Assignment 5

Classifying Movie Reviews using *Feature-based* Machine Learning &
Deep Learning with RNN

1. **Document the parameters you have used when defining all three models, and their respective obtained accuracies.**

Part 1 – Using sci-kit klearn

My code for solving the problem can be seen here, on Kaggle.

```
vector = HashingVectorizer(stop_words="english", binary=True)
```

I use stop_words = "English", to remove English stop words, and binary = True to set all nonzero counts to 1. "This is useful for discrete probabilistic models that model binary events rather than integer counts." – scikit-learn.org

```
'''DecisionTreeClassifier'''
dtc_classifier = DecisionTreeClassifier(criterion="entropy")
```

I used the parameter criterion = "entropy", because I wanted *information gain* instead of *Gini impurity.* The results are similar, and I chose it since it's the one we have used in previous assignments. Entropy takes a little longer to compute, because of the logarithm.

$$Gini : Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

$$Entropy : H(E) = -\sum_{j=1}^{c} p_j \log p_j$$

Figure 1: Equations for Entropy and Gini impurity

```
'''Bernulli Naive Bayes'''
bnb_classifier = BernoulliNB()
```

Nothing to comment here.

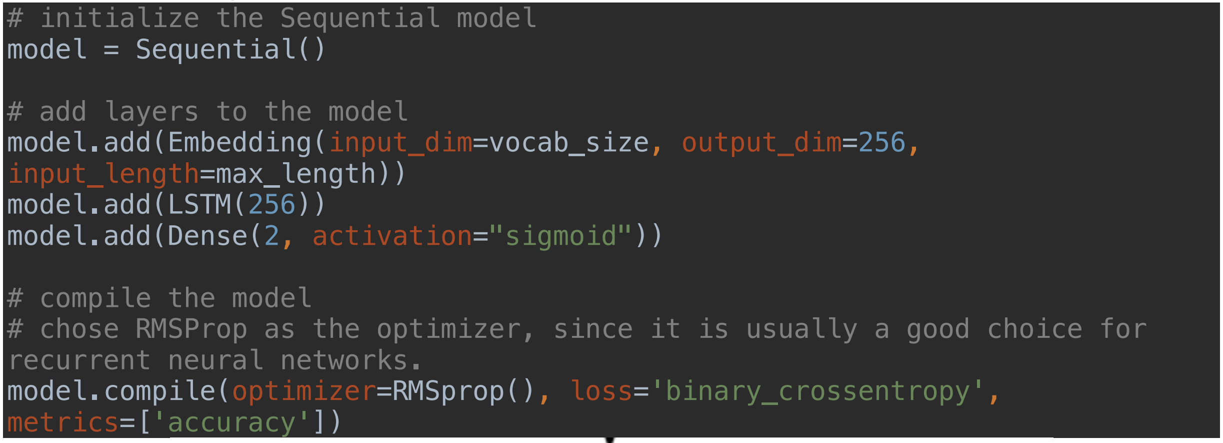
```
DecisionTreeClassifier    --> The accuracy for this classifier:
0.8645118288796274
BernoulliNB               -->  The accuracy for this classifier:
0.8380500735474381
```

As we can see here, the Decision Tree Classifier and Bernoulli NB achieves **86.45**% and **83.80**% **accuracy**, respectively.

Part 2 – Using Keras

My code for solving the problem can be seen here, on Kaggle.

Built the following neural network:



```
# initialize the Sequential model
model = Sequential()

# add layers to the model
model.add(Embedding(input_dim=vocab_size, output_dim=256,
input_length=max_length))
model.add(LSTM(256))
model.add(Dense(2, activation="sigmoid"))

# compile the model
# chose RMSProp as the optimizer, since it is usually a good choice for
recurrent neural networks.
model.compile(optimizer=RMSprop(), loss='binary_crossentropy',
metrics=['accuracy'])
```

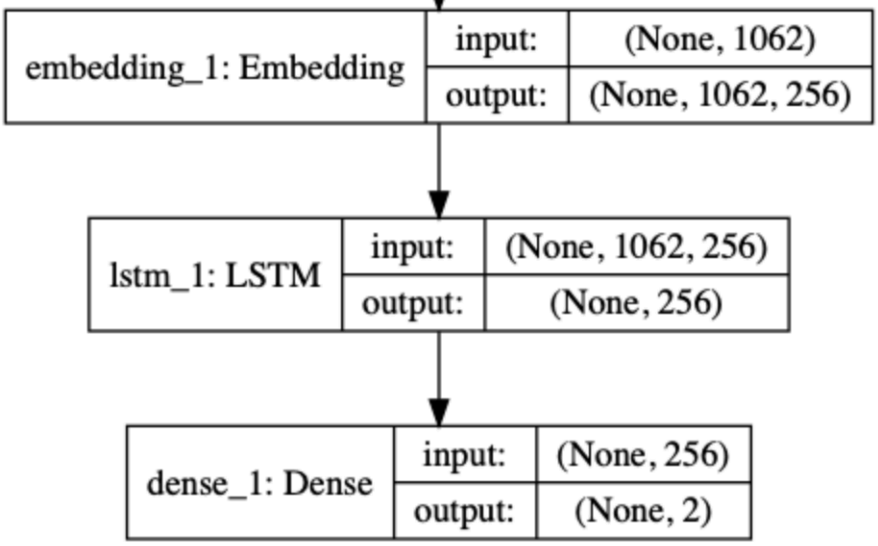


Figure 2: the LSTM model used in this assignment

For the dimension input- and output dimensions I had to do some "trial and error". Used Stackoverflow and our friends over at "the Internet", and watched how others adjusted their parameters to fit the training data.

A key factor was the amount of data we got to train on. It was **a lot** of data. So, I made the decision, after advice from the student assistant, to partition the data. I partitioned the data to fit the time gap I got at Kaggle (where I had a P100 to train on). Partitioned the data then by half the size of the X_train data.

In the code, I am using to_categorical, reason being that we have a Dense-layer outputting, through two "Densenodes", a vector with two values, e.g.: → [[0.2, 0.8], [0.25, 0.75]] which is Densonode1's and Densenode2's probabilities for the the input being class1 or class2 (positive or negative «movie review»).

The RNN (with LSTM layer) achieved the following accuracy on the test data:

Loss:	0.1628359776079664
Accuracy:	**0.9487619514586909**

Which I would say, is acceptable – considering we should acquire at least 90% accuracy.

Here is a graph for the training of the model used to achieve this result:

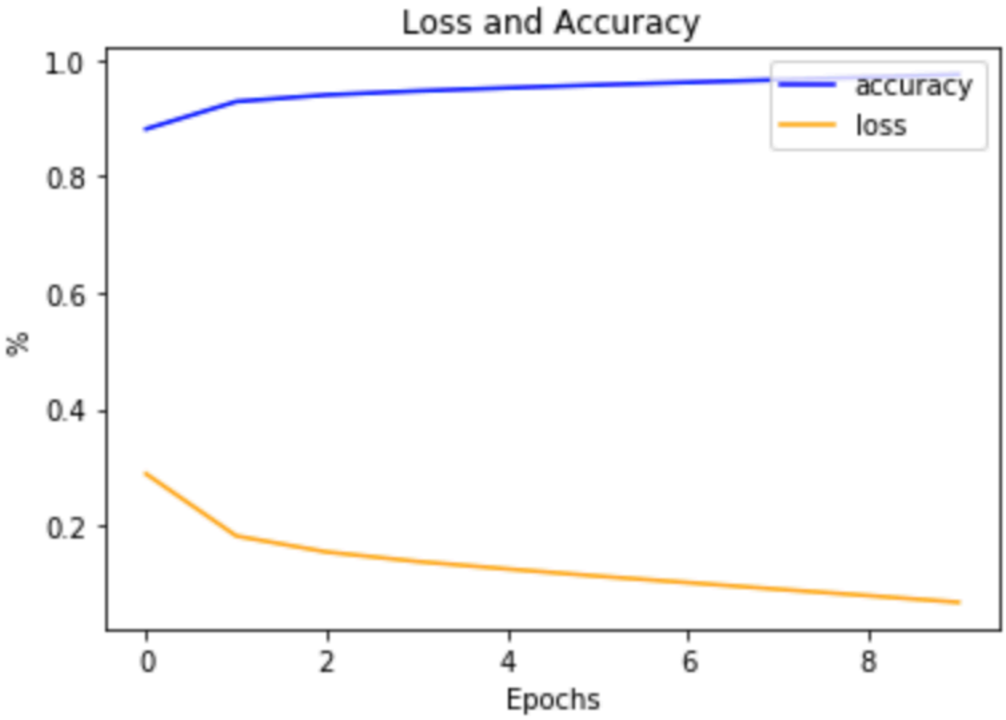


Figure 3: Evaluation of the model

2. **What is the reason for the large improvement in accuracy from the Naive Bayes/Decision Tree models to the LSTM? Give the most important reasons**

The biggest difference I believe, is that the LSTM network is capable of fetching the order of words, which makes a huge difference when the word "not" comes to play. "Nice" vs "not nice". There's a difference. The binarized vector of word-id's that we send in to the embedded layer is also a far more efficient representation of a review than sparse matrices.