Uncovering Sentiments using EDGAR Datasets

Team 9

Dawna Grace Raj

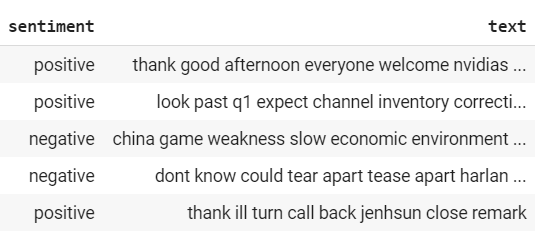
Jai Soni

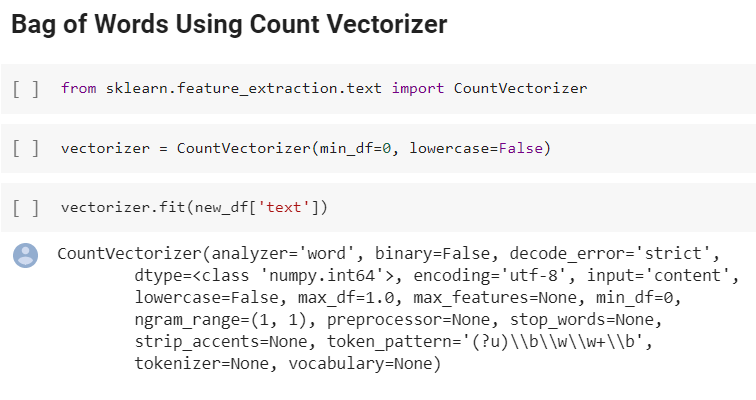
Nikhil Kohli

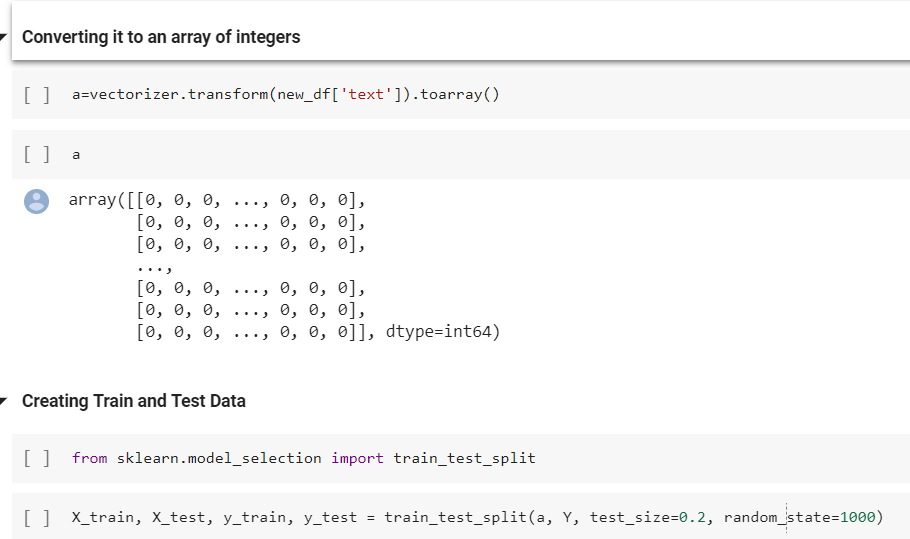
**Experiments:**

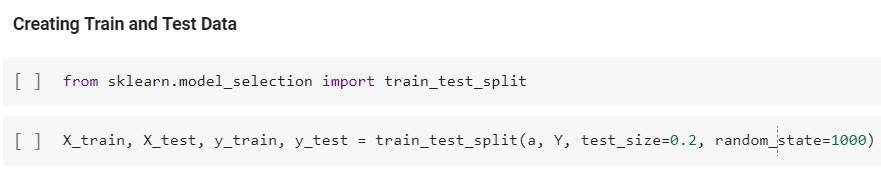
**Experiment 1:**

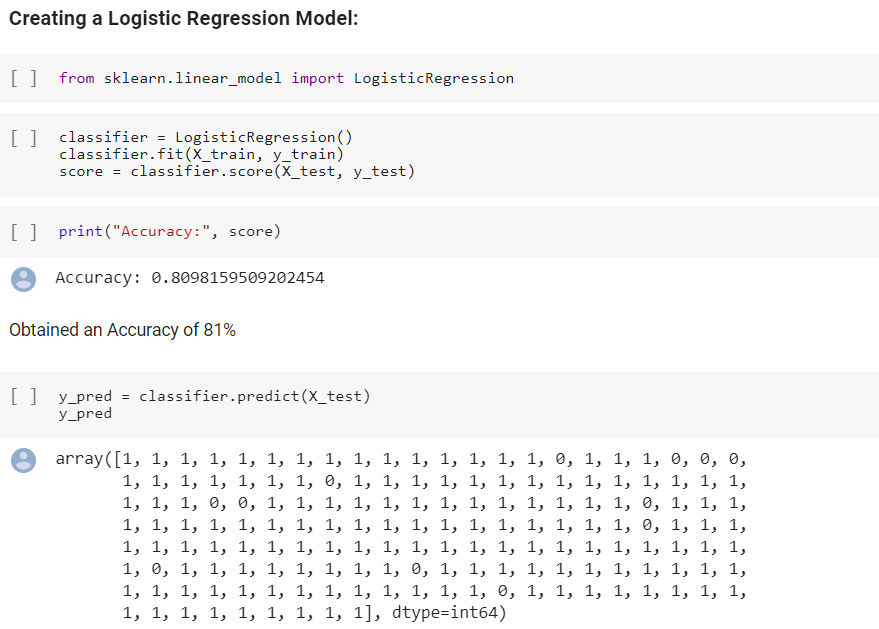
**Loading the data**



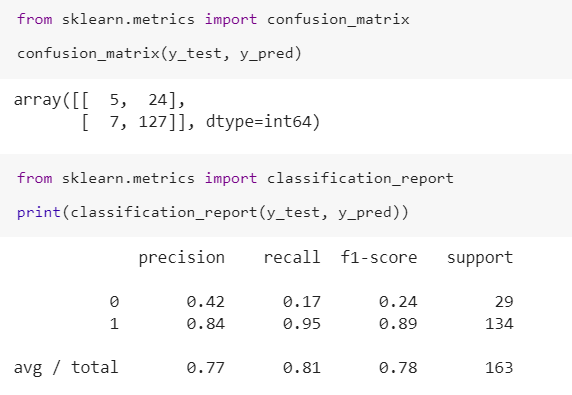


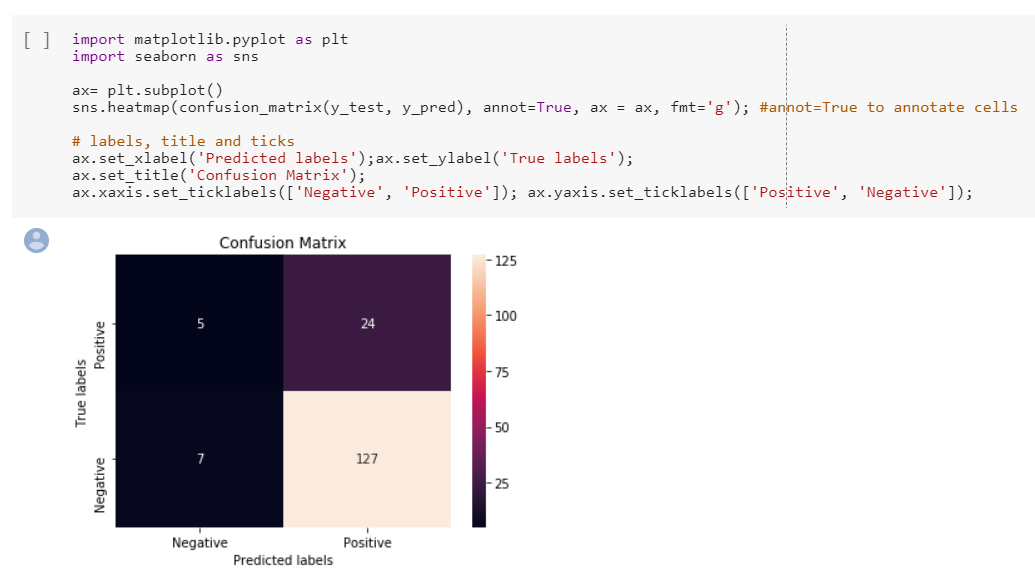




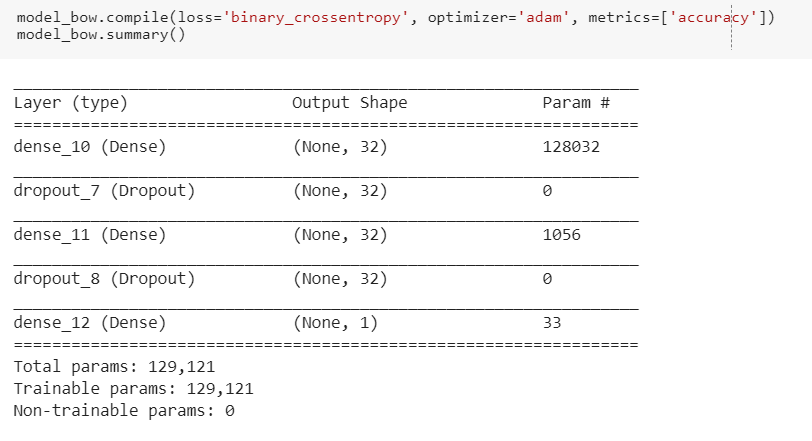


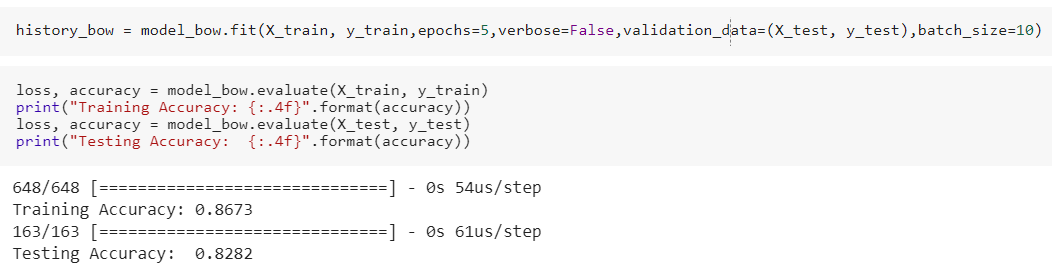
**Creating a metrics to measure model performance**

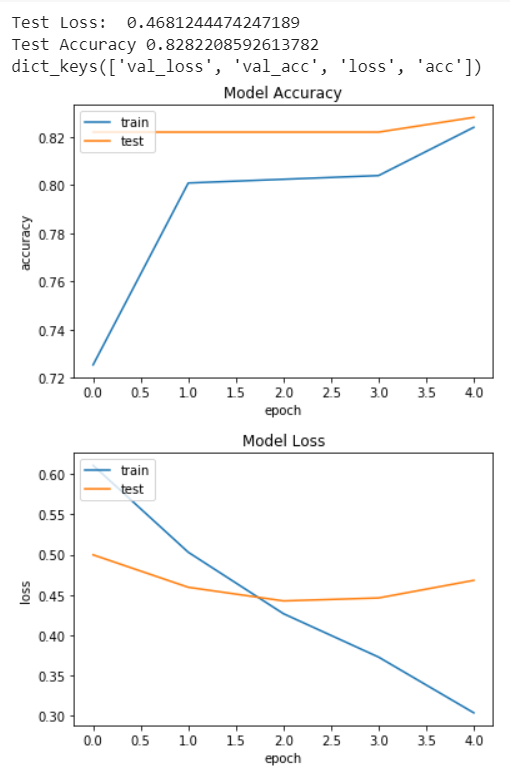


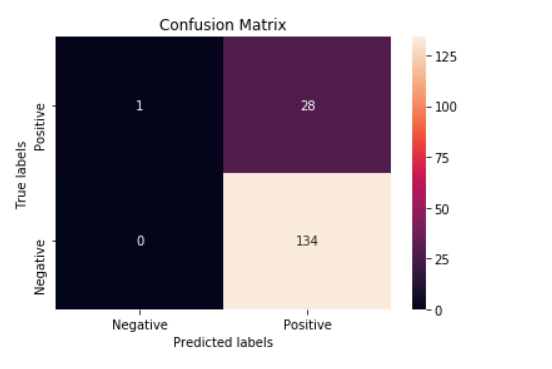


**Fully connected Model using Keras**

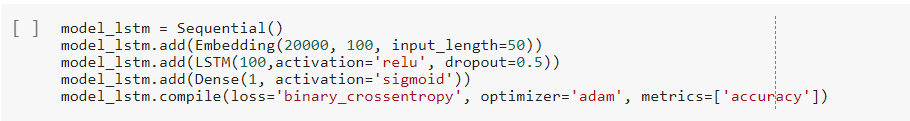


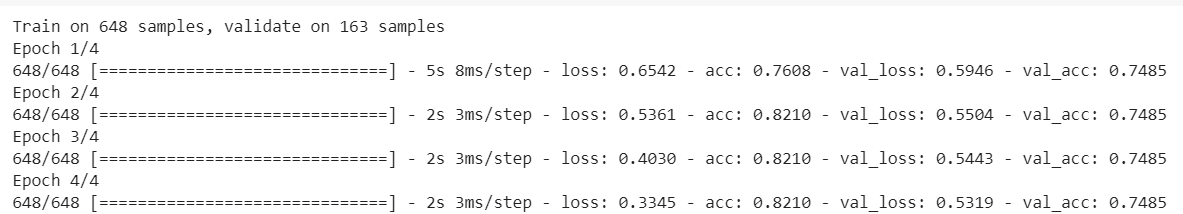


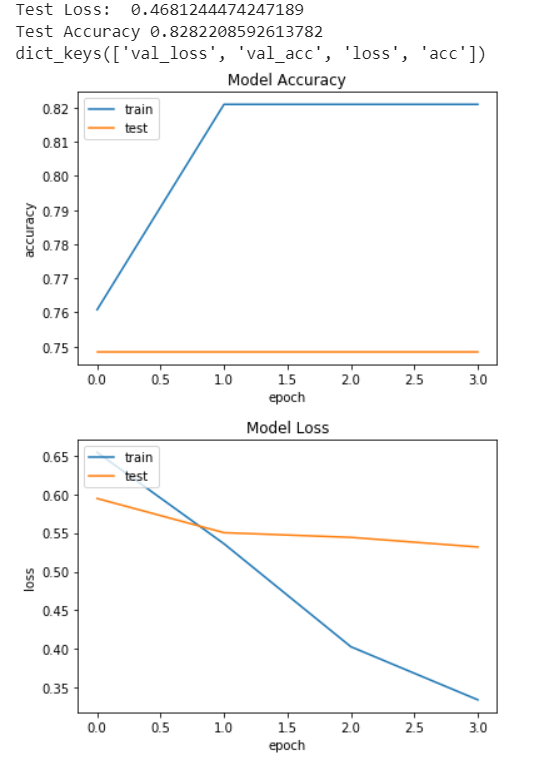




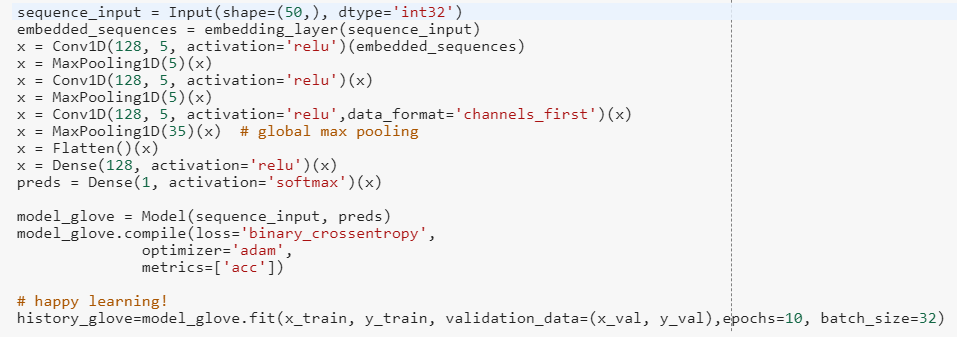
**Creating a LSTM model using keras**

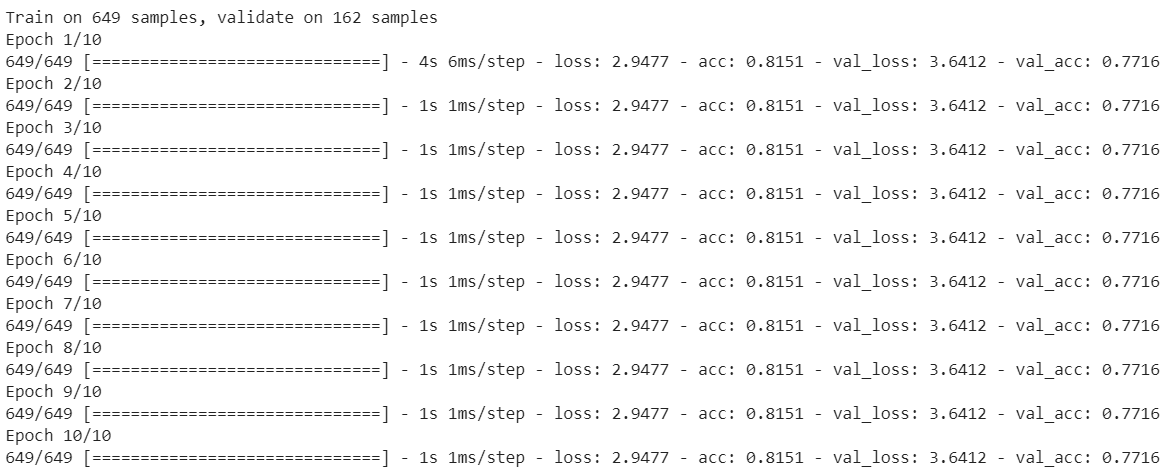






**Creating a Glove model**

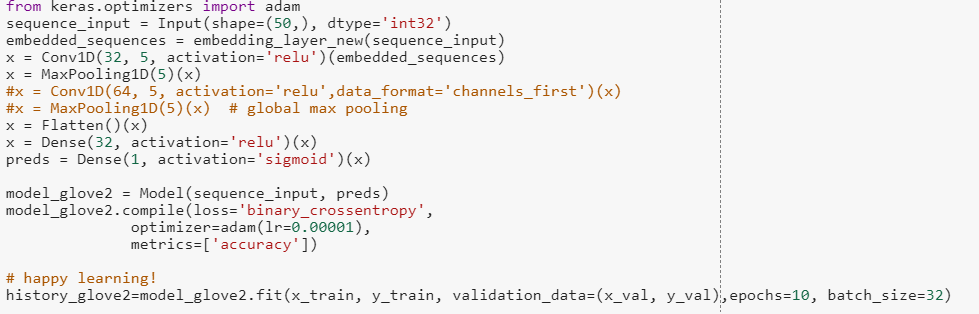


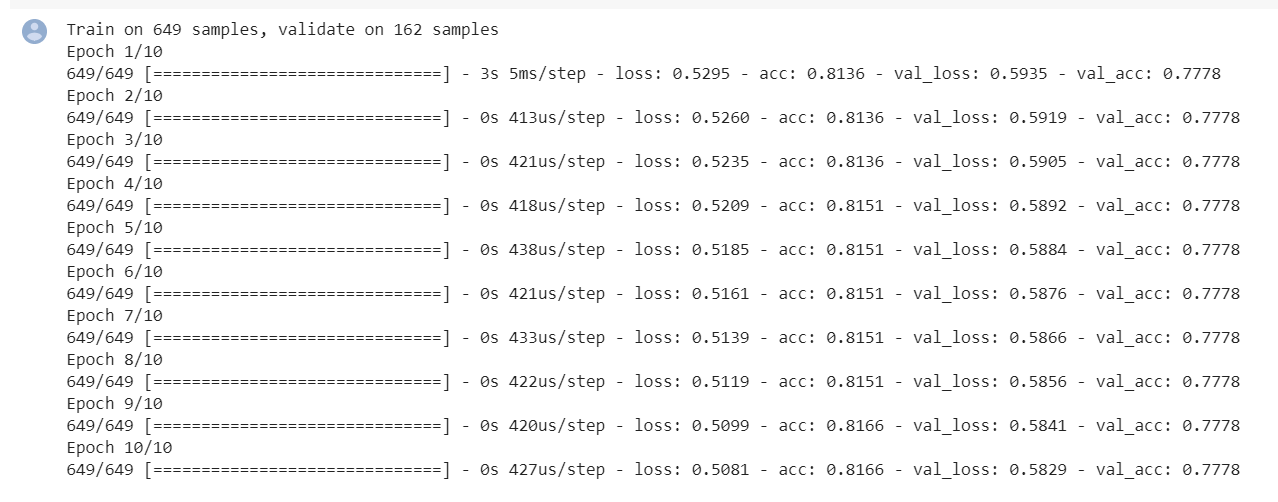


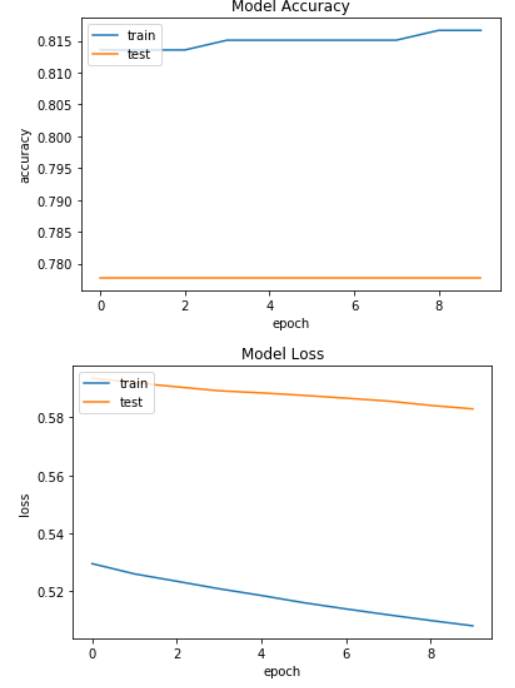
Getting a Test accuracy of 0.7716 with the Glove model

Trying out different models to get better accuracy

Model 2

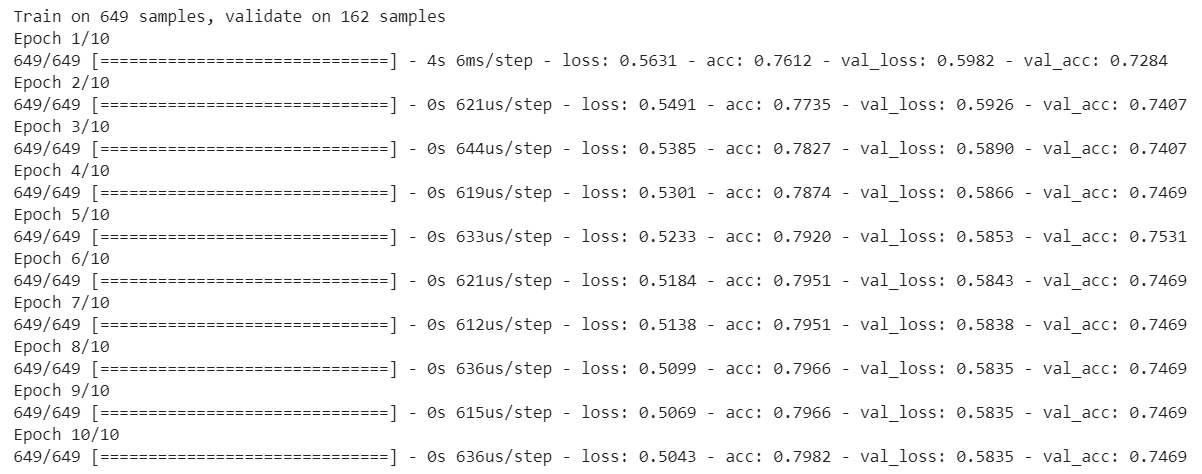


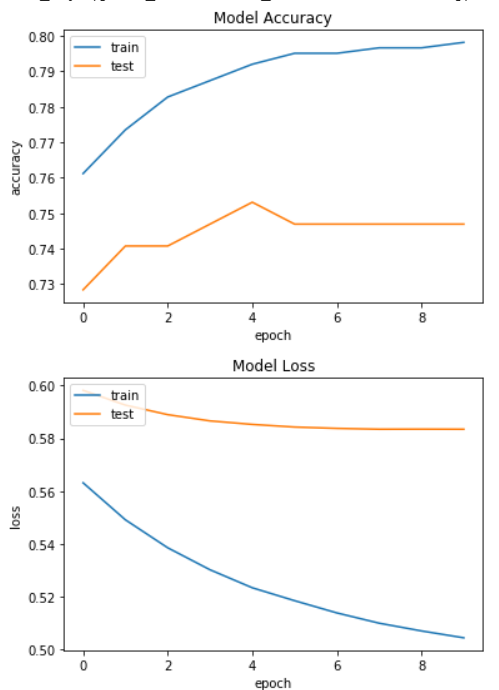




Model 3:

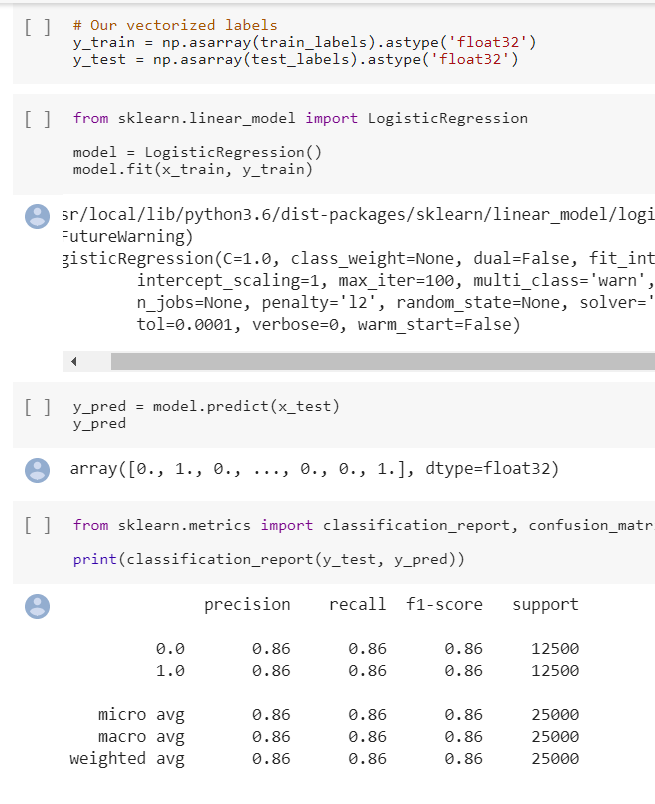




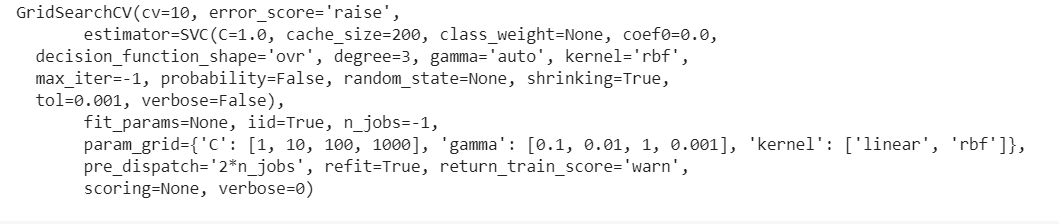


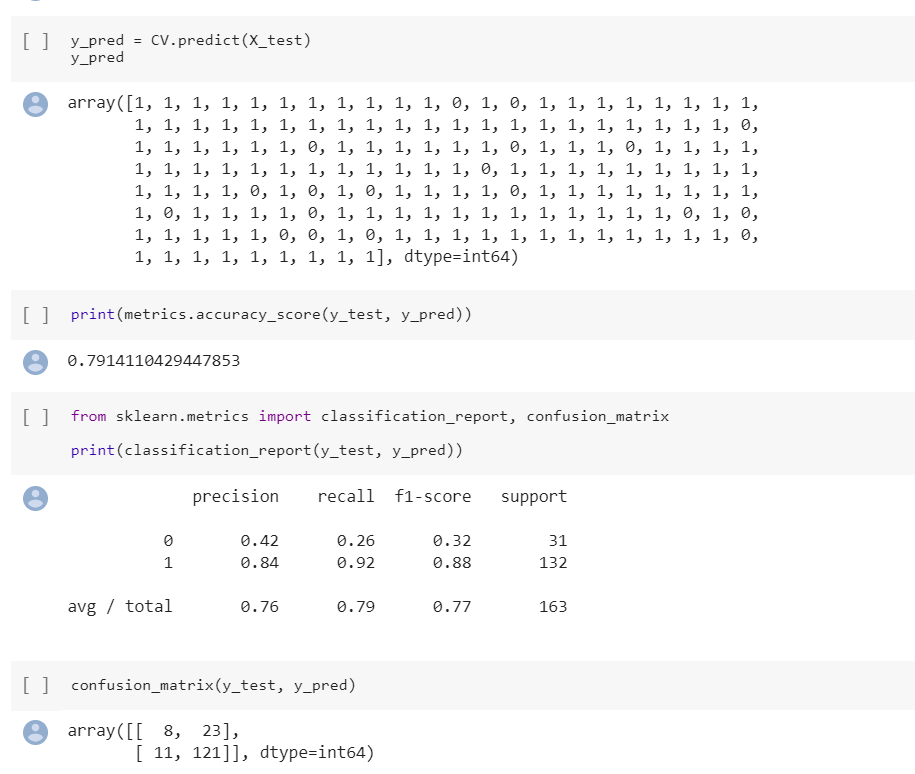
**Experiment 2: Transfer learning**

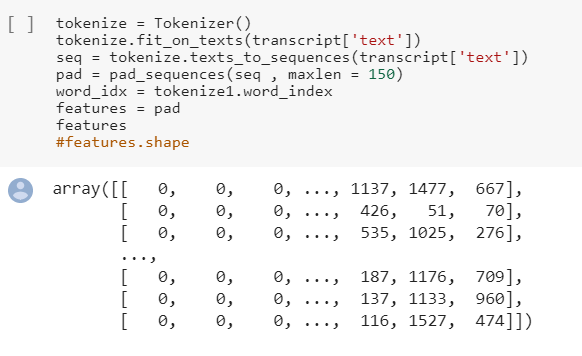
1. BOW



Used Grid Search and K-fold Cross Validation

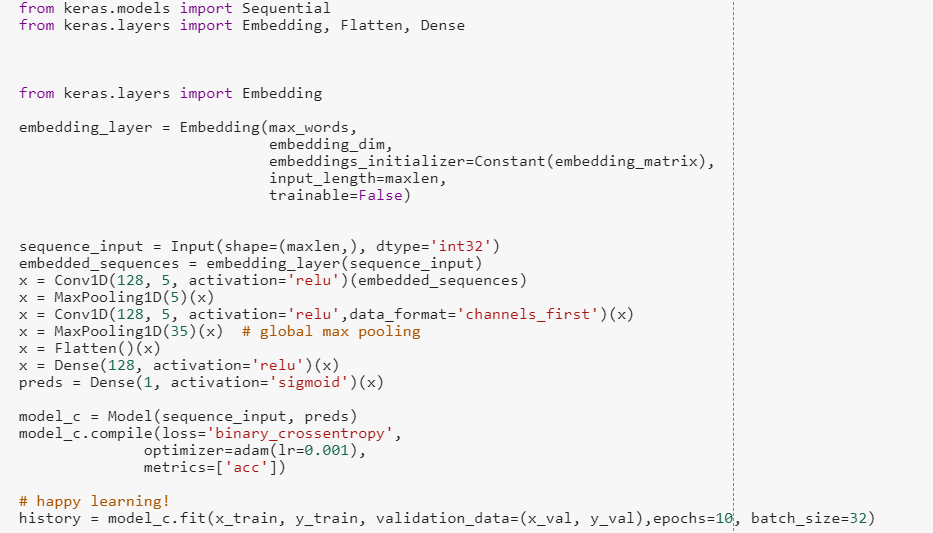


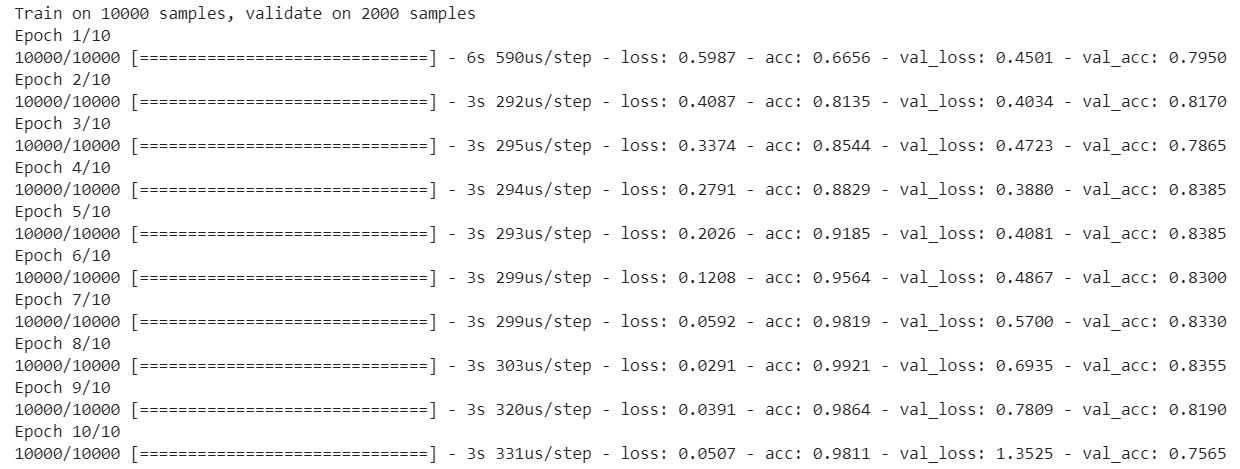


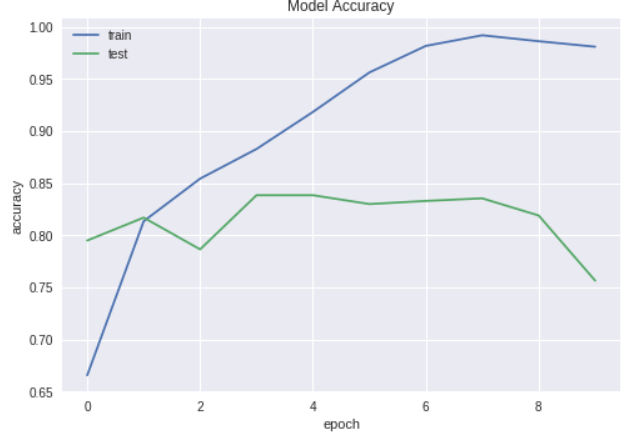


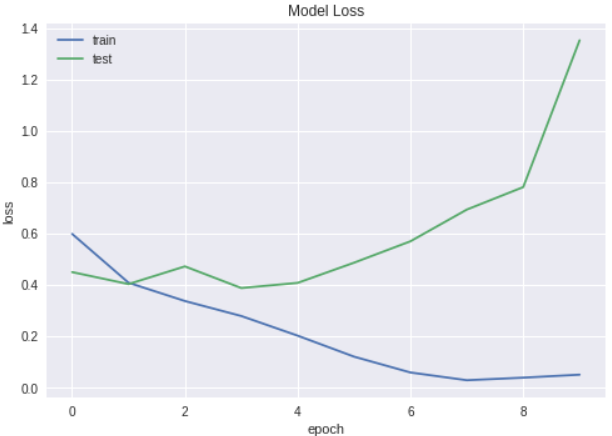
2. GLOVE

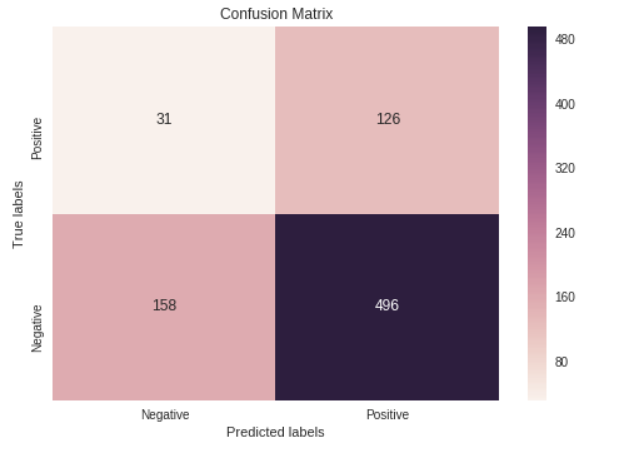
Used Pretrained model



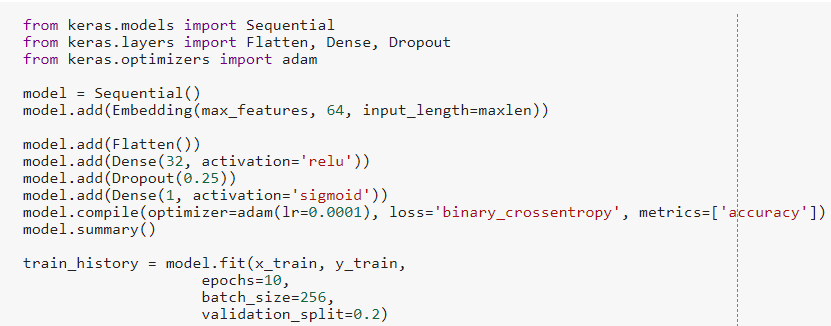


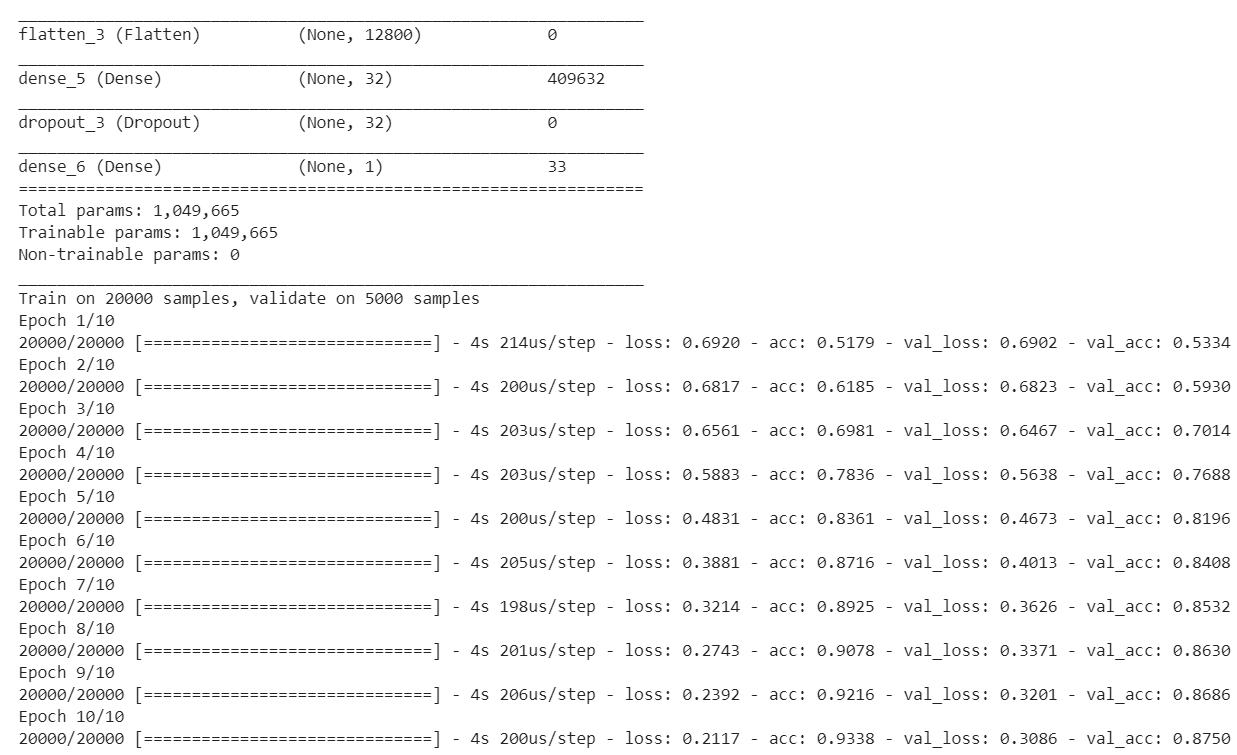


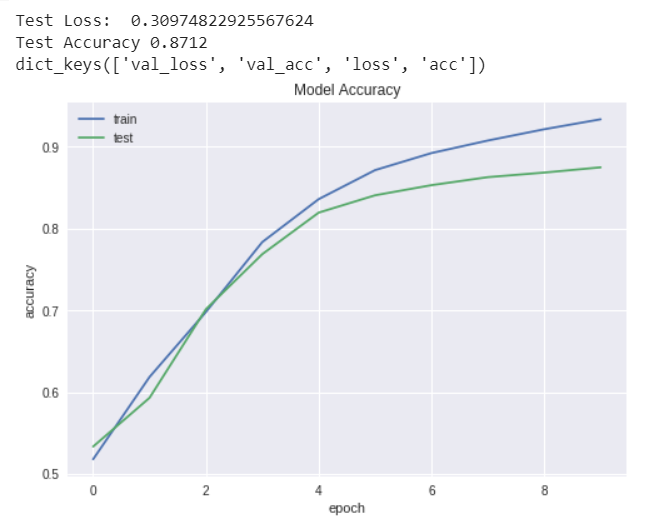


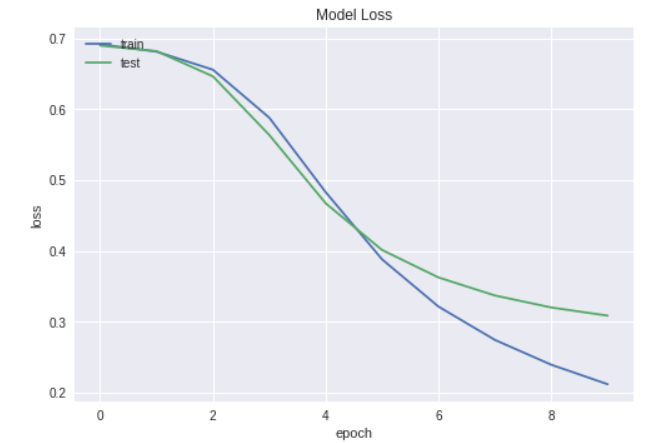


Word Embedding



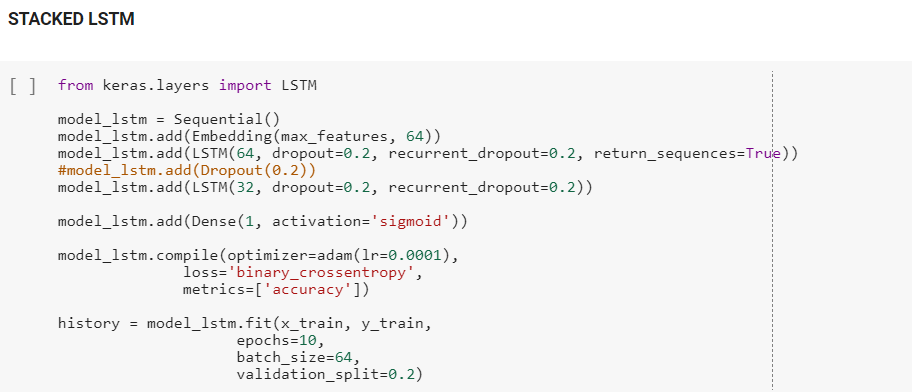


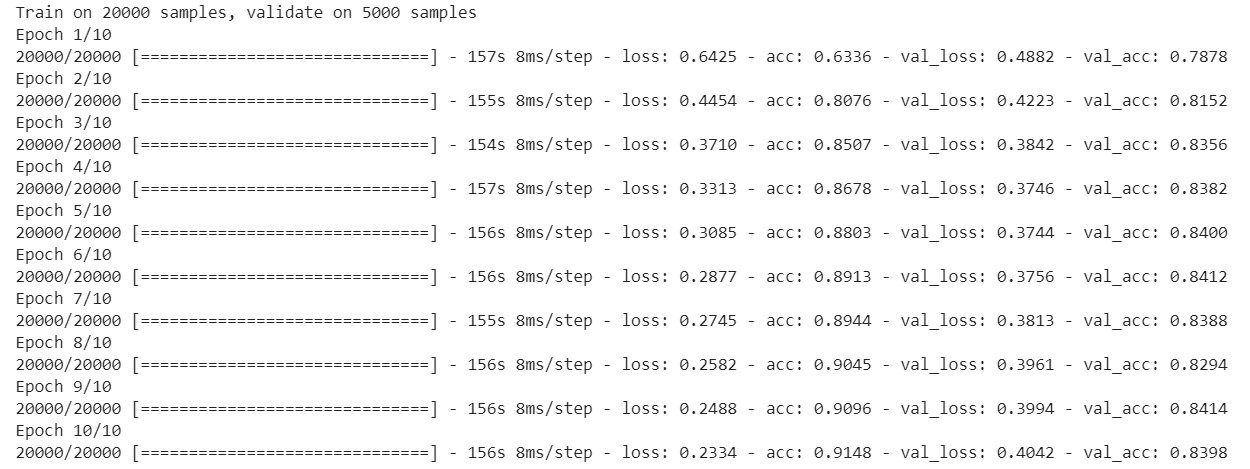




LSTM,

Stacked LSTM,



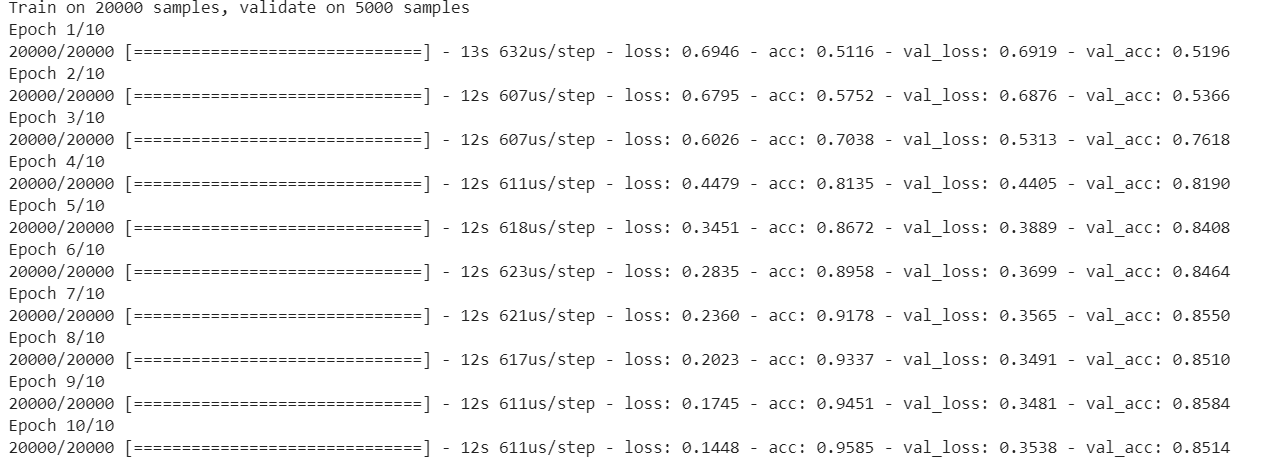


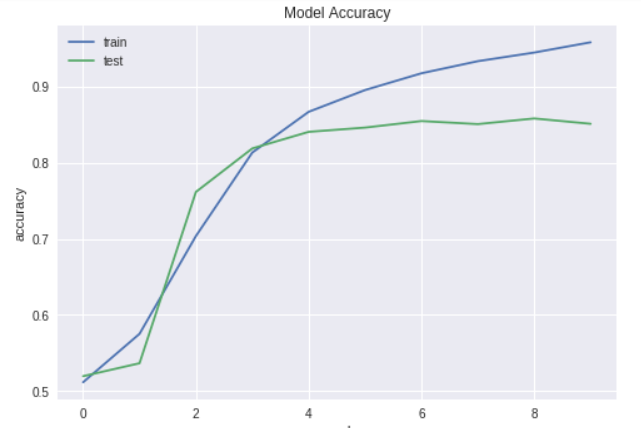


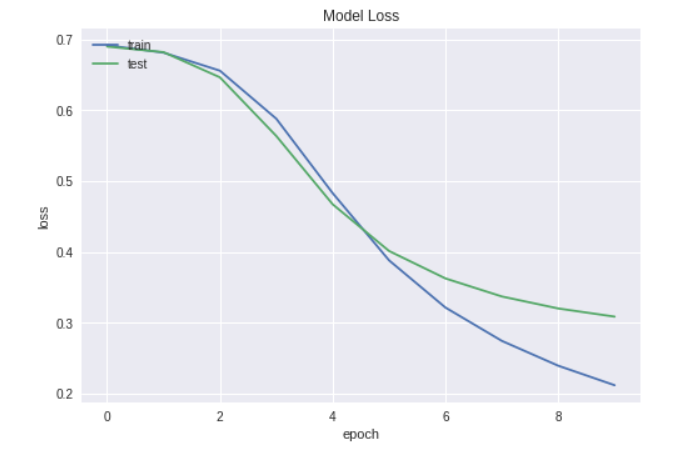
GRU(Gated Recurring Units),

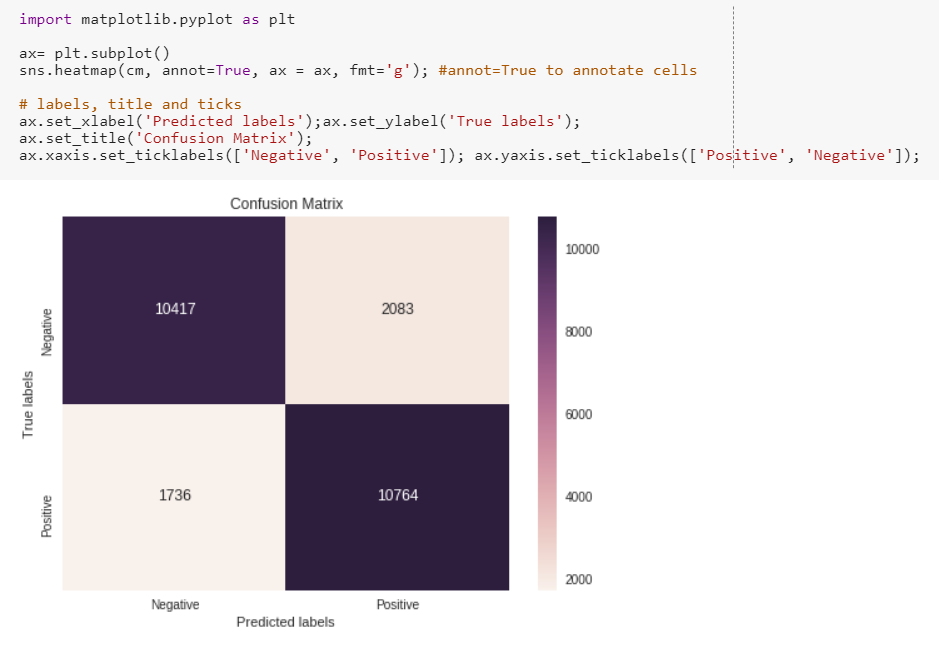
RNN,











Bi-directional RNN

We have created our own word embeddings on IMDB dataset and trained it

Used Transfer learning to predict sentiments for EDGAR datasets

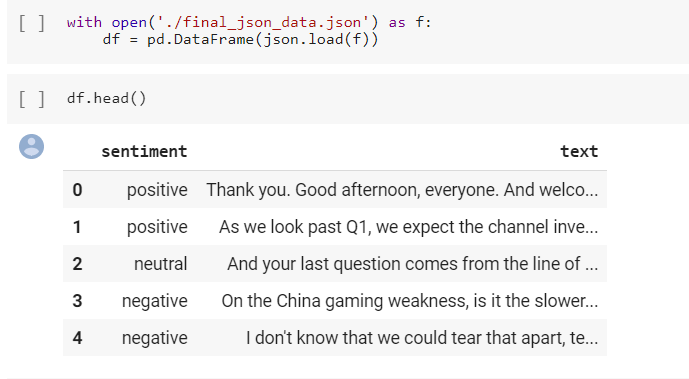
**Experiment 3: Using APIs**

Using the Amazon, Google, Microsoft and Watson APIs, obtain the sentiment

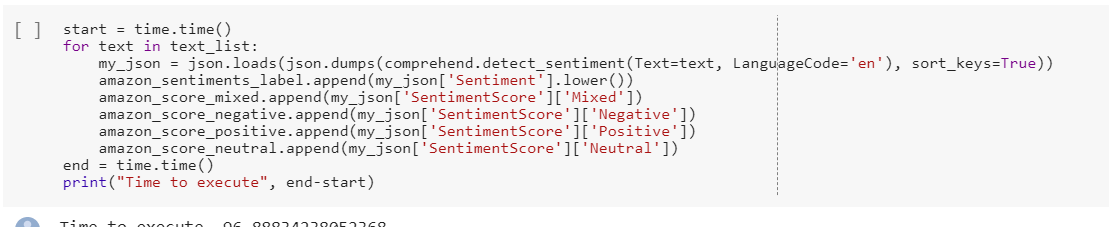
scores for your entire dataset.

Step 1:

Fetching the initial dataset with labelled transcripts

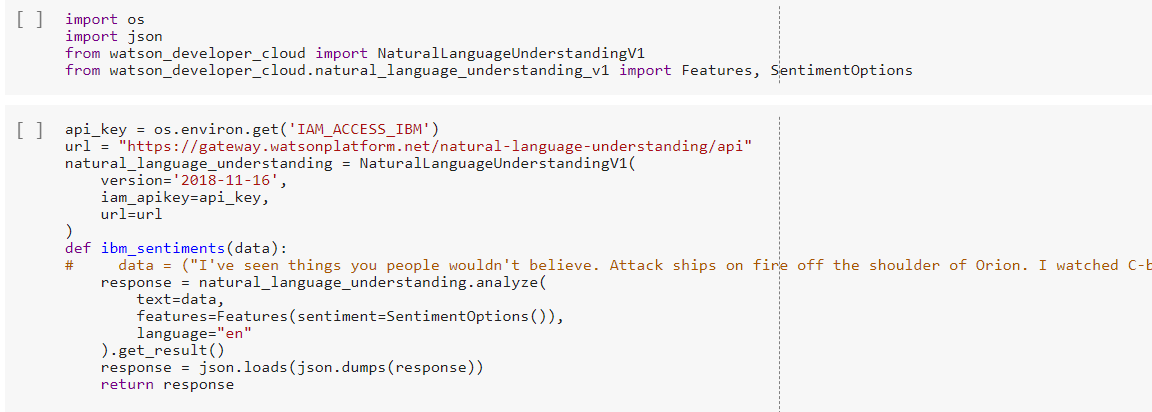


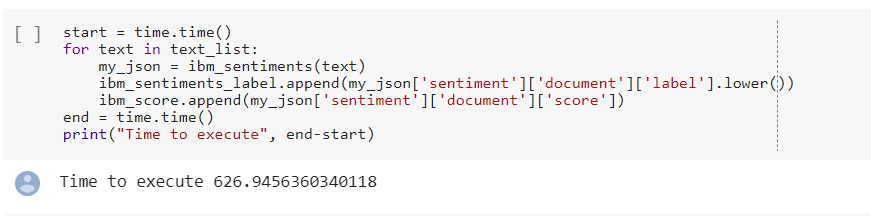
Step 2a : Using Amazon API - aws comprehend

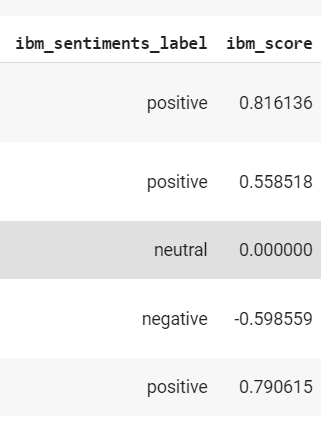




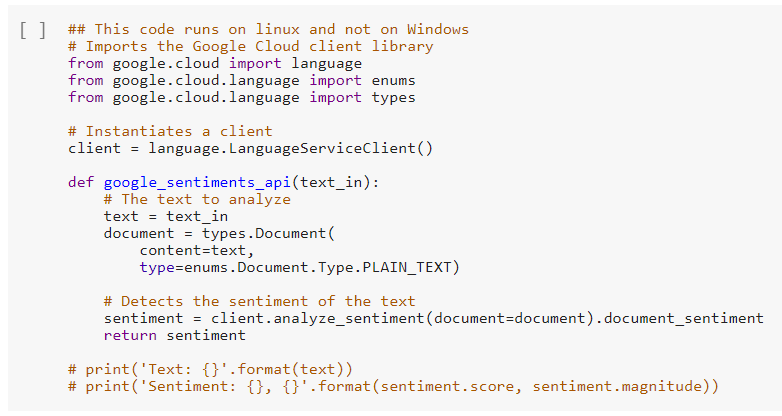
Step 2b : Using Watson API – natural language understanding







Step 2c : Google cloud language API



The score of a document's sentiment indicates the overall emotion of a document. The magnitude of a document's sentiment indicates how much emotional content is present within the document, and this value is often proportional to the length of the document.

It is important to note that the Natural Language API indicates differences between positive and negative emotion in a document, but does not identify specific positive and negative emotions. For example, "angry" and "sad" are both considered negative emotions. However, when the Natural Language API analyzes text that is considered "angry", or text that is considered "sad", the response only indicates that the sentiment in the text is negative, not "sad" or "angry".

A document with a neutral score (around 0.0) may indicate a low-emotion document, or may indicate mixed emotions, with both high positive and negative values which cancel each out. Generally, you can use magnitude values to disambiguate these cases, as truly neutral documents will have a low magnitude value, while mixed documents will have higher magnitude values.

When comparing documents to each other (especially documents of different length), make sure to use the magnitude values to calibrate your scores, as they can help you gauge the relevant amount of emotional content.

The chart below shows some sample values and how to interpret them:

Sentiment Sample Values

Clearly Positive\* "score": 0.8, "magnitude": 3.0

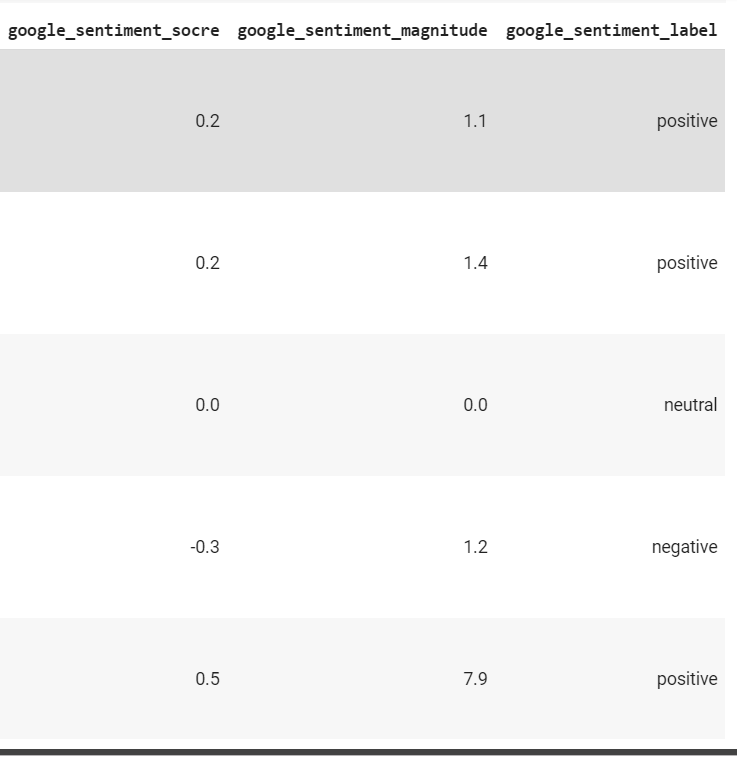
Clearly Negative\* "score": -0.6, "magnitude": 4.0

Neutral "score": 0.1, "magnitude": 0.0

Mixed "score": 0.0, "magnitude": 4.0

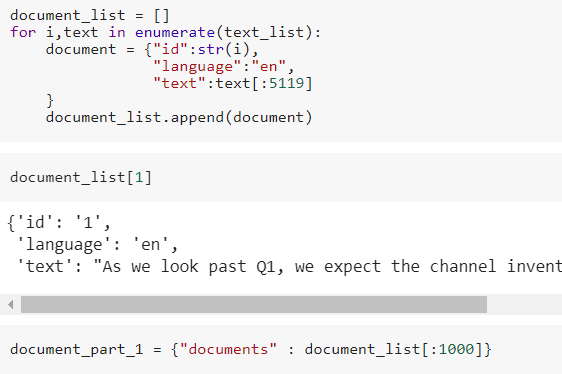
\* “Clearly positive” and “clearly negative” sentiment varies for different use cases and customers. You might find differing results for your specific scenario. We recommend that you define a threshold that works for you, and then adjust the threshold after testing and verifying the results. For example, you may define a threshold of any score over 0.25 as clearly positive, and then modify the score threshold to 0.15 after reviewing your data and results and finding that scores from 0.15-0.25 should be considered positive as well.





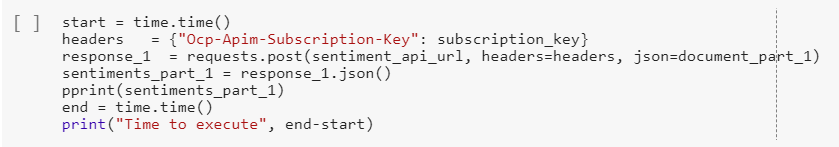
Step 2d: Azure text analysis API

Preparing documents



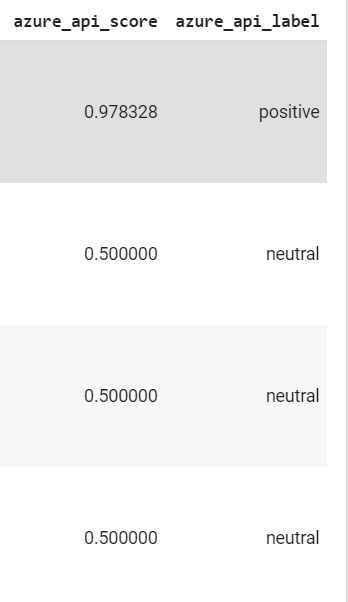


Getting sentiment score for part 1



Getting sentiment score for part 2





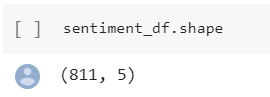
Saving the file with sentiment data from all API’s



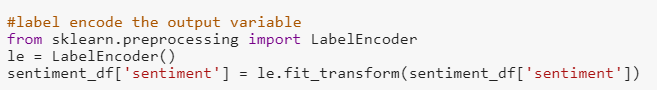
Step 3 : Normalizing sentiment scores

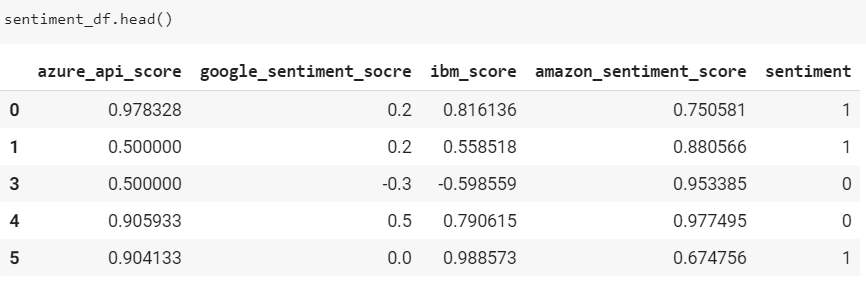
* Removing Neutral sentiments





* Build a model to map the output Sentiement label with the sentiment scores from all 4 apis

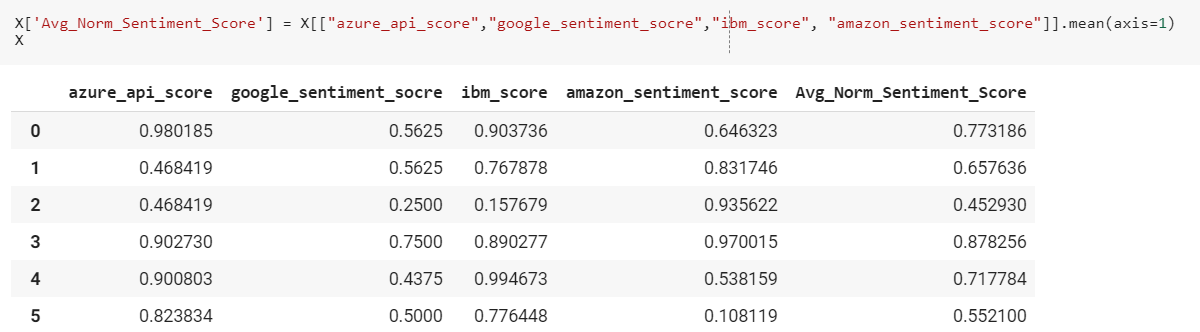




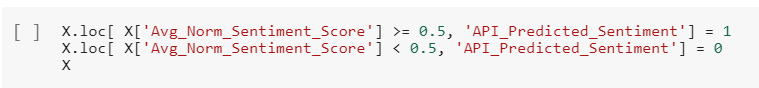
* Scaling the scores

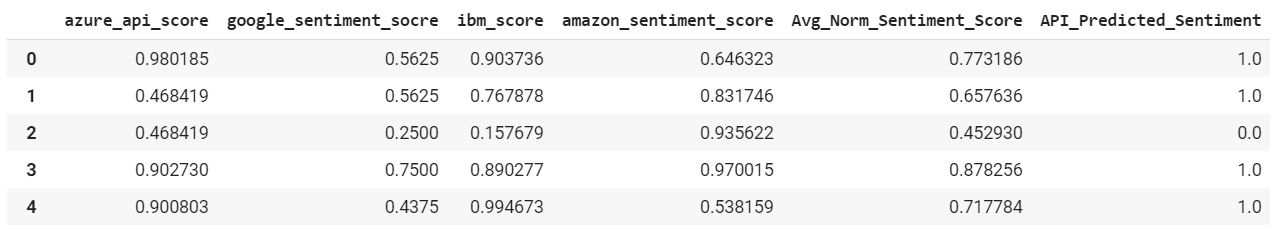


* After normalization



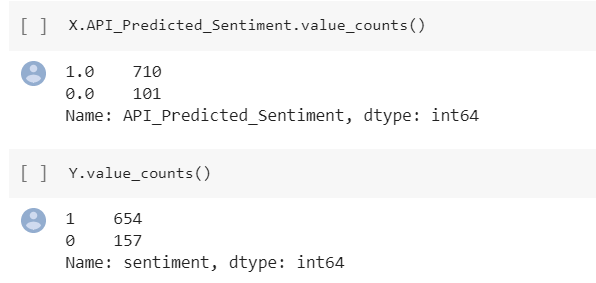
* Labelling w.r.t Average sentiment score

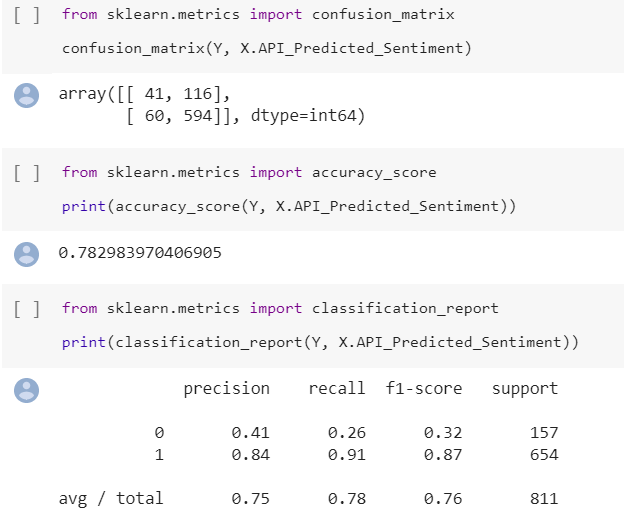


* 

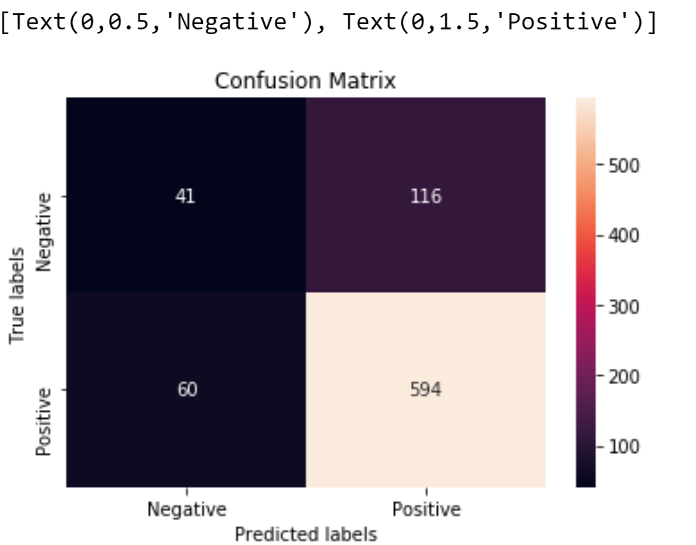
Step 4:

Getting Metrics



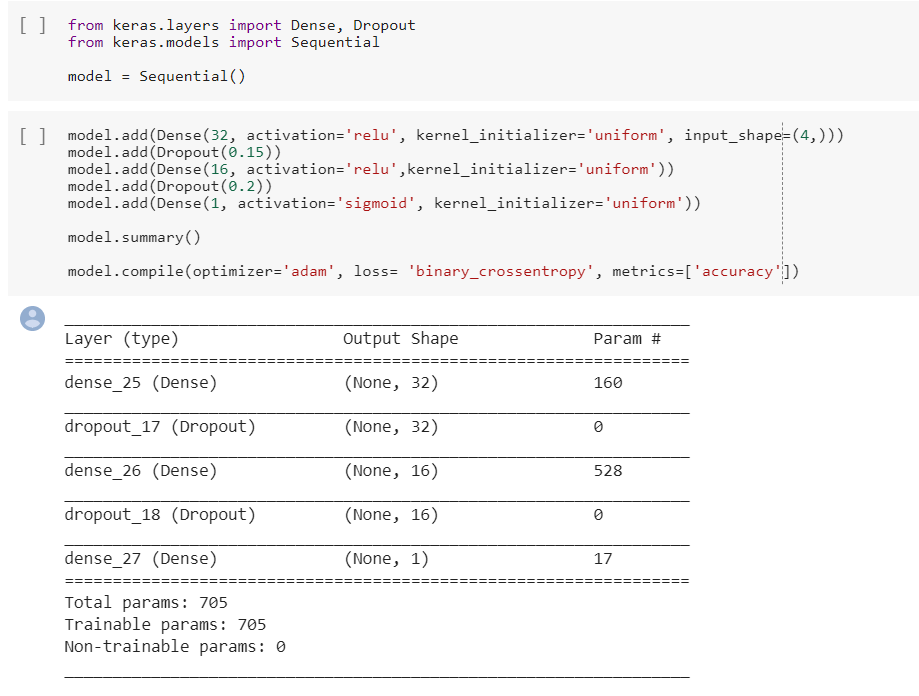




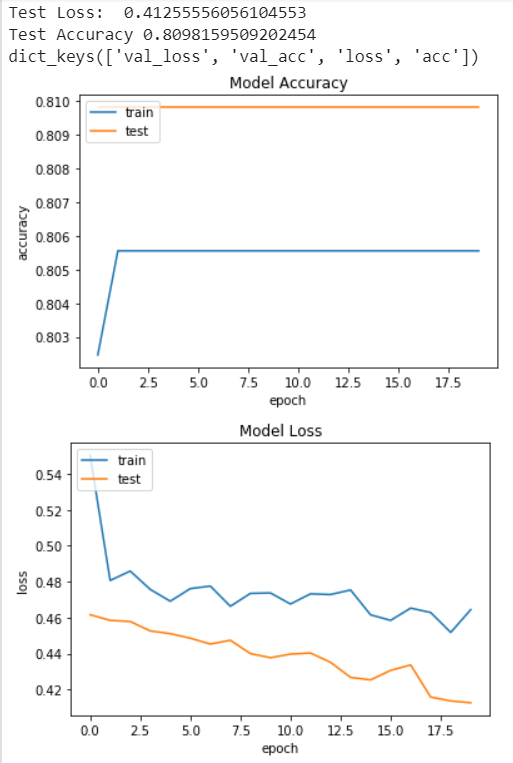


**Experiment 4: Ensemble learning using AutoML**

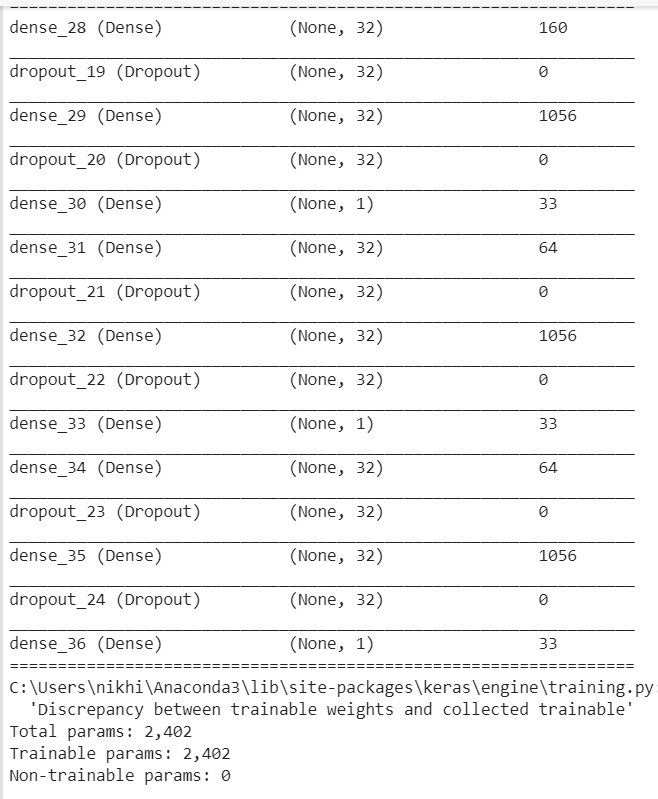
In order to map the sentiment scores from all 4 API’s to the sentiment’s we labelled manually, we created a FC Neural network

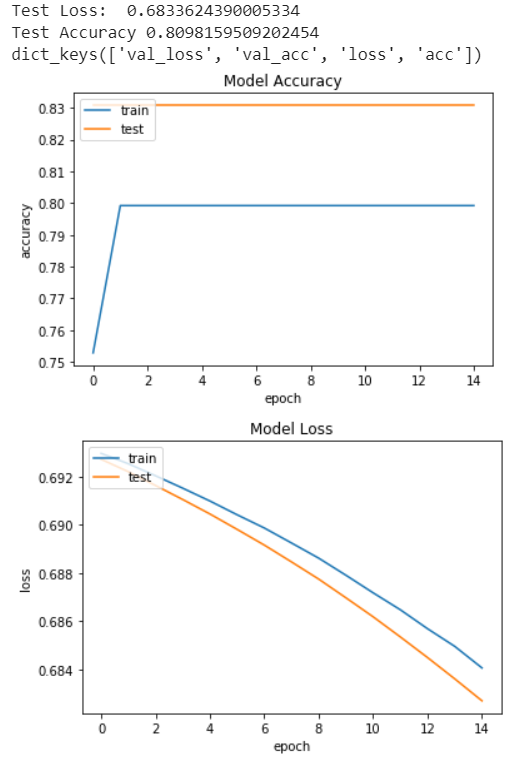


Followed by this we tried TPOT, AutoSKlearn , H20.ai in order to utilize autoML for mapping the same

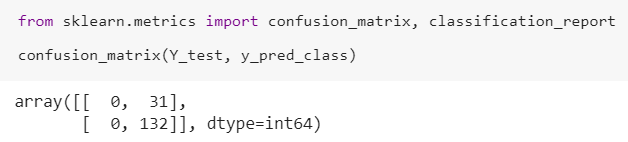


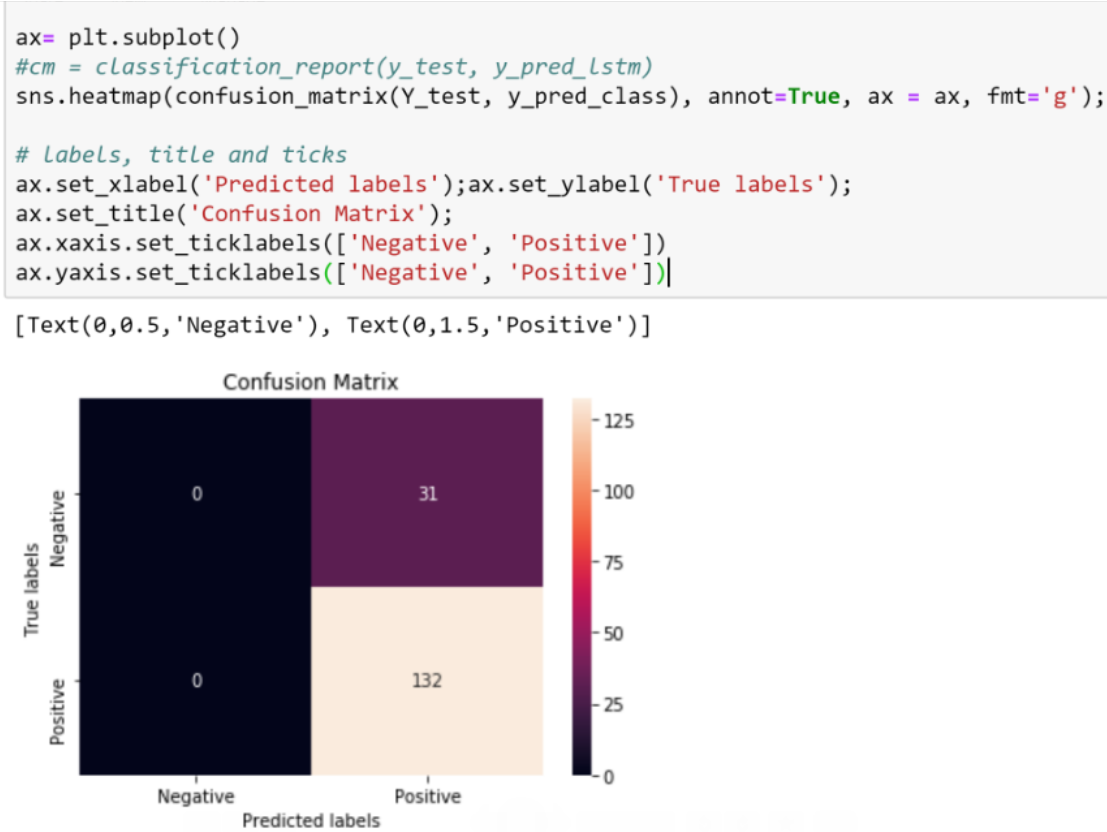
We did hyperparameter tuning for the model so that the loss converges better





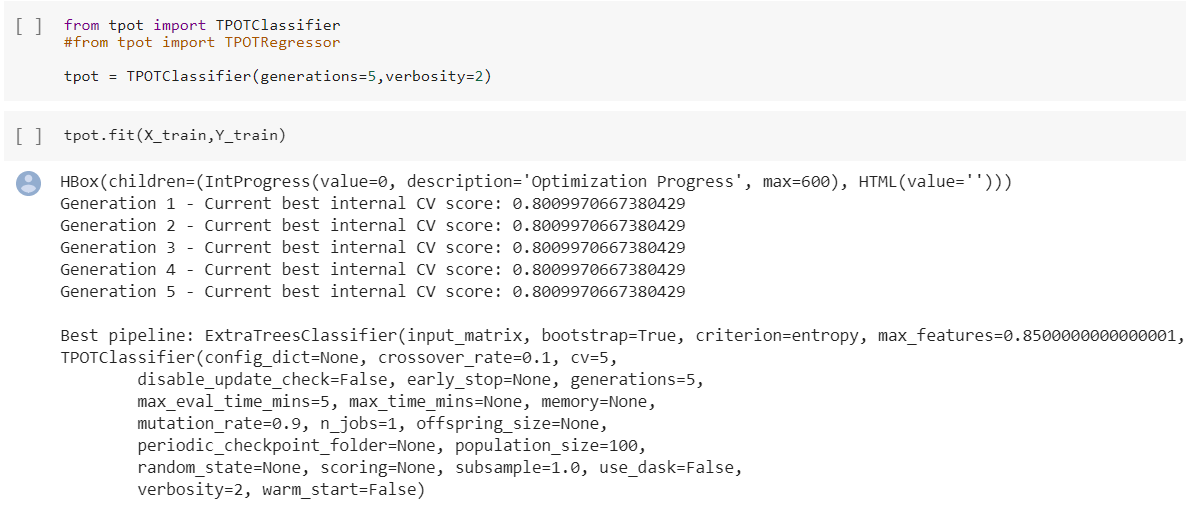
Since the model is trained on very few negative data, the model doesn’t predict the negative sentiments at all and we find 0 True Negatives as a result



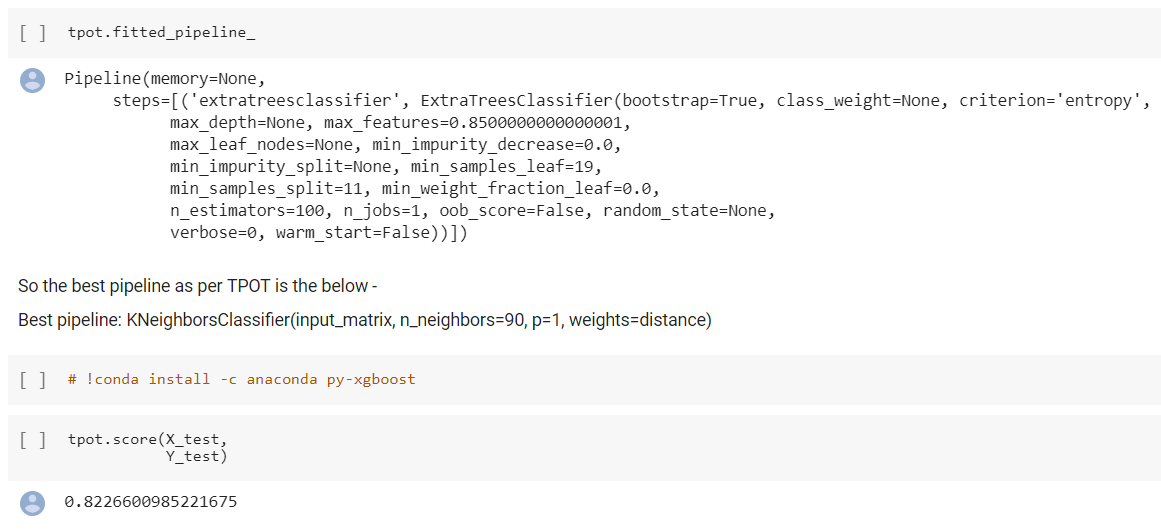


AUTO ML

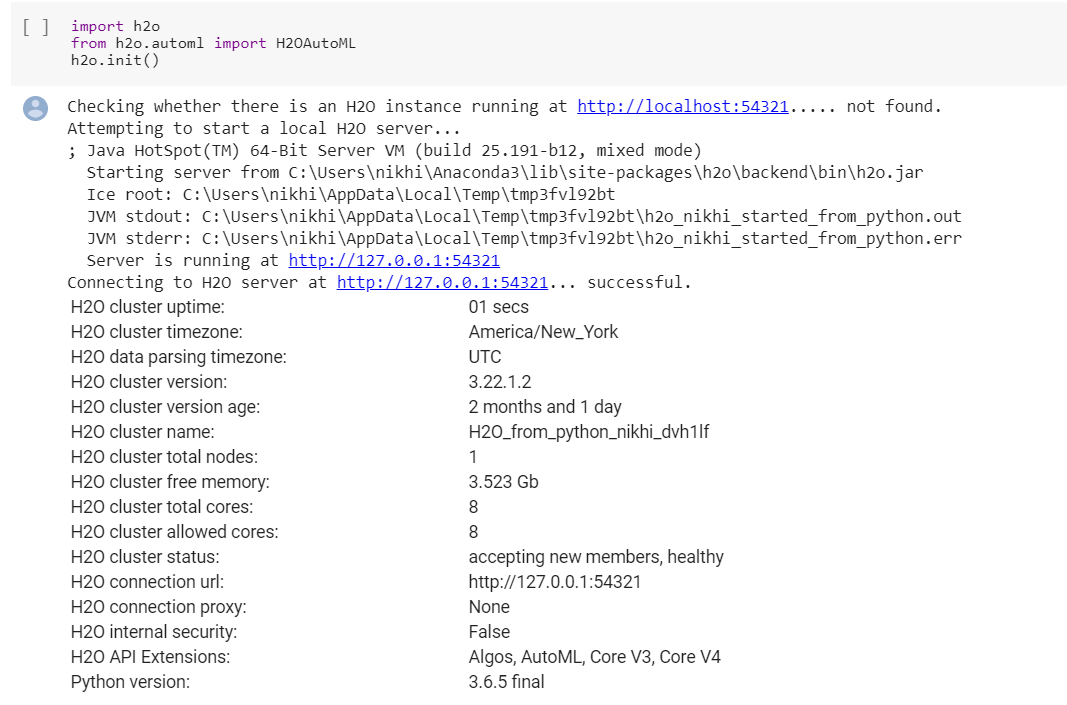
Implementing TPOT to get a model with tuned hyperparameters



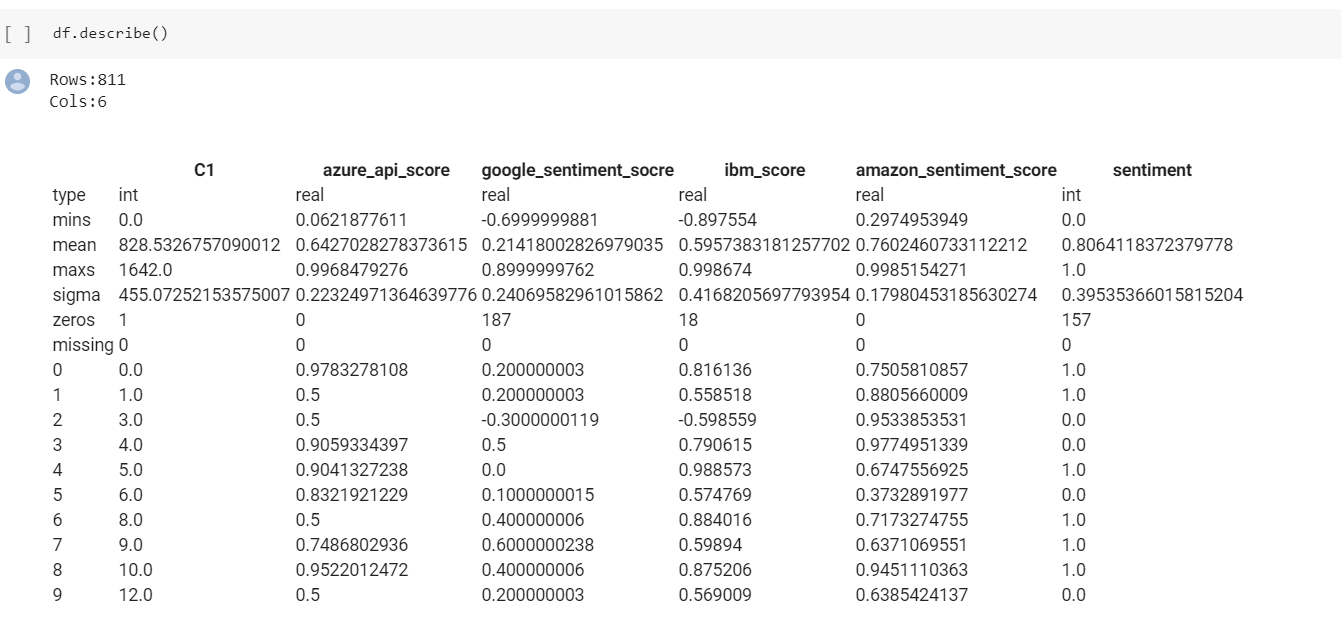
Output of TPOT:



Using H2O for creating a complete pipeline



Description about data

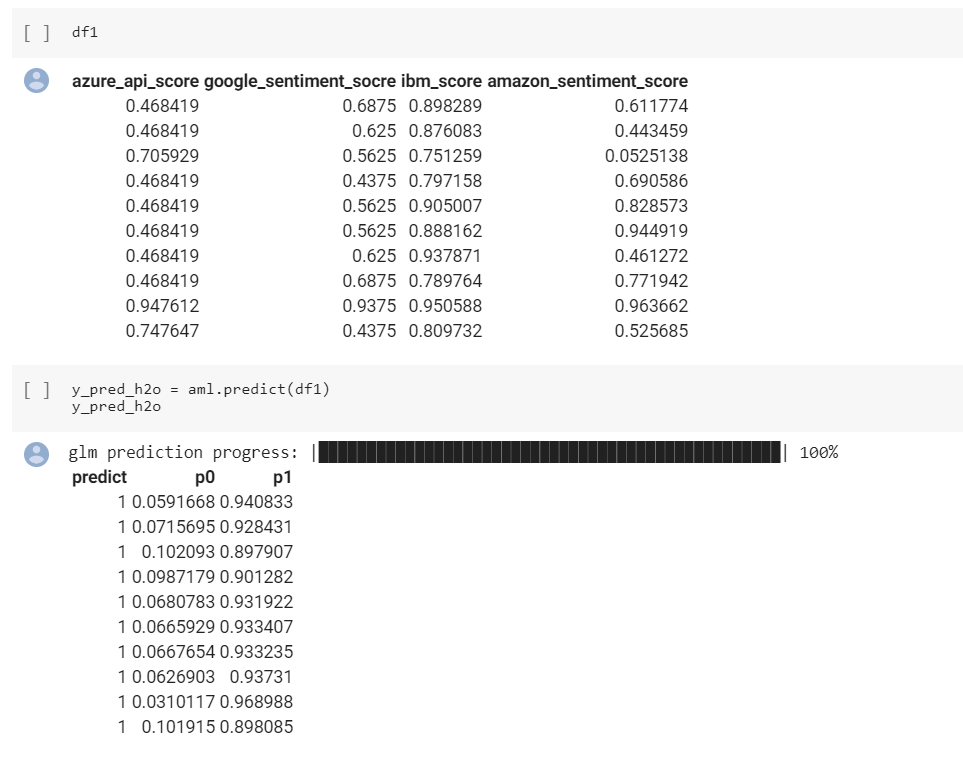


Training 10 models and getting there metrics score as follows:



The best auc score we got is : 0.742

Getting predictions for Text, p0 is Probability to be negative and p1 is Probability to be Positive



AutoSKlearn

Facing issue with autosklearn package

The issue is still open and the link to it is down below



https://github.com/automl/auto-sklearn/issues/520