Machine Learning and Content Analytics

M.Sc. in Business Analytics (Full-Time)

Department of Management Science & Technology

# MACHINE LEARNING PROJECT

**Student1's Name:** Mikes Manos & **Student1's ID:** f2821908

**Student2's Name:** Nikolopoulos-Gkamatsis Filippos & **Student2's ID:** f2821909

**Student3's Name:** Panagiotatos Konstantinos & **Student3's ID:** f2821911

## Contents

## Figures

## 1. Description/ Mission

This project is an end to end machine learning project. The main purpose of which is to be able to predict the annotated label of the sentences of the abstract of published scientific papers. The possible labels are Claim, Evidence or No Label. A claim is defined as a statement deriving from findings about a scientific entity or process. Also, a claim should be supported by one or more evidence, so as to prove its veracity. With these in mind, the importance and the usage of such an algorithm can easily be identified.

Although there are infinite published papers, not all of the produce-provide a scientific claim which is supported by evidence. In other words the fact that a paper has been published does not automatically mean that it generates a breakthrough or even new information and perspective. This is the reason and the inspiration for the creation of the algorithm that will be presented within this report.

The Deep-Learning algorithm that was implemented for this assignment could be a very useful tool for whoever needs to validate the "strength" of a scientific paper quickly via only it's abstract. Although the interested parties could be numerous and will not be elaborated the main ones could be the universities, the scientific journals and even more the government.

A very powerful example that highlights the potential strength of such a model is being unfold in front of us the days that this particular report is being written, the Covid-19 era. Nowadays multiple,tending to "infinite" papers are being published and are related to the virus. Reasonable enough, not all authors and researchers produce powerful insights through their papers. Thus, the suggested algorithm would be extremely useful for the government and its respective departments that are responsible for the measures and the treatment of the virus, in order to at least initially sort the papers that have claims supported by evidence and discard those that do not.

At this point it seems reasonable to break down the entire project into small pieces and to describe them by order of implementation. After identifying the goal of the project and the reasons led to it, it is necessary to also identify the steps followed. First of all, the data needed to be found, so multiple papers were collected from some sources that will be mentioned in the following chapter. After succeeding in the data collection, the next necessary task was to pre-process these papers, in order to produce the dataset. Based on the papers collected, the dataset contains all sentences, as well as their labels. Afterwards, the dataset was again processed in order to be valid as a model input, which will try to predict the label of a paper. Finally, the neural network models were created and further details about all these, as well as the results will be presented in the following chapter.

## 2. Data

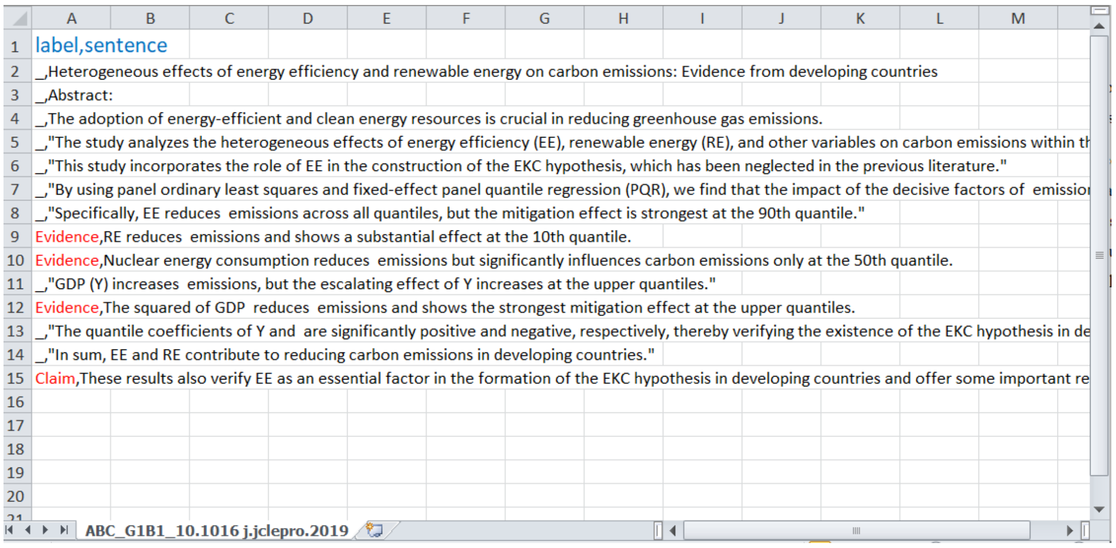
The data that were used for this project were actual published scientific papers. It should be mentioned at this point that the data collection part of the project was not executed entirely from our research team. Multiple other teams that are consisted of graduate students of the Msc in Business Analytics from Athens University of Economics and Business assisted us in the data collection and annotation phase.

It was necessary for the algorithm to include papers from different scientific aspects for accuracy reasons. Thus, each team extracted abstracts from published papers from different areas. The areas that were searched were inspired from the Sustainable Development Goals as they were enunciated from the United Nations (https://www.sightsavers.org/policy-and-advocacy/global-goals/?gclid=CjwKCAjwzvX7BRAeEiwAsXExo32UdU83PxJp6CdNA9K9kYPCPeKmsOR1lw9egebt-Lly_xneNaWu3hoCPAwQAvD_BwE).

More specifically the areas that were searched were: Good health (SDG3), Gender equality (SDG5), Clean energy (SDG7), No inequality (SDG10), Responsible consumption (SDG12) and Climate change (SDG13). In addition, the websites that were used for the abstract extraction were: arXiv.org ,pubmed.gov and semanticscholar.org.

After the extraction of all these abstracts each team member had to annotate the sentences of the abstract. In other words, he\she had to label each sentence as Claim, Evidence or No Label. This was achieved using INCEpTION of the clarin:el research infrastructure (https://inventory.clarin.gr/login/?next=/). Caution was taken for a "safety net" so that someone's annotation mistake would not have an effect. In order for a label to be finally kept, two out of three annotators had to agree in the label given at the annotation phase. In addition, abstracts that did not have a claim according to two out of three annotators were not kept also. The process ended with 889 annotated abstracts.

By the end of the annotation phase, the final dataset of the project was created by the remaining papers. Responsible for the creation of the dataset in its final structure was Mr. Aris Fergidis, who is a doctorate student of the University and was the project coordinator through the entire search. So the input (datasets) that were used in the project were 889 csv files that contained two columns which were label and sentence. A small example is being demonstrated below.

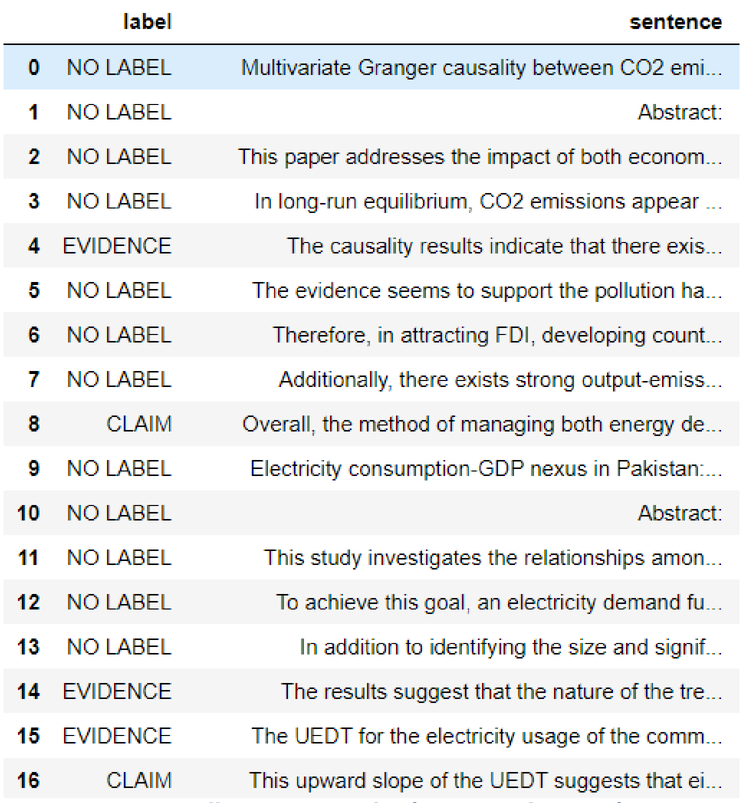


| | A |
|---|---|
| 1 | label,sentence |
| 2 | _,Heterogeneous effects of energy efficiency and renewable energy on carbon emissions: Evidence from developing countries |
| 3 | _,Abstract: |
| 4 | _,The adoption of energy-efficient and clean energy resources is crucial in reducing greenhouse gas emissions. |
| 5 | _,"The study analyzes the heterogeneous effects of energy efficiency (EE), renewable energy (RE), and other variables on carbon emissions within th |
| 6 | _,"This study incorporates the role of EE in the construction of the EKC hypothesis, which has been neglected in the previous literature." |
| 7 | _,"By using panel ordinary least squares and fixed-effect panel quantile regression (PQR), we find that the impact of the decisive factors of emissio |
| 8 | _,"Specifically, EE reduces emissions across all quantiles, but the mitigation effect is strongest at the 90th quantile." |
| 9 | Evidence,RE reduces emissions and shows a substantial effect at the 10th quantile. |
| 10 | Evidence,Nuclear energy consumption reduces emissions but significantly influences carbon emissions only at the 50th quantile. |
| 11 | _,"GDP (Y) increases emissions, but the escalating effect of Y increases at the upper quantiles." |
| 12 | Evidence,The squared of GDP reduces emissions and shows the strongest mitigation effect at the upper quantiles. |
| 13 | _,"The quantile coefficients of Y and are significantly positive and negative, respectively, thereby verifying the existence of the EKC hypothesis in de |
| 14 | _,"In sum, EE and RE contribute to reducing carbon emissions in developing countries." |
| 15 | Claim,These results also verify EE as an essential factor in the formation of the EKC hypothesis in developing countries and offer some important re |
| 16 | |
| 17 | |
| 18 | |
| 19 | |
| 20 | |

ABC_G1B1_10.1016 j.jclepro.2019

*Figure 2.1. Initial dataset's preview in Excel.*

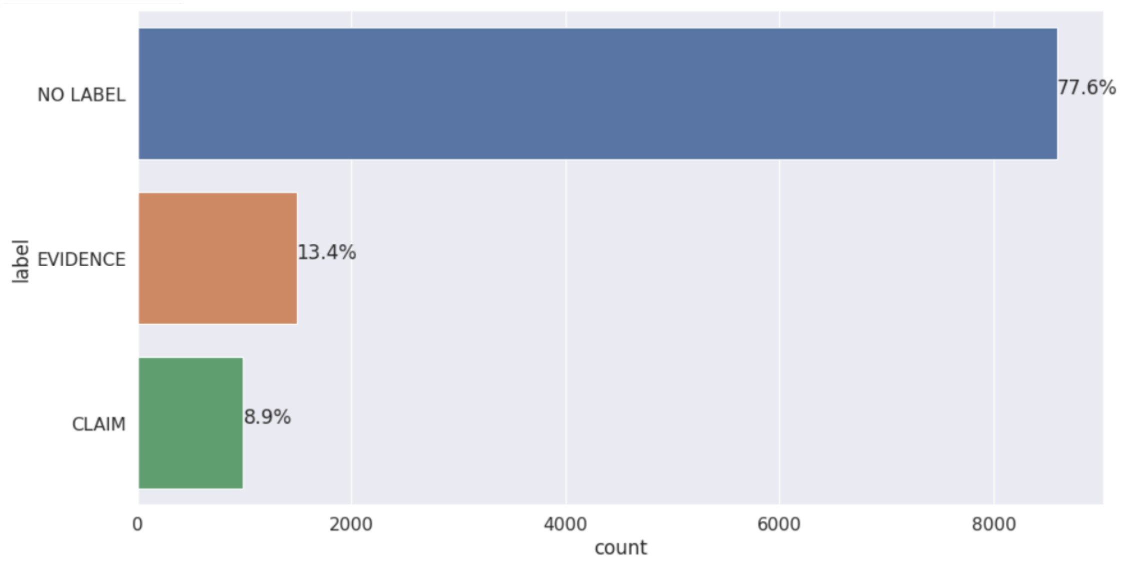
Thus, as also mentioned earlier a sentence can be labeled as a claim, evidence or nothing. Each one of these files was imported and then all of them were concatenated, in order to create the final dataset. This consisted of all sentences in these 889 scientific papers and their label. The first rows of the dataset are presented below:



| | label | sentence |
|---|---|---|
| 0 | NO LABEL | Multivariate Granger causality between CO2 emi... |
| 1 | NO LABEL | Abstract: |
| 2 | NO LABEL | This paper addresses the impact of both econom... |
| 3 | NO LABEL | In long-run equilibrium, CO2 emissions appear ... |
| 4 | EVIDENCE | The causality results indicate that there exis... |
| 5 | NO LABEL | The evidence seems to support the pollution ha... |
| 6 | NO LABEL | Therefore, in attracting FDI, developing count... |
| 7 | NO LABEL | Additionally, there exists strong output-emiss... |
| 8 | CLAIM | Overall, the method of managing both energy de... |
| 9 | NO LABEL | Electricity consumption-GDP nexus in Pakistan:... |
| 10 | NO LABEL | Abstract: |
| 11 | NO LABEL | This study investigates the relationships amon... |
| 12 | NO LABEL | To achieve this goal, an electricity demand fu... |
| 13 | NO LABEL | In addition to identifying the size and signif... |
| 14 | EVIDENCE | The results suggest that the nature of the tre... |
| 15 | EVIDENCE | The UEDT for the electricity usage of the comm... |
| 16 | CLAIM | This upward slope of the UEDT suggests that ei... |

*Figure 2.2. Full Dataset's sample after imported in Dataframe.*

Next, the frequency of each label needed to be calculated, so as to have a better view of the data. Thus, the following graph was created showing the percentage of each label:



*Plot 2.1. Frequency of each label occurrence in the whole dataset.*

According to the above graph, the majority of the sentences are labeled as no label. Specifically, 77.6% are not labeled neither as claim or evidence, In addition, it is observed that 13.4% of the sentences have been annotated as evidence and the 8.9% as claims. At this point, it should be mentioned that the model that will be trained must at least produce an accuracy higher than 77.6%. This percentage will be used as the minimum threshold, since is someone naively labeled all sentences as No Label he would accomplish an accuracy of that level. It should be also mentioned that the final shape of the data will not be presented here, but they will be reshaped in order to be valid inputs to the models. The tokenizations that were used in order to convert the sentences into vectors will be presented thoroughly in the next chapter of this report, since different techniques were used for different models.

# 3. Methodology

The main concept as far as the methodology is concerned was to implement various models with different architectures and search their hyper-parameters in order to find the one with the best behavior with our data. The techniques that were used where mainly from the Deep Learning canvas and secondary from the Machine Learning area. At this point it seems reasonable to dive in each different model area and explain its logic and present some basic information for the models that were used.

## 3.1 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

Numerous classifiers were explored and searched in order to clarify if they fit in an appropriate manner with our specific dataset. In order to avoid being tedious and repetitive not all of them were included in the report and in the code. Since many of them produced an extremely poor fit. Those that were explored and will not be commented for the aforementioned reasons are: Multinomial Naïve Bayes, Random

Forest, Decision Trees, Gaussian Naïve Bayes, LogisticRegression. It should be mentioned that for most of them it is rather logical and it was expected that the fit would not be good.

On the other hand there were two classifiers that produced some worth of mentioning predictive power. Those were the multi LinearSVC and the SGDClassifier. The multi linearSVC can output a categorical class value, and uses the OneVsAllModel meta-classifier to output any CategoricalAttribute value. It works by training n binary LinearSVC classifiers one for each given class and classifying an instance as a given class when one of the underlying LinearSVC classifiers reports.

The SGDClassifier is a linear classifier which implements regularized linear models with stochastic gradient descent (SGD) learning. Stochastic Gradient Descent (sgd) is a solver. It is a simple and efficient approach for discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression.

For both models there were two different approaches as far as the tokenization is concerned. Those two methods were the CountVectorizer which simply counts the word frequencies and the TFIDFVectorizer which differentiates since the value increases proportionally to count but is inversely proportional to frequency of the word in the corpus; that is the inverse document frequency (IDF) part.

Then the pipelines continued with the TfidfTransformer which computes the tf-idf scores for our sentences. Another part of our pipelines was the technique of oversampling by exploiting the RandomOverSampler. This seemed a good idea since the dataset of the project was rather imbalanced. Practically it did not improve the models significantly.

Furthermore, Cross validation was used in order to make our results more certain and concrete. Last but not least, Grid search was executed for the aforementioned models in order to discover their best parameters. The Grid search did help the model to improve its accuracy but only through the no-label sentences. It behaved poorly with the evidences and claims.

All in all, the best model from all the effort that was made from our team in the Machine learning neighborhood was the multi LinearSVC. The complete pipeline of the model contained the CountVectorizer, the TfidfTransformer, the RandomOverSampler and finally the model. The results of this model will be presented in the next chapter of this report. Although, it should be mentioned that they were not very impressive. Our next implementations will be models from the Deep Learning area.

## 3.2 CNN

The first model that was used was the well-known CNN model. In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

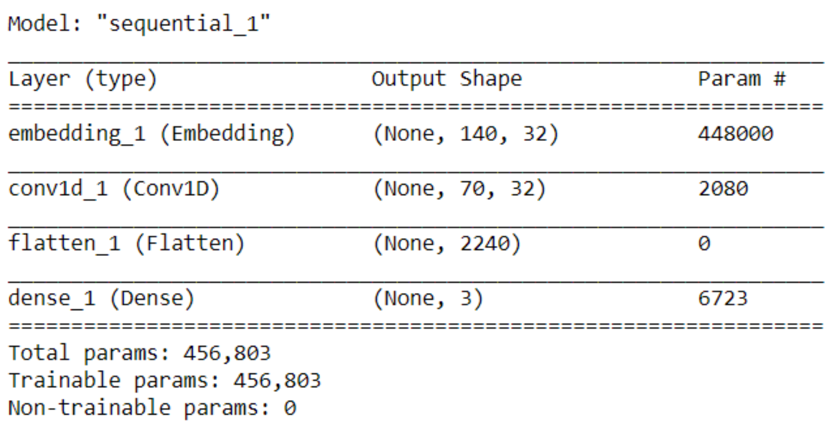
CNNs have also been explored for natural language processing problems, such as ours. CNN models are effective for various NLP problems and have achieved excellent results in semantic parsing, search query retrieval, sentence modeling, classification, prediction and other traditional NLP tasks.

The first step of this model's creation was to prepare and change the format and shape of the data. This includes some sub steps that will play a vital role in the deployment and the results of the model. The first of them was to clean the sentences by removing some unnecessary values and stopwords and punctuations. Subsequently, the dataset was split into train and test datasets, where the last one was the 20% of the original dataset or 2213 sentences.

After that, the train and test sentences were transformed via the Tokenizer, which found 11710 unique tokens. Then, the transformed data were padded, so as to have the same length with the sentence that contained the maximum number of words (140). It should be mentioned that the padding was executed in the end of each sequence, as it produces relatively better results.

As far as the sentences' labels are concerned, they were transformed with the usage of OneHotEncoder. A one hot encoder allows the representation of categorical data to be more expressive. Many machine learning algorithms cannot work with categorical data directly. The categories must be converted into numbers. More specifically, each sentence label was converted to an array, where the existence of 1 represented the correct label and the other two positions were represented by 0.

Consequently, the final shape of the train sentences and their labels were (8851,140) and (8851,3) respectively. Also, the part of the dataset kept for the final testing had a shape of (2213,140) and (2213,3) respectively. Last but not least, we created some class weights that were based on the size of each class in order to help the model to cope with the imbalanced dataset. At this point, the data pre-processing is done, and the input of the model is ready. The model that was finally deployed and kept since it produced the best results was the one that has the architecture that is demonstrated below.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 140, 32)           448000
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 70, 32)            2080
_________________________________________________________________
flatten_1 (Flatten)          (None, 2240)              0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 6723
=================================================================
Total params: 456,803
Trainable params: 456,803
Non-trainable params: 0
```

*Figure 3.1. Cnn's  model final shape.*

As it is shown from the above figure, the architecture is quite simple.  Various more complicated architectures were explored and tested but none of them produced better results. The structure of the model contains 4 layers. The first layer is the embedding layer. The Embedding layer is initialized with random weights and will learn an embedding for all of the words in the training dataset. The first input of this layer represents the input dimension. In other words, this is the vocabulary size (14000). Then, its next input is the output dimension, which represents the size of the vector space in which words will be embedded (32). Finally, the layer requires an input length, which in our case is the maximum number of words in a sentence (140). These inputs resulted in 448000 parameters.

The second layer is the convolutional layer. The layer defines a filter (or also called feature detector) of height 2 (also called kernel size). Only defining one filter would allow the neural network to learn one single feature in the first layer. This might not be sufficient, therefore we will define 32 filters. This allows us to train 32 different features on the first layer of the network. The output of the first neural network layer is a 70 x 32 neuron matrix. Each column of the output matrix holds the weights of one single filter. With the defined kernel size and considering the length of the input matrix, each filter will contain 70 weights.

The third layer is the necessary flattening component. Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully connected layer which produces 6723 parameters. The activation function that was used in the fully connected layer was the SoftMax function. SoftMax is a function that turns a vector of K real values into a vector of K real values that sum to 1. These are the probabilities of each sentence to be either No-label, evidence or claim. The total parameters of the model are 456.803.

Another very important aspect of the deployment of the model is the compiler that will be used. Our compiler exploited the usage of the Adam optimizer with a learning rate equal to 0.001 as well as the Accuracy metric and the categorical crossentropy loss. Other optimizers that were explored were: Adadelta ,Adamax , Nadam, Ftlr, RMSprop and SGD. From all those only the RMSprop and the Adam managed to predict claims and evidences. All the rest predicted every sentence as no-label. In addition, multiple other metrics were explored such as recall, precision, recall at precision and precision at recall but none of them produced anything worthy.

Now, the model has been entirely defined and built. The next logical step is to fit the model to the training data and define the proportion of the data that will be used for the validation within the training process. The percentage of the data that was used for that purpose was the 20%. In addition, the fit of the model included the class weights that were mentioned previously. Some other parameters that were used were the number of epochs that were 10 and the batch size which was 32. These numbers seem a bit strange since they are relatively small, but with higher numbers the results became only worse. Furthermore, we exploited the usage of a checkpoint which kept the best version of the model that was created within the different epochs. Lastly, the model was evaluated via the test dataset and the results will be demonstrated in the next chapter of this report.

## 3.3 MLP

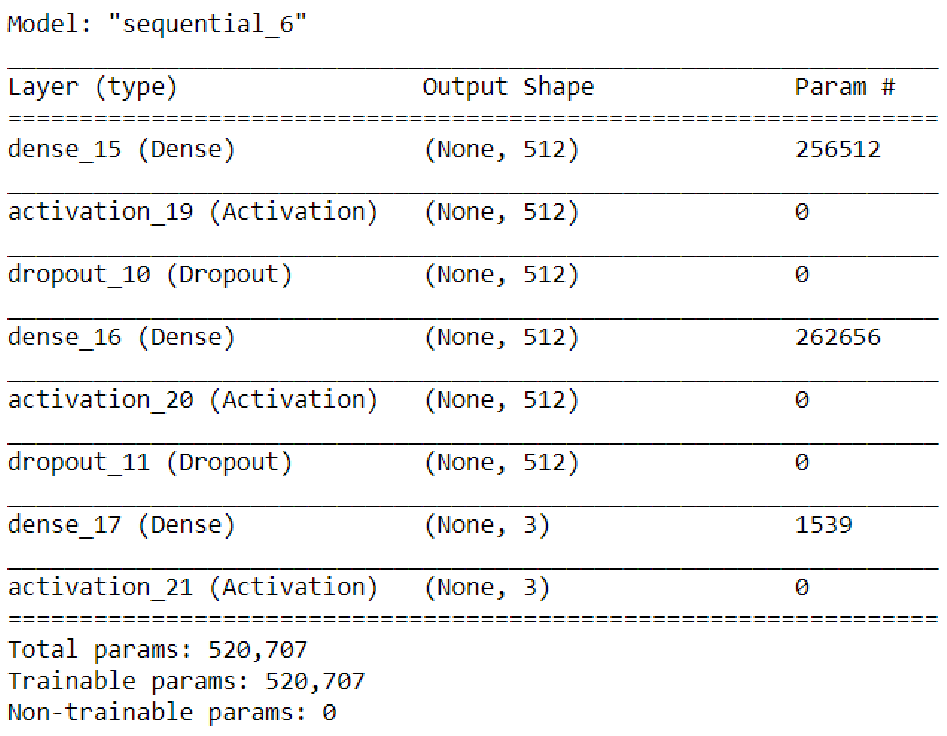
A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptrons (with threshold activation). Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable. The structure of the model is presented in the following figure:

The first step of this model's creation was, once again, to prepare and change the format and shape of the data. Most of the actions here were identical with the respective ones that were made in the CNN model. The first of them was to clean the sentences by removing some unnecessary values and stopwords and punctuations. Subsequently, the dataset was split this time into train, validation and test datasets, where the validation and test datasets each contained the 10% of the original dataset, 1106 and 1107 respectively

After that, the train and test sentences were transformed via the Scikit-learn's CountVectorizer. CountVectorizer is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text. With CountVectorizer we are converting raw text to a numerical vector representation of words and n-grams. This makes it easy to directly use this representation as features. The maximum words that were used as the vocabulary were 500. This number once again seems relatively small, but after a lot of search, it provided the best results.

As far as the sentences' labels are concerned, they were transformed with the usage of OneHotEncoder with the same way they were in the CNN model. More specifically, each sentence label was converted to an array, where the existence of 1 represented the correct label and the other two positions were represented by 0. Consequently, the final shape of the train sentences and their labels were (8851,500) and (8851,3) respectively. In addition, the part that was kept for the validation of the model had a shape of (1107,500) and (1107,3) respectively Also, the part of the dataset kept for the final testing had a shape of (1106,500) and (1106,3) respectively. At this point, the data pre-processing is done, and the input of the model is ready. The model that was finally deployed and kept since it produced the best results was the one that has the architecture that is demonstrated below.



```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_15 (Dense)             (None, 512)               256512    
_________________________________________________________________
activation_19 (Activation)   (None, 512)               0         
_________________________________________________________________
dropout_10 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_16 (Dense)             (None, 512)               262656    
_________________________________________________________________
activation_20 (Activation)   (None, 512)               0         
_________________________________________________________________
dropout_11 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_17 (Dense)             (None, 3)                 1539      
_________________________________________________________________
activation_21 (Activation)   (None, 3)                 0         
=================================================================
Total params: 520,707
Trainable params: 520,707
Non-trainable params: 0
```

*Figure 3.2.MLP's model final shape.*

According to the table above, the first layer consists of 512 neurons. Also, another input was necessary, and this is the maximum number of words, which is chosen to be equal to 500 after various tests. Then, the activation function follows before the dropout layer. The dropout step is required, because neural networks are likely to quickly overfit the train data. In fact, this is a computationally cheap and effective regularization method to reduce overfit. To explain this, dropout needs a rate that will define the percentage of nodes that will be dropped randomly during training.  In our case, this percentage was chosen to be equal to 35%. Then, these layers were again added one more time, before the activation of the SoftMax function, which will produce the probability of each sentence's label. Again, the usage of a checkpoint keeping the best version of the model that was created within the different epochs was found necessary.

As far as the compliler is concerned, the best optimizer was found to be RMSprop and the Accuracy metric among with the categorical crossentropy loss. Again, the other optimizers did not produce satisfying results, as most of them predicted all the sentences as No Labels.

Now, the model has been entirely defined and built. Some other parameters that were used were the number of epochs that were 30 and the batch size which was 700. Furthermore, we exploited the usage of an early stoper which actually stopped the deployment of the model when three different epochs did not conclude to an improvement in the validation loss of the prediction in the validation set. Validation loss is the value of cost function for your cross-validation data and loss is the value of cost function for our training data. The details about this model's predictive results will be demonstrated in detail in the next chapter.

## 3.4 LSTM (RNN)

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

The term "recurrent neural network" is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled.

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones. Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.

Long short-term memory (LSTM) is artificial recurrent neural network (RNN) architecture used in the field of deep learning. 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). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

The advantage of an LSTM cell compared to a common recurrent unit is its cell memory unit. The cell vector has the ability to encapsulate the notion of forgetting part of its previously stored memory, as well as to add part of the new information. To illustrate this, one has to inspect the equations of the cell and the way it processes sequences under the hood.

The data preprocessing phase of the LSTM model was exactly the same as the CNN model. Thus, in order to avoid being repetitive the details will not be presented thoroughly. The only difference was in the tokenization and padding phase of the were the length that was chosen was 40 instead of the 140 length that was chosen in the CNN model. Consequently, the final shape of the train sentences and their labels were (8851,40) and (8851,3) respectively. Also, the part of the dataset kept for the final testing had a shape of (2213,40) and (2213,3) respectively. Last but not least, we created some class weights that were based on the size of each class in order to help the model to cope with the imbalanced dataset. At this point, the data pre-processing is done, and the input of the model is ready. The model that was finally deployed and kept since it produced the best results was the one that has the architecture that is demonstrated below.

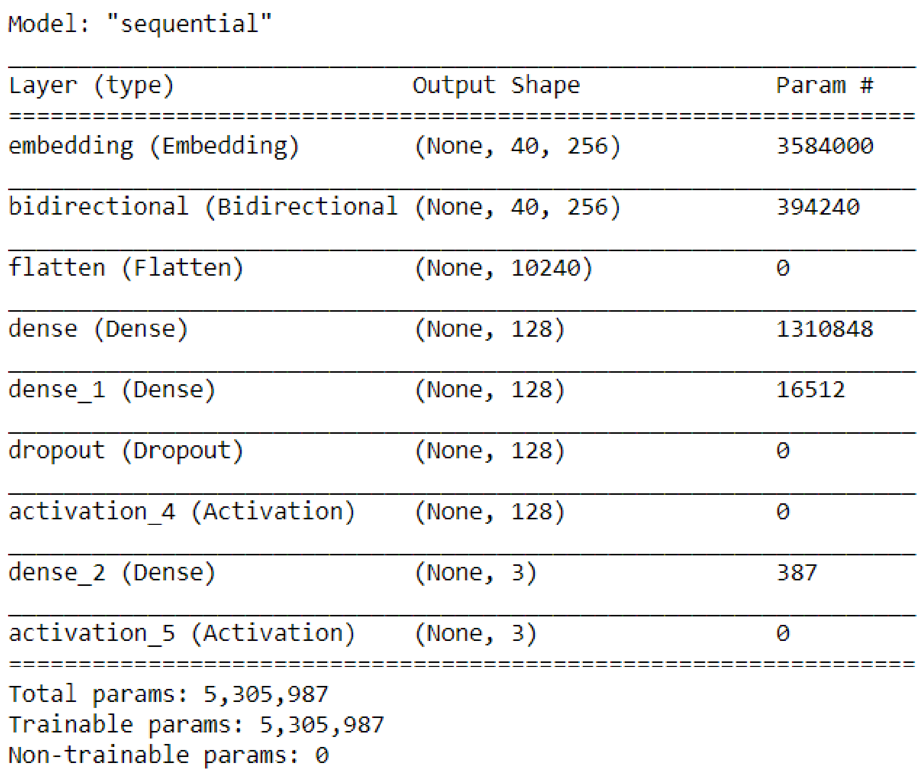

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding (Embedding)        (None, 40, 256)           3584000   
_________________________________________________________________
bidirectional (Bidirectional (None, 40, 256)           394240    
_________________________________________________________________
flatten (Flatten)            (None, 10240)             0         
_________________________________________________________________
dense (Dense)                (None, 128)               1310848   
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512     
_________________________________________________________________
dropout (Dropout)            (None, 128)               0         
_________________________________________________________________
activation_4 (Activation)    (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 387       
_________________________________________________________________
activation_5 (Activation)    (None, 3)                 0         
=================================================================
Total params: 5,305,987
Trainable params: 5,305,987
Non-trainable params: 0
```

*Figure 3.3. Rnn - LSTM's model final shape.*

As shown in the figure above, the first layer is an Embedding layer like the one used in the CNN Model. However, the number of words defined as the maximum in this case was equal to only 40. This is might seem a small number compared to the one used in the CNN model, but it was producing the best results. The size of the vocabulary representing the input dimension of this layer remained the same (14000). Also, the output dimension was changed to a much higher number equal to 256. Next, the bidirectional LSTM layer's inputs were chosen to be 128 units and a dropout rate equal to 50%. This led to 394240 parameters, which comes from the following formula: [(units + embedding_dimnesion + 1) * units] * 4*2.

The third layer is the flattening component. Thus, the data were converted into a 1-dimensional array, whose length is equal to 10240 (40*256) for inputting it to the next layer. After that, two more denses with dimension equal to 128 ,the relu activation function and the same dropout rate were added. The final dense's dimension was equal to the number of classes, so as to adjust the shape as a valid input of the SoftMax function.

Another very important aspect of the deployment of the model is the compiler that will be used. Our compiler exploited the usage of the Adam optimizer with a learning rate equal to 0.005 as well as the Accuracy metric and the categorical crossentropy loss. All the rest optimizers and metrics were once again tested but did not produce better results.

Now, the model has been entirely defined and built. The next logical step is to fit the model to the training data and define the proportion of the data that will be used for the validation within the training process. The percentage of the data that was used for that purpose was the 20%. In addition, the fit of the model included the class weights that were mentioned previously. Some other parameters that were used were the number of epochs that were 15 and the batch size which was 200. Furthermore, we once again exploited the usage of a checkpoint which kept the best version of the model that was created within the different epochs. Lastly, the model was evaluated via the test dataset and the results will be demonstrated in the next chapter of this report.

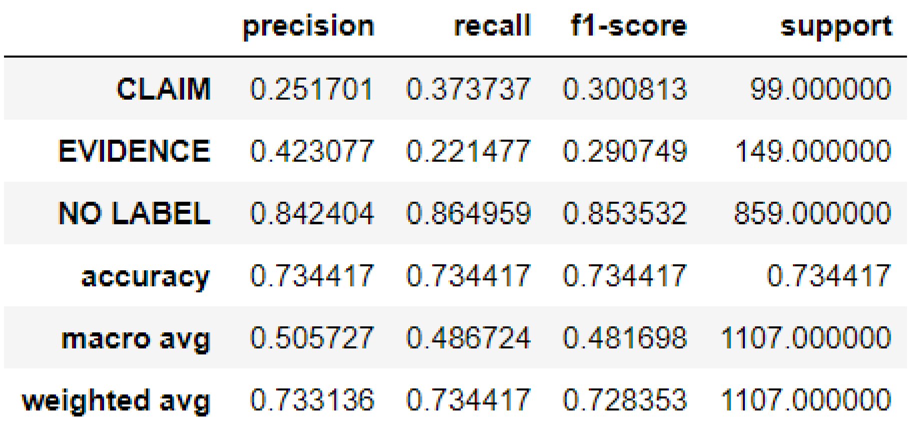
# 4. Results

At this point, all the algorithms have been deployed and the results have been obtained. The results will be presented separately for each model category and in the end of this chapter the best one will be chosen as the team's suggestion and proposal. The order of presentation will be the same as the previous chapter of this report. Thus, the firstly presented model will be the one that was obtained from the Machine learning classifiers.

The criteria upon which we will choose the best model from all the procedures will be the recall of the arguments instead of total accuracy. Recall, also known as sensitivity, is the fraction of the total amount of relevant instances that were retrieved. The reason behind this is that the goal of this project is to predict scientific arguments, thus it seems more valuable to predict those labels instead of predicting the sentences that had not labeled. In other words, the most important aspect of our predictive model will be that it does not lose any arguments. It is one thing to predict some No-labels as claims for instance and it is much worse to predict a claim as a No-label.

## 4.1 Machine Learning

As it has been mentioned earlier the algorithms that produced some kind of worth were the multi LinearSVC and the SGDClassifier. For both classifiers different approaches were explored also as mentioned previously. In addition, grid search was deployed in order to find the best hyper-parameters. Although with the hyper-parameters that were produced the accuracy became higher, the prediction of the claims and evidences dropped significantly, and the model became better only for the prediction of the No-labels. Thus, the next figure shows the classification report produced by the predictions of the best LinearSVC model, according to the selection criteria that were mentioned in previous chapters.



| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **CLAIM** | 0.251701 | 0.373737 | 0.300813 | 99.000000 |
| **EVIDENCE** | 0.423077 | 0.221477 | 0.290749 | 149.000000 |
| **NO LABEL** | 0.842404 | 0.864959 | 0.853532 | 859.000000 |
| **accuracy** | 0.734417 | 0.734417 | 0.734417 | 0.734417 |
| **macro avg** | 0.505727 | 0.486724 | 0.481698 | 1107.000000 |
| **weighted avg** | 0.733136 | 0.734417 | 0.728353 | 1107.000000 |

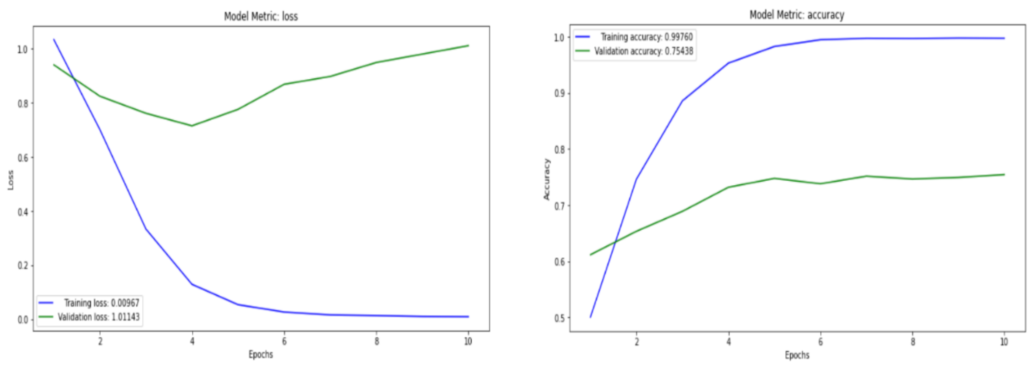
*Table 4.1. Linear SVC model classification report.*

This figure shows four metrics, which are precision, recall, f1-score and support. The precision, also called positive predictive value, is the fraction of relevant instances among the retrieved instances the percentage of the right predictions classified as each label. Furthermore, recall stands for the percentage of the actual labels that are being predicted correctly by the model. The f1-score is the harmonic mean of the previous two metrics and the support represents the number of claims in the test dataset.

As mentioned before the most important metric according to our business goal in order to evaluate the models is the recall. As far as the claims are concerned, the respective score of this metric is equal to approximately 37%. This means that the model could predict more than 1 out of 3 claims. This number was significantly higher than the respective one of the Evidence label, which was about 22%. Now, the 86.4% of the sentences that were no label were correctly predicted. The total accuracy of the model is 73.4%, which is not very satisfying.

## 4.2 CNN

At this point, the results of the CNN model will be presented. First of all, two plots will be demonstrated, so as to present the accuracy and loss in both train and validation datasets. In fact, the history of the model fit is visually presented in these graphs.



*Plot 4.1. Cnn model Accuracy and Loss per epoch.*

From the above plot of loss and accuracy, we can see that the model does not have comparable performance on both train and validation datasets. Since these parallel plots start to depart consistently after a certain epoch, it might be a sign of over fit and to stop training at an earlier epoch. Actually, by exploiting the usage of the model checkpoint as mentioned before, our model that was kept was from the final epoch, although the model did over fit. Since it provided the best results at the validation accuracy it seemed like the correct thing to do despite the clear over fitting.

The next thing that needs to be presented is the confusion matrix. This is the matrix indicating the actual and predicted classified sentences. This is shown below and it should be mentioned that the actual labels are presented in the rows of the matrix, while the predicted in the respective columns.

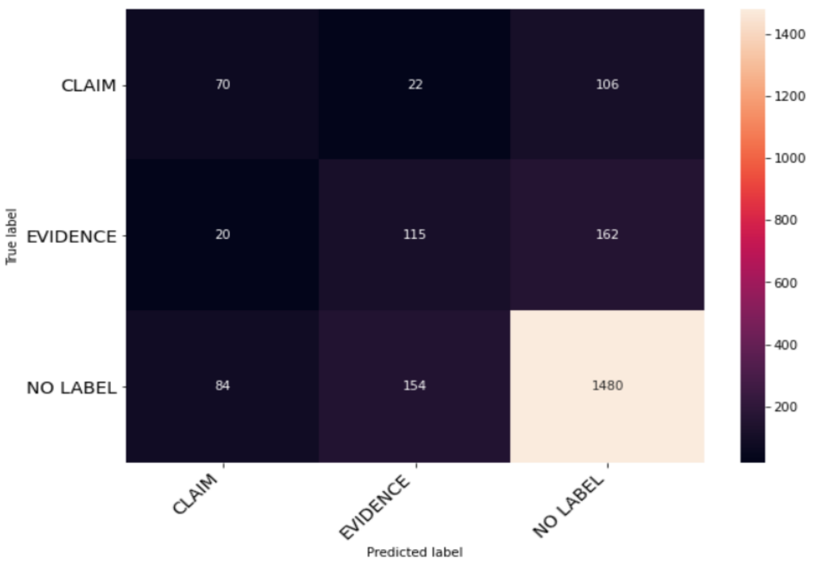
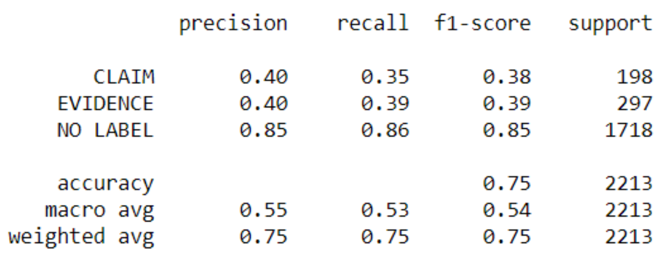


*Figure 4.1. Cnn model Confusion Matrix.*

From the above demonstrated confusion matrix there are two cells that are very important for our business intentions. Those are the ones that present the values of 70 and 115. The number 70 represents the number of actual claims that the model has successfully predicted, while number 115 represents the number of actual evidences that the model has successfully predicted. These numbers are relatively good. In addition, the correctly predicted no-labels are 1480. The only thing that is not preferable but is true is that most of the claims and evidences that are classified wrong, have been classified as no-labels. Thus, the most disturbing values of the above matrix are the 106 and 162 sentences that are either claims or evidences but have been classified as no-labels. Only 42 (22 +20) sentences have been classified in the other component of the argument (evidence-claim).

All these observations are presented in another way in the following figure. Specifically, this figure shows the classification report produced by these predictions, in order to have a better view, as the numbers represented above should be translated into percentages.

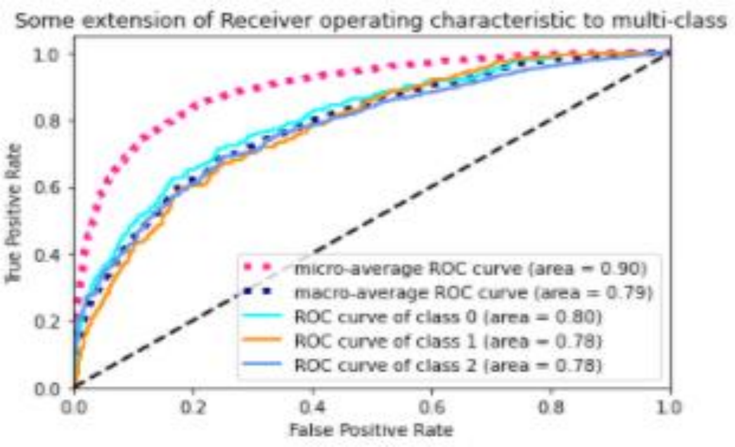


```
              precision    recall  f1-score   support

       CLAIM       0.40      0.35      0.38       198
    EVIDENCE       0.40      0.39      0.39       297
    NO LABEL       0.85      0.86      0.85      1718

    accuracy                           0.75      2213
   macro avg       0.55      0.53      0.54      2213
weighted avg       0.75      0.75      0.75      2213
```

*Table 4.2. Cnn model Classification report.*

As stated before, the metric of main interest is recall and more specifically the recall of the claims and the evidences. As demonstrated from the above plot the recall of the claims is 35% and the recall of the evidences is 39%. In addition, the harmonic mean of precision and recall (f1-score) is 38% and 39 % respectively. Although these values are relatively pleasant, the accuracy of the model is rather poor (75%). On the bright side, there is a balance between precision and recall for all three labels.

Finally, another visual that is presented is the ROC Curve. The ROC curve visualizes the quality of the ranker or probabilistic model on a test set, without committing to a classification threshold. The curves presented are the ones representing the general predictions of the model, as well as the ones representing the predictions of each class. The area under the ROC curve (AUC) is a useful tool for evaluating the quality of class separation for soft classifiers. In the multi-class setting, we can visualize the performance of multi-class models according to their one-vs-all precision-recall curves. The AUC can also be generalized to the multi-class setting



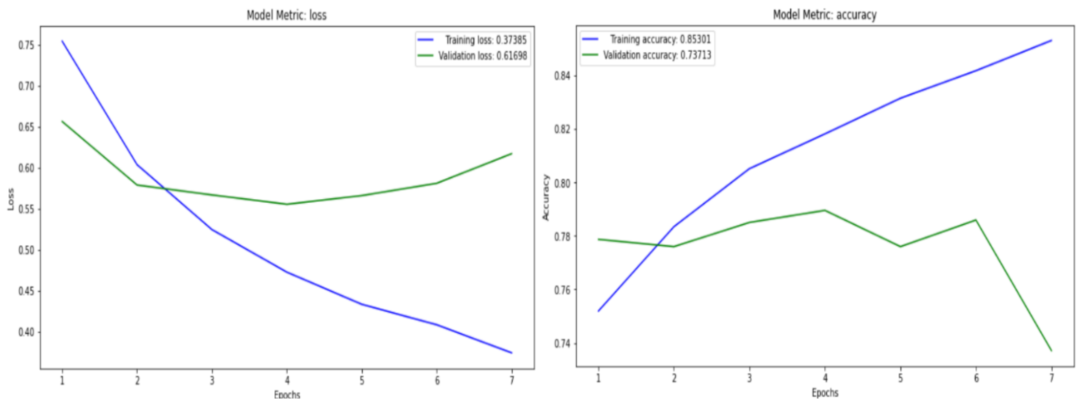
*Plot 4.2. CNN model ROC curve.*

A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally), whereas a micro-average will aggregate the contributions of all classes to compute the average metric. As demonstrated from above the micro average is equal to 90% and the macro average is equal to 79%. In addition, the ROC and AUC values of each class are only slightly different, that gives us a good indication of how good our model is at classifying individual class.

The results of the CNN model have been finally presented as well as their respective visualizations. In conclusion, we are somewhat positive for the evaluation of the model. At this point, we will proceed with the presentation of the results of the MLP model.

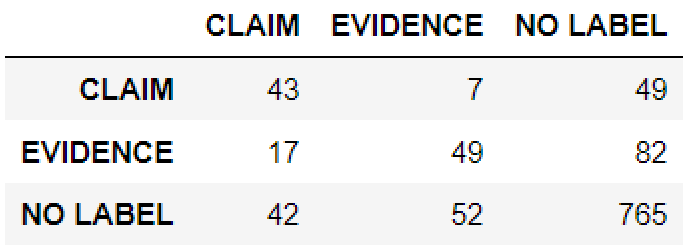
## 4.3 MLP

At this point, the results of the MLP model will be presented. The first two plots represent the accuracy and loss in both train and validation datasets. In other words, the below two figures demonstrate the history of the training of the model.



*Plot 4.3. MLP model Accuracy and Loss per epoch.*

From the above plot of loss and accuracy, we can see that the model does not have comparable performance on both train and validation datasets for once more. Since these parallel plots start to depart consistently after a certain epoch, it might be a sign of over fit. Actually, by exploiting the usage of the early stopping algorithm, the training of the model stops before the over fitting becomes extreme. The next figure that will be presented is the confusion matrix.

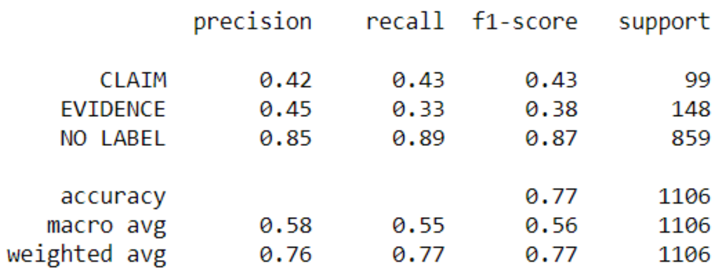


| | CLAIM | EVIDENCE | NO LABEL |
|---|---|---|---|
| CLAIM | 43 | 7 | 49 |
| EVIDENCE | 17 | 49 | 82 |
| NO LABEL | 42 | 52 | 765 |

*Table 4.3.MLP model Confusion Matrix.*

From the above demonstrated confusion matrix the values of the two cells that are very important for our business intentions are 43 (out of 99) and 49 (out of 148). These values represent the number of actual claims and evidences respectively that the model has successfully predicted. These numbers are relatively satisfying. In addition, the correctly predicted no-labels are 765 out of 859. The only thing that is not preferable but is true is that most of the claims and evidences that are classified wrong, have been classified as no-labels as in the CNN model.

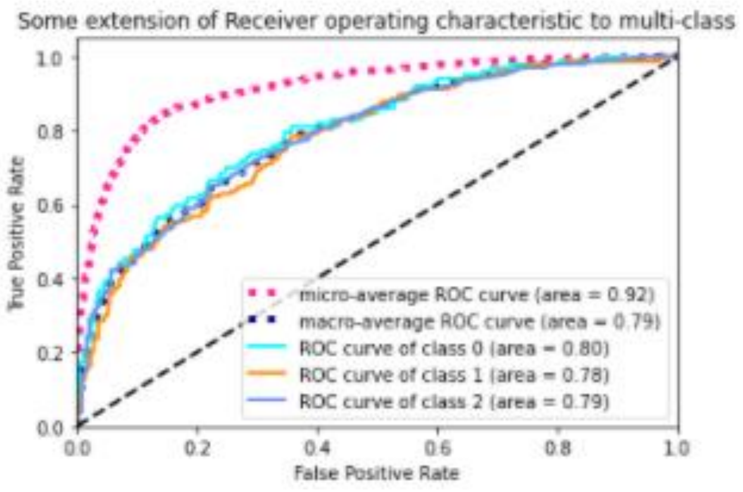
All these observations are presented in a more summary way in the following figure. More specifically, this figure shows the classification report produced by these predictions, in order to have a better view, as the numbers represented above should be translated into percentages.



```
              precision    recall  f1-score   support

       CLAIM       0.42      0.43      0.43        99
    EVIDENCE       0.45      0.33      0.38       148
    NO LABEL       0.85      0.89      0.87       859

    accuracy                           0.77      1106
   macro avg       0.58      0.55      0.56      1106
weighted avg       0.76      0.77      0.77      1106
```

*Table 4.4. MLP model Classification report.*

As stated before, the metric of main interest is recall and more specifically the recall of the claims and the evidences. As demonstrated from the above plot the recall of the claims is 43% and the recall of the evidences is 33%. In addition, the harmonic mean of precision and recall (f1-score) is 43% and 38 % respectively. Although these values are relatively good, the accuracy of the model is mediocre (75%). In comparison with the CNN model, this model seems to be slightly better when it comes to claims and no labels and slightly worse when it comes to evidences. In conclusion, generally it seems to be slightly better than the CNN model since it also presents a higher level of accuracy. Finally, the last visualization that will be presented is the ROC Curve.



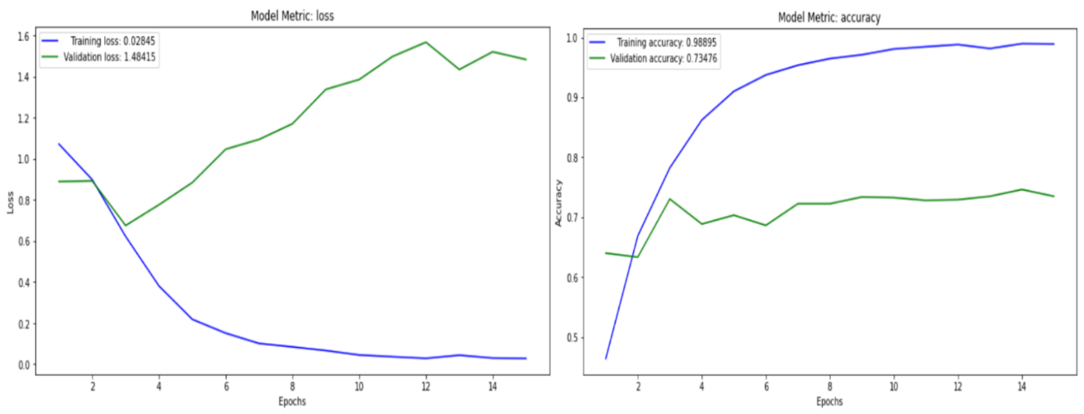
*Plot 4.4.MLP model ROC curve.*

A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally), whereas a micro-average will aggregate the contributions of all classes to compute the average metric. As demonstrated from above the micro average is equal to 92% and the macro average is equal to 79%. In comparison with the CCN model these values are equal or slightly hier. In addition, the ROC and AUC values of each class are exactly the same with the CNN model and only slightly different between them.

The results of the MLP model have been finally presented as well as their respective visualizations. In conclusion, we are somewhat positive for the evaluation of the model and it also seems to be performing and fitting our data slightly better than the CNN model. At this point, we will proceed with the presentation of the results of the LSTM model.

## 4.4 LSTM

The last neural network, whose results should be presented is the Bidirectional LSTM. Again the first view of the results will be given by the accuracy and loss through each training epoch plots.



*Plot 4.5. LSTM  model Accuracy and Loss per epoch.*

As shown in these graphs, the results of this model seem to be similar to the ones obtained by the CNN model. The accuracy of the validation data seems to be almost constant after a certain epoch, while the one of the training data is increasing each time. As a model checkpoint according to the highest validation accuracy was used, the model kept was produced by the 14th epoch. These plots indicate overfitting, but this epoch's model produced the best results for this neural network.

Next, a better view about the predictions will be given by the confusion matrix, which is demonstrated below:

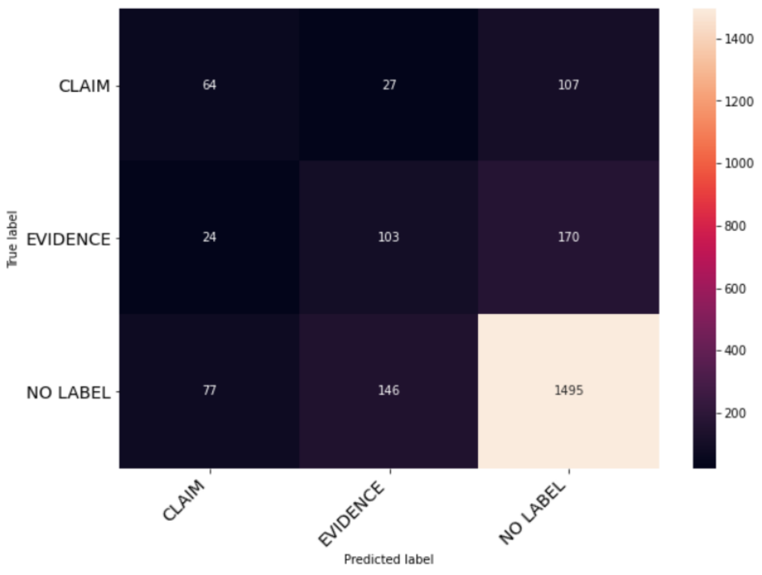


*Figure 4.2.LSTM model Confusion Matrix.*

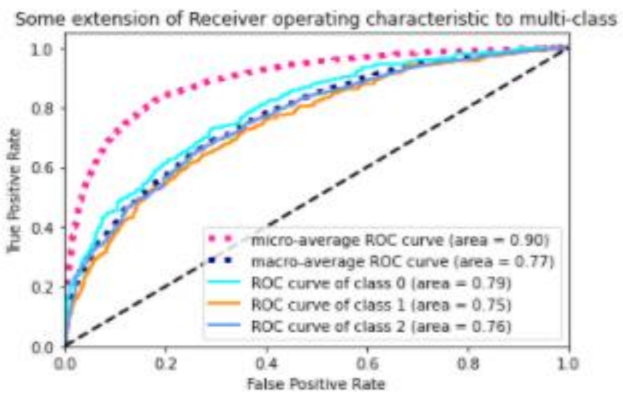
As expected from earlier observations about the data and the models' predictions, most of the successful classifications come from the sentences that are not labeled.In fact, 1495 sentences were right predicted as No Label. However, attention will be given once again to the predictions of claims and evidences. The number of successful claims classification was 64 and 103 evidences were successfully predicted by the model. Another 27 claim sentences were classified as evidences, while 24 of the last category were predicted as claims.

At this time, the classification report is presented.

```
              precision    recall  f1-score   support

       CLAIM       0.39      0.32      0.35       198
    EVIDENCE       0.37      0.35      0.36       297
    NO LABEL       0.84      0.87      0.86      1718

    accuracy                           0.75      2213
   macro avg       0.53      0.51      0.52      2213
weighted avg       0.74      0.75      0.74      2213
```

*Table 4.5. LSTM model Classification report.*

The numbers demonstrated before are translated to percentages and the results seem to be slightly worst compared to the ones of the CNN model. To be more specific, the model successfully predicted 32% of the claims ,35% of the evidences and 87% of the non labelled sentences. The accuracy is equal to 75%, which is not satisfying, but the aforementioned results do not seem so bad. Finally, the visual of the ROC Curve will be presented to evaluate this model.



*Plot 4.6.LSTM  model ROC curve.*

It is indicated that the micro average is equal to 90% and the macro average is equal to 77%. Also, the ROC and AUC values of each class do not seem to different significantly, which is a sign that the model classifies individual classes.

Thus, the presentation of the LSTM's results and visuals have been presented. To conclude, this neural network achieved similar results with CNN. At this time, the best model will be selected, and the reasons will be presented in the next chapter.

## 4.5 Model Selection

In this Chapter, all models will be reviewed in order to select the best one. Firstly, it should be mentioned that the results of a model creation contain a bit of randomness, which comes from the data split, the node dropout as well as other components that contain random features. This means we cannot be sure for the model selection, especially when the differences between the results are not so significant.

The previous Chapter contained each model's results in detail and it was observed that most of them did not differ so much. The machine learning models resulted in satisfying classifications, especially in the sentences labeled as claims. However, the recall metric of the evidences was not as high as expected, so we proceeded to the construction of some deep learning models.

As far as the deep learning models are concerned, the two of them produced almost identical results. Those models were CNN and the Bidirectional LSTM. All metrics of those models were approximately the same and were pleasant, but the ones obtained from the MLP were slightly better. In fact, all metrics and evaluation figures produced by its predictions were higher compared to the ones obtained from the previous two. Thus, the model that our team selected as the best was the MLP.

## 5. Members/Roles

The team contributing to this project was consisted of three members, namely Konstantinos Panagiotatos, Manos Mikes, Filippos Nikolopoulos. All of these members are students of the M.Sc in Business Analytics of Athens University of Economics and Business. The first degree varied among the members, as Konstantinos Panagiotatos is a Mechanical Engineer, Manos Mikes a Mathematician and Fillipos Nikolopoulos has studied Economics.

As far as the roles of each member are concerned, first all three of the members contributed to the collection of the scientific papers and their annotation. As also mentioned in a previous chapter, the dataset was not created by the collected papers of this team, but it was created combining them with the respective ones collected by the rest of the teams.

Then, the data preparation, so as to convert them to valid input for a model was again made by all three members. The creation of the three models was achieved by splitting each one of them to a single member. After that, the results of these models were discussed in some meetings and all changes in the models were made by the entire team. The same case stands for this report that was delivered together with the code.

## 6. Bibliography & Acknowledgements

This project was much more than a simple cooperation of the three members that consisted the team. As mentioned previously multiple other teams assisted us in the annotation phase and in addition a lot of help and guidance was given from Mr. Perrakis and Mr. Fergiadis, the two assistant professors of Mr. Chatzigeorgiou. Furthermore Mr. Perrakis provided us with python notebooks that included code that were heavily used in the project. Except from the aforementioned assistance our team explored the power of internet by absorbing a lot of theory wise and code wise help. Some worthy noticing sites were the following:

https://medium.com/@xzz201920/multi-layer-perceptron-mlp-4e5c020fd28a https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

http://deeplearning.net/tutorial/mlp.html

https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

https://medium.com/@jorgesleonel/hyperparameters-in-machine-deep-learning-ca69ad10b981

https://towardsdatascience.com/guide-to-choosing-hyperparameters-for-your-neural-networks-38244e87dafe

http://neuralnetworksanddeeplearning.com/

https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f

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

https://medium.com/@djajafer/multi-class-text-classification-with-keras-and-lstm-4c5525bef592

https://medium.com/swlh/step-by-step-building-a-multi-class-text-classification-model-with-keras-f78a0209a61a

https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590bee1bd17

https://stackabuse.com/python-for-nlp-multi-label-text-classification-with-keras/

https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/

https://lionbridge.ai/articles/using-deep-learning-for-end-to-end-multiclass-text-classification/

https://realpython.com/python-keras-text-classification/

https://www.kaggle.com/kadhambari/multi-class-text-classification

## 7. Time Plan

As mentioned in previous Chapters, the steps followed for the final results are:

1) Paper Extraction (Data Collection)
2) Paper Annotation
3) Model Implementation
4) Discussion and assembly of results

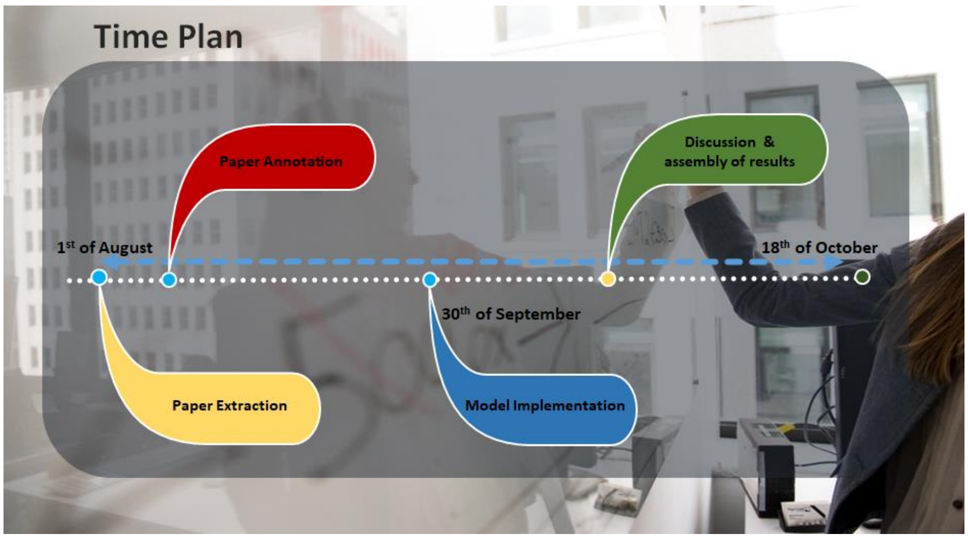


*Figure 3: Time Plan of the project*

First of all, the data collection was split in three phases. In each phase 50 scientific papers were collected, creating in this way a batch. Then, each document of the batch was annotated by members of all teams working on each SDG. Each batch needed about one week to be collected and annotated by all members. So, the duration of the first two steps was approximately three weeks. After all papers were annotated, some members of the teams (adjudicators) checked every paper to check for possible errors in the annotation phase. This was done because, as mentioned before, a paper was considered as valid only if 2 out of 3 team annotators' labels were matching.

The next step began in the middle of September and it was the programming part of the project. The data were pre-processed, and a few models were trained with some of them, in order to achieve the project's goal. The duration of this step was about one and a half week.

After the models were created, a discussion about the results was necessary. Thus, the team reviewed the outcomes of the previous steps and made any changes that were required. The time needed for the completion of this step was approximately half a week. Finally, one more week was needed, in order to prepare the presentation.

## 8. Conclusions/comments

In conclusion the entire implementation of this project was quite demanding and at the same time fascinating. Through, its process the members of our team learned a lot of important aspects of the Deep Learning field and its challenges. It was an end to end project which made us realize the difficulties and the aspects of an entire Machine/Deep Learning project, which is a very educating experience.

The difficulties that we faced where not so many but were crucial. First of all, the time that was given for this project was very short and did not respond to the difficulty and complexity level of the subject. Another aspect of challenges that we faced and sometimes managed, and some others did not were the inconsistencies in the TensorFlow library. Although, it is understandable that TensorFlow is a continuously developing library with many progressive and new aspects of Data Science, there were many problems as far as its functions are concerned. Some of them could only be executed in a particular version, where some others could not.

On the other hand, there were many positive assisting aspects to the project. First, the literature and the help that can be found for the subject in the web is rather impressive. In addition, the guidance that was provided to our team was very helpful and crucial. Furthermore, the cooperation between the members of our team was very efficient and harmonic.

As far as the results of the models that were presented in this report and are our team's suggestions are concerned, there seems to be a certain worth. All models have a somewhat good predictive ability. The one that was chosen for us and is suggested is the MLP model, since it performed slightly better than the rest. At this point, we would like to mention that creating a model that performs extremely well in real case text classification problems is not an easy task. In order to prove this point, the example of Greek banks seems very relative. For a lot of years they used an algorithm in order to determine if someone should be able to take a loan or not, that was only 2% better than complete randomness.

On the other hand, it should be mentioned that there is room for improvement. It is our team's belief that the annotation phase of the project could be done better. In addition, provided more time more algorithms could be explored and even an assembly of these algorithms that could work with the majority rule. Unfortunately, all these are left for future work.