LNR Project
Data Science, course 2023-2024
Written by **Juan Tomás Moreno**

Oppositional thinking analysis: Conspiracy theories versus critical thinking narratives

In this report, we aim to solve the PAN shared task presented to us: *"Oppositional thinking analysis: Conspiracy theories versus critical thinking narratives"*. Our goal is to give a clear solution to one of the main problems affecting democracies in the West: **misinformation**. How can we know what is right and what not, if we cannot even spot the difference between a critic and a misleading conspiracy?

To approach this matter, we first tried to find out the best Machine-Learning-based classification algorithm for our dataset, using our knowledge in the field of Natural Language processing. This is why we tried to classify our English and Spanish corpora using different text representation techniques, and evaluating them with several classifying methods, which were chosen due to their extended usage in Data Science studies. Firstly, we will detail our four classifiers:

- Support Vector Machines (**SVM**): Classifier that finds the optimal hyperplane to separate classes in a given feature space.
- Multi-layer Perceptron (**MLP**): Type of artificial neural network consisting of multiple layers of nodes (so-called neurons), used for classification and regression.
- **Logistic Regression**: Statistical model that predicts the probability of a class using a logistic function applied to a given linear combination of features.
- **Hard Voting**: Ensemble method where each classifier casts a vote and the final class is determined by the majority of votes. In our case, the classifiers participating in the ensemble will be all the three aforementioned methods.

On the other hand, text representations are used to convert words and documents into numerical formats that computers can process, enabling the application of machine learning and natural language processing algorithms for tasks such as classification, translation, and text generation. For the present task, we used **Bag-of-Words** (representing text by the frequency of each word in a document, ignoring order) and **Word2Vec** (transforming words into numeric vectors in a vector space, capturing semantic relationships).

For each of the sixteen combinations deployed, always considering the standard hyperparameters for each model, we compared the results to the respective ground truth established from a 80-20 train-test partition built by calling a Python function [`clasif`], using the **Matthew Correlation Coefficient** (or MCC). In the attached table below, we can see the obtained coefficients, for both **Spanish** and **English** corpora:

	Bag-of-Words	*Word2Vec*
SVM	0.515 — 0.616	0.221 — 0.249
MLP	0.621 — 0.64	0.29 — 0.455
Logistic Regression	**0.654 — 0.69**	0.179 — 0.287
Hard voting	0.072 — 0.689	0.073 — 0.352

Highlighted in bold, we see the highest MCC result for each language analysed: it is the **Logistic Regression** model, using a **Bag-of-Words** text representation. This was the top-performing approach for both corpora.

To ensure we got relatable results, we want to validate our results by performing a **10-fold cross-validation**, with and without stratification. Although, we must take into account two changes we introduced:

- Applying **one-hot encoding**: Reformatting the “CRITICAL” and “CONSPIRACY” labels to binary format, treating each category independently without assuming ordinal relationships.
- Adjusting **hyperparameters**: Optimizing our model performance, by establishing the `max_iter` parameter on 1000 (instead of the default value, 100). This was adjusted in order to avoid warning triggering and to ensure a complete and thorough training of our model.

We built a Python function [`re_eval`] to obtain the respective **MCC and F1-score** for our models, after applying the aforementioned cross-validation. When comparing how the models performed under K-Fold and Stratified K-Fold validation techniques, we observed remarkable consistency in the evaluation metrics, both in terms of MCC and F1-score, for both corpora. This suggests that stratification had minimal impact on the models' performance in our already fairly balanced datasets. Although, we must take into account that stratification is a recommended practice for data with class imbalance, so this might be useful for future projects. All in all, this comparison highlights the robustness of the models and supports our choice. We can see that both F1-scores are near **0.9**, meaning that our model is, in fact, capable of correctly classifying most positive samples while maintaining a low number of false positives and false negatives.

Finally, after checking the value of our approach, we had to face the challenge itself. To prepare the results, firstly we had to modify the `clasif` function in order to, instead of doing a train-test split, just train the model with all the training data provided. Afterwards, we predicted the labels from the test dataset using the newly-trained model, and saved them in a JSON file as specified.

Technologies used:

- Python
- Jupyter
- Google Colab
- Pandas
- Numpy
- Scikit Learn
- Gensim
- NLTK
- JSON