**GROUP ID: 6**

**TOPIC: FAKE NEWS DETECTION USING MACHINE LEARNING**

Members:

1814023 - Purvi Harniya

1814024 - Neelay Jagani

1814025 - Esha Gupta

GitHub Link: <https://github.com/Purviharniya/Fake-news-detection>

**Introduction**

Data has been increasing at an unprecedented range in an exponential manner and is producing 2.7 quintillion bytes of data everyday.

The definition of fake news is information that pushes people down the wrong road. Fake news is spreading like wildfire these days, and people are sharing it without confirming it. This is frequently done to promote or impose specific views, and it is frequently accomplished through political agendas.

As a result, it is vital to recognise fake news.

**Problem Definition:**

Fake News have become more prevalent in recent years and with great amount of dynamism in internet and social media, differentiating between facts and opinions, relating to commercial or political upheavals has become more difficult than ever.

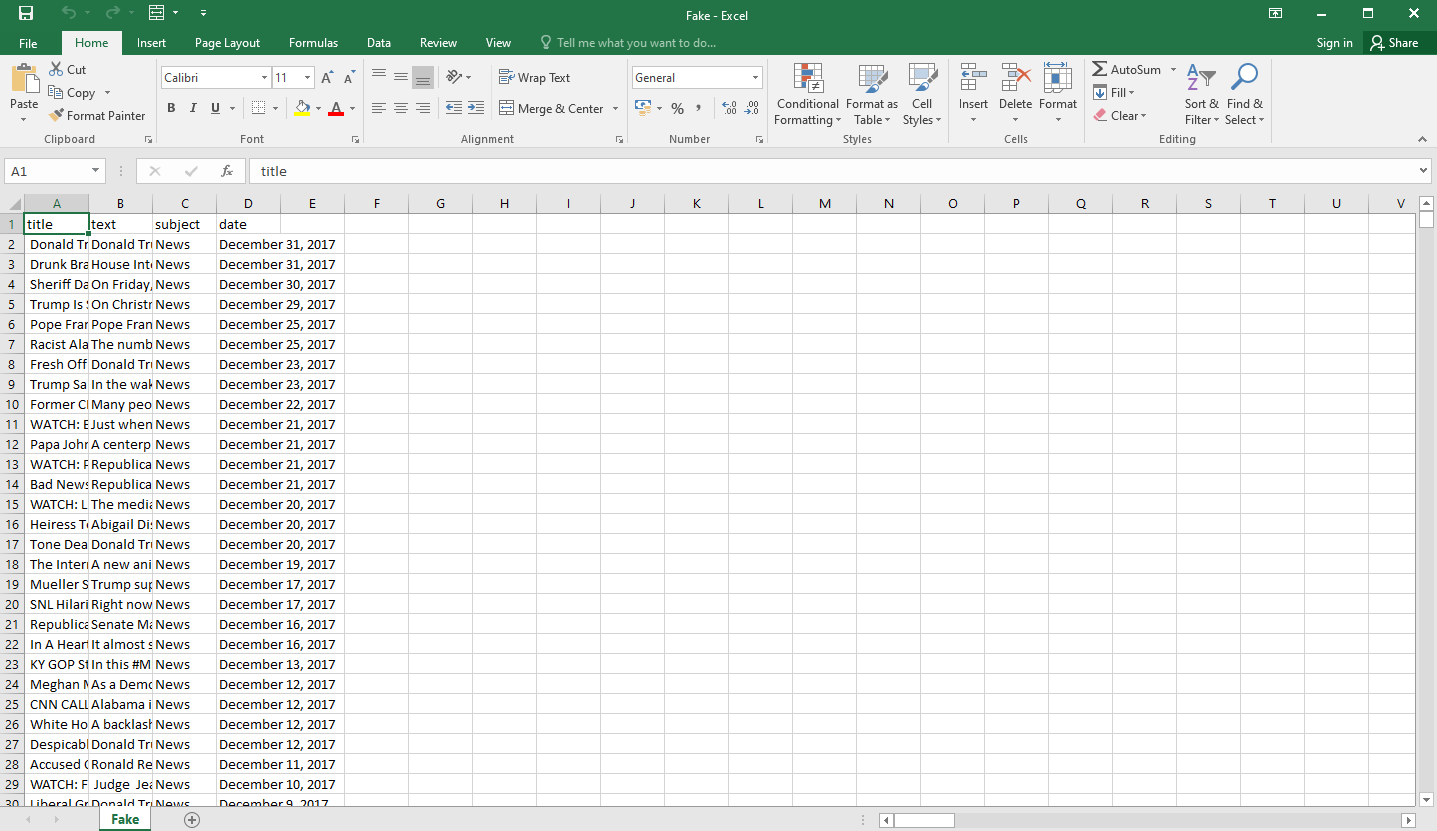
Fake information is purposely or unintentionally spread throughout the internet. The massive dissemination of fake news has left an indelible mark on people and culture.

We use various NLP and preprocessing methodologies like tokenization, stop words removal, lemmatization, stemming and machine learning classification algorithms - logistic regression, pac, ada, naive bayes, svm, random forest, xgboost, decision trees and rnn, to build a model that differentiates between fake news and real news and also analyze the performance of these various classification methodologies to choose the best classifier on out dataset.

**Dataset:**

We have used the ISOT dataset which can be downloaded from - <https://www.uvic.ca/ecs/ece/isot/datasets/index.php>

1. Fake news DataSet

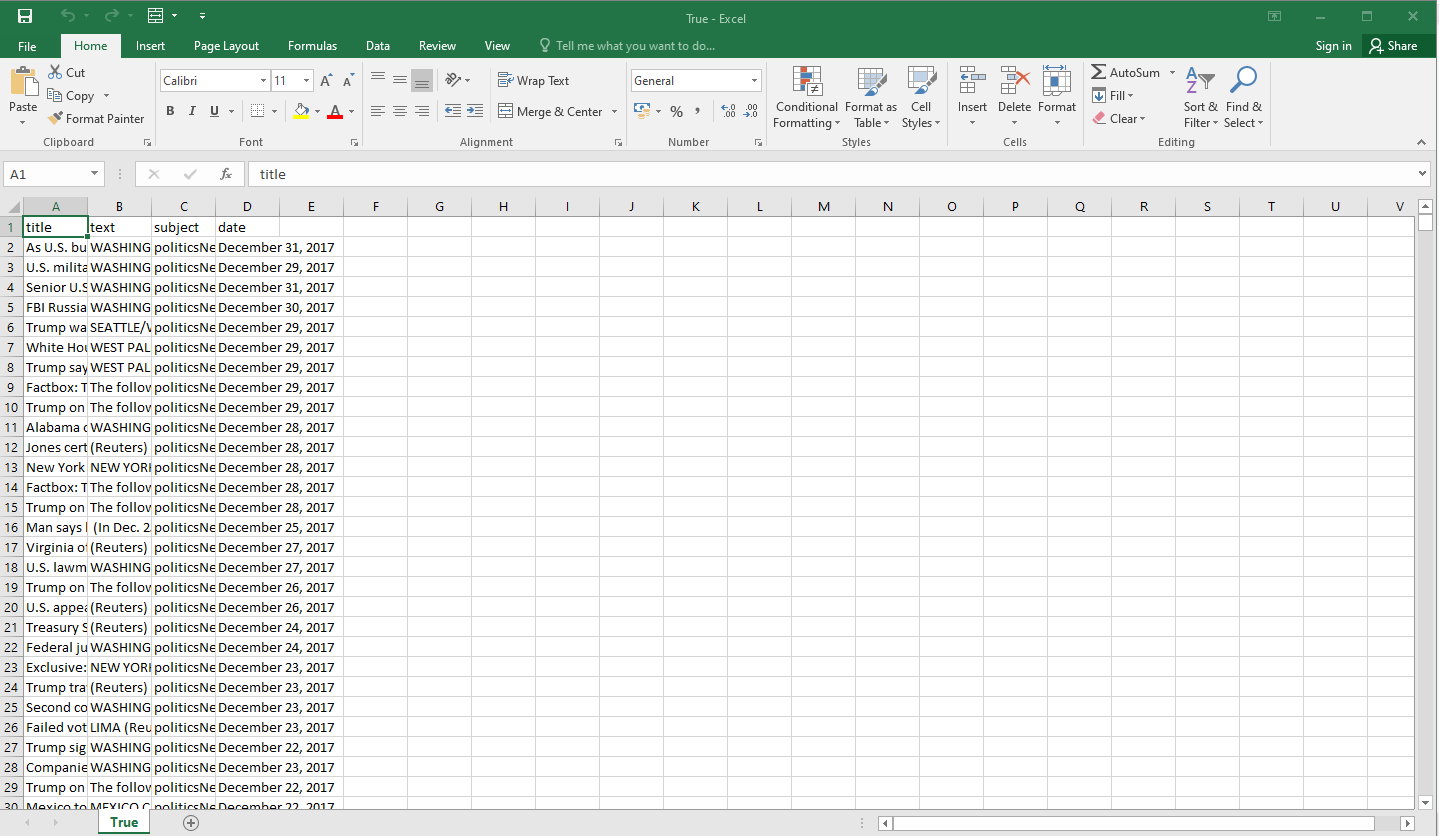
****

Rows: 23481

Columns: 4

Column Description: The four columns include the title of the news, the text in the news, the subject of the news and the date of the news.

1. True news DataSet

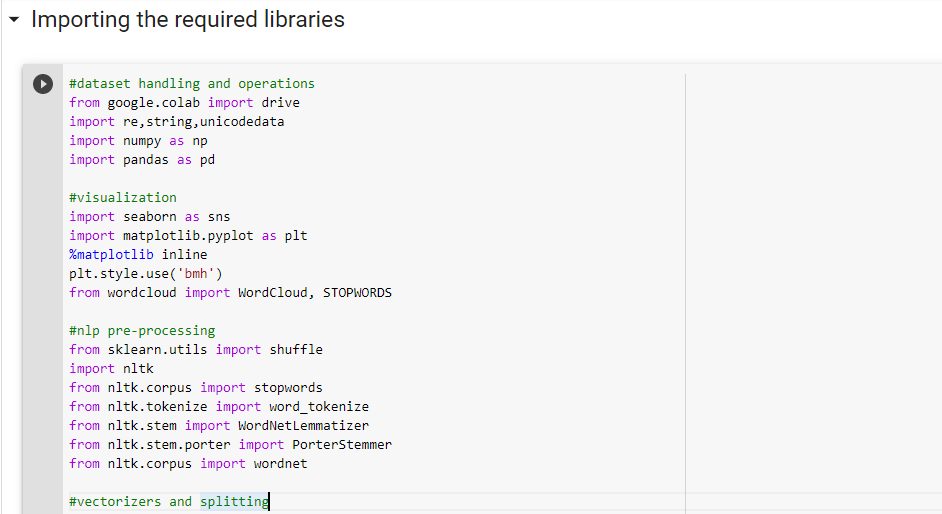


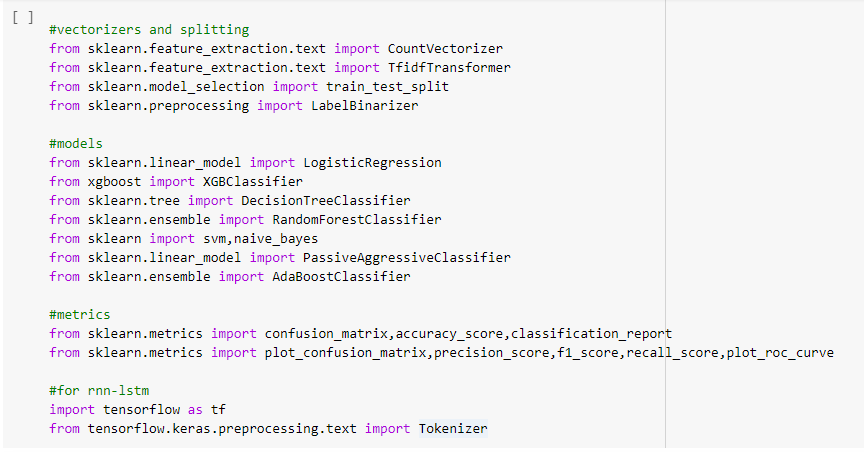
Rows: 21417

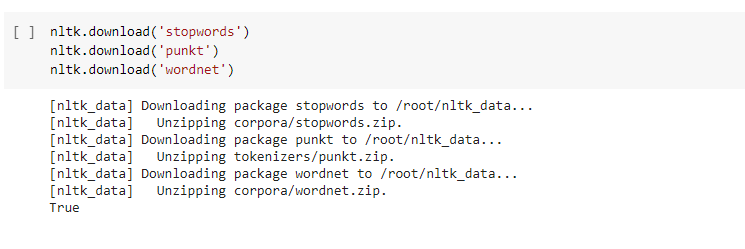
Columns: 4

Column Description: The four columns include the title of the news, the text in the news, the subject of the news and the date of the news

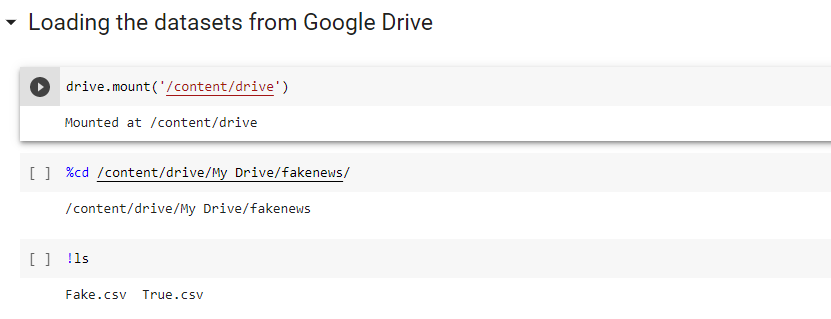
**Implementation:**

1. Importing relevant libraries



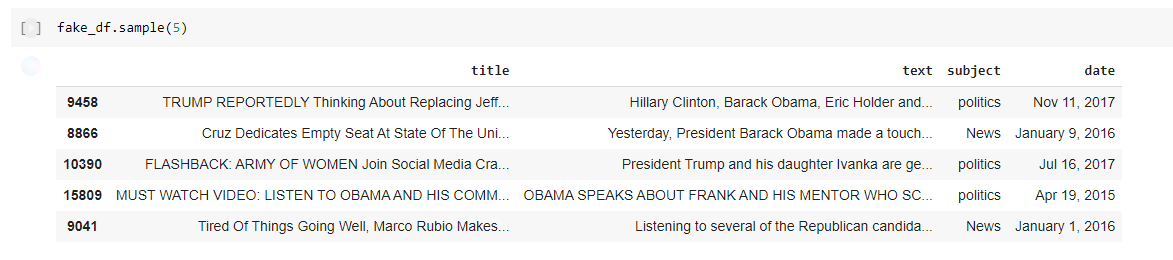


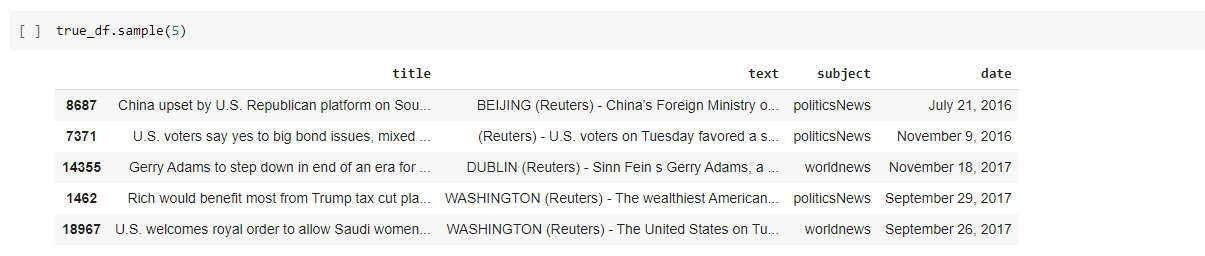
1. Our dataset is uploaded on Google Drive. To read the dataset, we mount our drive by authorizing the google account and then moving to the folder where our dataset is present using ‘%cd’. We check for the folder content using ‘!ls’



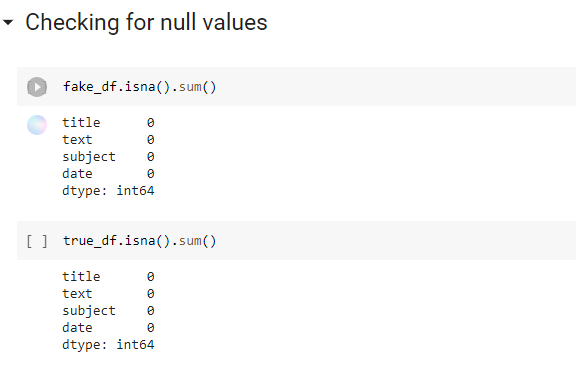
1. Now that our drive has been mounted, we read the dataset. We have two different datasets for Fake and True news. The fake news database consists of 23481 items and the true news dataset contains 21417 items. We display a few records using the sample() function.



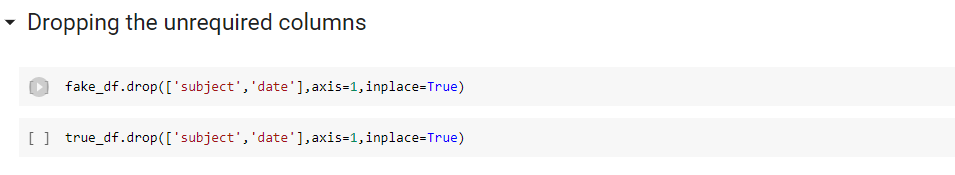




1. We check for null values in both datasets.

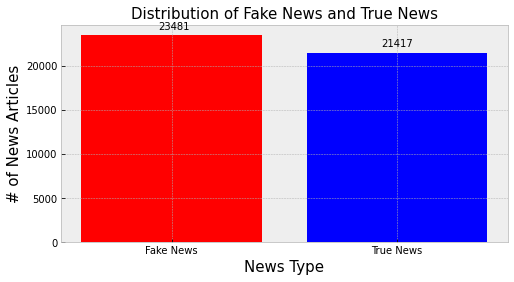


1. We drop the columns that aren’t relevant for fake news detection. These columns have no effect on determining whether the news is fake or true.

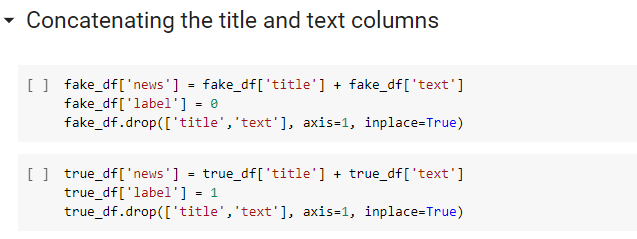


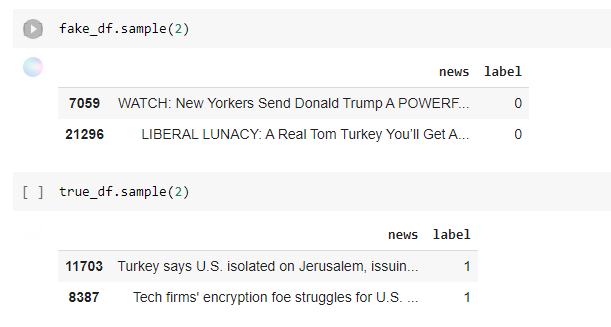
1. Then we check the distribution of fake news and true news by plotting a bar plot. We see that the bar for fake news is higher since it has more records than true news, but the distribution is perfect to train our model as both the datasets have records in the range of 21-23k.





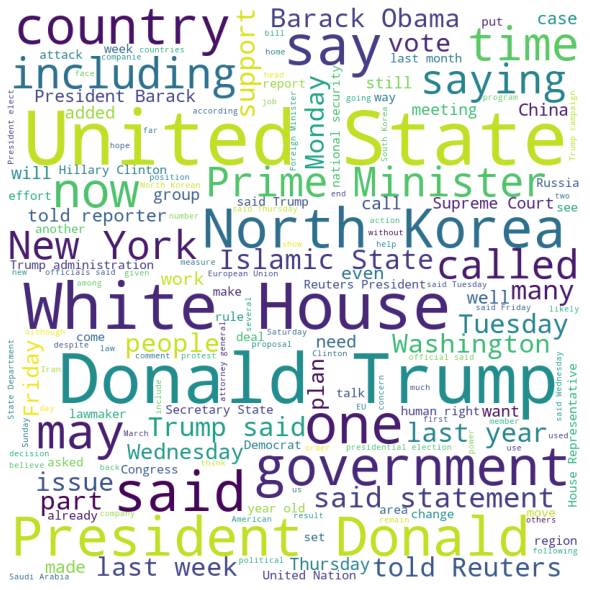
1. To simplify the data, we concatenate the title and text columns into news column in both the datasets and drop the former two. We create a new column ‘label’ which has values of 0-fake and 1-true.



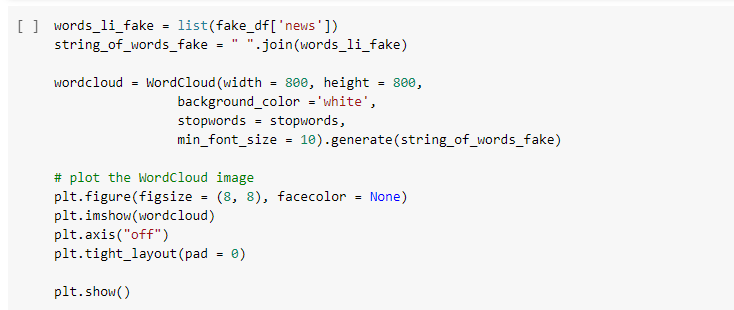


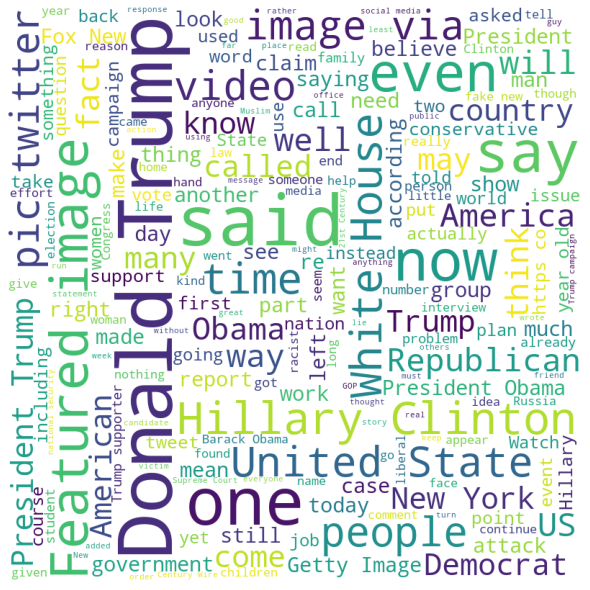
1. We visualize the news present in the True news dataset by creating a Wordcloud to highlight the popular words and phrases based on frequency and relevance.





1. Similarly, to visualize the text or news present in fake news dataset, we create another word cloud.



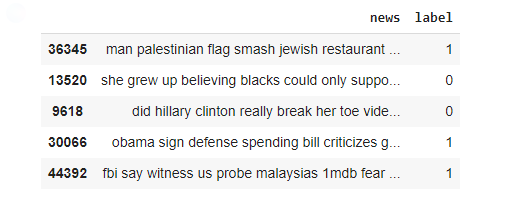


1. In order to build, train and test our models, we need to concatenate the datasets into a single dataset. We concatenate the datasets in the new dataset.

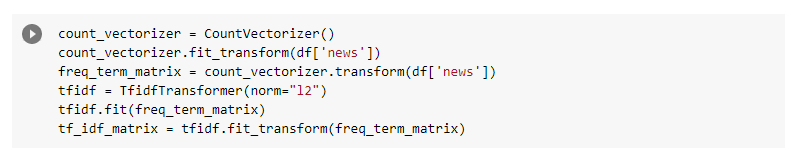


1. In order to preprocess the news text, we use NLP, where we first tokenize the text and then remove stop words, after which we lemmatize the text.

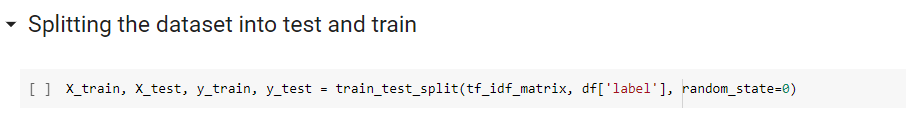




After that we use count vectorizer to transform our data from a collection of text documents to a matrix of token counts. We then use tf-idf vectorizer to convert the collection of raw documents to a matrix of TF-IDF features.

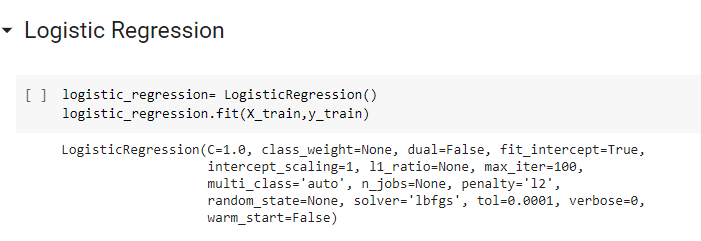


1. Now that we have pre-processed our data, we split the data into testing and training data using train\_test\_split.

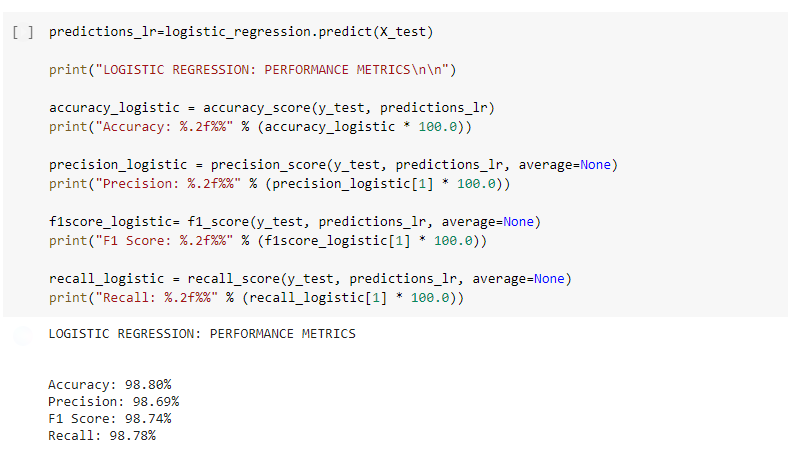


**I. Logistic Regression**

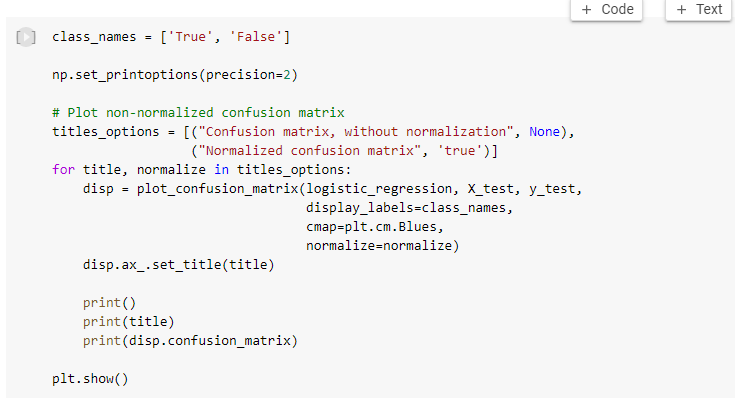
1. We create our logistic regression model and fit our training data to it.

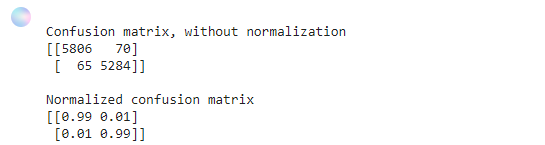


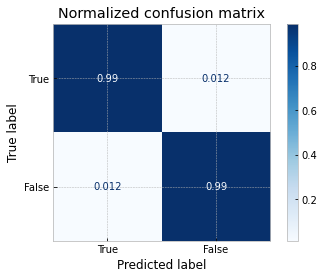
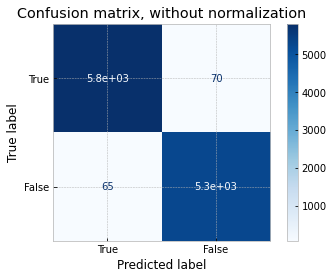
1. We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 score and recall. We get Accuracy - 98.80%, Precision- 98.69%, F1 Score- 98.74% and Recall- 98.78%.



1. We plot a confusion matrix and a normalized confusion matrix.

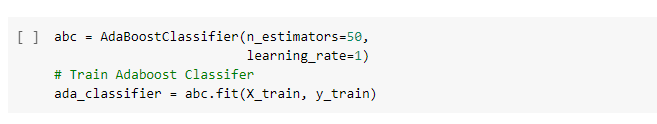




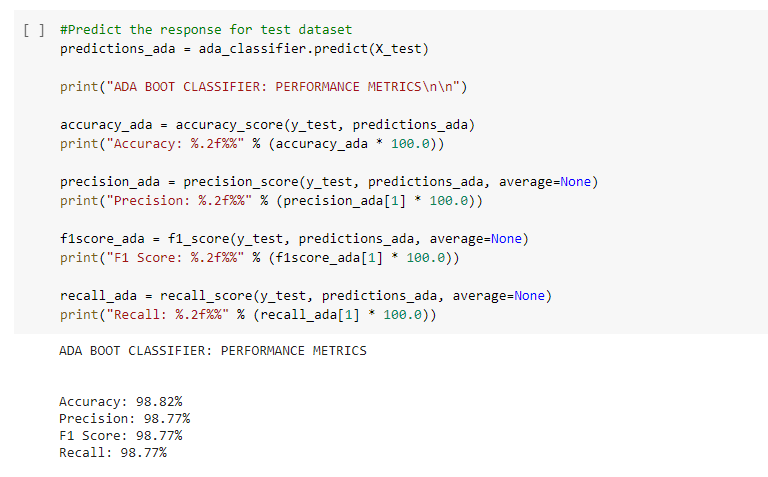


**II. ADA**

1. We create our AdaBoost Classifier model and fit our training data to it.

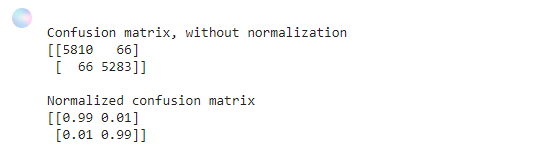


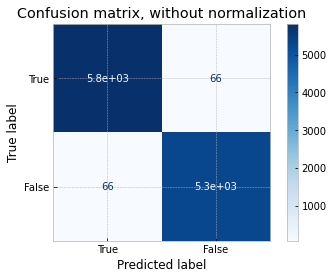
1. We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 score and recall. We get Accuracy - 98.82%, Precision- 98.77%, F1 Score- 98.77% and Recall- 98.77%.

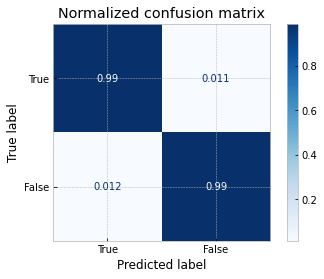


1. We plot a confusion matrix and a normalized confusion matrix.



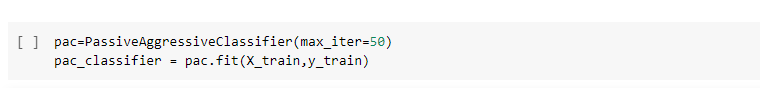




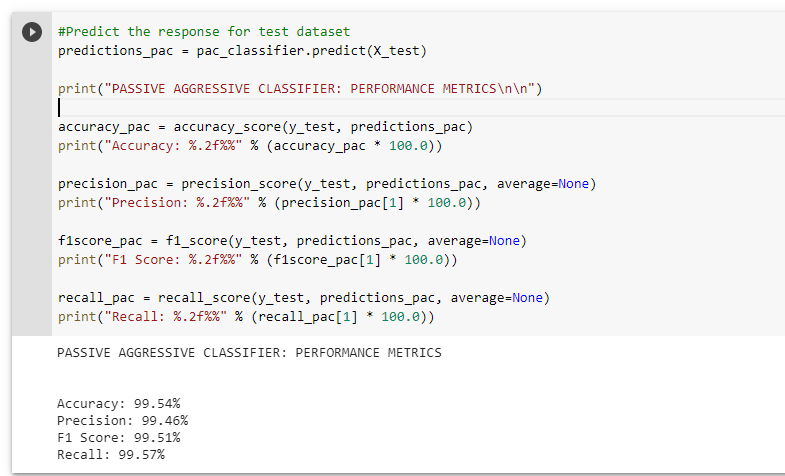


**III. PAC**

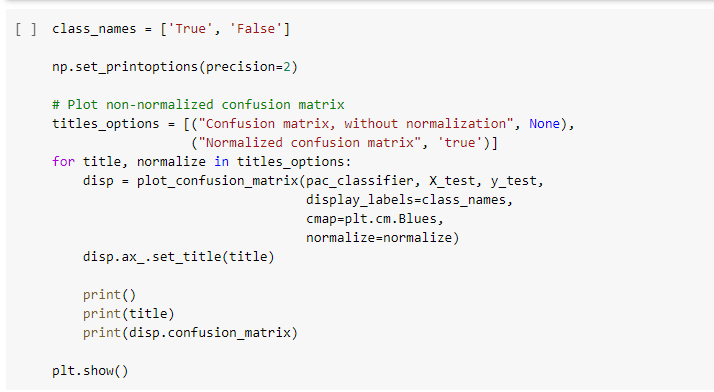
1. We create our Passive Aggressive Classifier model and fit our training data to it.

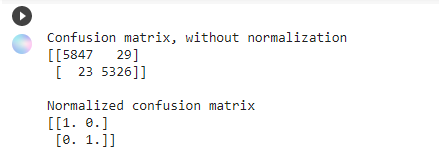


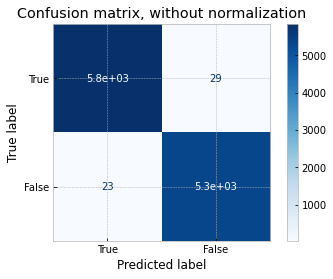
1. We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 score and recall. We get Accuracy - 99.54%, Precision- 99.46%, F1 Score- 99.51% and Recall- 99.57%.

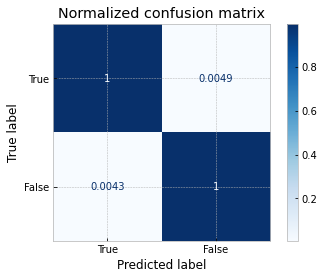


1. We plot a confusion matrix and a normalized confusion matrix.



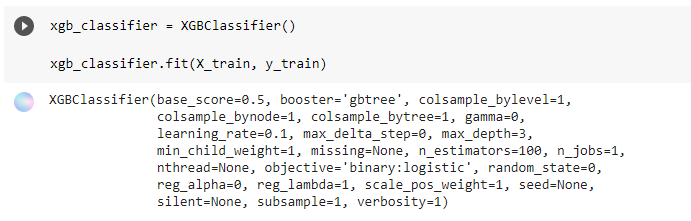




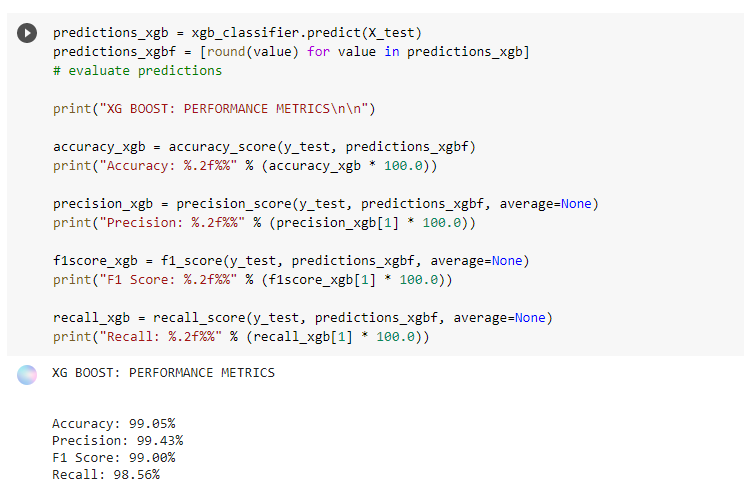


**IV. XGBoost**

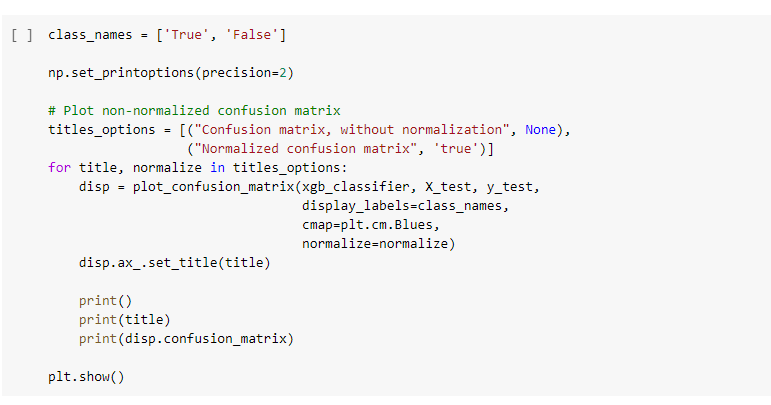
1. We create our XG Boost Classifier model and fit our training data to it.

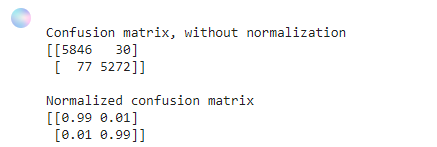
****

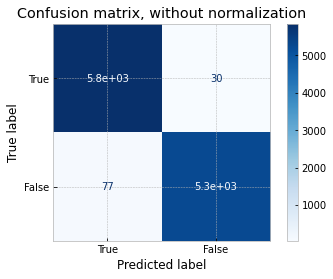
1. We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 score and recall. We get Accuracy - 99.05%, Precision- 99.43%, F1 Score- 99.00% and Recall- 98.56%.

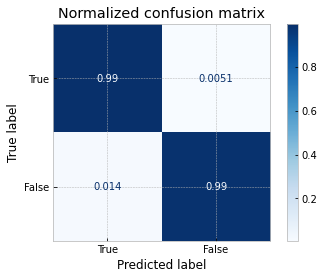


1. We plot a confusion matrix and a normalized confusion matrix.



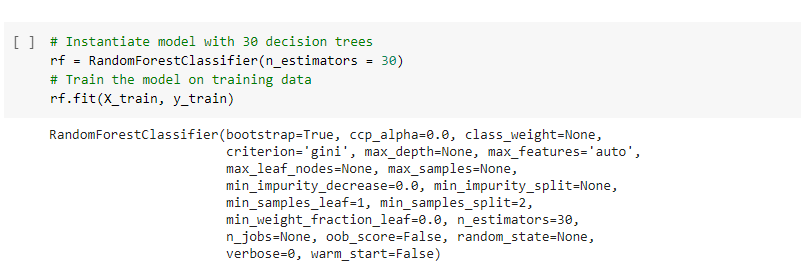




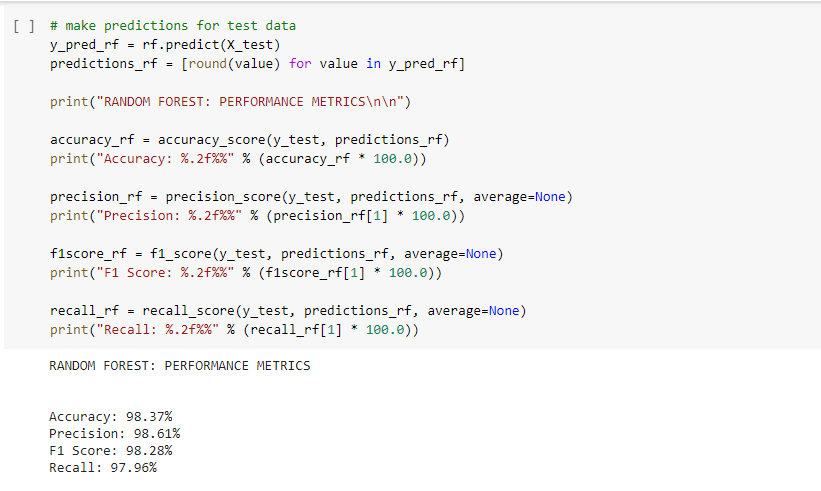


**V. Random Forest**

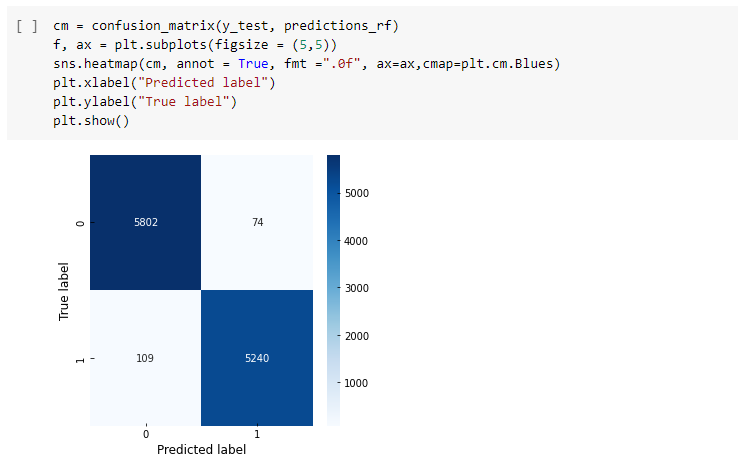
1. We create our Random Forest Classifier model and fit our training data to it.



1. We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 score and recall. We get Accuracy - 98.37%, Precision- 98.61%, F1 Score- 98.28% and Recall- 97.96%.

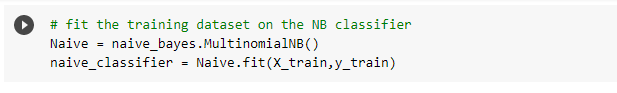


1. We plot a confusion matrix.

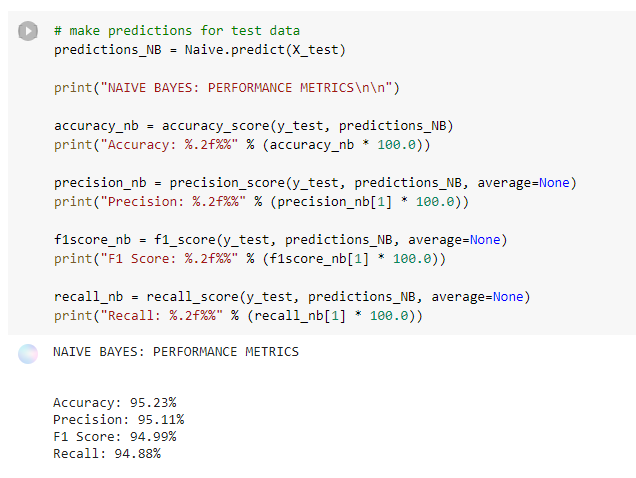


**VI. Naive Bayes**

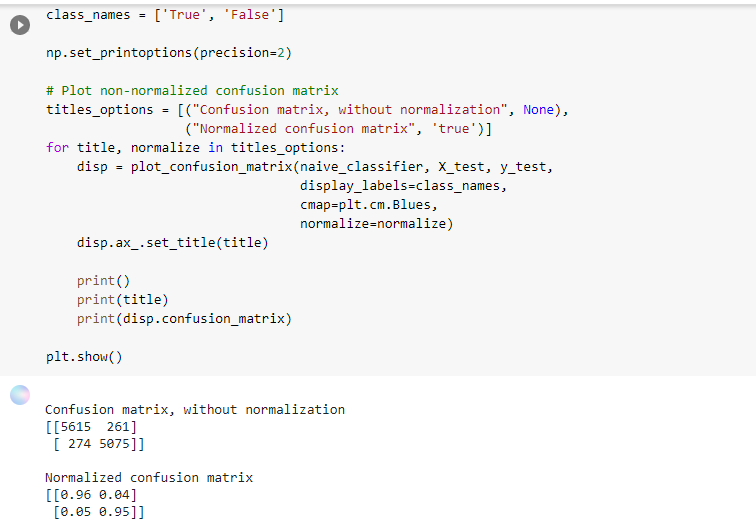
1. We create our Naive Bayes Classifier model and fit our training data to it.

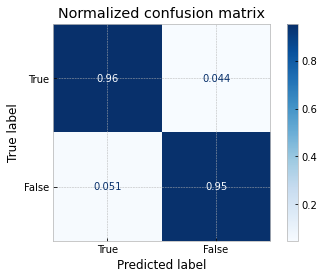
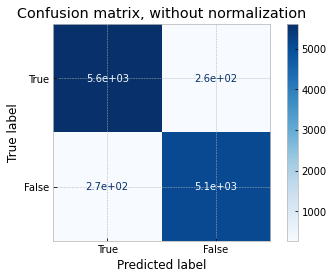


1. We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 score and recall. We get Accuracy - 95.23%, Precision- 95.11%, F1 Score- 94.99% and Recall- 94.88%.



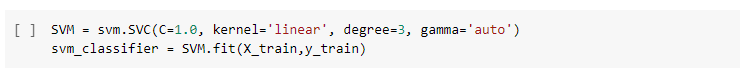
1. We plot a confusion matrix and a normalized confusion matrix.



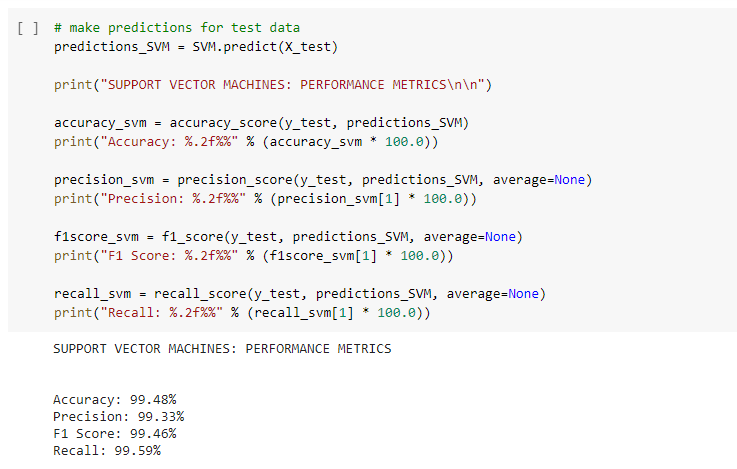


**VII. Support Vector Machine (SVM)**

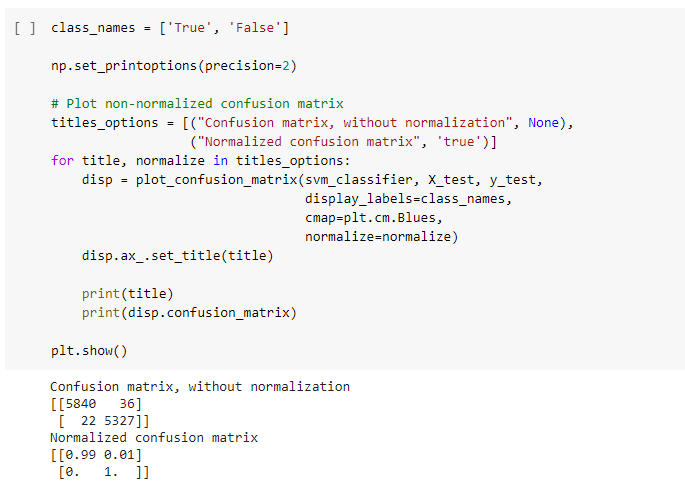
1. We create our SVM Classifier model and fit our training data to it.

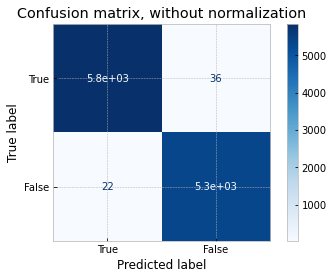


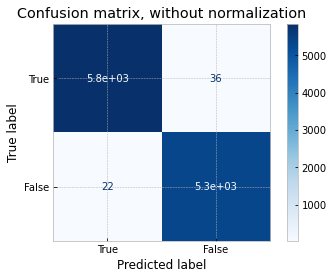
1. We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 score and recall. We get Accuracy - 99.48%, Precision- 99.33%, F1 Score- 99.46% and Recall- 99.59%



1. We plot a confusion matrix and a normalized confusion matrix.

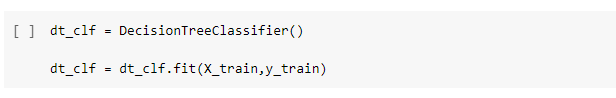




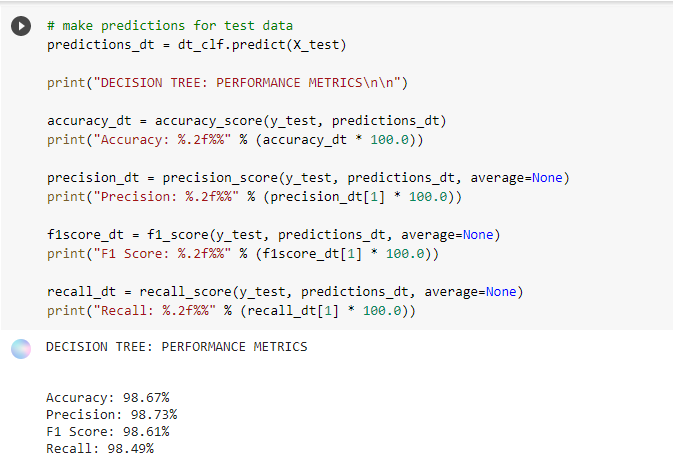


**VIII. Decision Tree**

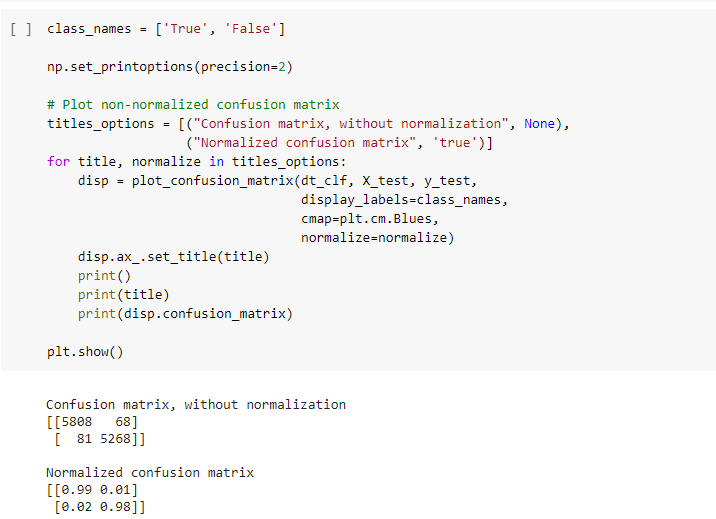
1. We create our Decision Tree Classifier model and fit our training data to it.

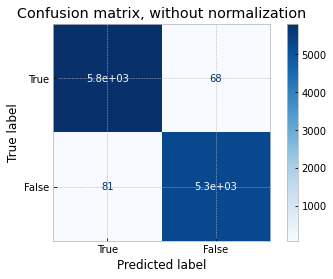


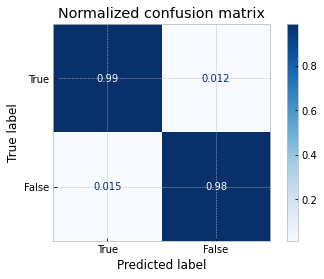
1. We then predict using our testing data. We use our predictions and testing labels to calculate the metrics - accuracy, precision, F1 score and recall. We get Accuracy - 98.67%, Precision- 98.73%, F1 Score- 98.61% and Recall- 98.49%



1. We plot a confusion matrix and a normalized confusion matrix.



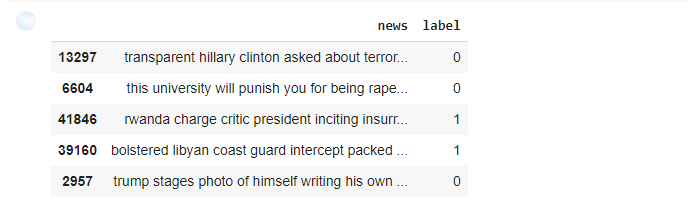




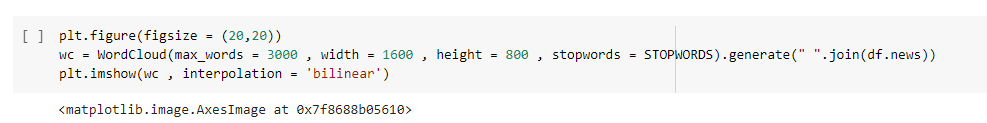
**VIII. RNN**

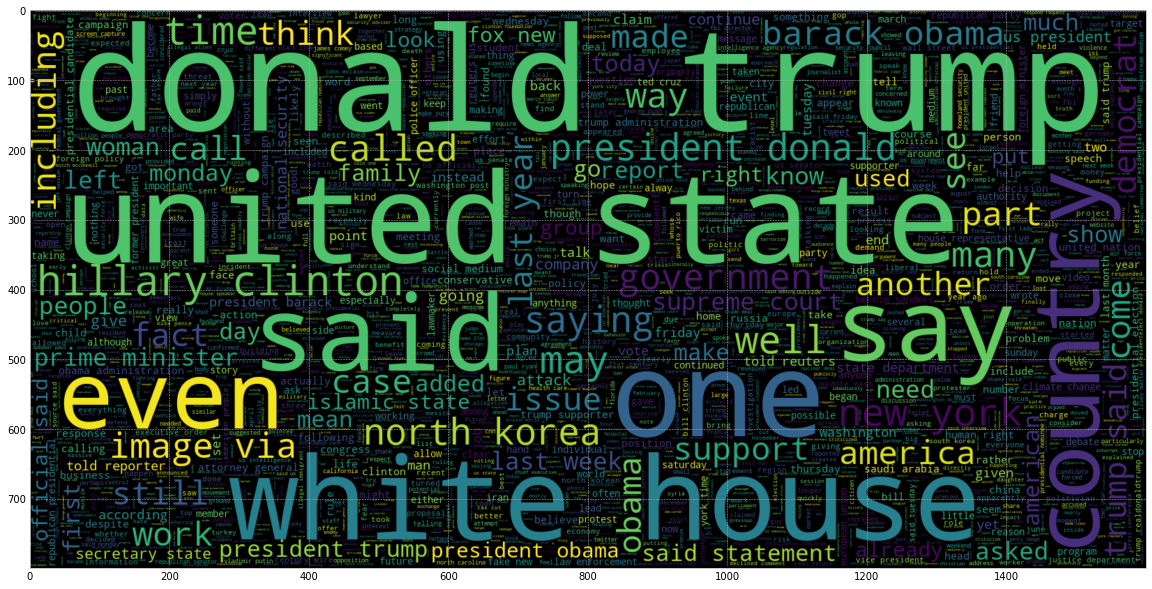
1. For our RNN model, we first preprocess the dataset news by normalizing it by removing non-words and extra spaces



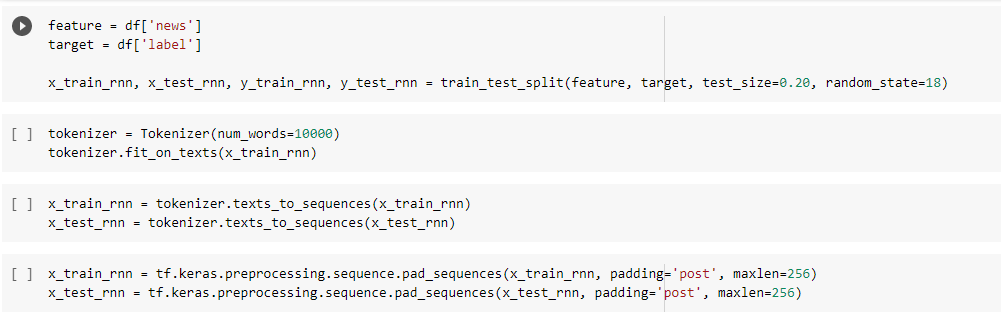


1. We then create a Wordcloud to highlight the popular words and phrases based on frequency and relevance.

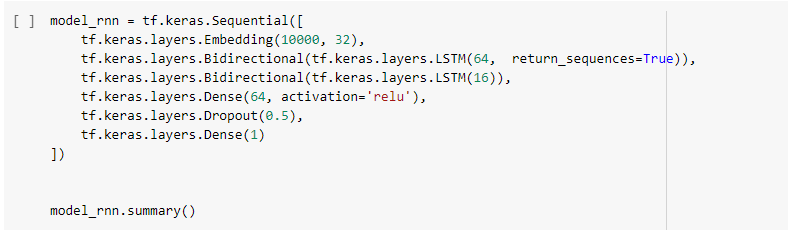




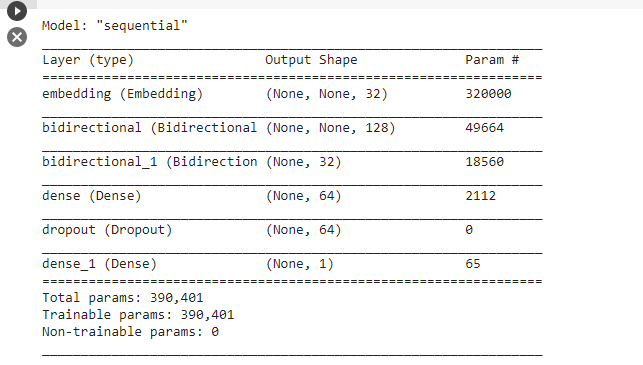
1. After that we divide the dataset features and target into testing and training data.



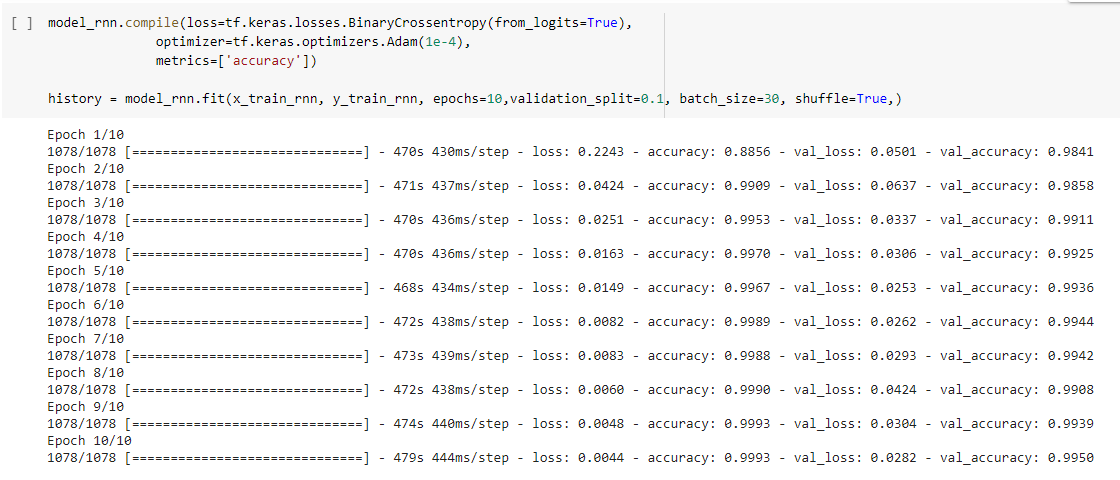
1. Then we build our RNN sequential model and add the layers to it



1. This is our model summary:

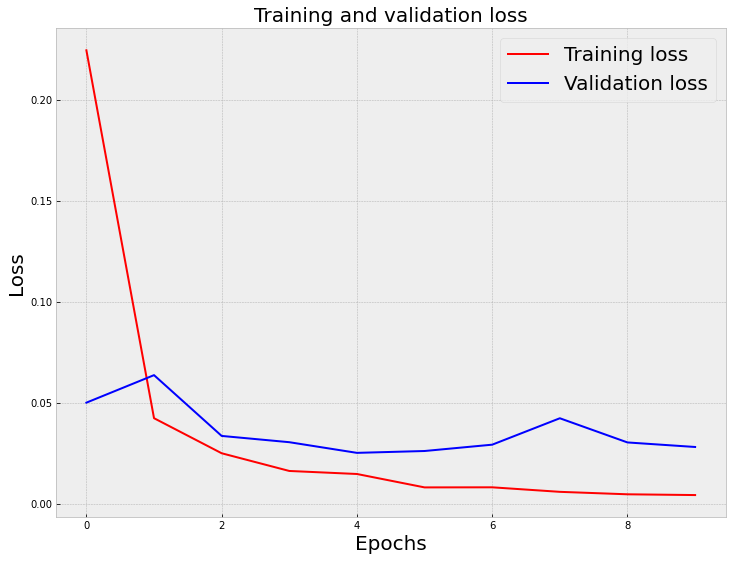


1. We compile our RNN model and fit the training data to it for 10 epochs. We see that with every epoch, the loss decreases and the accuracy increases. At the 10th epoch, we get a validation accuracy of 99.50% and the validation loss is 0.028, whereas the loss is of 0.0044 and accuracy is 99.93%



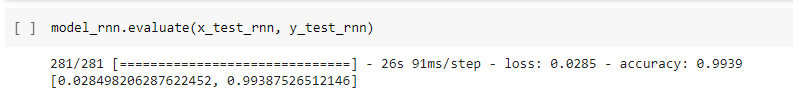
1. We then use the model history to plot the training loss and accuracy:



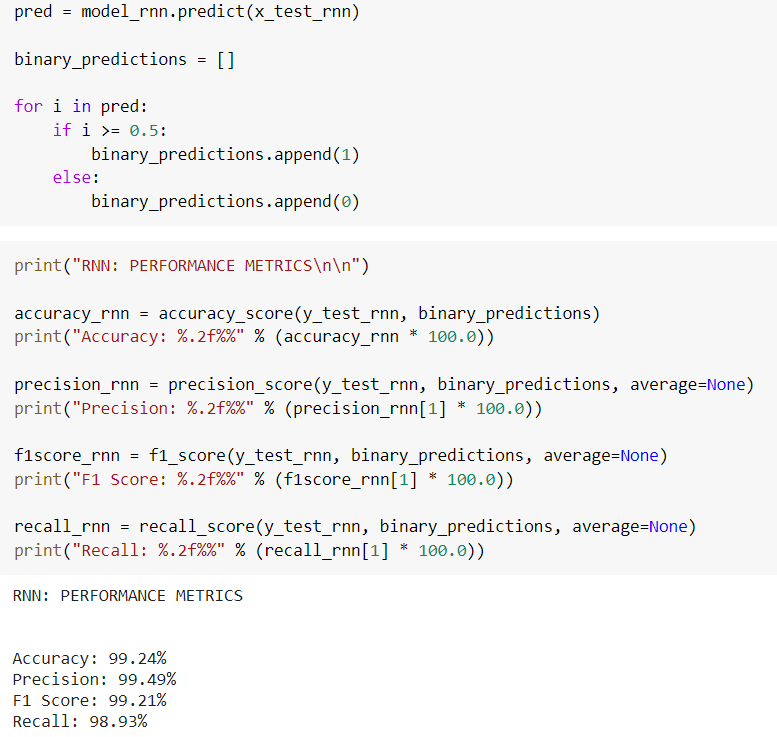


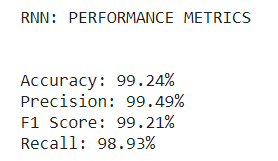


1. We evaluate the model with the testing data and the loss turns out to be 2.84% and an accuracy of 99.387%

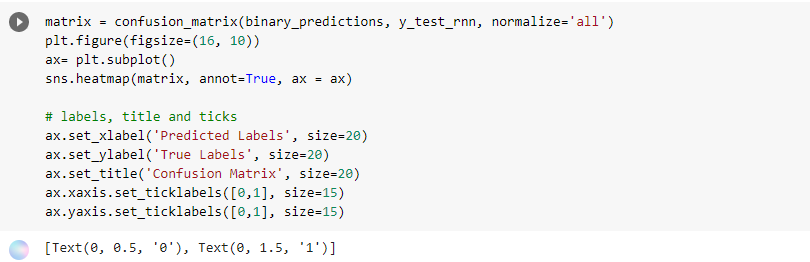


1. We then calculate other performance metrics. The accuracy as mentioned earlier is 99.39%, the precision is 99.35%, the f1 score is 99.36% and the recall is 99.37%.



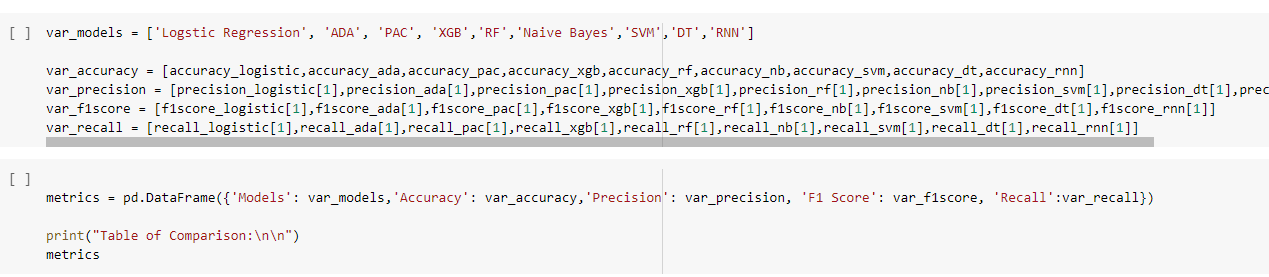


1. We also plot a confusion matrix to see how well has our model classified the records.

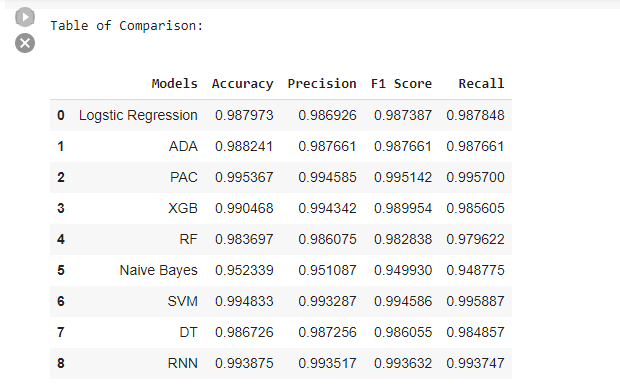


**X. Comparing all the Models**

1. We create separate lists to store the performance metrics of each model and create a dataframe for it.

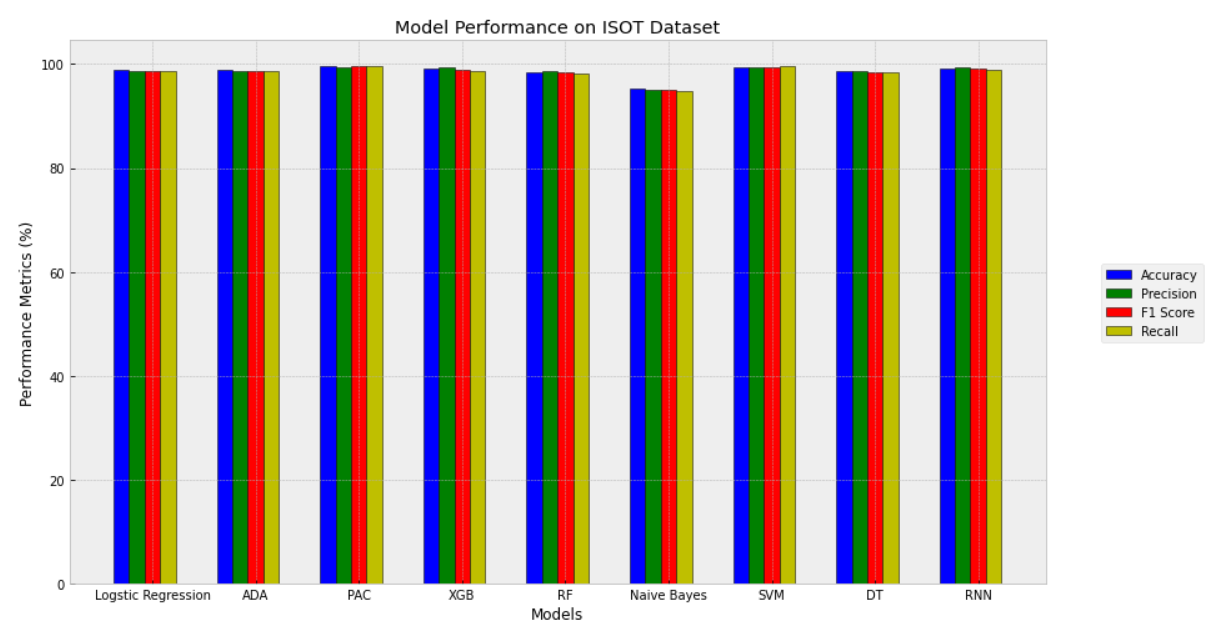


1. This is the dataframe that we have created which shows the models and their performance metrics.

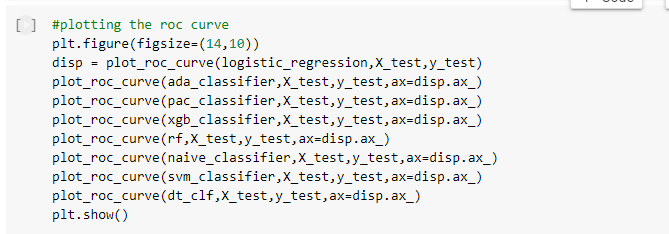


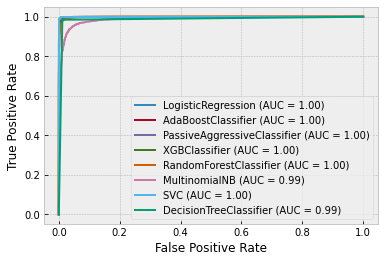
1. We plot a bar graph to compare the performance metrics of all the models on the ISOT dataset.





1. We also plot the ROC curve to better understand the performance of each model. We see that SVM is ideally the best classifier for our dataset.





**Outcome:**

**CO3:** Comprehend radial-basis-function (RBF) networks and Kernel learning method work.

**Conclusion: (Conclusion to be based on the objectives and outcomes achieved)**

We downloaded the ISOT dataset from <https://www.uvic.ca/ecs/ece/isot/datasets/index.php> and uploaded it to our drive, and then loaded it and preprocessed it using various NLP algorithms like tokenization, stop words removal, lemmatization and stemming. We vectorized the text documents using count vectorizer and tf-idf vectorizer. After preprocessing, we split the data into testing and training and we built nine models using nine different classification algorithms and used the predictions to calculate the performance metrics. The details of each are given below:

| **Sr** | **Models** | **Accuracy** | **Precision** | **F1 Score** | **Recall** |
| --- | --- | --- | --- | --- | --- |
| 1 | Logistic Regression | 0.987973 | 0.986926 | 0.987387 | 0.987848 |
| 2 | ADABoost Classifier | 0.988241 | 0.987661 | 0.987661 | 0.987661 |
| 3 | Passive Aggressive Classifier | 0.995367 | 0.994585 | 0.995142 | 0.995700 |
| 4 | XG Boost | 0.990468 | 0.994342 | 0.989954 | 0.985605 |
| 5 | Random Forest | 0.983697 | 0.986075 | 0.982838 | 0.979622 |
| 6 | Naive Bayes | 0.952339 | 0.951087 | 0.949930 | 0.948775 |
| 7 | SVM | 0.994833 | 0.993287 | 0.994586 | 0.995887 |
| 8 | Decision Tree | 0.986726 | 0.987256 | 0.986055 | 0.984857 |
| 9 | RNN | 0.993875 | 0.993517 | 0.993632 | 0.993747 |

From the ROC curve and the bar plot which compares the performance of all the models, we conclude that the SVM (accuracy-99.48%, precision - 99.33%, f1 score-99.46%, recall-99.59%) is the best algorithm on our ISOT dataset for the task of fake news detection and classification. The colab notebook of our implementation is available at - <https://colab.research.google.com/drive/1Jgn0uP2frisa9GS1AaI5iUlsHnt9ziO5?usp=sharing> .