Detect False News

# Summary

# For the implementation of the unreliable news article detection model, two specific classification algorithms were used. Specifically, we are performing experimental analysis between two models, Logistic Regression and Support Vector Machines. Preprocessing of the data contained in the training and test datasets was necessary. Steps were taken to avoid overfitting, and evaluation of the results was performed.

# Introduction

# The categorization was done with the aim of finding the accuracy with which we can predict the category to which each new piece of news that is entered into the system will belong.

# Data Set Used

# The dataset refers to a collection of articles that can be found on www.kaggle.com. It includes articles from real news, and each article corresponds to an author, a unique identifier (id), the title of the article, and the content (text) of that article. Those articles that are characterized as "true" have the value in the label column, while those characterized as "false" have the value 0 in the label column.

# Preprocessing

In the preprocessing of our data using Python libraries, we initially merged the columns "author" and "title". In the next stage, the "label" column was assigned to a separate variable from the other columns. Furthermore, each text underwent multiple text conversion processes, such as removing punctuation marks, converting uppercase letters to lowercase, filling in the gaps with an empty string, and finally removing any spaces between words.

# Classifiers

**Logistic Regression:**

Since the categorization we will perform is binary (true or false news), the specific method of categorization is perfectly suited to be used. This is because logistic regression produces results on a scale from zero to one. Close to zero is considered "false," while close to one is considered “true”.

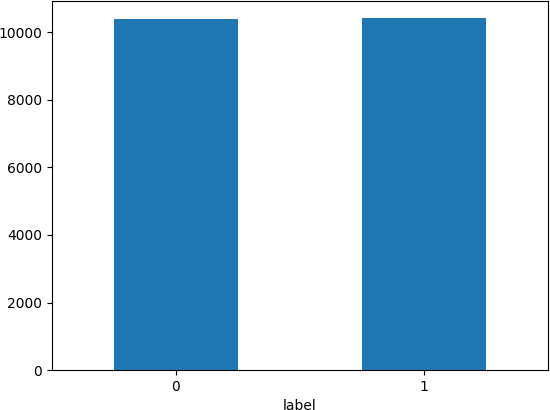
**Support Vector Machines:**

Like Logistic Regression, this classifier defines the result as "true" if it is close to 1 and "false" if it is close to -1. Additionally, the dataset is small, so there won't be a problem with scaling the model.

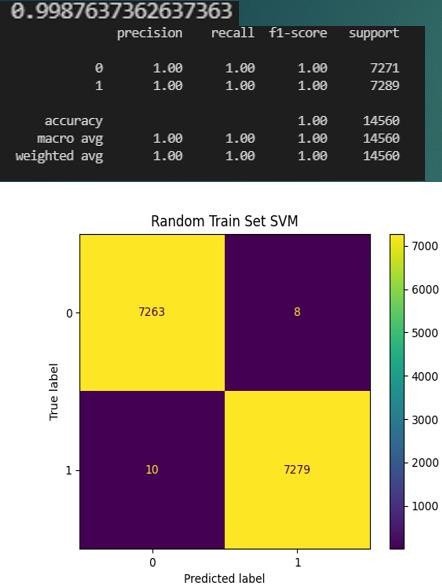
# Methodology

The Scikit-Learn library was used, specifically the train\_test\_split() method, to create random training and testing datasets. This means that two sets of data were created. Then, the data in these sets are transformed into vector form using the TdifVectorizer model provided by the library. Furthermore, two main methods were created for logistic regression and the SVM algorithm, within which the declaration of the respective model to be implemented takes place. Additionally, tests are conducted within the functions on both the testing and training sets, aiming to identify potential overfitting issues. In other words, it was finally assessed whether there is an overfitting problem in both algorithms. The same exact approach was implemented regarding the random selection of a set (e.g., 1000) of samples. As for the timing of model training and the timing of estimating a new text, they were calculated using the time library. Finally, in terms of evaluating user-inputted text, the flask package/library was used to create a relatively simple application at the level of a webpage, which accepts a text input from the keyboard. Then, this text undergoes data preprocessing as mentioned previously, and after being inputted into the model, its truthfulness is evaluated.

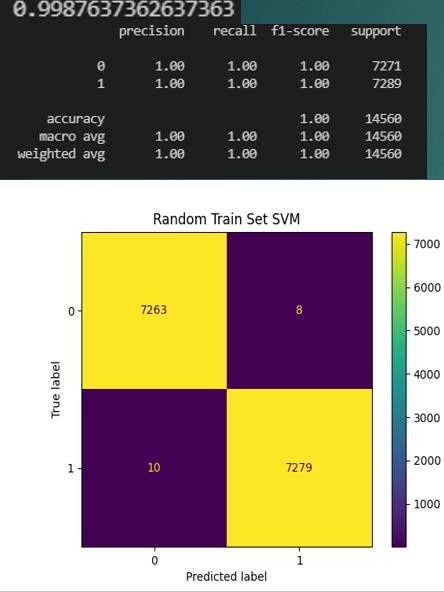
# Chart, treemap chart Description automatically generatedExperimental Evaluation



**Logistic Regression (Training)Logistic Regression (Testing)**

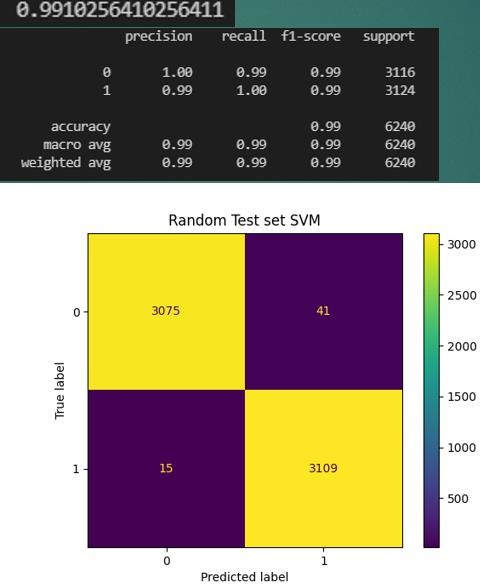


**Support Vector Machine (Training)**

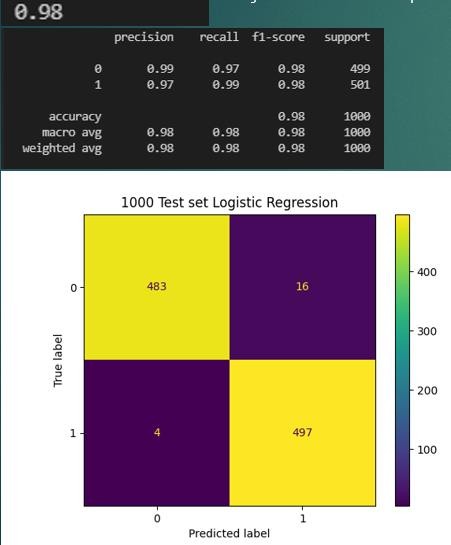
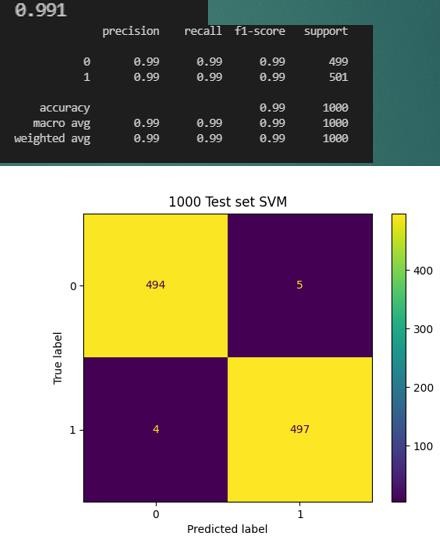


**Logistic Regression (1000)**

**Support Vector Machines (Training)**



**Support Vector Machines (1000)**

**REFERENCES**

* https://flask.palletsprojects.com/en/2.0.x/api/
* <https://scikit-learn.org/stable/>