ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Machine Learning & Content Analytics

Doctoranytime

User reviews moderation with deep learning

Tutor: Haris Papageorgiou

Postgraduate Students

Vangelis Christou - P2821805
Brikena Kokalari – P2821811
Konstantinos Kolovos – F2821905

Athens, October 2020

Contents

Introduction

Established in 2012, Doctoranytime is a Greek company which redefined the health system in Greece by offering detailed information on the experience and specialization of doctors, as well as evaluations from patients to thousands of them. In fact, Doctoranytime is addressed to all those who are looking for a doctor and wish to visit him in his private practice by making use of their public (EOPYY) or private insurance or by making a private appointment.

When the need arises for a doctor, the patient can at any time, regardless of office hours, enter the doctoranytime, seek either a specific doctor or find the doctor he needs using the search filters. The patient can then see the profiles of all the doctors returned to him by the system as search results, and select the most suitable doctor for him, evaluating information such as the doctor's CV, the services offered and his experience in specific areas of the specialty, the cost of a visit or medical operation, the insurance companies that are accepted, the evaluations and the comments of patients who visited the doctor. If the patient chooses his doctor, he can make an online appointment at any time of the day or night, in less than 1 minute, without unnecessary phone calls and hassle.

The benefits that the patient derives are that he can find doctors who accept EOPYY and visit them completely free of charge in their private practice, already seeing through the platform if they have reached the maximum number of appointments set by public insurance and when is the next available one. At the same time, in order to eliminate the waiting time and the inconvenience that may arise with the EOPYY (free) appointments and to serve the patient more directly and with less burden, doctoranytime.gr has contracted with hundreds of doctors who receive visits at a cost of 10 up to 25 euros. It is also extremely useful for patients to see the reviews and comments of other previous patients, as well as a range of information such as doctor services and costs, that ultimately help in deciding which doctor to visit.

Last but not least, it is very important and interesting that the patient can at the time he chooses an examination to see what preparation is needed for the examination, something that will determine the time he has to do it, as well as to choose other related examinations to make sure he does all the necessary exams and avoid repeat visits.

Problem description

One of the main advantages for Doctoranytime's users is the ability for a new patient to check the history of previous patients' reviews and evaluations for the health specialist he / she is interested in.

Doctoranytime users give ratings and write reviews about the services of health specialists. These reviews and ratings help doctoranytime evaluate services provided and take necessary action to improve customer service. While ratings are useful to convey the overall experience, they do not convey the context which led a reviewer to that experience. If we look at only the rating, it is difficult to guess why the user rated the service as 4 stars. However, after reading the review, it is not difficult to identify that the review talks about good 'service' and 'experience'.

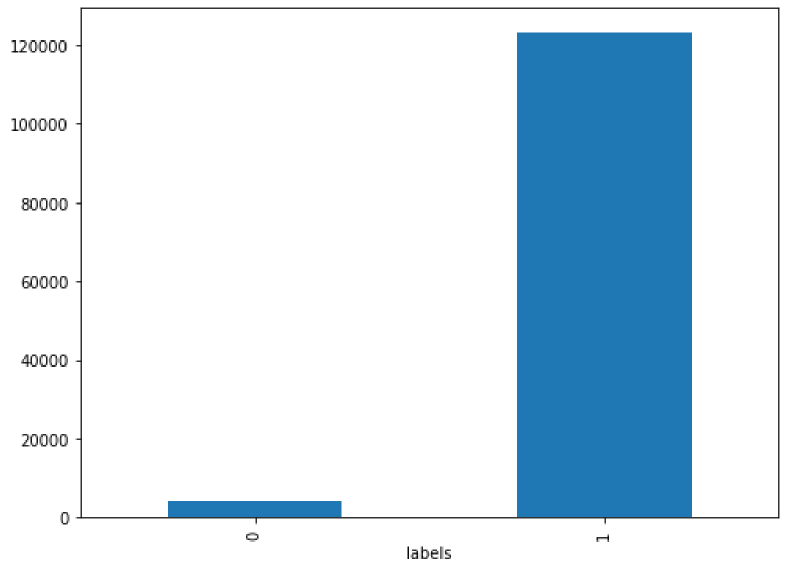
To date, approximately 150,000 comments have already been published, and more than 600 new ones are submitted daily. For their processing and publication, it is estimated that more than 2.5 hours of continuous work is daily needed, so we understand that this is a time-consuming process. The company year by year increases its turnover and the above sizes have increasing trends. This procedure is highly costly as it takes several hours of employee work that could be used on more substantial tasks, and also a problem for the doctors whose evaluation is delayed, especially for the new subscribed ones with few evaluations, as generally they are not preferred by clients.

Mission

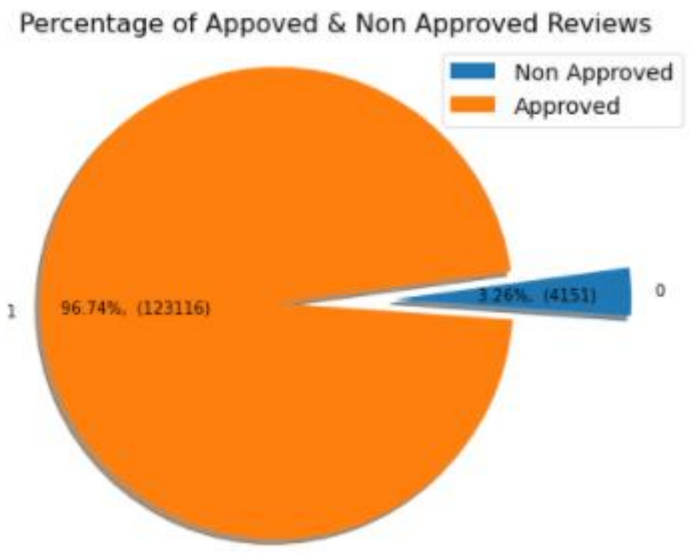
The purpose of this assignment is to use machine learning to provide an automated solution that manages the approval procedure of patients' reviews, that is, whether a review is suitable for publication or not. So far, the company does the above process manually and there is no model that makes the above process automatically with the Greek language. In addition, some restrictions must be taken into account in terms of the form and content of the reviews which will be in line with the policy set by the company. Based on the evaluation management history, we will train a neural network to do the above process automatically without human intervention.

Data

We currently have a database of 127.267 reviews and their end result in a csv file. The data is recorded in two columns where in the first is the customer evaluation and the second column concerns the evaluation category (whether approved or rejected).



However, the above two classes for the final conclusion of the reviews are not represented accordingly in our dataset, as is often the case in most real-world classification problems. The data shows a strong level of class imbalance, which occurs when there are insufficient views of the data corresponding to any of the class tags. In our case, the approved reviews far outweigh the unapproved reviews. If we train a binary classification or neural network model without correcting this problem, the model will be completely biased.



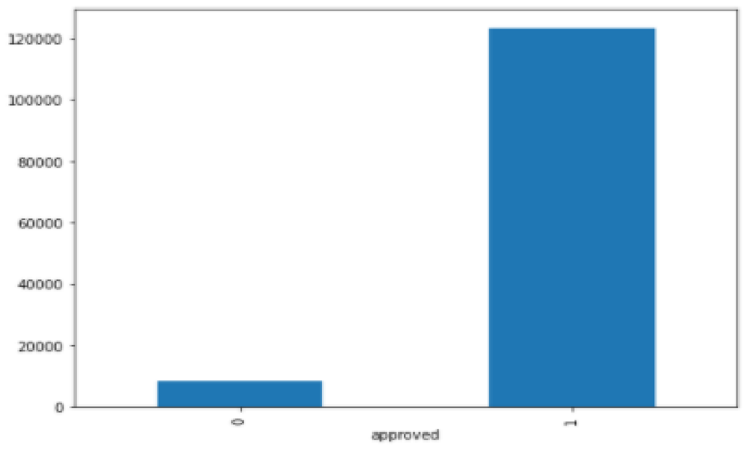
It would be good to mention that “Doctoranytime” has published guidelines and terms of use for a review to be published[1]. Any comment that is offensive or profane is not published. Almost every single one of the reviews that have not been published, lie in this rule. Therefore, in order to achieve data augmentation, we had to add “toxic” data/comments in the initial dataset to make the classes more balanced.

Another restriction concerns receipts or personal data: Doctoranytime do not publish reviews regarding the issuance of receipt by the experts. Concerning the privacy of personal data, we will never see published: surname, email, phone number. Again, we have to add random data which contains emails, telephone numbers or names.

Data augmentation

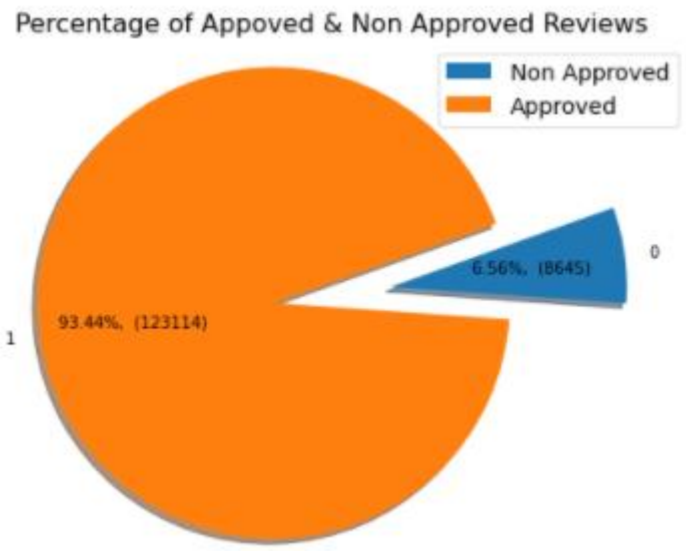
In order to fulfill the above rules, we had to add some extra reviews with specific content. First, we thought of adding toxic content in the dataset from YELP. Yelp is an American public company headquartered in San Francisco, California. The company develops, hosts, and markets the Yelp.com website and the Yelp mobile app, which publish crowd-sourced reviews about businesses. We realized that this method would give a lot of useless information and would not be good for our model, as our case study is quite specialized, so we rejected it. Instead we manually created about 128 records with the restricting rules of the company’s policy.

Finally, we more than doubled the toxic comments, following the below procedure. We translated the texts/entries in English and then using the **text attack** python’s tool, we created extra entries based on the synonyms and then we translated it back in Greek.



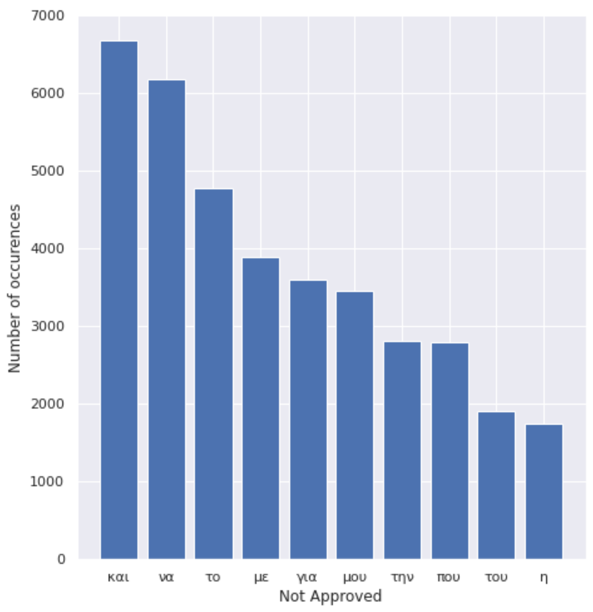
[1] https://www.doctoranytime.gr/us/reviews-policy

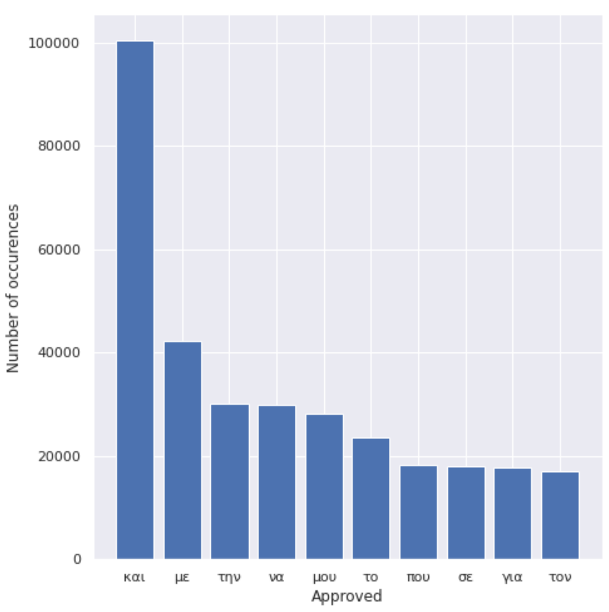
The final dataset that was created after the above manipulation consists of 123114 reviews that were approved and 8645 not approved reviews. Still the gap in the percentage of each class does exist, but we are going to tackle this problem later on.



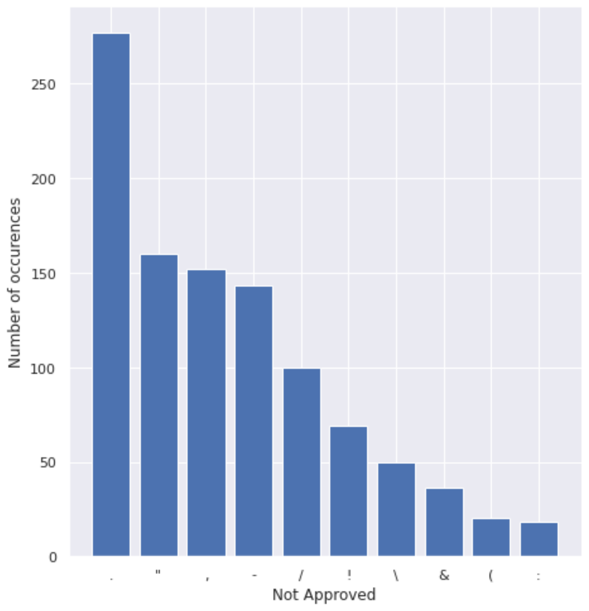
Dividing the consolidated text into tokens, we find the most common words which will most likely be the Stop words. For tasks like text classification, where the text is to be classified into different categories, stop words are removed or excluded from the given text so that more focus can be given to those words which define the meaning of the text.

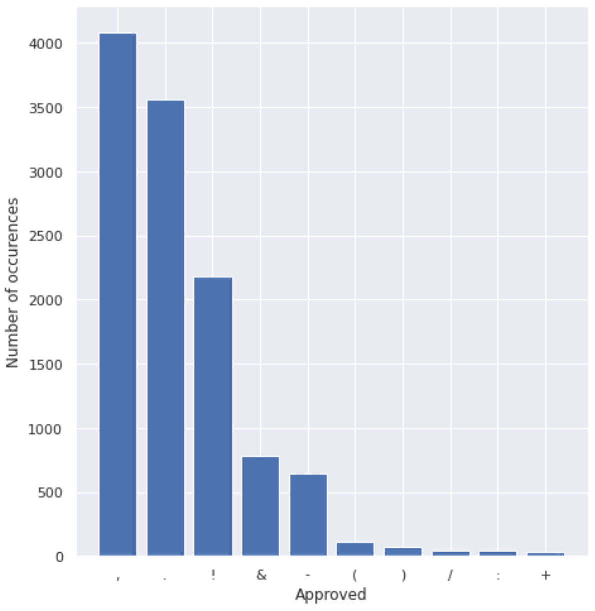
We can observe that the most used words are common in the two classes and depicted below for the class 0 and 1 respectively.





We can observe that the most used punctuations are common in the two classes and depicted below for the class 0 and 1 respectively.





Because individual words are not able to convey the meaning of a sentence, we identify the trigram that is three consecutive words in a sentence to better extract the meaning.

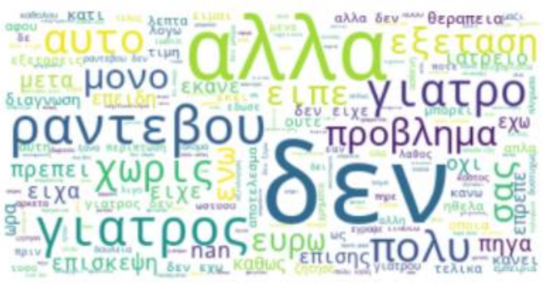
Most frequent tri-grams:

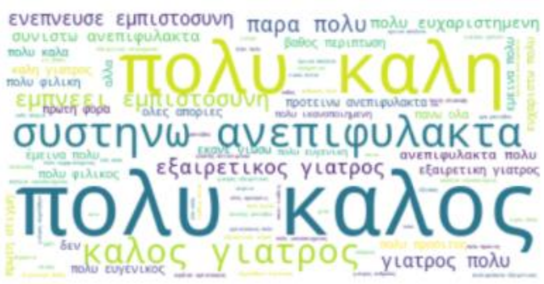
```
"(('πολύ', 'καλός'), 'γιατρός')": 3446
"(('τον', 'συστήνω'), 'ανεπιφύλακτα')": 3206
"(('μου', 'ενέπνευσε'), 'εμπιστοσύνη')": 2909
"(('την', 'περίπτωση'), 'μου')": 2011
"(('το', 'πρόβλημα'), 'μου')": 1871
"(('την', 'περίπτωσή'), 'μου')": 1857
"(('με', 'έκανε'), 'να')": 1776
"(('τον', 'συνιστώ'), 'ανεπιφύλακτα')": 1486
"(('πολύ', 'καλή'), 'γιατρός')": 1390
"(('έμεινα', 'πολύ'), 'ευχαριστημένη')": 1360
"(('με', 'την'), 'περίπτωση')": 1341
"(('στην', 'ώρα'), 'του')": 1334
"(('σου', 'εμπνέει'), 'εμπιστοσύνη')": 1308
"(('από', 'την'), 'πρώτη')": 1302
"(('σε', 'κάνει'), 'να')": 1236
"(('ο', 'γιατρός'), 'ήταν')": 1216
"(('το', 'πρόβλημά'), 'μου')": 1191
"(('με', 'το'), 'πρόβλημα')": 1176
"(('την', 'συστήνω'), 'ανεπιφύλακτα')": 1054
"(('θα', 'τον'), 'πρότεινα')": 1047
"(('τον', 'προτείνω'), 'ανεπιφύλακτα')": 1031
```

Data Preprocessing

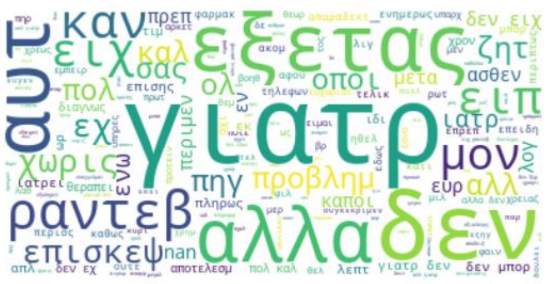
To supply the neural network with clean data we performed a series of preprocessing processes. We started with a list of stop words that had to be removed. Stop words are the words in any language which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. We removed the Greek Stop Words as well, keeping the word "δεν", as in Greek this word gives a negative meaning in the sentence and is quite important for classification as we may observe below.
Deleting NA values, tones, punctuation, emoji, URL, html and ascii were the next steps.

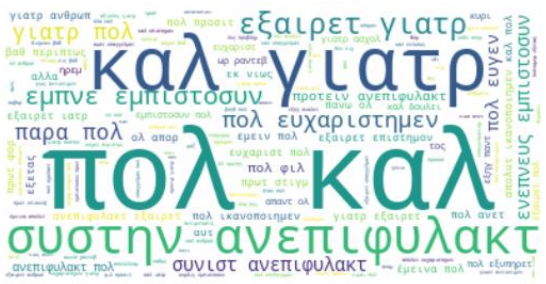
Having all the data cleaned we created a word cloud for both classes to observe which words are shown the most in the dataset. The below pictures show the most frequent word in our data for the class 0 (not approved reviews) and 1 (approved reviews) respectively.



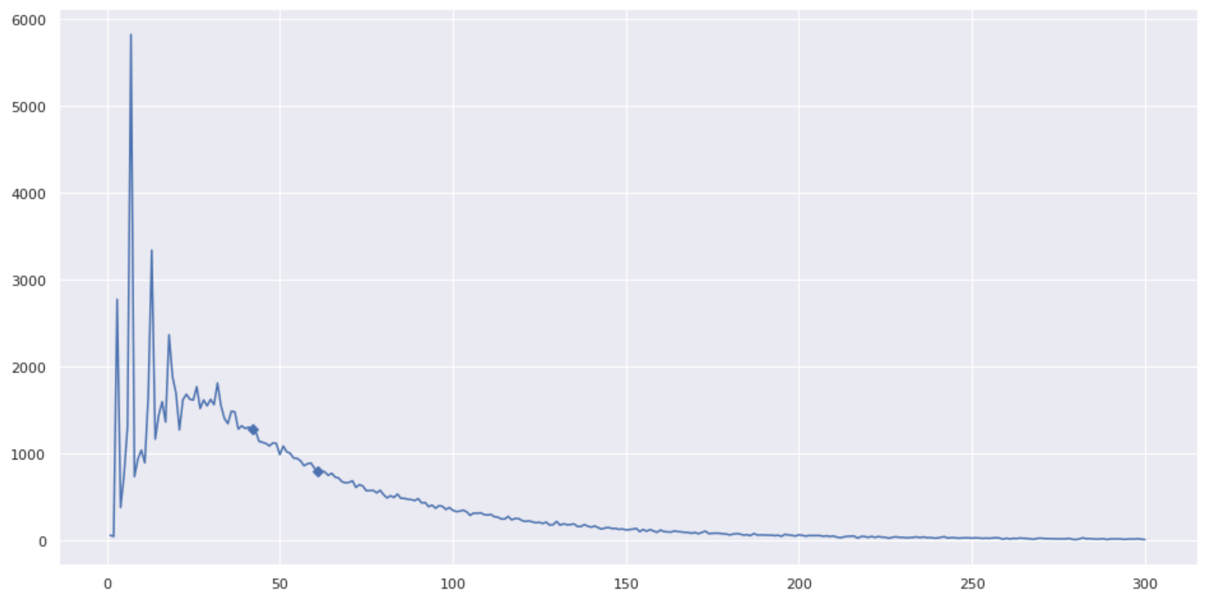


With stemming, the word cloud is formed as below:





For statistical reasons and in order to set LSTM's max_length parameter, we plotted below the mean and the median of the text length of the reviews , which are 60 and 41 respectively.



Methodology

Word Embedding

There are some contrasting reviews about the experts as well as the length and pace of the review. Clearly, there are a lot of interesting insights we can draw from them. However, we cannot simply give these sentences to a machine learning model and ask it to tell us whether a review was positive or negative. We need to perform certain text preprocessing steps. Bag-of-Words and TF-IDF are two examples of how to do this. These techniques could be used to vectorize a sentence at the beginning, retaining most of the linguistic information present in the sentence

Embedding is one such technique where we can represent the text using vectors. The more popular forms of word embeddings were implemented in the data are shown below:

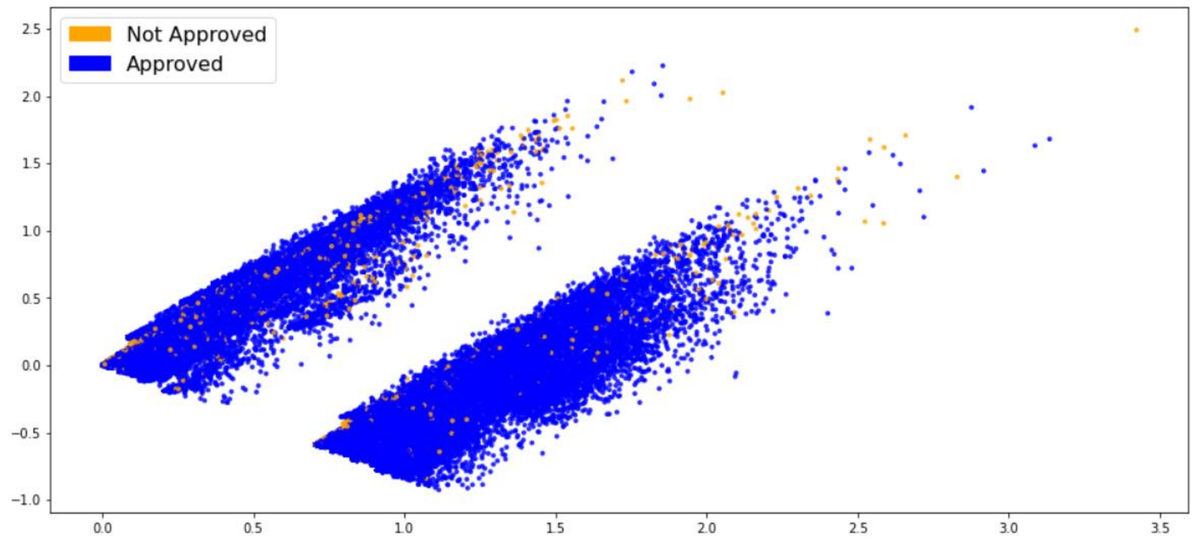
- **BoW, which stands for Bag of Words**

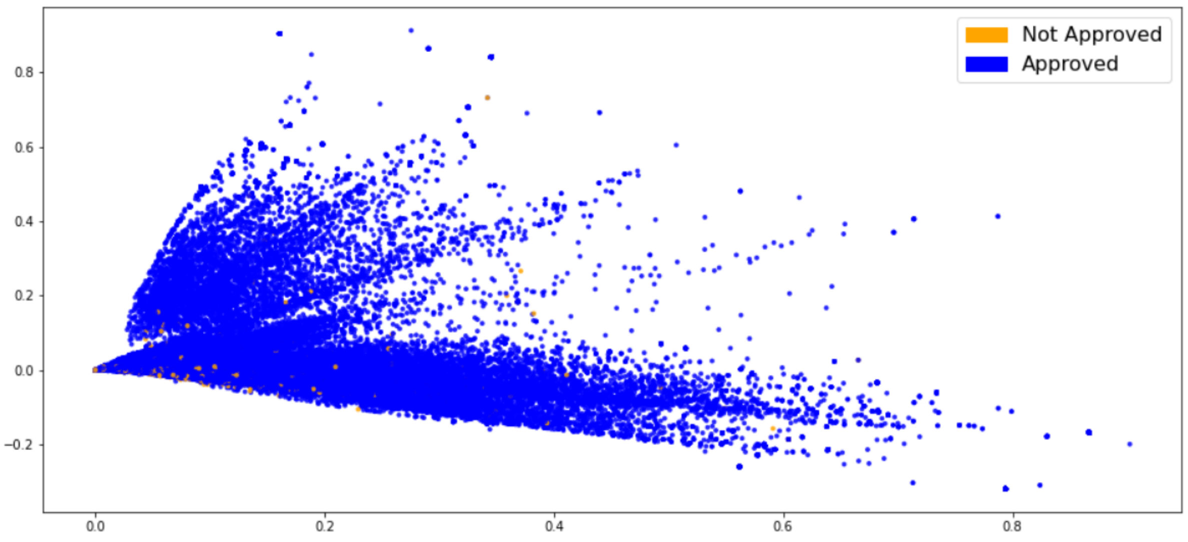
The Bag of Words (BoW) model is the simplest form of text representation in numbers. Like the term itself, we can represent a sentence as a bag of words vectors (a string of numbers).

- **TF-IDF (which stands for Term Frequency-Inverse Document Frequency)**

TFIDF or tf–idf is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

We compared the two above mentioned methods in using **Latent Semantic Analysis**(LSA). LSA is a technique for creating a vector representation of a document. Having a vector representation of a document gives you a way to compare documents for their similarity by calculating the distance between the vectors. This in turn means you can do handy things like classifying documents to determine which of a set of known topics they most likely belong to. Latent Semantic Analysis took BoW and TF-IDF one step further. We used a function that first uses SK-Learn's truncated SVD (LSA) class to transform the high dimensionality (number of columns) of the BoW embedding down to 2 dimensions. Then the two dimensions are used to plot each review, colored by the approval status (class). The first picture is depicting the BoW approach and the second picture the TF-IDF approach. As we see the first method gives more clear results.





Greek approach models

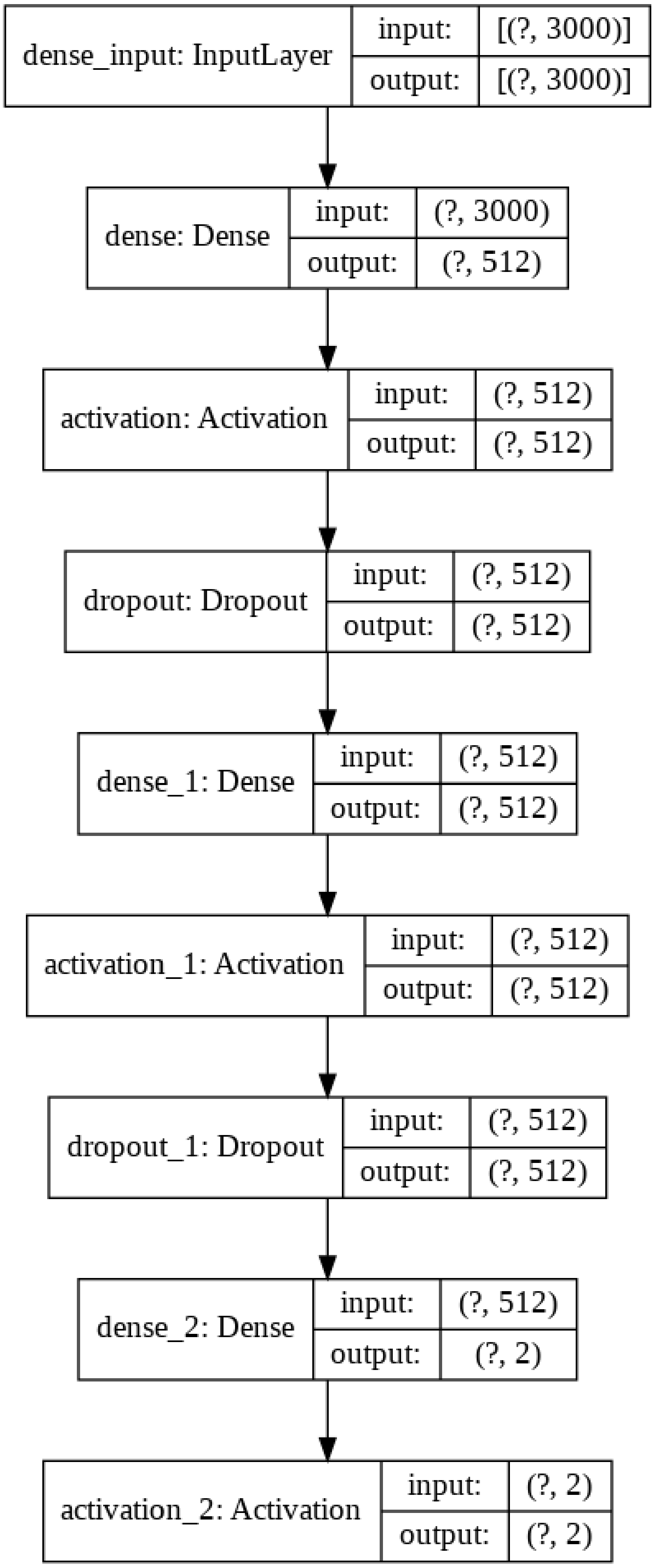
Feed Forward Neural Network

FFNN(Feed Forward Neural Network) model is the simplest form of artificial neural network. Information flows in one direction from first input layer to hidden layer to output layer. You can have any number of hidden layers with different sizes. Output layer in case of classification will be the same size as that of the number of classes(2 outputs as the number of the predicting classes). Firstly, we created an FFNN model that consists of the following parts:

1. Input layer(Feature selection technique)
2. Hidden layers (Dense layer (Relu technique))
3. Output layer. (Sigmoid classification technique)

Setting the maximum number of features at 3000, means that we have a vocabulary of 3000 words. We will use the *rectified linear unit* activation function referred to as ReLU on the first two layers and the Sigmoid function in the output layer. Because this is a binary classification problem, a common choice is to use the sigmoid activation function in a one-unit output layer to ensure our network output is between 0 and 1 and easy to map to either a probability of class 1 or snap to a hard classification of either class with a default threshold of 0.5. Also, dropout layers with a 40% dropout have been added to check for over-fitting. The last layer outputs a one hot encoded vector which gives the character output
Training of the model occurs over epochs(pass through all of the rows in the training dataset) and each epoch is split into batches(number of observations to propagate through the network before updating the parameters).

After defining the model, we compiled it. When compiling, we specified some additional properties required to find the best set of weights to map inputs to outputs in our dataset. We specified cross entropy as the loss argument, which is used for binary classification problems and is defined in Keras as “binary_crossentropy“. We defined the optimizer as the efficient stochastic gradient descent algorithm “adam“. This is a popular version of gradient descent because it automatically tunes itself and gives good results in a wide range of problems. Finally, because it is a classification problem, we collect and report the classification accuracy, defined via the metrics argument as “**binary_accuracy**”.
The shape of the final mode is depicted below.


dense_input: InputLayer
input: [(?, 3000)]
output: [(?, 3000)]
dense: Dense
input: (?, 3000)
output: (?, 512)
activation: Activation
input: (?, 512)
output: (?, 512)
dropout: Dropout
input: (?, 512)
output: (?, 512)
dense_1: Dense
input: (?, 512)
output: (?, 512)
activation_1: Activation
input: (?, 512)
output: (?, 512)
dropout_1: Dropout
input: (?, 512)
output: (?, 512)
dense_2: Dense
input: (?, 512)
output: (?, 2)
activation_2: Activation
input: (?, 2)
output: (?, 2)

Long short-term memory (LSTM)

Our second approach was using Long short-term memory (LSTM). LSTM is a special type of Recurrent Neural Network (RNN) that can learn long term patterns. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).

Having split the data set in training, validation and test, tokenizer does all the heavy lifting for us. It will take all the length of the corpus as most common words. After tokenization, the next step is to turn those tokens into lists of sequences. When we train neural networks for NLP, we need sequences to be in the same size, that’s why we use padding. Our max_length is 80, so we use pad_sequences to make all of our articles the same length which is 80. As a result, we will see that if a review was 70 in length, it becomes 80, a 2nd review was 53 in length, it becomes 80, and so on. In addition, there is padding_type and truncating_type, there are all “post”, means for example, if a review was 75 in length, we padded to 80, and we padded at the end, that is adding 5 zeros. Then we do the same for the validation and test sequences.

We build a tf.keras.Sequential model and start with an embedding layer. An embedding layer stores one vector per word. When called, it converts the sequences of word indices into sequences of vectors. After training, words with similar meanings often have similar vectors. We defined an Embedding layer with a vocabulary of all the corpus of the dataset, a vector space of 32 dimensions in which words will be embedded, and input documents that have 80 words each.

The Bidirectional wrapper is used twice with a LSTM layer, which propagates the input forwards and backwards through the LSTM layer and then concatenates the outputs. This helps LSTM to learn long term dependencies. We then fit it to a dense neural network to do classification. Since LSTM generally has the problem of overfitting, dropout can be applied.

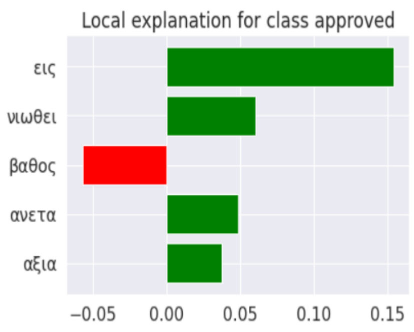
Finally, because this is a classification problem, we use a Dense output layer with a single neuron and a sigmoid activation function to make 0 or 1 predictions for the two classes (approved and not approved) in the problem. Because it is a binary classification problem, log loss is used as the loss function (binary_crossentropy in Keras). The efficient ADAM optimization algorithm is used.

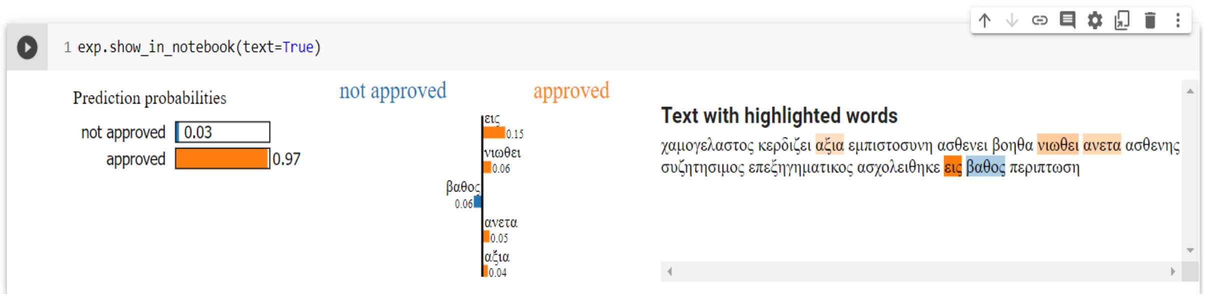
Classical Machine Learning with Logistic Regression Classifier

In this section we will present the classification results that were generated from a classical machine learning algorithm, Logistic Regression Classifier. The following table displays the key classification metrics (Accuracy/Precision/ Recall/ AUC). In our classification we will monitor the Precision of class 0 which makes more sense due to the nature of the problem at hand.

```
Logistic Regression Accuracy: 0.9565194642109669
Logistic Regression Precision: 0.9811162634784066
Logistic Regression Recall: 0.972693032015066
Logistic Regression AUC: 0.8266111662910869
```

But, what machine learning with Logistic Regression Classifier classifiers is doing? We installed the LIME package to generate labels for our two classes and see some of the visualization of the explanations. We randomly select a review in the test set, it happens to be a review of class 1, and our model predicts it as of class 1 as well. Notice that for each class, the words on the right side on the line are positive, and the words on the left side are negative.





```
1 exp.show_in_notebook(text=True)
```

Text with highlighted words

χαμογελαστος κερδιζει αξια εμπιστοσυνη ασθενει βοηθα νιωθει ανετα ασθενης συζητησιμος επεξηγηματικος ασχολειθηκε εις βαθος περιπτωση

English Approach models

RoBERTa

In addition to the above methods, we decided to work on English data, trying to obtain better results. After translating with text attack all the reviews in English, we used Robustly optimized BERT approach **RoBERTa**, a retraining of BERT with improved training methodology, 1000% more data and compute power. BERT (Bidirectional Encoder Representations from Transformers) is a technique for natural language processing (NLP) pre-training developed on large corpora and open-sourced by Google.

RoBERTa builds on BERT's language masking strategy and modifies key hyperparameters in BERT, including removing BERT's next-sentence pretraining objective, and training with much larger mini-batches and learning rates. RoBERTa was also trained on an order of magnitude more data than BERT, for a longer amount of time. This allows RoBERTa representations to generalize even better to downstream tasks compared to BERT.

The strong performance on RoBERTa on a broad range of NLP tasks is well-established, and various works have investigated the cause of these good results. One promising approach has been to study the structure of attention weights in the self-attention layers in BERT/RoBERTa and mapping it to the syntactic structure of text.

On top of RoBERTa, we used Simple Transformers which let us quickly train and evaluate Transformer models. Simple Transformers is a Natural Language Processing (NLP) library designed to simplify the usage of Transformer models without having to compromise on utility. Only 3 lines of code are needed to initialize a model, train the model, and evaluate a model. There are two task-specific Simple Transformers classification models, *Classification Model* and *MultiLabelClassificationModel*. The ClassificationModel class is used for all text classification tasks except for multi label classification.

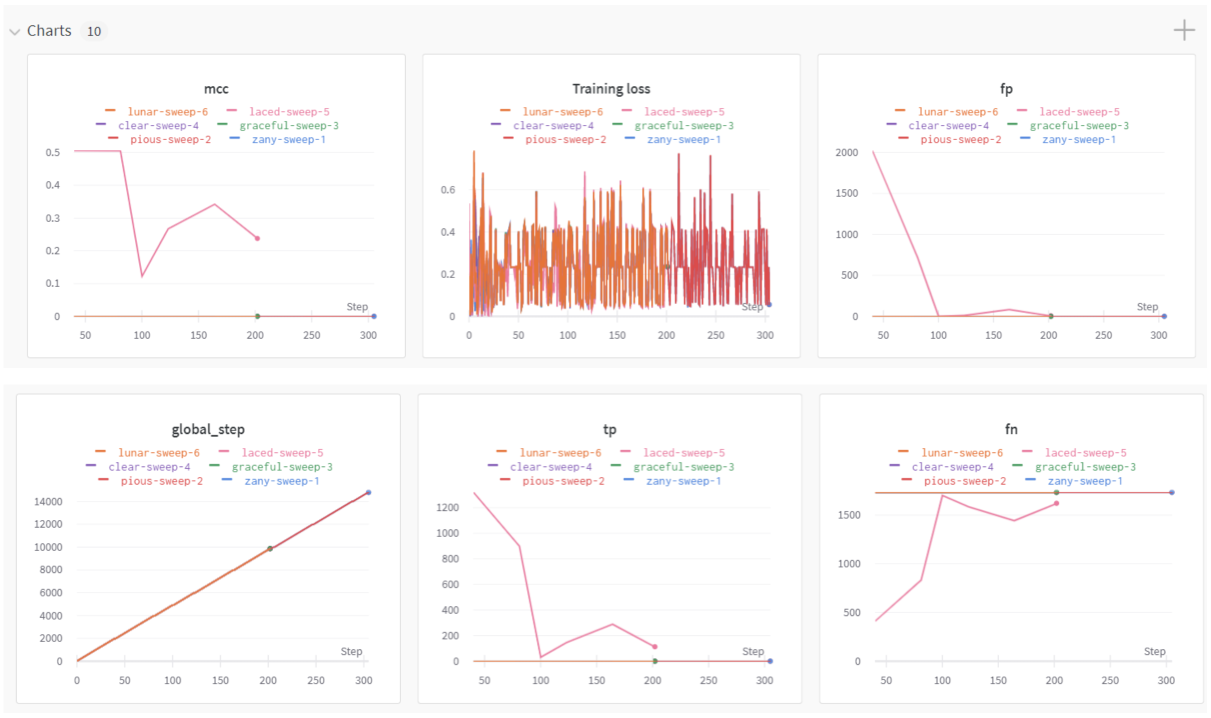
The data split ratio is 60:20:20, which consists of the first 60% of data for train, the data corresponding to the interval between 60% and 80% for validate and the last 20% corresponding to 80%-100% for test.

To create a ClassificationModel, we specified a model_type, which should be one of the model types from the supported models. As mentioned above, we used the pre-trained model

RoBERTa. Having the default simple transformers parameters and changing the overwrite_output_dir to True, the trained model will be saved to the ouput_dir and will overwrite existing saved models in the same directory. The model can be used for further prediction without re-running the whole notebook, since each epoch needs 25-30 minutes to run.

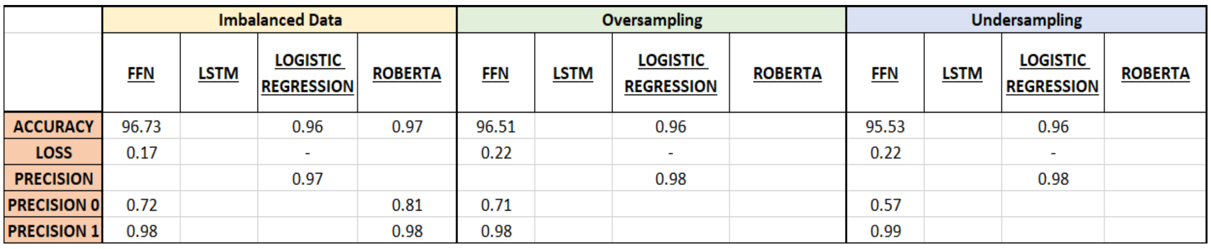
Hyperparameters optimization using Sweeps

We used Weights & Biases Sweeps to automate hyperparameter optimization and explore the space of possible models. Hyperparameter Sweeps offer efficient ways of automatically finding the best possible combination of hyperparameter values for our machine learning model with respect to the dataset. Choosing a good set of hyperparameter values played a huge role in developing of our model. Simple Transformers has native support for the W&B Sweeps feature for automated hyperparameter optimization.



Oversampling & Undersampling

The problem of unequal distribution of approved/ not approved reviews lead us changing the dataset in order to have more balanced data. There are two main methods to even-up the classes, Oversampling and Undersampling. **Undersampling** is the process where you randomly delete some of the observations from the majority class in order to match the numbers with the minority class and **Oversampling** is the process of generating synthetic data that tries to randomly generate a sample of the attributes from observations in the minority class (or more formally sampling with replacement). We have experimented with both methods on each model concluding that undersampling provides better results.



	Imbalanced Data				Oversampling				Undersampling			
	FFN	LSTM	LOGISTIC REGRESSION	ROBERTA	FFN	LSTM	LOGISTIC REGRESSION	ROBERTA	FFN	LSTM	LOGISTIC REGRESSION	ROBERTA
ACCURACY	96.73		0.96	0.97	96.51		0.96		95.53		0.96	
LOSS	0.17		-		0.22		-		0.22		-	
PRECISION			0.97				0.98				0.98	
PRECISION 0	0.72			0.81	0.71				0.57			
PRECISION 1	0.98			0.98	0.98				0.99			

Results

Learning Curves

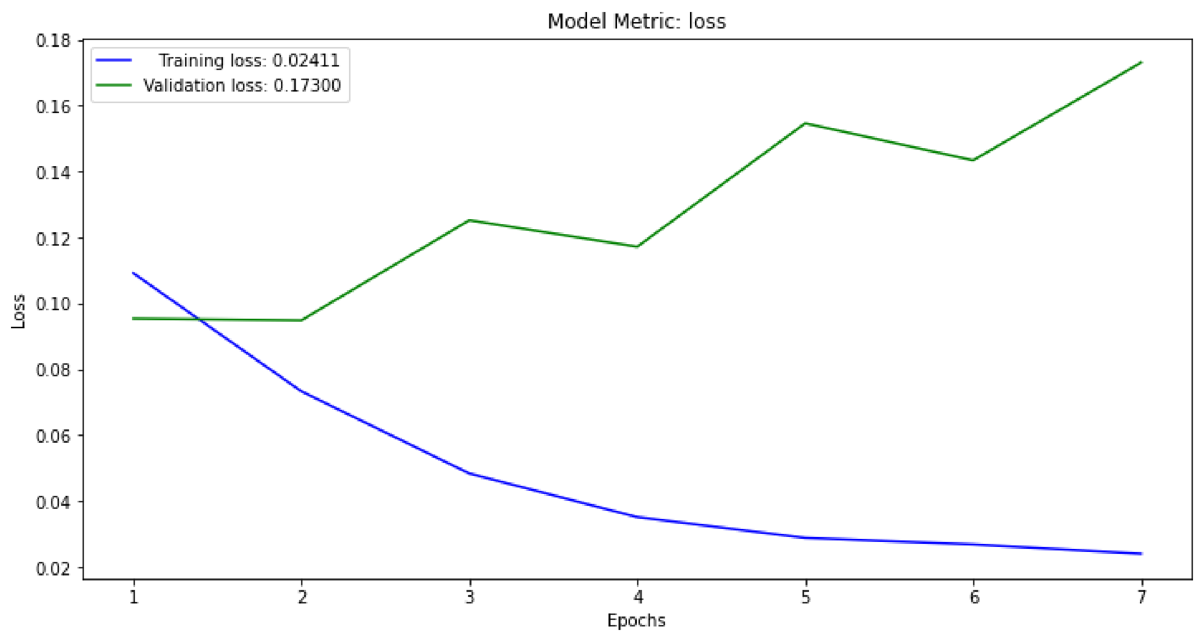
A learning curve is a plot of model learning performance over experience or time. Learning curves are a widely used diagnostic tool in machine learning for algorithms that learn from a training dataset incrementally. The model can be evaluated on the training dataset and on a holdout validation dataset after each update during training and plots of the measured performance can be created to show learning curves.

During the training of our machine learning models, the current state of each model at each step of the training algorithm is being evaluated. It is being evaluated on the training dataset to give an idea of how well the model is "*learning*." It is also being evaluated on a hold-out validation dataset that is not part of the training dataset. Evaluation on the validation dataset gives an idea of how well the model is "*generalizing*."

Thus, in the below diagrams the *Train Learning Curve* is calculated from the training dataset that gives an idea of how well the model is learning whereas the *Validation Learning Curve* is calculated from a hold-out validation dataset that gives an idea of how well the model is generalizing.

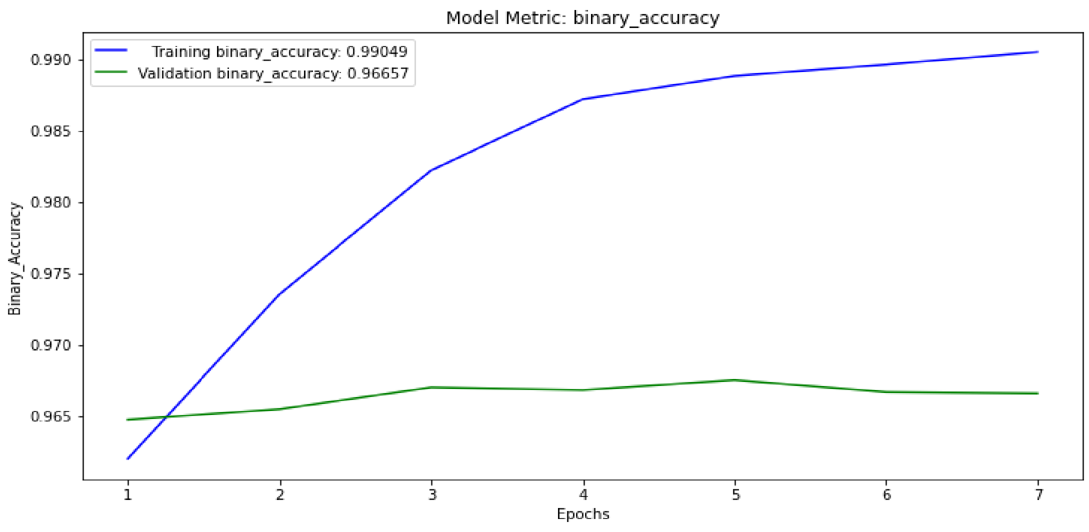
The models are optimized according to cross-entropy loss and their performance is evaluated using classification accuracy. In this case, two plots are created, one for the learning curves of each metric, and each plot can show two learning curves, one for each of the train and validation datasets.

Feed forward model



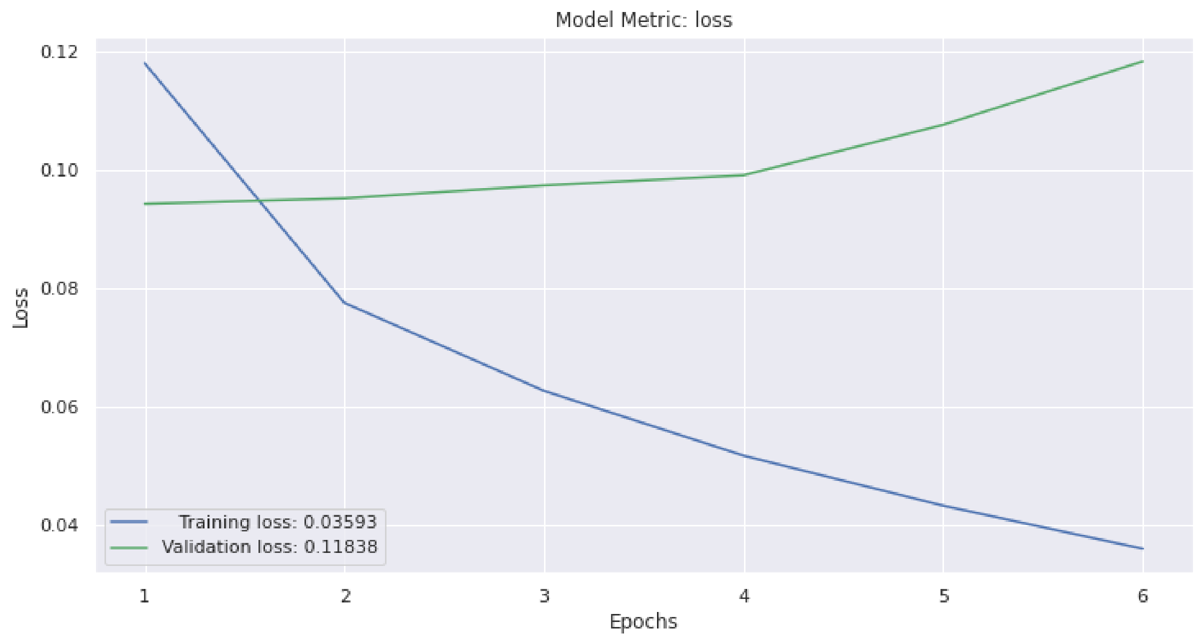
The shape and dynamics of a learning curve can be used to diagnose the behavior of the feed forward model and in turn perhaps suggest the type of configuration changes that may be made to improve learning and/or performance. From the above metric, the loss, we can observe a flat line in the first epochs and a growth trend of the loss values from the 5th epoch indicating that the model was unable to learn the training dataset at all. This may happen because the model does not have a suitable capacity for the complexity of the dataset and model that cannot learn the training dataset.

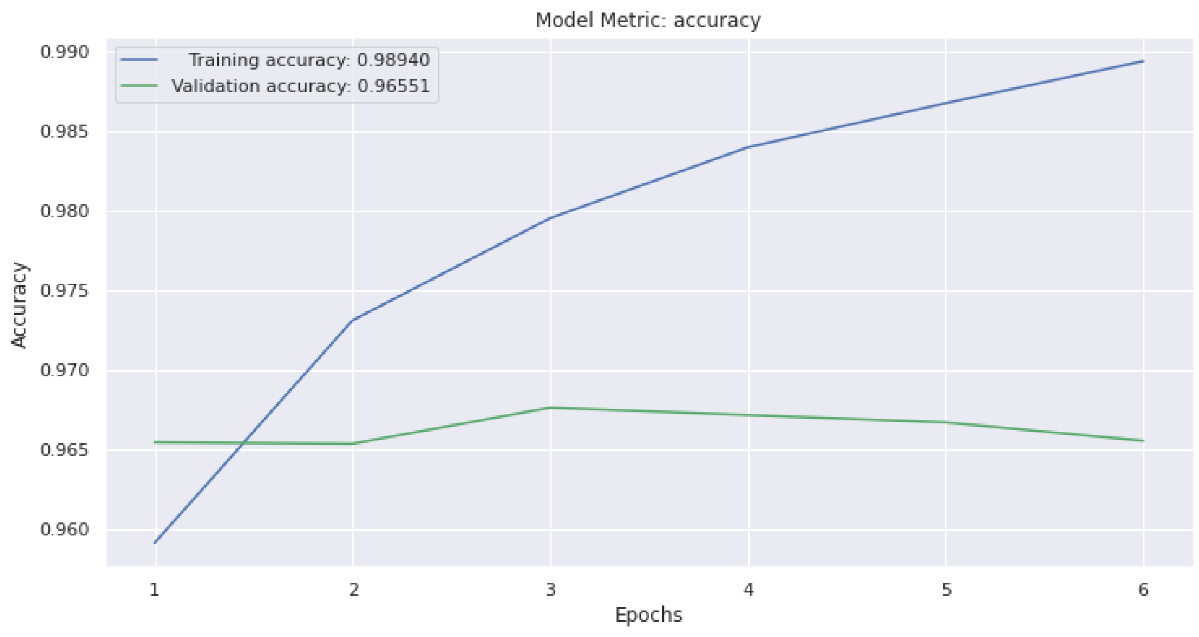
Regarding model accuracy, in the majority of epochs, validation accuracy is constantly less than the training dataset and goes lower as we move on from one epoch to another. This means that the model is not well-trained and not learning.



LSTM model

Concerning LSTM model and after observing the below diagrams we can conclude that the model has memorized the large class (approved reviews) and is returning that class achieving high accuracy. We probably only need 1 or 2 epochs. At the end of the training, we can see that there is a little bit overfitting.

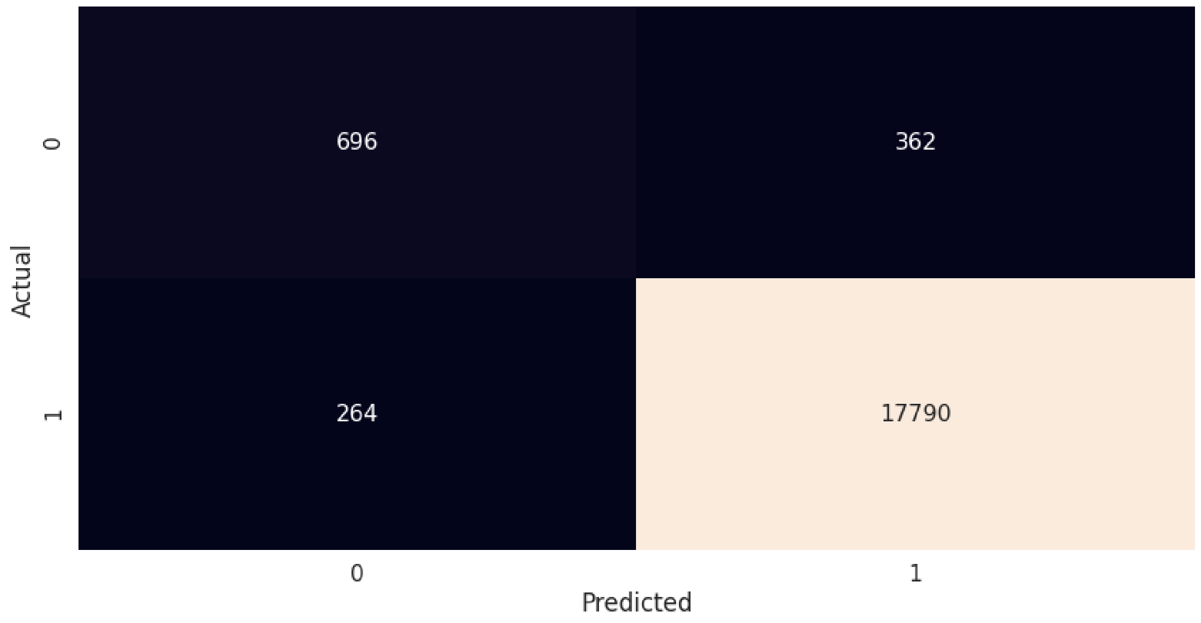




Confusion matrix & Classification Report

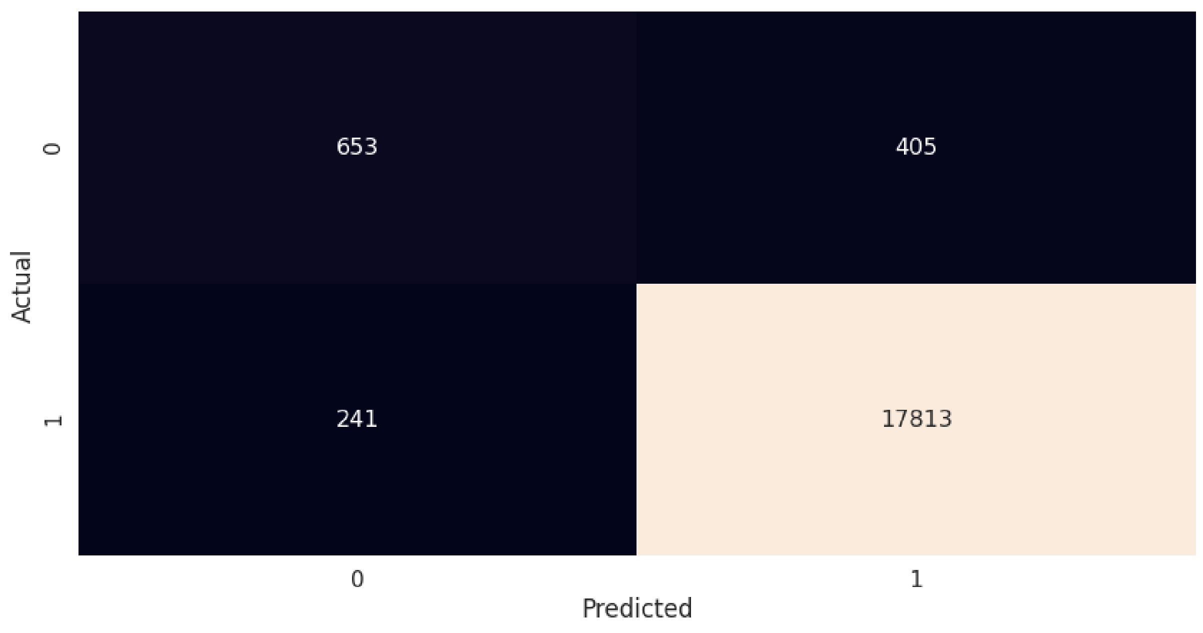
Confusion matrix is used as a performance measurement for machine learning classification problems where the output can be two or more classes. In the below tables, 4 different combinations of predicted and actual values are depicted.

Feed forward model

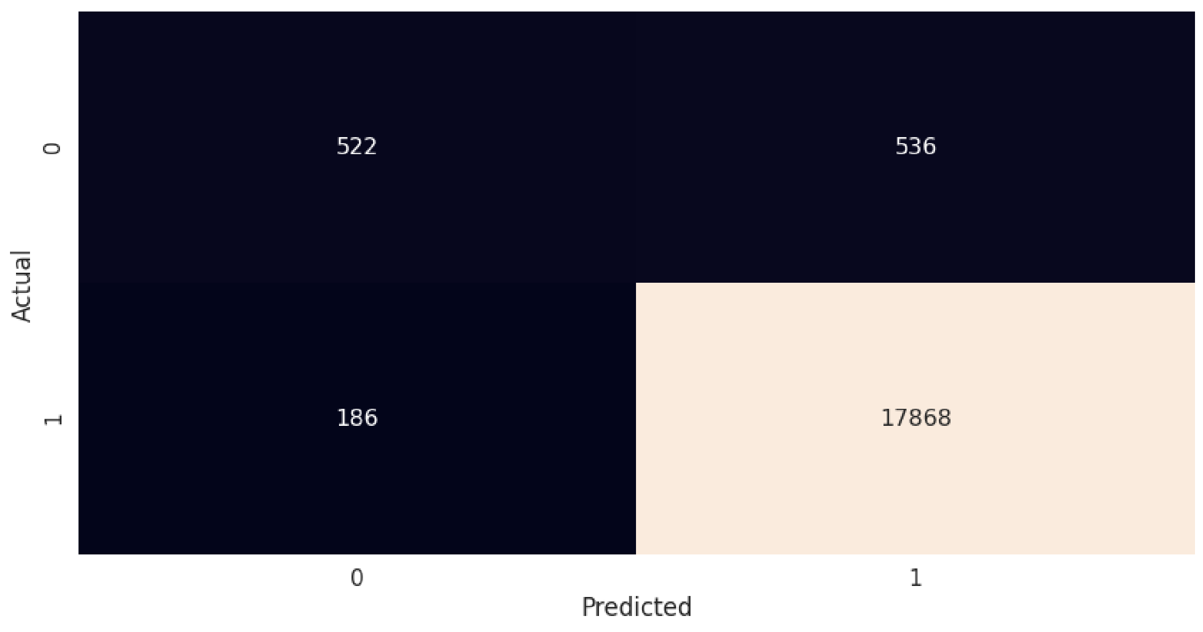


```
              precision    recall  f1-score   support

           0       0.72      0.66      0.69      1058
           1       0.98      0.99      0.98     18054

    accuracy                           0.97     19112
   macro avg       0.85      0.82      0.84     19112
weighted avg       0.97      0.97      0.97     19112
```

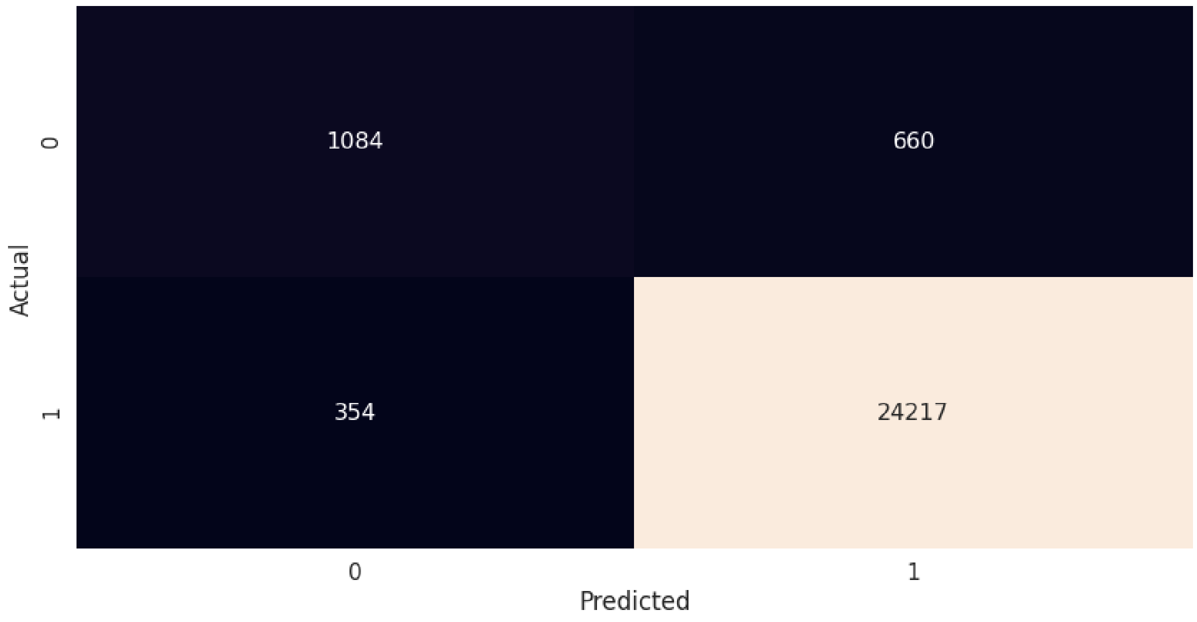
LSTM



Logistic regression

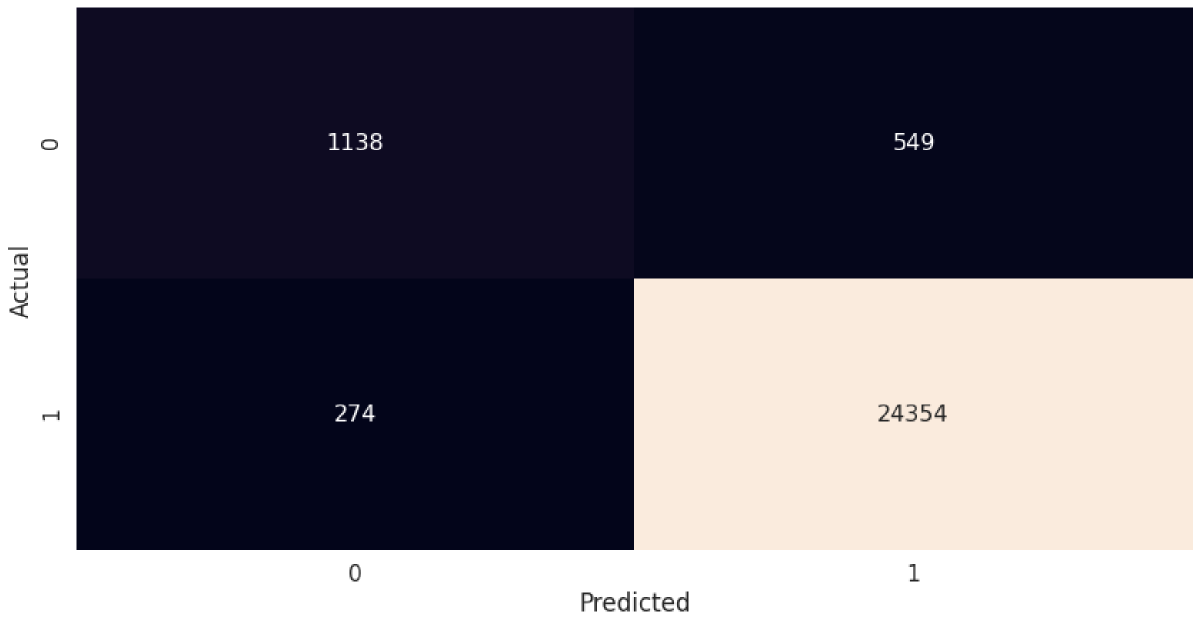


RoBERTa (1 epoch)



```
              precision    recall  f1-score   support

           0       0.75      0.62      0.68      1744
           1       0.97      0.99      0.98     24571

    accuracy                           0.96     26315
   macro avg       0.86      0.80      0.83     26315
weighted avg       0.96      0.96      0.96     26315
```

RoBERTa (2 epochs)



```
              precision    recall  f1-score   support

           0       0.81      0.67      0.73      1687
           1       0.98      0.99      0.98     24628

    accuracy                           0.97     26315
   macro avg       0.89      0.83      0.86     26315
weighted avg       0.97      0.97      0.97     26315
```

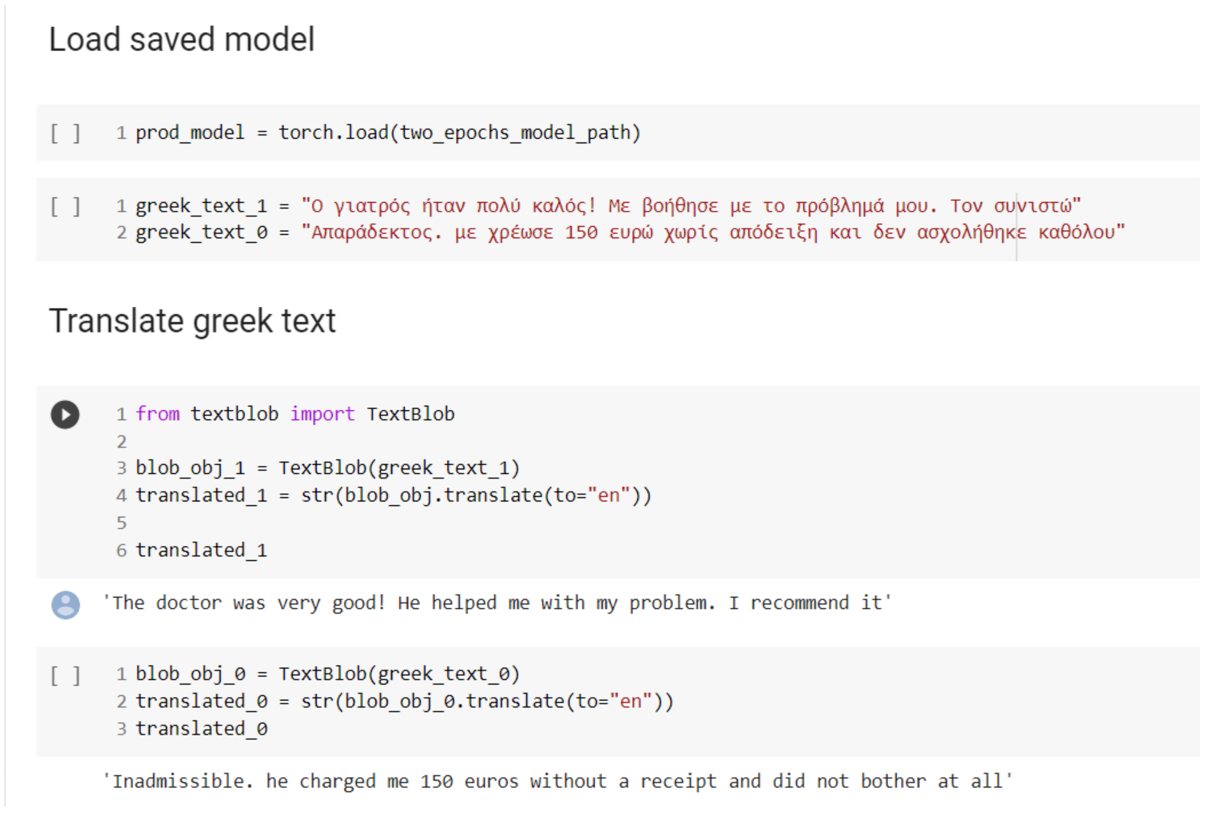
From the above generated classification reports and confusion matrices, we can see that high precision is not achieved for both classes. From a business prospect, we are more susceptible to the precision of class 0. This means that we would rather tolerate a mistake and let a review that is actually of class 1 and is predicted as of class 0 than vice versa. The precision for the class 0, which is the ratio of correct negative predictions to the total predicted negatives is higher in the RoBERTa model of 2 epochs. This makes sense, since the RoBERTa model is trained on large amounts of data and is really well pre-trained. Thus, we consider that other approaches would fit better for which more investigation would be needed.

The Google Colab is used to develop the X model on the GPU. The training process lasted about X hours. The major problem encountered during this assignment was related with the distribution of the reviews that are not represented accordingly in our dataset, as is often the case in most real-world classification problems. The data shows a strong level of class imbalance, which occurs when there are insufficient views of the data corresponding to any of the class tags. In our case, the approved reviews far outweigh the unapproved reviews. Due to this problem, a significant time effort was invested to increase the sample of rejected reviews.

We were wondering what stop words to set and while we had used some, like “δεν”, we came to the conclusion that these were crucial to the interpretation of the sentence and re-included them in the text.

Production

The results of the model are completely satisfactory, which allows us to test it in real data and make a first pilot production tool. We tried with two reviews, one of class 0 (non approved) and one of class 1 (approved) and we desire to examine if the right prediction will be made. The texts are translated into English and then fed into the tool. As we see in both cases we have correct results.



Load saved model

```
prod_model = torch.load(two_epochs_model_path)
```

```
greek_text_1 = "Ο γιατρός ήταν πολύ καλός! Με βοήθησε με το πρόβλημά μου. Τον συνιστώ"
greek_text_0 = "Απαράδεκτος. με χρέωσε 150 ευρώ χωρίς απόδειξη και δεν ασχολήθηκε καθόλου"
```

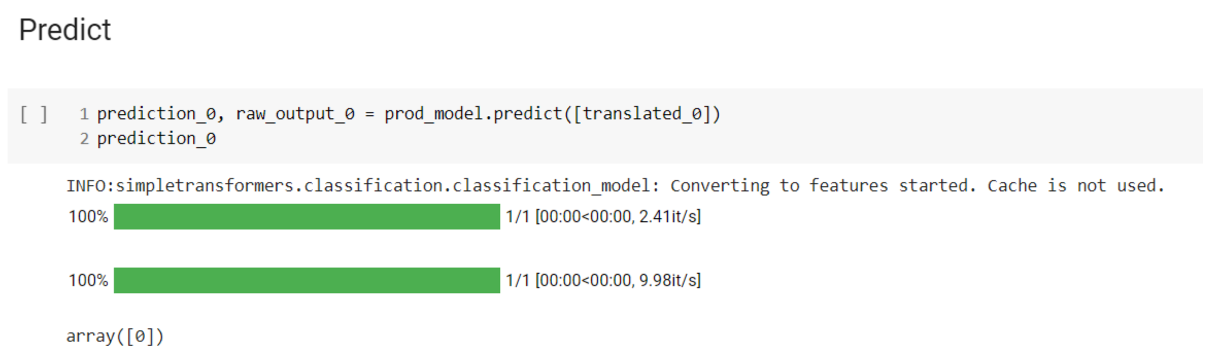
Translate greek text

```
from textblob import TextBlob

blob_obj_1 = TextBlob(greek_text_1)
translated_1 = str(blob_obj.translate(to="en"))

translated_1
```

```
'The doctor was very good! He helped me with my problem. I recommend it'
```

```
blob_obj_0 = TextBlob(greek_text_0)
translated_0 = str(blob_obj_0.translate(to="en"))
translated_0
```

```
'Inadmissible. he charged me 150 euros without a receipt and did not bother at all'
```



Predict

```
[ ]  prediction_0, raw_output_0 = prod_model.predict([translated_0])
     prediction_0
```

```
INFO:simpletransformers.classification.classification_model: Converting to features started. Cache is not used.
100% 1/1 [00:00<00:00, 2.41it/s]

100% 1/1 [00:00<00:00, 9.98it/s]

array([0])
```

Members / Roles

Our team consists of three members:

- Vangelis Christou who is CTO at Doctoranytime is a graduate student of Computer Science at Athens University of Economics & Business.
- Brikena Kokalari who is a graduate student of Accounting & Finance department at Athens University of Economics & Business with a specialization in Finance.
- Konstantinos Kolovos who is a graduate student of Applied Mathematical & Physical Sciences department at National Technical University of Athens with a specialization in Applied Mathematics.

The object of our work and the model we were called to create, required the cooperation of people with different backgrounds. The problems we were called upon to deal with, were many as it is a realistic business problem that has not been addressed before. In order to provide solutions to the various obstacles, there was a group contribution with multiple meetings in order to provide feasible and effective solutions.

Bibliography

1. "Doctoranytime.gr". How the system of evaluation works in doctoranytime. Retrieved 25 September 2020 from: https://www.doctoranytime.gr/us/reviews-policy?fbclid=IwAR0twUW4t6RWO3d9ten7z59EIVK_4TZMfDCkyOXqiC2ltKeArjaf4qBpl8
2. George S. (2020). Handling Imbalanced Datasets in Deep Learning. Towards Data Science. Retrieved 27 September 2020 from:

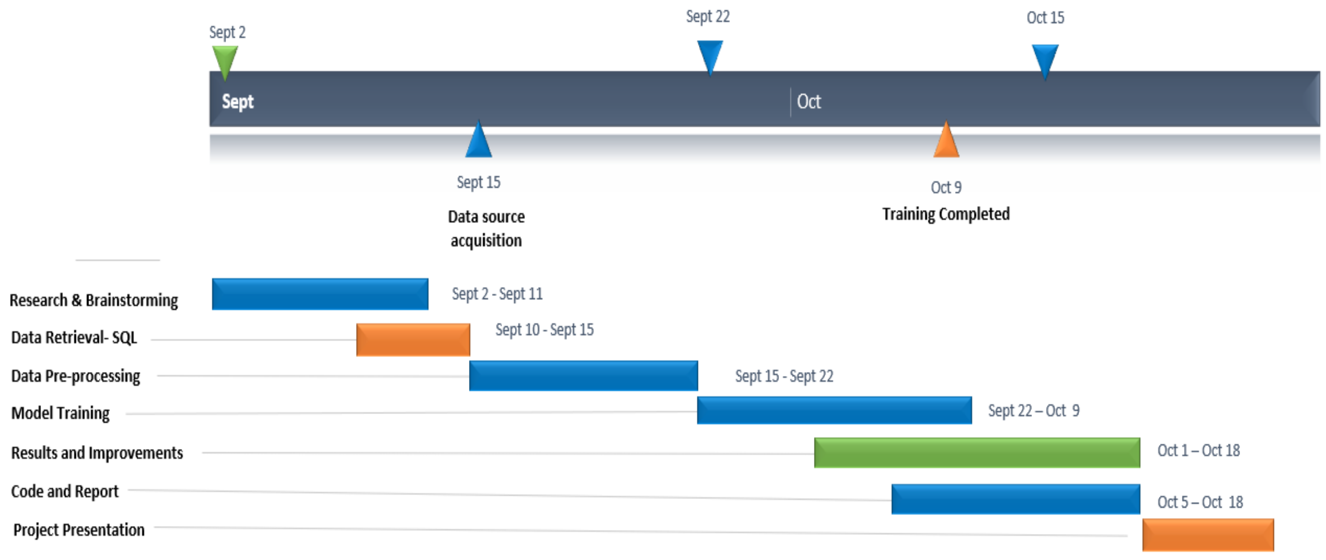
https://towardsdatascience.com/handling-imbalanced-datasets-in-deep-learning-f48407a0e758

3. Sayak P. (2018). Diving Deep with Imbalanced Data. “Datacamp.com”. Retrieved 27 September 2020 from: https://www.datacamp.com/community/tutorials/diving-deep-imbalanced-data?utm_source=adwords_ppc&utm_campaignid=898687156&utm_adgroupid=48947256715&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=332602034352&utm_targetid=aud-299261629574%3Adsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=9061578&gclid=CjwKCAjw8MD7BRArEiwAGZsrBXMmlIh4X-VYzooqJDUe8-2YvF_ltGyKqFkm_a9Ursqi6h7ktKu3qBoC-dgQAvD_BwE&fbclid=IwAR2GxQS2a5FKEl0ypwy4GsvynBAUveIU2mneEJP-mtZmN83dlB3DrDHopA8
4. Elite Data Science. How to Handle Imbalanced Classes in Machine Learning. Retrieved 27 September 2020 from: https://elitedatascience.com/imbalanced-classes
5. Morris J., Lifland E. et al. (2020). QData/TextAttack: TextAttack is a Python framework for adversarial attacks, data augmentation, and model training in NLP. Retrieved 15 September from: https://arxiv.org/pdf/2005.05909.pd
6. Diaz G. (2020). Stopwords ISO. Retrieved 21 September 2020 from: https://github.com/stopwords-iso/stopwords-el/blob/master/stopwords-el.txt
7. Basic text classification. Retrieved 21 September 2020 from: https://www.tensorflow.org/tutorials/keras/text_classification
8. Train, Test, Validate split Python. Three sets. Retrieved 24 September 2020 from: https://stackoverflow.com/questions/59077550/train-test-validate-split-python-three-sets?fbclid=IwAR3FlGU5VWv8BlJ0BBlckY5iMdPaUqxfgGZbLJhMqJig95hUDyaBVkwb4pY
9. Brownlee J. (2019). How to use Learning Curves to Diagnose Machine Learning Model Performance. Retrieved 28 September 2020 from: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/
10. Long short-term memory. Retrieved 28 September 2020 from: https://en.wikipedia.org/wiki/Long_short-term_memory
11. Anomaly detection. Retrieved 10 October 2020 from:

https://en.wikipedia.org/wiki/Anomaly_detection

12. “Medium.com”. Hyperparameter Optimization using sweeps with W&B. Retrieved 15 October 2020 from: https://medium.com/six-ways-to-debug-a-machine-learning-model/hyperparameter-optimization-using-sweeps-with-w-b-1e9d94dda46
13. Sweep da reviews: https://wandb.ai/vagelious/Sweep%20da%20reviews?fbclid=IwAR0WNfDm53HLVc3IYgsBtkBoI8YvezSU9IpUi7OEpIKofpYIIBT4gJcrCes&workspace=user-

Time Plan



Comments

The model that brought the best results is Roberta with 2 epochs so far. In Word Embedding, it seems that the method Bow works better than Tf-idf. During our data collection period we have realized that there was a problem from the beginning with the database which was unbalanced in the two classes. In addition, the rejection class is not representative of the company's policies, i.e. there are reviews that have not been approved but do not break the rules. Often, users called the company and asked for their review to be dropped. This creates a problem with data clarity. In the future, there should be different labeling for reviews that violate the policy and for reviews that do not break the rules but for some reason do not go up. Moreover, it would be useful to apply anomaly detection while in the pre-processing of the data to remove observations that deviate greatly from the majority of the data. Also, in our future plans we want to include a word corrector-predictor function, to correct misspelling words e.g. “ιαρτός”--> ”ιατρός”. We are not machine learning analysts; however, we understand the situation and can communicate with a team of data analysts and lead them to the final solution.