 - val_loss: 3.0152 - val_acc: 0.0375
     Epoch 2/5
     8485/8485 [==============================] - 127s 15ms/sample - loss: 3.0063 - acc: 0.0500 - val_loss: 2.9939 - val_acc: 0.0559
     Epoch 3/5
     8485/8485 [==============================] - 127s 15ms/sample - loss: 2.9784 - acc: 0.0574 - val_loss: 3.0011 - val_acc: 0.0604
     Epoch 4/5
     8485/8485 [==============================] - 127s 15ms/sample - loss: 2.9676 - acc: 0.0673 - val_loss: 2.9778 - val_acc: 0.0633
     Epoch 5/5
     8485/8485 [==============================] - 132s 16ms/sample - loss: 2.9420 - acc: 0.0704 - val_loss: 2.9549 - val_acc: 0.0675
```

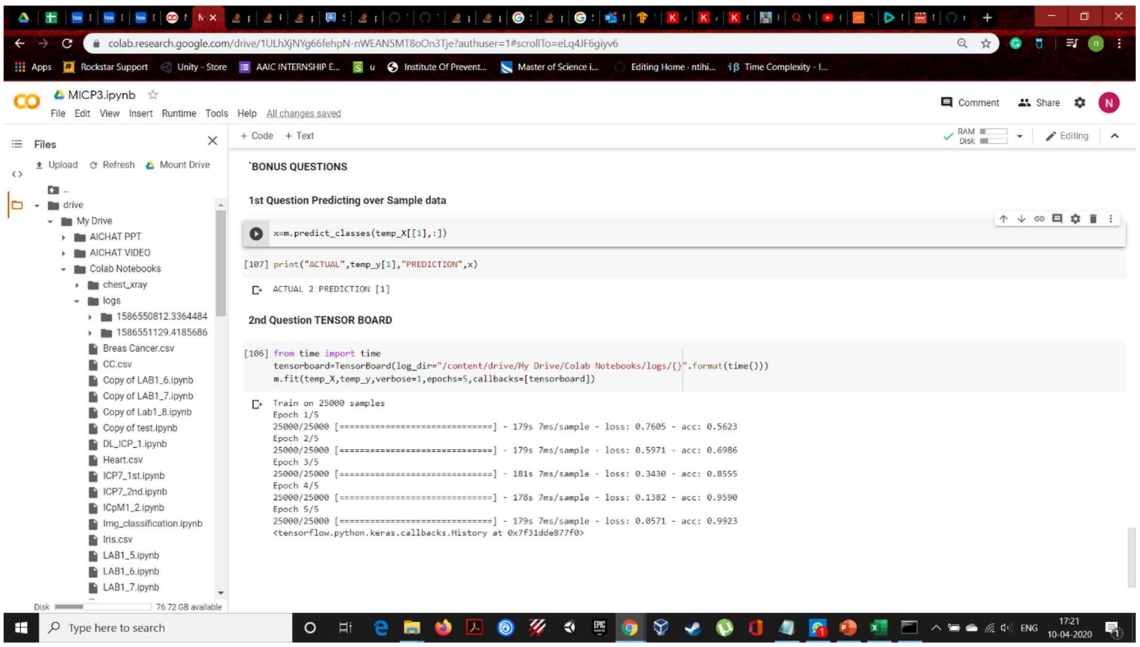
4.Plot the loss and accuracy using history object



*Bonus

Question

1.Predict over one sample of data and check what will be the prediction for that.



2.Plot loss and accuracy in Tensorboard.