Salva Veritate Manual

# Introduction

In the modern day, information travels at extremely high speeds, and in very large quantities as well. Either passively or actively, we are constantly being overwhelmed with new information at every moment. In particular, the news is a prolific distributor of information, from far and wide and of every variety. However, with political preferences, biases, and lies in the background, as well as the various other contextual variables of every news source, it can become nearly impossible to distinguish the truth from a sea of falsities. That is why Salva Veritate was created: to help mitigate the spread of potentially dangerous misinformation across the world, and to help you, the people, distinguish what is correct and what is not.

# Operation

# Salva Veritate’s power primarily comes from the use of a few very popular Python libraries, namely NumPy, Pandas, and specific parts of SciKitLearn, including train\_test\_split, TfidfVectorizer, and PassiveAggressiveClassifier. The PassiveAggressiveClassifier is the backbone of how Salva Veritate determines the validity of the news texts it analyzes, by *classifying* them into distinct categories.The News Dataset.csv included in the base kit was used to alter various parts of the program, including the parameters used for modelling, and to ensure that everything was functional. NumPy and Pandas were used to store and manipulate the data included in the .csv file. Train\_test\_split, as the name implies, splits up various pieces of data in matrices created by Pandas into training sets and testing sets, or training sets and validation sets, while TfidfVectorizer transforms the split data into a form that is usable by PassiveAggressiveClassifier in order to both train it and use it to classify other items (primarily those in the test set designate by train\_test\_split). Salva Veritate allows the user to specify the many parameters involved in PassiveAggressiveClassifier and customize the model to their preference, such as specifying how much of the training data is to be used as a validation set on each training iteration. The two primary functions of Salva Veritate are creating models and using them to classify new news articles. Models are created based on the inputted values of the user, and trained based on data provided in csv files, which are also specified by the user. In particular, the paths of the directories and names of the files must be provided with perfect accuracy, or else the program will not recognize that the specified directory or file exists. Afterward, there is also the option to save a created model to a designated directory, which contains the values for every variable that the user specified the value for. In terms of using a model, paths are inputted to find the model and data used to train it, and the path of a given news file is used to find the news file itself, which is selected to be classified. The user can then select the attribute of the news item to be used as a predictor, and following an analysis of the provided information, the model determines whether or not the news item is “real” or “false”.

# UI Navigation

Currently, the UI implementation uses the console in Python 3.7. The steps involved in the two major functions of Salva Veritate are similar to what is specified above in the “Operation” section. For model creation, a prompt is given, asking the user to designate each variable with their preferred value. If the response is invalid based on the current parameter, then a default value will be applied instead. As well, a few prompts involving predictor and response specification (x and y values, in essence) exist, and these values must match the column names/headers in the specified .csv file perfectly, or else the program will be forced to return to the main menu. For prompts involving paths, these can be copy-pasted directly from Windows Explorer while navigating inside of the desired directory. Name prompts are case sensitive and will need to match the file names exactly. If either the directory path or name is incorrectly inputted, the program will automatically exit to the main menu to prevent an error from occurring. The prompts for the menus when using a model to classify a news item follow the same rules as those in model creation.

# Error Prompts

For the most part, error prompts have been replaced with print statements that explain what issue occurred instead of causing the program to crash. However, Salva Veritate, in general, assumes that the user is not entering a null/empty value when prompted to provide one. On certain prompts, if an empty value is entered, the program will actually crash and stop running entirely, ending in an error message. For this reason, it would be optimal if users avoided attempting to enter null values at any of the prompts in the program.

# Limitations

As of the current version of Salva Veritate, only .csv files are supported, for both input and output. There will be more file formats, such as .xlsx (Excel) files, accommodated for in future versions.

While classification models are very useful for news validation, so too are regression models, which return numerical, quantitative values instead of classes that are instead qualitative values. However, there is no current implementation of regression models in Salva Veritate, and this may or may not be added in future versions, depending on if only using classification models such as PassiveAggressiveClassifier is sufficient to validate if news texts are true or not.

Finally, the news validation function currently only supports news texts in the form of .txt files, with a very specific format as well, containing all of its attributes in a certain order so that the according news item object can be created without issue. The information is then transferred easily from the created object into a format usable by the PassiveAggressiveClassifier, which returns its verdict of “REAL” or “FAKE” for the provided news text.

# Closing Notes

Salva Veritate can never always be correct in its predictions, no matter how accurate the models it creates may be. It is ultimately up to the users, and the people, to be aware of red flags and certain features of false news texts, as well as other forms of misinformation. Therefore, it is paramount that all users, and hopefully all people, do their best to stay as informed as possible about the truth.