UNIVERSITY OF LONDON
INTERNATIONAL PROGRAMMES

BSc Computer Science and Related Subjects

<u>CM3070 PROJECT</u>
<u>FINAL PROJECT REPORT</u>

MISINFORMATION MITIGATION USING FAKE NEWS DETECTION EMPLOYED BY NATURAL LANGUAGE PROCESSING

Author: NIRMAL PARTHASARATHY
Student Number : 190428228
Date of Submission: 13TH MARCH 2023
Supervisor: SARITA SINGH

Contents

CHAPTER 1: INTRODUCTION

1.1 Abstract

Fake news. Misinformation. Disinformation. Wrong information. These are several interchangeable terms that are used to describe articles, stories or in general, information that appear to be legitimate news on the internet or other propagatable media. They are usually disseminated with a motivation to influence opinions, political views, or merely as a joke, causing indirect or direct negative consequences. This project report discusses the various approaches I have researched, evaluated, and employed by using natural language processing (NLP), machine learning and deep learning to address the fake news problem. Using evaluation metrics such as precision/accuracy/recall/F1-scores, I would then develop classification workflows which involve machine learning and deep learning based classifiers to improve on the metrics to achieve a sustainable and resilient fake news detection text-based classifier.

1.2 Project Template & Idea

The project template that I have chosen to tackle for my Final Year Project is Fake News Detection, under the CM3060 Natural Language Processing module.

I am going to be making use of the techniques, concepts, and course material I have learnt and explored over the course of my university term with UOL to tackle the task of developing and implementing a software module which would effectively be able to distinguish between real and fake news on an increasingly accurate basis, compared to currently available NLP and machine learning classifiers which attempt to do the same.

1.3 Motivation

According to statistics published on statisa.com, it has been calculated that 62% of adults believe that fake news is prevalent on online news websites and platforms, compared to TV/radio sources at 52% and newspapers/magazines at 52%. The demographic of this study spans across 27 countries among 19541 respondents; age group 16-74 years (Watson, A. 2021).

Current fake news detection models or tools aim to replace human judgement instead of being augmented to be a collaborative tool in making the decision. Hence it is important to consider and address the above public responses as it emphasizes on the issue of fake news and the necessity to train better and improved models to effectively achieve an accurate detection workflow of the several forms of fake information that is being published every second, every day through various online channels.

The motivation behind undertaking this project is that the leading limitation with currently deployed NLP pipelines and classification models is that insufficient or inaccurately labelled datasets and categorizations are used to train and test news data as an attempt to optimize fake news detection workflows.

1.4 Concept

The concept of this project starts off by firstly focusing on evaluating currently developed and available natural language processing pipelines (n-grams, count vectorization, Bag odf Words etc.) and classification models (Naïve Bayes, SVM, Random Forest) based on set evaluation metrics. With recall, precision, accuracy and F1-scores, we would be able to analyze and develop baseline creations to initiate a benchmark of the classification models which can be used to train and test the data on and implement improved variations of data processing pipelines on the chosen dataset, efficient machine learning or deep learning based algorithms which are able to handle huge data pre-processing or NLP pipelines.

The final deliverable has been targeted for me is to output a software module embodied as a Jupyter Notebook using Python. However, I would like to put forth here that to extend my interest in this project and explore newer scope of work, I would also be aiming to develop a Chrome extension which can be deployed to detect fake news on Chrome browsers. The fake news detection algorithm I am hoping to deploy within the Chrome extension will be based on my research and findings that I build and develop upon over the course of this final year project.

1.5 Aims

1. Build an apt NLP pipeline for the chosen dataset
2. Understanding why currently available NLP and machine learning models and classifiers may or may not be as effective in distinguishing between fake and real news articles
3. Exploring the cost to benefit ratios of implementing and developing new and improved NLP pipelines and ML classifiers on the chosen dataset
4. Achieving an improved fake news detection model to deliver a viable solution for users using deep neural network models. (More on this in Chapter 3: Project Design)

1.6 Objectives

1. Employing multiple word embedding techniques for NLP pipelines
2. Feature analysis and extraction using dependency evaluation
3. Conducting the appropriate literature reviews on existing models, research studies and techniques
4. Creating a baseline model for performance evaluation against improved models
5. Developing improved models for future model evaluation and hyperparameter tuning to achieve optimal fake news detection accuracies

6. Using the effective NLP pipeline and model to modularly implement a software output for applicative use
7. Packaging research findings and results into an outlined and structured report to achieve proper documentation

1.7 Deliverables

1. Jupyter Notebook using Python – well documented NLP pipelines, baseline models, self-developed classification models and algorithms, software module which performs the relevant fake news detection application
2. Preliminary Project Report
3. Final Project Report
4. Presentation Video

1.8 Preliminary Research Questions

1. Is it necessary to build an increasingly accurate NLP pipeline and model system to detect fake news when considering the capabilities of currently available options?
2. How can the content of the fake news be analyzed using feature extraction methods to improve model performance in detecting the presence of fake news identifiers?
3. Are deep learning neural network models better in comparison to classification models for the chosen dataset and to achieve better accuracies when it comes to evaluating the improved models?

CHAPTER 2: LITERATURE REVIEW

In this section, we are going to explore my research and evaluation on the available related work in the field of fake news detection. This is such that I would be able to identify and apprise myself of knowledge gaps that I may possess, which could limit the theoretical concepts and perspectives I could make use of to further my research and application in this project.

Firstly, I have chosen to explore a similar project to the project I have chosen to undertake. Next, we will be evaluating a study assessing the significance and effectiveness of big data and the quality of the data being used for fake news detection in the current digital landscape. Finally, I have conducted critical evaluation on the NLP techniques and models that I plan to employ in this project, hence requiring reviews on the relevant methodologies, code documentation and libraries which would help support the algorithms I plan to develop.

2.1 DeepFakE: Fake News Detection by coalescing social, user and content based features with deep neural networks

The first paper we are going to be looking at is part of a project which was published by Rohit Kumar Kaliyar, Anurag Goswami and Pratik Narang in May 2020, as part of a published journal known as *The Journal of Supercomputing* (Kaliyar. 2020).

The principal concept of the DeepFakE project registers the significance of the relationships that can be made between content based features found in news articles and the features which encompass the social networks that the said news articles are being propagated through to spread fake information.

Kaliyar et al. also explore the different approaches to fake news detection, namely learning, knowledge and feature based approaches. Figure 1 below outlines the various approaches that they have considered as part of their project (Kaliyar. 2020).

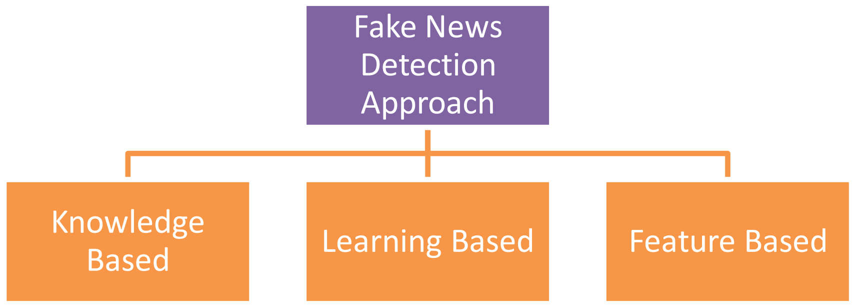


Figure 1 Fake News Detection Approaches (Kaliyar. 2020)

Out of the 3 approaches Kaliyar et al. have decided to focus on the feature based approach to applying natural language processing on selected features of the news data and using machine learning models for training and data classification (Kaliyar. 2020).

Content based features generally refer to syntactic and linguistical features of the news articles they have analyzed. This also includes the use of parts of speech (PoS) tagging, lemmatization, and stemming as part of multiple NLP pre-processing pipelines which enable Kaliyar et al. to perform the necessary engagements (Feng S. 2012).

Social context based features refer to user based engagements within the social media platforms. Kaliyar et al. have studied and explained about the patterns which arise when a fake news is created and spread across the social media platforms (Shu K. 2019). Kaliyar et al. explains about the echo chamber phenomenon, which is a very important user based feature that they have employed to support the project in. The echo chamber concept is explained as a *'a group of like-minded people'* (Kaliyar. 2020), where users do not always exist as just 1 unique entity and hence another alternative to how and where fake news is spread from is to analyze community related data, where users of the same motivations are settled into and these circles usually have a democracy on the ideas that are circulated around (Kaliyar. 2020).

Hence, the technical approach Kaliyar et al. have used to address the fake news detection problem is to first understand the relations between user profiles on social media platforms (social context features) and the communities which the user profiles participate in (echo chamber concept).

This relationship enables them to build a triple pronged tensor (Rabanser S. 2017) which they would then be able to use with the news content based tensor to conduct and test their hypothesis with machine learning and deep learning based models.

Tensors are defined as multi-dimensional arrays that enable a matrices to be formed in higher dimensions so that related data can be coalesced into a single data structure (Papanastasiou F. 2019).

With the above tensor formations, Kaliyar et al. are able to perform different types of fake news detection workflows. Figure 2 below shows the various classification workflows that they have proposed in their methodology (Kaliyar. 2020).

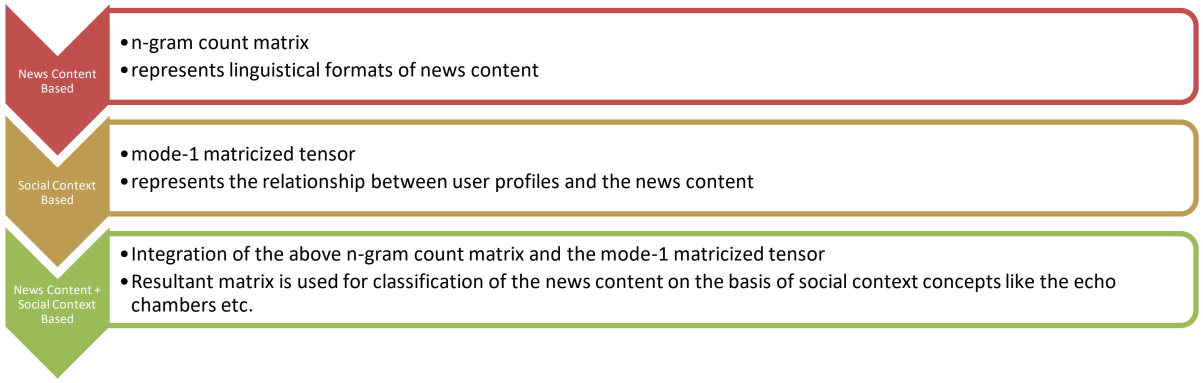

News Content Based
• n-gram count matrix
• represents linguistical formats of news content
Social Context Based
• mode-1 matricized tensor
• represents the relationship between user profiles and the news content
News Content + Social Context Based
• Integration of the above n-gram count matrix and the mode-1 matricized tensor
• Resultant matrix is used for classification of the news content on the basis of social context concepts like the echo chambers etc.

Figure 2 Classification based on feature selection types (Kaliyar. 2020)

In another vital section in this paper, Kaliyar et al. also explain the machine learning and deep learning network and model they have used to train the datasets on and classify the news data.

1. XGBoost Machine Learning Algorithm
    a. Decision Tree based
    b. Gradient boosting model

Due to its gradient boosting characteristic, the XGBoost algorithm is able to reduce feature splitting incidents by analyzing feature distributions among the data nodes in concern.

2. DeepFakE – Proposed Deep Learning Multi-Layered Neural Network
    a. Designed as 4 hidden layers
    b. Alternates between 128 input/output size dense layers and 128 input/ouput size dropout layers
    c. Final 2 layers are 32 input/output size dense/dropout layers with a binary output in the final dense layer

The Deep Neural Network (DNN) based DeepFakE model allows features to be input and classified into a binary categorization of fake or real news in the dense layers. The dropout layers allow the network to regularize any overfitting of the data being applied during the training phase.

The dataset that Kaliyar et al. have used has been acquired from BuzzFeed and Politifact, both being credible sources of data which have a reasonable distribution of fake and real news articles from a huge user pool.

The table below encompasses the evaluative results of the experiments that Kaliyar et al. have conducted to assess the performance of their classifiers when using the matrices formed from the features selection process that I have explained earlier in this section as inputs and hence how accurate their machine learning based classifier, XGBoost and the

DNN based proposed model DeepFakE performs in the task of detecting fake news in the dataset they have acquired. Note here that they have indeed done the entire evaluation workflow that is normally done for ML classification tasks, deriving a confusion matrix for the accuracy, precision, recall and F1-score calculations, but I have only decided to include the accuracy and precision scores here rounded to 3 s.f. (Natekin A. 2013).

Table 1 Experimental Evaluative Results using the Politifact dataset (Kaliyar. 2020)

Classification Approach	Accuracy	Precision
News Content & XGBoost	78.8%	74.5%
News Content & Social Context & XGBoost	86.7%	80.3%
News Content & Social Context & Proposed DNN (DeepFakE)	88.6%	82.1%
Social Context & XGBoost	84.5%	78.7%

From the above results, it is evident that the proposed DeepFakE DNN based network was instrumental in improving the accuracy and precision scores of the fake news detection workflow that Kaliyar et al. have decided to implement in collaboration with the feature based approach they have used.

Hence it can be put forth that an important aspect that can be taken away from this methodology is that the feature selection process is an significant and constructive phase of the NLP project that I am going to be undertaking in the field of fake news detection. Kaliyar et al. have beautifully outlined the benefits of using the impact of the social constructs and circles on the news content that is circulated and propagated and how the content of said articles could be designed and fabricated to disseminate fake information.

2.2 Analyzing Big Data and quality of data used to mitigate misinformation

Studies and research over the years since the culmination of misinformation being a problem; have proved that there are several ways to approach the fake news detection process. Natural language processing has increasingly become the forefront of analyzing text and linguistical data that is input into the system. Hence, it is important to understand here that fake news is not just an issue that pertains to information being disseminated with inaccuracies or negative intentions but a data problem. Specifically, a big data problem which this paper discusses in detail (Torabi Asr, F. 2019).

Fatemeh Torabi Asr and Maite Taboada (Torabi Asr, F. 2019) have collaborated on this paper to research and address the topic of data acquisition and maintaining a balanced distribution of news categories and articles that the data represents as inputs which are then fed into NLP pipelines, machine learning models and networks for classification purposes.

Fatemeh et al. have emphasized on the need of fake news data from general media and social media entities and companies. The paper clearly outlines that fake news detection is increasingly a text-based classification problem and natural language processing concepts and experiments do need to be applied vitally to distinguish fake news and real news (Torabi Asr, F. 2019). To propagate this workflow, sufficient training data is always required.

Hence, the paper also focuses on where fake news data/datasets can be found, through multiple sourcing methods and databases. MisInfoText (Torabi Asr, F. 2019), is a repository of news articles containing links and source paths to several publicly available datasets which are comprehensively labelled. This data has been web scraped and cleaned as a service by MisInfoText (Torabi Asr, F. 2019), and it is available in the public domain. The labelling has done by reliable fact checking sources (Pavleska T. 2018), which are proficient in collating and distributing large amounts of data. Fatemeh et al. have also reviewed the availability of heavily researched and trained datasets from other related papers, which have been tabulated with the labelling systems, topical characteristics of news articles and data size and types present in the datasets (Torabi Asr, F. 2019). It also makes comparisons between the famous LIAR dataset (Wang. 2017) and FEVER (Thorne et al. 2018) dataset.

Using the above dataset analysis on the different types of data and sources that are being used for fake news detection, I am able to derive a clearer picture as to how to choose my dataset to approach the my fake news detection project, based on the purposes laid out for each dataset in correlation to the topics that are collated, and the labelling systems which are put in place to possibly enable a more comprehensive text classification solution.

Fatemeh et al. have also delved into the various approaches that are currently undertaken in domains other than the text classification, NLP and machine learning space to address the fake news problem.

1. Public education
    a. Improving media literacy and etiquette across the relevant institutions and entities which are in the news dissemination ecosystems (Torabi Asr, F. 2019)
2. Spread control
    a. Understanding that fake news spreads like wildfire due to the filter bubble and echo chamber concepts, where individuals are only exposed to singular perspectives and taking steps to prevent the reinforcement of the said perspectives which may cause the spread of misinformation (Torabi Asr, F. 2019)
3. Manual checking
    a. Use of fact checking websites or frameworks
4. Automatic checking

a. Usage of text classification workflows comprising of NLP, machine learning, or deep learning networks

To summarize, it can be said that the main takeaway from this paper is that regardless of dataset acquisition methods, actions that are taken to prevent fake news from spreading, it is vital to ensure that to develop a resilient fake news classification system, accurately labelled fake news articles must be made possible. There must be a clear distinction, regimentation, and conviction towards classifying a news article or story by subject matter experts as real or fake in the labelling process, in order to train sustainable classification systems.

CHAPTER 3: DESIGN

3.1 Revisiting Project Aims

Before examining the project design, let us first have a refresh on the aim and motivation behind this Final Year Project idea.

Aim:

To achieve an improved, highly accurate fake news detection model which can be deployed and delivered as a viable, robust solution for users by employing the use of NLP, machine learning and deep learning research and implementations.

3.2 Domain and User Justification

To expand on the user domain of this project, we must be able to assess if the fake news detection models I have implemented address and justify the user domain of this project.

The project's ultimate goal is to mitigate the occurrence of fake news and bring an awareness to users of news websites, social media platforms that the information they are consuming is not always legitimate; and that the development of NLP workflows and building various classification models are an attempt to regulate the dissemination of groundless statements and hoaxes being published online.

3.3 Project Relevant Techniques

There are 3 main techniques which are instrumental to developing an accurate model to address the fake news detection problem I have posed in this project:

1. Feature Extraction
2. Classification Algorithm Analysis
3. Model Performance Evaluation

Feature Extraction:

Feature extraction techniques can be generally divided into 3 core levels:

1. Social
    a. Features which relate to the characteristics of the digital circles and groups which are formed under a social agreement based on common interests, inclinations and topics. The echo chamber and filter bubble concepts I have explained in Section 2 plays an important role here, where data can be analyzed based how users are consuming and spreading the fake information that they encounter.

2. User
    a. Features which relate to the demographics of entities/individuals which interact with fake news data in the digital landscape, be it news media channels, or social media platforms
3. Content
    a. Features which relate to the lexical and linguistical features of the text content found in the bodies of news articles and headlines. These are the most powerful and weightage heavy features to fake news detection due to their importance in text classification processes

Thanks to the literature review I have conducted in section 2, I have gained a better understanding of the benefits and drawbacks that arise due to the collaboration of social and user features from the Kaliyar et al. paper from the published journal source (Kaliyar. 2020). Content features provide a more direct textual and linguistical approach to feature selection and model construction.

There are several methods we can use to engineer features into vectors from the raw dataset; as inputs for model construction:

1. Count Vectorization
2. Word Embedding
3. Linguistical Analysis using NLP
4. TF-IDF Vectorization
5. Topic Modelling

The relevant tools for the above techniques can be generally found in the sklearn library based on Python.

Classification Algorithms:

In order to develop a software module which is able to perform fake news detection, a very important step is to execute the text classification phase. There is a huge pool of machine learning models which we can use to train classifiers with the use of the feature data we have extracted in the precedent phase.

Machine Learning Classifier Examples:

1. Support Vector Machine (SVM) Classifier
2. Naïve Bayes Classifier
3. Passive Aggressive Classifier
4. Bidirectional Encoder Representation from Transformers (BERT)
5. Deep Neural Networks

Deep Neural Network Examples:

1. Convolutional Neural Networks (CNN)
2. Recurrent Convolutional Neural Networks (RCNN)
3. Long-Short Term Model (LSTM)

I will be expanding more on these classification models and deep neural networks as part of my project structure and methodology in the following sections. These models can mostly be found from the sklearn library based on Python. As for the Deep Neural Networks, depending on the case, we would be able to access the relevant libraries from publicly available sources.

Model Performance Evaluation:

To evaluate the performance and effectiveness of the implemented models effectively and sustainably among the various workflows in this project, it is important to have a proper evaluation system to compare the accuracies, recall and precision of the text classification approaches.

The standard library that can achieve this sklearn, using the sklearn.metrics module. It enables us to calculate and plot confusion matrices, calculate accuracy, recall, precision and F1 scores.

Accuracy scores would quantify and measure the number of news data points that the classification model in question has predicted as fake or real news as exactly as what is represented in the test data set. Recall scores measure the ratio of the number of correctly predicted news data points to the sum of correctly predicted real news and wrongly predicted fake news. This helps us evaluate the ability of the classification model in question to correctly detect all real news data points. Precision scores would measure the ratio of correctly predicted real news data points to the sum of correctly and wrongly predicted real news data points. This helps us evaluate the ability of the classification model in question to avoid classifying a fake news data point as real news.

Using these metrics, I would be able to tabulate and compare the effectiveness and robustness of the classification models I propose to implement as part of this project.

3.4 Project Methodology

Dataset Acquisition and Analysis:

The dataset I am using for this project has been ethically considered and appropriately sourced from Kaggle under the public domain (Kajal Yadav [Kaggle]. 2020).

1. Text corpus of news articles and metadata scraped across 600 webpages
2. Approximately 10K news article data and metadata

3. 6 feature attributes: News Headline, News URL, Source, Stated On, Date, Label
4. Multi-Class Labelling: False, Pants on Fire, True, Mostly True, Barely True, Half True
5. Licensing Type: Public Domain (Creative Commons License: CC BY-SA 4.0) (Kajal Yadav [Kaggle]. 2020)

Once this step is accomplished, the project design is going to require the dataset to be split into 2 pipeline versions.

1. Non web scraped, news headline Fake-Real News Dataset
2. Web scraped, news headline + news content Fake-Real News Dataset

Further details and outline will be provided in Chapter 4, Implementation.

Dataset Pre-Processing:

There are going to be 2 main pre-processing pipelines using NLP and other text data cleaning techniques to prepare the raw data to be input into classification models and workflows for this project.

1. SimpleNLP
2. Extensive Data Cleanse NLP (EDC-NLP)

The SimpleNLP pipeline is going to be representing a rudimentary as needed basis of pre-processing characteristic, with simple case conversions, lemmatization and stopword removals.

As for the Extensive Data Cleanse NLP (EDC-NLP) pipeline, it is going to be designed according to the linguistical characteristics of the news headlines and news bodies found in the dataset I have acquired in the previous step. This is such that the data cleaning process is able to efficiently tackle the data point necessities for text classification while preserving the context of the news data.

The analysis of the news headline and news content features of the dataset has posed some pre-processing requirements as follows:

1. Removal of →
    a. Links, Whitespaces, Newlines, Tabs, Accented Characters, Special Characters, Stopwords, HTML tags
    b. Conversion of resultant text to lower case text
2. Reduction of →
    a. Repetitive Characters, Punctuations
3. Expansion of contracted words
4. Spelling Correction using Autocorrection Python module

Feature Analysis:

In this step, I will be employing the relevant feature extraction techniques I have explained in the Relevant Techniques section of this chapter, to select the relevant data types and features to be trained and classified as part of model implementation for dependency evaluation.

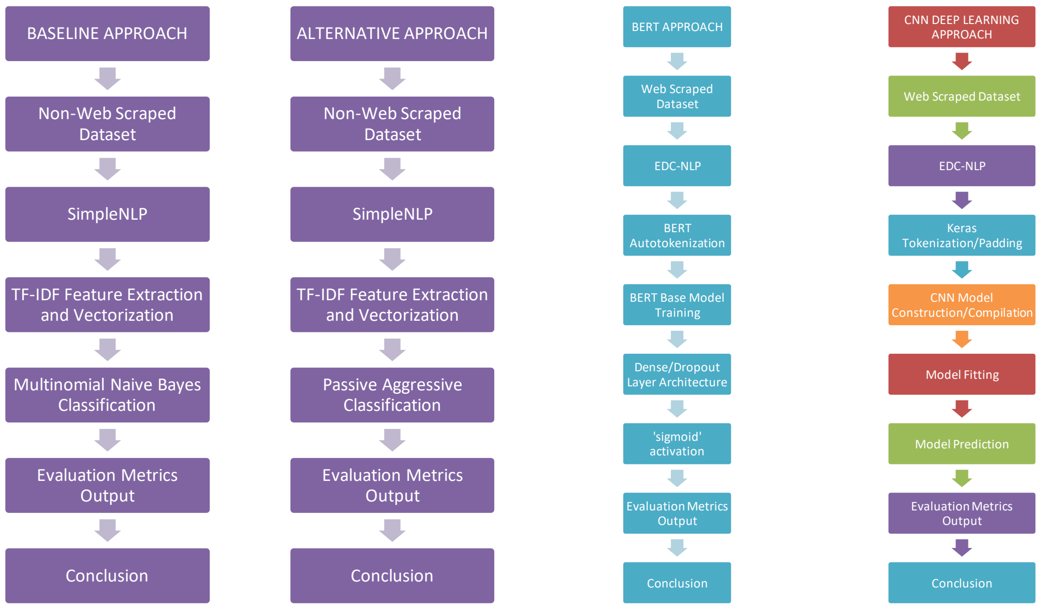
My chosen feature extraction techniques comprise of:

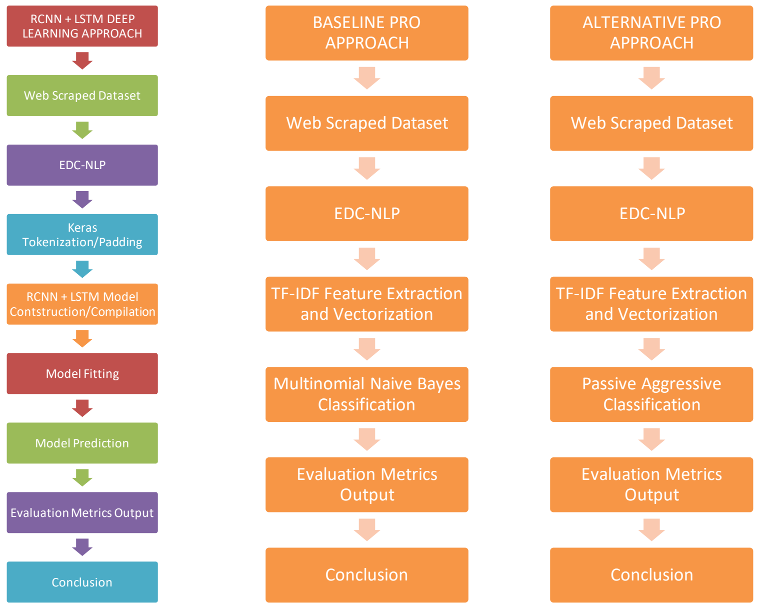
1. TF-IDF Vectorization
2. Count Vectorization
3. Linguistical Analysis using NLP

3.5 Fake News Detection Workflow Design

The remainder of the project methodology will be explained as the multiple constructive machine learning / deep neural network architectures I have proposed to implement.

There are going to be 7 main text classification workflows in this project, aimed at optimizing the effectiveness of using NLP and machine learning at detecting fake news or information.


BASELINE APPROACH
Non-Web Scraped Dataset
SimpleNLP
TF-IDF Feature Extraction and Vectorization
Multinomial Naive Bayes Classification
Evaluation Metrics Output
Conclusion
ALTERNATIVE APPROACH
Non-Web Scraped Dataset
SimpleNLP
TF-IDF Feature Extraction and Vectorization
Passive Aggressive Classification
Evaluation Metrics Output
Conclusion
BERT APPROACH
Web Scraped Dataset
EDC-NLP
BERT Autotokenization
BERT Base Model Training
Dense/Dropout Layer Architecture
'sigmoid' activation
Evaluation Metrics Output
Conclusion
CNN DEEP LEARNING APPROACH
Web Scraped Dataset
EDC-NLP
Keras Tokenization/Padding
CNN Model Construction/Compilation
Model Fitting
Model Prediction
Evaluation Metrics Output
Conclusion


RCNN + LSTM DEEP LEARNING APPROACH
Web Scraped Dataset
EDC-NLP
Keras Tokenization/Padding
RCNN + LSTM Model Contstruction/Compilation
Model Fitting
Model Prediction
Evaluation Metrics Output
Conclusion
BASELINE PRO APPROACH
Web Scraped Dataset
EDC-NLP
TF-IDF Feature Extraction and Vectorization
Multinomial Naive Bayes Classification
Evaluation Metrics Output
Conclusion
ALTERNATIVE PRO APPROACH
Web Scraped Dataset
EDC-NLP
TF-IDF Feature Extraction and Vectorization
Passive Aggressive Classification
Evaluation Metrics Output
Conclusion

The fundamental workflow for each of the proposed model approach is as follows:

1. Importation of relevant libraries
2. Dataset Pre-Processing
3. Feature Extraction
4. Input Generation
5. Model Construction and Architecture Analysis
6. Model Prediction
7. Model Performance Evaluation

3.5.1 Baseline Model Approach: SimpleNLP + TF-IDF + Multinomial Naïve Bayes Classifier

Term Frequency refers to the frequency of a certain word which appears in a document, or text body (Sammut C. 2011). Inverse Document Frequency refers to the frequency of the occurrence of a certain word amongst an entire collection of documents, or text corpus (Sammut C. 2011).

Using this feature extraction technique, I will be training and testing the cleansed data on the Naïve Bayes classifier as part of my baseline implementation. The predictions will be evaluated according to Model Performance Evaluation metrics.

3.5.2 Alternative Approach: SimpleNLP + TF-IDF + Passive Aggressive Classifier

This classifier ultimately works as a family of algorithms based on the Perceptron model (T. Zhai. 2022), and is usually used in social media companies such as Twitter (T. Zhai.

2022), where there is information and data being constantly in fluxed into the framework and an algorithm which is able to handle the huge data inflow is required.

The Passive Aggressive classifier foundationally works such that if the prediction of the classifier is correct, the machine learning model remains unchanged, and if the prediction is otherwise, then the classifier alters the model to achieve a more accurate prediction.

3.5.3 BERT Classification Approach (Using EDC-NLP Pipeline)

The BERT (Bi-Directional Encoder Representation from Transformers) approach is a slightly more unique approach. This partially supervised learning model can comprehend and analyze the context and relationships between words in a text string, or sentence structures amongst a huge text corpus.

I propose to use the BERT (Base) model, which has 12 Encoding layers in its stack. Using the BERT (Base), we will automatically tokenize the text corpus of the news headlines being input and pass the input through the encoder layers to perform the classification of the fake/real news data.

3.6 Deep Learning Approach using Neural Networks

Deep neural networks are networks with a complex architecture, where obscure layers perform extremely complicated tasks. Using deep neural networks in the field of fake news detection has proven to be a heavily researched and experimental task.

For this project, I propose to implement 2 familiar, different types of neural networks, which are expanded on as shown below.

3.6.1 Deep Neural Network Approach – EDC-NLP + CNN Deep Learning Model

For this approach, we are going to adopt a convolutional neural network (CNN) approach on the dataset which has been cleaned using the EDC-NLP process.

Even though, convolutional neural networks are usually used for image and video recognition, my decision to use the CNN approach to detect fake news would be as such:

1. After the word embedding process, CNN networks are optimal in extracting important features and conducting supervised learning on those vector data inputs
2. CNNs are able to provide a dense network along with the pooling layer architecture hence increasing the efficiency of feature recognition and hence prediction amongst large datasets.

There is going to be a sequential compilation of spatial dropouts, convolutional 1-Dimensional layers and Dense layers which will employ the use of a binary cross entropy loss function as it is a binary classification dataset and focus on the accuracy metrics to isolate the effectiveness of the module in detecting fake news.

3.6.2 Deep Neural Network Approach – EDC-NLP + RCNN + LSTM Deep Learning Model

Using deep learning neural networks such as these enable us to construct varied model architectures to optimize the text classification workflow process in project designs.

For the second deep learning approach, I propose to implement a Recurrent Convolutional Neural Network (RCNN) collaborated with a Long Short Term Memory (LSTM) network.

My decision to choose the above 2 configurations to build and compile a Deep Learning network would be as such:

1. RCNN models are adept at learning contextual information, which is important in our case of distinguishing the text contained within several news article data points
2. LSTM networks are effective with modelling sequential data points and hence good at handling long scope dependencies

However, where we are going to develop on these networks are the variations of the layers and layer implementation structures across the network model architecture, with the goal of achieving the best possible accuracy in detecting fake news from a dataset.

3.7 Baseline PRO Model Approach: EDC-NLP + TF-IDF + Multinomial Naïve Bayes Classifier

This is a supplemental approach to the initial Baseline approach I have included in my project design at 3.5.1.

The difference between this approach and the Baseline approach is that this approach uses the EDC-NLP pipeline to pre-process its data, and the dataset is derived from the web scraped Fake-Real news dataset.

The rationale behind this approach is to be able to identify the relationship between using a web-scraped, comprehensive and well formatted, data cleaned dataset on the accuracy of Fake News Detection as compared to using a less competitive dataset.

3.8 Alternative PRO Approach: EDC-NLP + TF-IDF + Passive Aggressive Classifier

This is a supplemental approach to the initial Alternative approach I have included in my project design at 3.5.2.

The difference between this approach and the Alternative approach is that this approach uses the EDC-NLP pipeline to pre-process its data, and the dataset is derived from the web scraped Fake-Real news dataset.

The rationale behind this approach is to be able to identify the relationship between using a web-scraped, comprehensive and well formatted, data cleaned dataset on the accuracy of Fake News Detection as compared to using a less competitive dataset.

3.9 Fake News Detection Chrome Extension

Due to project time constraints owing to the training of the Deep Learning models for fake news detection, I decided to not go ahead with the Chrome Extension Deployment, as initially reported. This decision was made by me so that I was able to allocate sufficient headroom and duration for the completion of the project proper first.

This aspect of the project is beyond the scope, which I have undertaken as a personal interest to implement a robust, deployable software module for fake news detection.

I propose to implement a Chrome Extension, which will be deployed using Python and Flask.

3.9.1 Design Notes: Future of the Prototype in Preliminary Stage

The 3-month outlook for this prototype requires me to take a series of incremental and model corrective steps to develop an improved, highly accurate fake news detection model.

As mentioned in the Project Design section, the plan is to develop a BERT classification model and Deep Learning model and to effectively tune and refine the classification confidence of the models to achieve a higher detection accuracy, so that we are aligned with the aims of this project. The subsequent models are currently a work in progress, and further details, methodologies and the outline of the implemented models will be included in the Final Report submission of this project.

The high-level scope below shows some of the important milestones that is to be taken for the further propagation of this prototype:

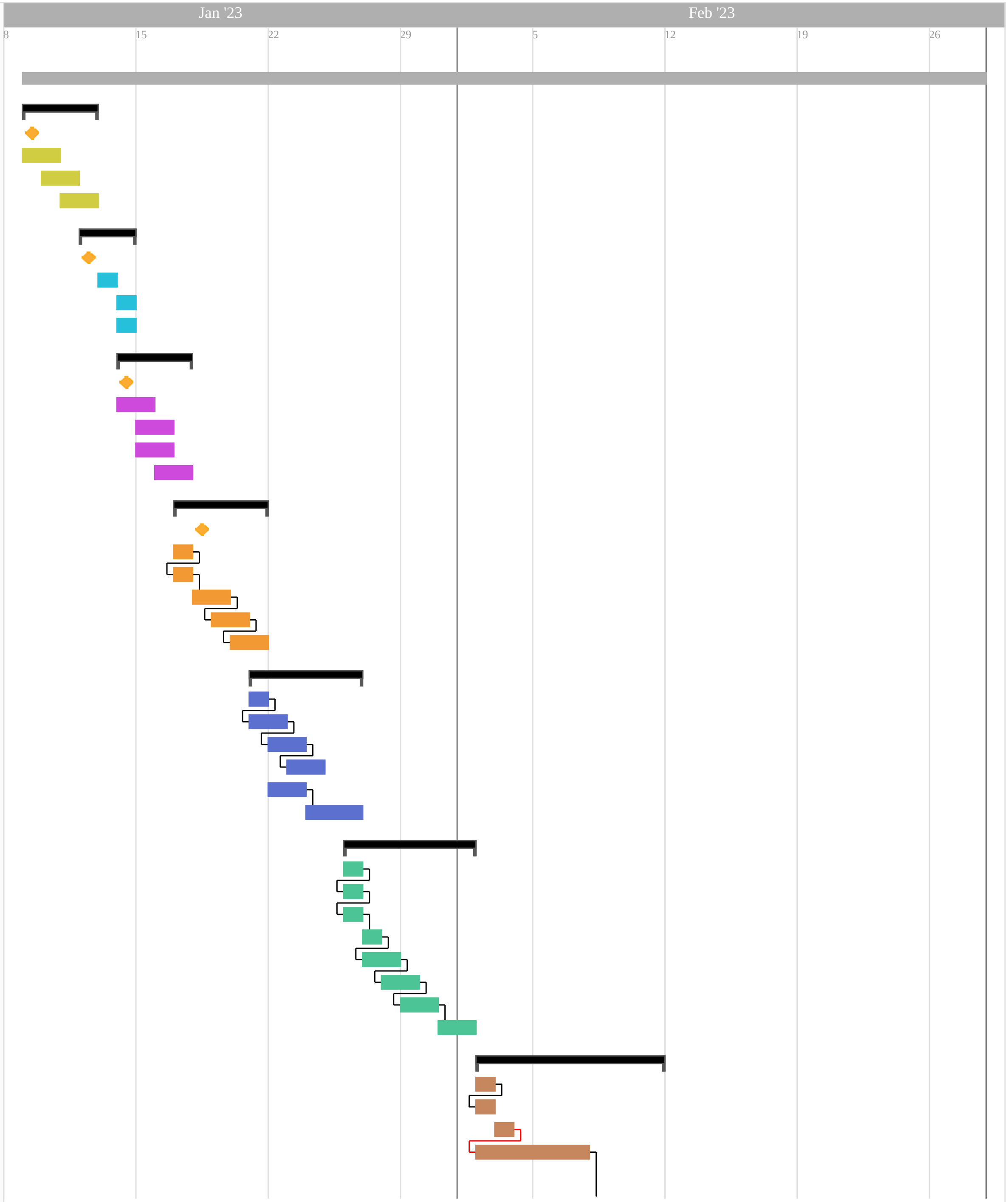
1. Improving text classification models
    a. Hyperparameter Tuning of models – model performance refinement
    b. Models & Output Coalescing - improve performance based on combining multiple models and their outputs
    c. Improved dataset cleaning pipelines and text data cleansing
2. Packaging improved model structures into a more comprehensive software module for fake news detection

3. Beyond the scope: Implementation of a chrome extension module based on the most accurate fake news detection model I can implement *(not included in Final Submission)*

Gantt Chart

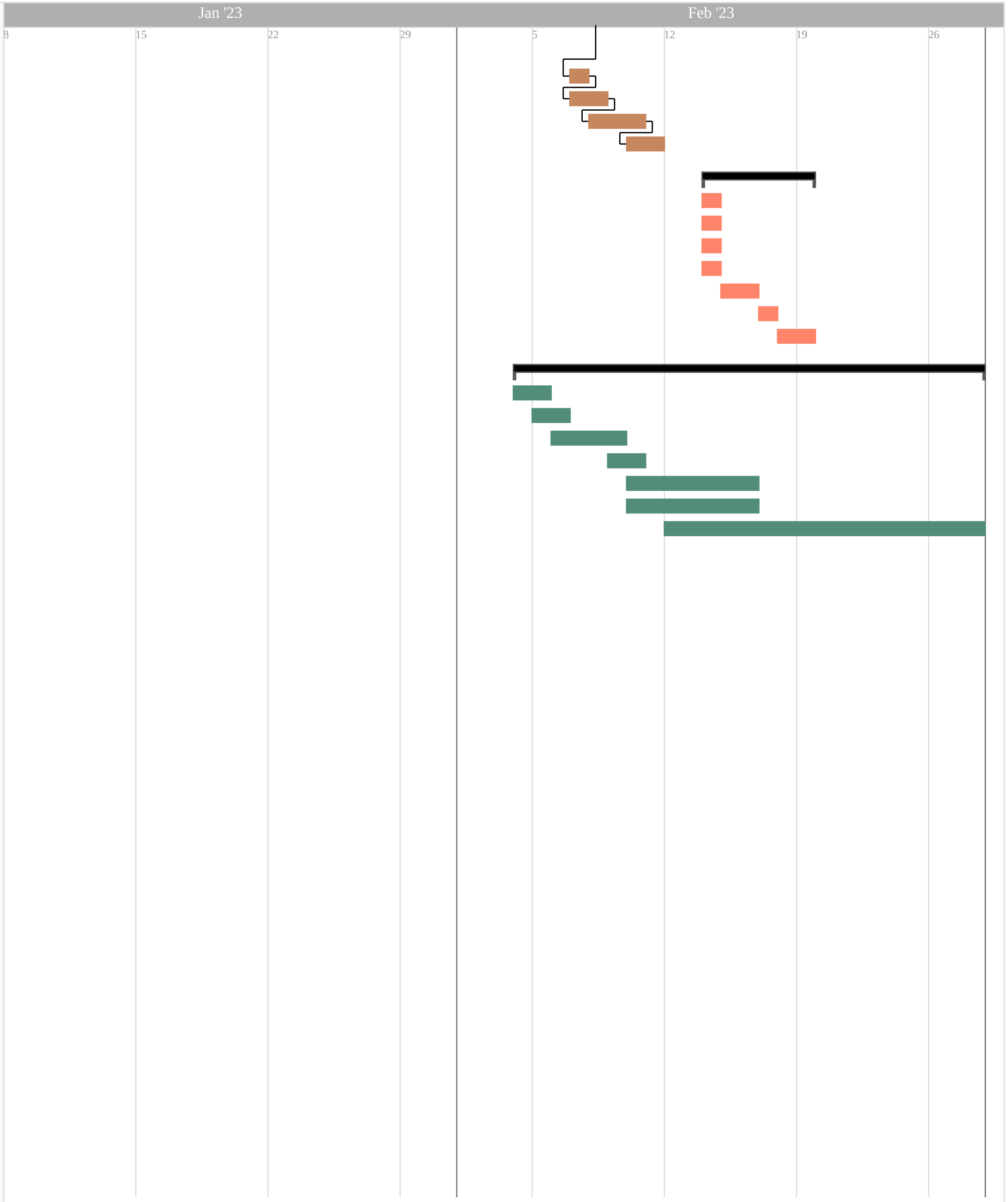
The Gantt Chart below shows the plan of action I have taken to lead the Final Year project to completion by executing the project design and plan I have laid out and polishing up and rolling out the relevant deliverables by the project deadline.

Final Year Project - NLP Fake News...	start	end	0h	100%
Dataset Analysis	**09/01/23**	**12/01/23**	**0h**	**100%**
Acquire dataset from Kaggle	09/01	09/01	0	100%
Evaluate dataset characteristics	09/01	10/01	0	100%
Evaluate ethics of dataset acquisiton	10/01	11/01	0	100%
Prepare dataset for pre-processing phase	11/01	12/01	0	100%
Dataset Pre-Processing	**12/01/23**	**14/01/23**	**0h**	**100%**
Text Data Cleaning	12/01	12/01	0	100%
Unnecessary semantics and linguistical feature ...	13/01	13/01	0	100%
Spelling Correction	14/01	14/01	0	100%
Lemmatization	14/01	14/01	0	100%
Exploratory Data Analysis	**14/01/23**	**17/01/23**	**0h**	**100%**
Label Classification Analysis	14/01	14/01	0	100%
Conversion from Multi to Binary Classification	14/01	15/01	0	100%
Label Distribution Analysis	15/01	16/01	0	100%
Wordcloud Analysis	15/01	16/01	0	100%
n-gram Analysis	16/01	17/01	0	100%
Baseline Model (TF-IDF + NB Classifier)	**17/01/23**	**21/01/23**	**0h**	**100%**
Library Importation	18/01	18/01	0	100%
Dataset Preparation for Model Approach	17/01	17/01	0	100%
Feature Extraction & Input Generation	17/01	17/01	0	100%
Model Architecture Construction	18/01	19/01	0	100%
Model Prediction	19/01	20/01	0	100%
Model Performance Evaluation	20/01	21/01	0	100%
Alternative Model (TF-IDF + Passive Aggressi...	**21/01/23**	**26/01/23**	**0h**	**100%**
Library Importation	21/01	21/01	0	100%
Dataset Preparation for Model Approach	21/01	22/01	0	100%
Feature Extraction & Input Generation	22/01	23/01	0	100%
Model Architecture Construction	23/01	24/01	0	100%
Model Prediction	22/01	23/01	0	100%
Model Performance Evaluation	24/01	26/01	0	100%
BERT Classification Approach	**26/01/23**	**01/02/23**	**0h**	**100%**
Library Importation	26/01	26/01	0	100%
Dataset Preparation for Model Approach	26/01	26/01	0	100%
Feature Extraction & Input Generation	26/01	26/01	0	100%
Model Architecture Construction	27/01	27/01	0	100%
Model Prediction	27/01	28/01	0	100%
Model Performance Evaluation	28/01	29/01	0	100%
Hyperparameter Tuning	29/01	30/01	0	100%
Re-Evaluation	31/01	01/02	0	100%
Deep Learning Models (CNN + Variational R...	**02/02/23**	**11/02/23**	**0h**	**100%**
Library Importation	02/02	02/02	0	100%
Dataset Preparation for Model Approach	02/02	02/02	0	100%
Feature Extraction & Input Generation	03/02	03/02	0	100%
Model Architecture Construction	02/02	07/02	0	100%



teamgantt
Created with Free Edition

Model Prediction	07/02	07/02	0	100%
Model Performance Evaluation	07/02	08/02	0	100%
Hyperparameter Tuning	08/02	10/02	0	100%
Re-Evaluation	10/02	11/02	0	100%
Model Performance Evaluation	**14/02/23**	**19/02/23**	**0h**	**100%**
Plot Confusion Matrices	14/02	14/02	0	100%
Calculate Accuracy Scores	14/02	14/02	0	100%
Calculate Recall Scores	14/02	14/02	0	100%
Calculate Precision & F1-Scores	14/02	14/02	0	100%
Tabulate Results	15/02	16/02	0	100%
Compare Model Performances	17/02	17/02	0	100%
Evaluate model with higest accuracy in Fake N...	18/02	19/02	0	100%
Final Report (Deliverable)	**04/02/23**	**28/02/23**	**0h**	**100%**
Project Abstract	04/02	05/02	0	100%
Introduction	05/02	06/02	0	100%
Collate related work and Literature Reviews	06/02	09/02	0	100%
Research Notes	09/02	10/02	0	100%
Project Design	10/02	16/02	0	100%
Project goals and tasks accomplished	10/02	16/02	0	100%
Fake News Detection Model Designs	12/02	28/02	0	100%



CHAPTER 4: IMPLEMENTATION

Before we dive into the project implementation I have developed as part of the Misinformation Mitigation project, let us see the outline of how I have approached my build.

I also want to note here that I have broken down my Jupyter Notebook into Chapters for readability.

I have implemented 2 NLP workflows and 7 Fake News Detection Classification models, whilst also including an alternative approaches which provides a different perspective to the fake news detection problem I am trying to address with this prototype.

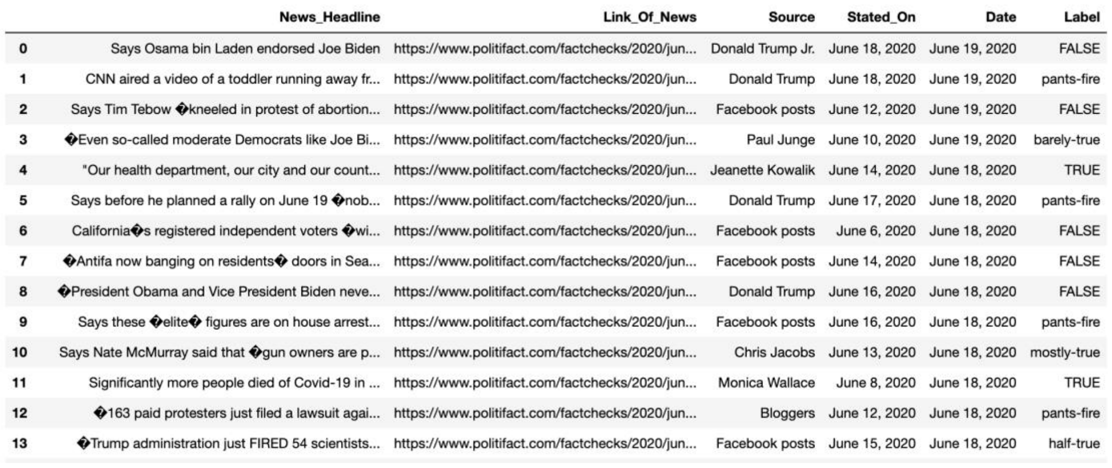
Note that I intentionally did not use the Bag of Words model for my text feature analysis and selection, as it is a heavily over used and primitive approach, which does not help to address the scope of work I am trying to develop and improve on.

4.1 Dataset Preparation

Dataset Acquisition:

I started off with the dataset acquisition process from Kaggle as can be seen in Appendix A.1 / Jupyter Notebook Chapter 2.

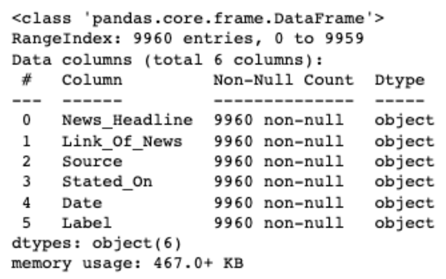
Acquired Dataset (Raw):



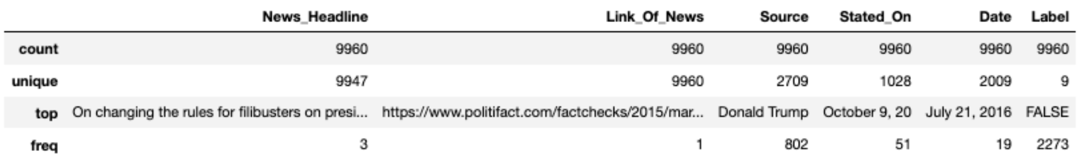
	News_Headline	Link_Of_News	Source	Stated_On	Date	Label
0	Says Osama bin Laden endorsed Joe Biden	https://www.politifact.com/factchecks/2020/jun...	Donald Trump Jr.	June 18, 2020	June 19, 2020	FALSE
1	CNN aired a video of a toddler running away fr...	https://www.politifact.com/factchecks/2020/jun...	Donald Trump	June 18, 2020	June 19, 2020	pants-fire
2	Says Tim Tebow �kneeled in protest of abortion...	https://www.politifact.com/factchecks/2020/jun...	Facebook posts	June 12, 2020	June 19, 2020	FALSE
3	�Even so-called moderate Democrats like Joe Bi...	https://www.politifact.com/factchecks/2020/jun...	Paul Junge	June 10, 2020	June 19, 2020	barely-true
4	"Our health department, our city and our count...	https://www.politifact.com/factchecks/2020/jun...	Jeanette Kowalik	June 14, 2020	June 18, 2020	TRUE
5	Says before he planned a rally on June 19 �nob...	https://www.politifact.com/factchecks/2020/jun...	Donald Trump	June 17, 2020	June 18, 2020	pants-fire
6	California�s registered independent voters �wi...	https://www.politifact.com/factchecks/2020/jun...	Facebook posts	June 6, 2020	June 18, 2020	FALSE
7	�Antifa now banging on residents� doors in Sea...	https://www.politifact.com/factchecks/2020/jun...	Facebook posts	June 14, 2020	June 18, 2020	FALSE
8	�President Obama and Vice President Biden neve...	https://www.politifact.com/factchecks/2020/jun...	Donald Trump	June 16, 2020	June 18, 2020	FALSE
9	Says these �elite� figures are on house arrest...	https://www.politifact.com/factchecks/2020/jun...	Facebook posts	June 16, 2020	June 18, 2020	pants-fire
10	Says Nate McMurray said that �gun owners are p...	https://www.politifact.com/factchecks/2020/jun...	Chris Jacobs	June 13, 2020	June 18, 2020	mostly-true
11	Significantly more people died of Covid-19 in ...	https://www.politifact.com/factchecks/2020/jun...	Monica Wallace	June 8, 2020	June 18, 2020	TRUE
12	�163 paid protesters just filed a lawsuit agai...	https://www.politifact.com/factchecks/2020/jun...	Bloggers	June 12, 2020	June 18, 2020	pants-fire
13	�Trump administration just FIRED 54 scientists...	https://www.politifact.com/factchecks/2020/jun...	Facebook posts	June 15, 2020	June 18, 2020	half-true

Next, I ensured to understand the characteristics and information present in the dataset, before I commence on the Exploratory Data Analysis (EDA) and pre-processing steps.

The code for these can be found in Appendix A.2.



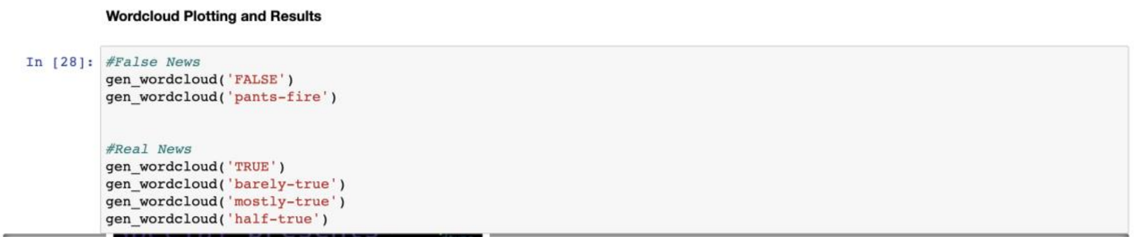
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9960 entries, 0 to 9959
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   News_Headline  9960 non-null   object
 1   Link_Of_News   9960 non-null   object
 2   Source         9960 non-null   object
 3   Stated_On      9960 non-null   object
 4   Date           9960 non-null   object
 5   Label          9960 non-null   object
dtypes: object(6)
memory usage: 467.0+ KB
```



	News_Headline	Link_Of_News	Source	Stated_On	Date	Label
count	9960	9960	9960	9960	9960	9960
unique	9947	9960	2709	1028	2009	9
top	On changing the rules for filibusters on presi...	https://www.politifact.com/factchecks/2015/mar...	Donald Trump	October 9, 20	July 21, 2016	FALSE
freq	3	1	802	51	19	2273

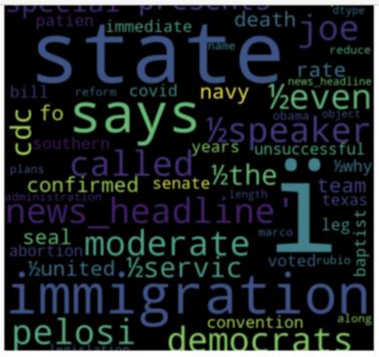
I then proceeded to analyze the characteristics of the Fake-Real news dataset by using Exploratory Data Analysis methods, which include:

Word Clouds:



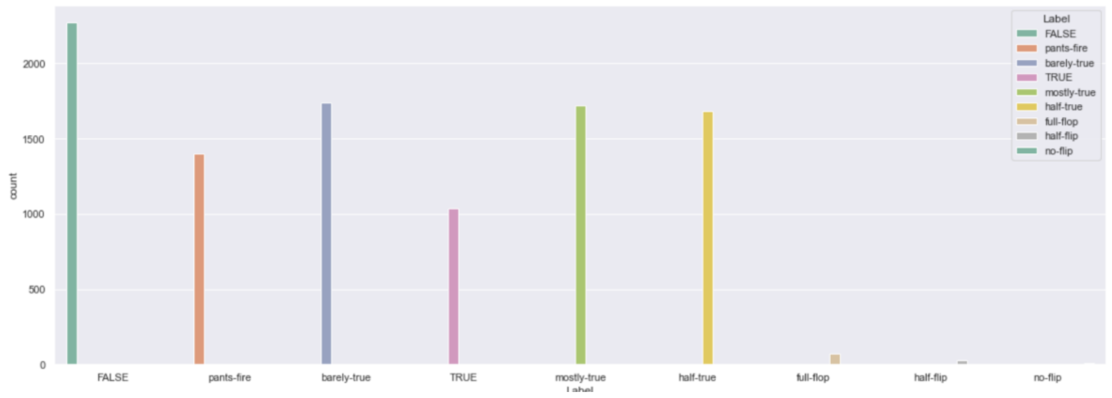
Wordcloud Plotting and Results

```
In [28]: #False News
         gen_wordcloud('FALSE')
         gen_wordcloud('pants-fire')


         #Real News
         gen_wordcloud('TRUE')
         gen_wordcloud('barely-true')
         gen_wordcloud('mostly-true')
         gen_wordcloud('half-true')
```

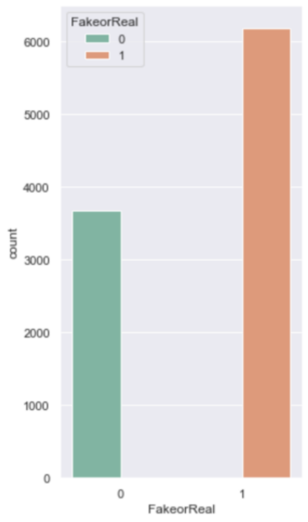


Label Distributions:

Multi-Class Labelled Format:



Binary Labelled Format (Post Conversion):



Since the dataset is of a multi-labelled classification nature, with 6 main labels to describe the legitimacy of the news articles, I have decided to convert the dataset into a binary classified dataset with 0 to represent fake news and 1 to represent real news. This also simplifies the machine learning classification process, as we already have 10K+ data points to train and test through. The code for the above can be found in Appendix A.3 / Jupyter Notebook Chapter 4.1.

4.2 Dataset Split, Web Scraping – Splitting the dataset for variational usage

The next step in the implementation, would be splitting and distinguishing the raw dataset into 2 separate datasets as can be understood from Figure 4.2.1.

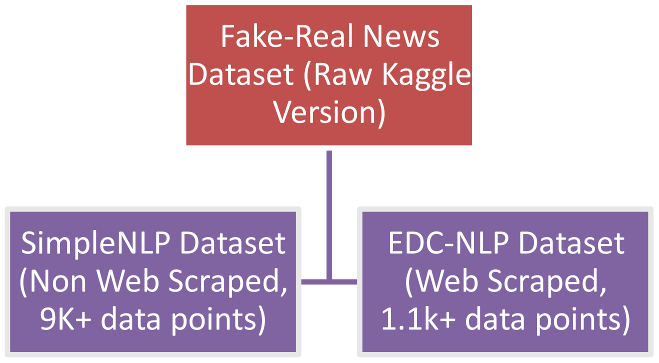
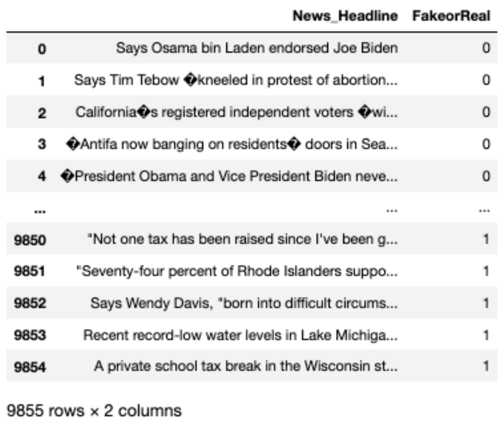

Fake-Real News Dataset (Raw Kaggle Version)
SimpleNLP Dataset (Non Web Scraped, 9K+ data points)
EDC-NLP Dataset (Web Scraped, 1.1k+ data points)

Figure 4.2.1: Dataset Split Plan

Simple NLP Dataset:



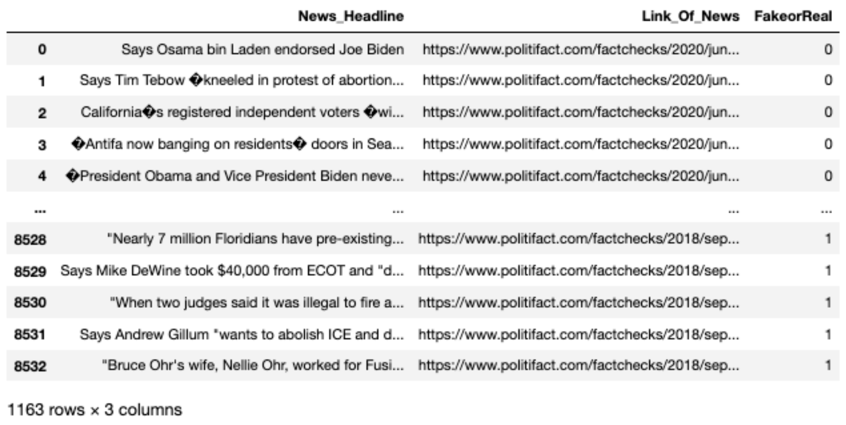
	News_Headline	FakeorReal
0	Says Osama bin Laden endorsed Joe Biden	0
1	Says Tim Tebow �kneeled in protest of abortion...	0
2	California�s registered independent voters �wi...	0
3	�Antifa now banging on residents� doors in Sea...	0
4	�President Obama and Vice President Biden neve...	0
...	...	...
9850	"Not one tax has been raised since I've been g...	1
9851	"Seventy-four percent of Rhode Islanders suppo...	1
9852	Says Wendy Davis, "born into difficult circums...	1
9853	Recent record-low water levels in Lake Michiga...	1
9854	A private school tax break in the Wisconsin st...	1

9855 rows × 2 columns

As we can see from the above dataframe, this dataset consists of just the news headline and binary information of the news headline, with 9855 data points.

EDC-NLP Dataset:

The screenshot below shows the dataframe before the web-scraping process, as I have preserved the ‘Link_Of_News’ column of the original dataset.

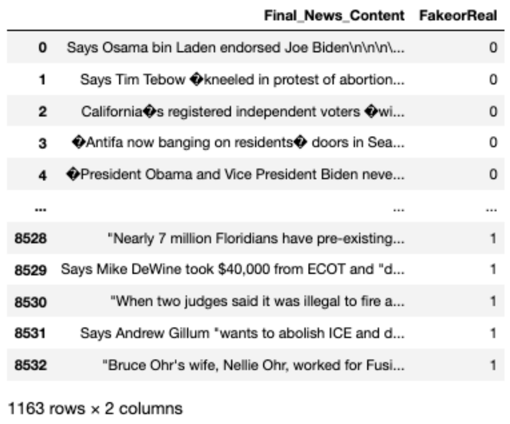


	News_Headline	Link_Of_News	FakeorReal
0	Says Osama bin Laden endorsed Joe Biden	https://www.politifact.com/factchecks/2020/jun...	0
1	Says Tim Tebow �kneeled in protest of abortion...	https://www.politifact.com/factchecks/2020/jun...	0
2	California�s registered independent voters �wi...	https://www.politifact.com/factchecks/2020/jun...	0
3	�Antifa now banging on residents� doors in Sea...	https://www.politifact.com/factchecks/2020/jun...	0
4	�President Obama and Vice President Biden neve...	https://www.politifact.com/factchecks/2020/jun...	0
...	...	...	...
8528	"Nearly 7 million Floridians have pre-existing...	https://www.politifact.com/factchecks/2018/sep...	1
8529	Says Mike DeWine took $40,000 from ECOT and "d...	https://www.politifact.com/factchecks/2018/sep...	1
8530	"When two judges said it was illegal to fire a...	https://www.politifact.com/factchecks/2018/sep...	1
8531	Says Andrew Gillum "wants to abolish ICE and d...	https://www.politifact.com/factchecks/2018/sep...	1
8532	"Bruce Ohr's wife, Nellie Ohr, worked for Fusi...	https://www.politifact.com/factchecks/2018/sep...	1

1163 rows × 3 columns

Note that I have also, truncated the dataset to about 1.1K+ data points. This was an executive decision made by me such that we would be able to train the dataset on complex BERT and Deep Learning models without occupying too much of the project duration as we have to also maintain our progress with the Gantt Chart Project Timeline.

The screenshot below shows the dataframe after the web-scraping process has been executed.



	Final_News_Content	FakeorReal
0	Says Osama bin Laden endorsed Joe Biden\n\n\n\...	0
1	Says Tim Tebow �kneeled in protest of abortion...	0
2	California�s registered independent voters �wi...	0
3	�Antifa now banging on residents� doors in Sea...	0
4	�President Obama and Vice President Biden neve...	0
...	...	...
8528	"Nearly 7 million Floridians have pre-existing...	1
8529	Says Mike DeWine took $40,000 from ECOT and "d...	1
8530	"When two judges said it was illegal to fire a...	1
8531	Says Andrew Gillum "wants to abolish ICE and d...	1
8532	"Bruce Ohr's wife, Nellie Ohr, worked for Fusi...	1

1163 rows × 2 columns

As can be observed, I have taken the scraped HTML text content (the news article body) of each news article, and concatenated the columns to make a 'Final_News_Content' column. This column contains information on the news headline + news body content, hence providing us with richer textual data for classification uses. The code for the web-scrape process can be found in Appendix A.4 / Jupyter Notebook Chapter 4.1.1.

4.3 SimpleNLP, Lemmatization, n-gram analysis

SimpleNLP:

Now for the pre-processing proper. I utilized simple regex expressions, case conversion functions, lemmatization and stopword removal libraries to process the data. The code for this can be found in Appendix A.5 / Jupyter Notebook Chapter 4.3.

Lemmatization:

Next, I performed lemmatization to cleanse the news headline data present in the dataset. Lemmatization is usually referred to as the process of converting an evolved word into its root word with the consideration of the context of the word in a sentence or text corpus, and the characteristics of the text corpus.

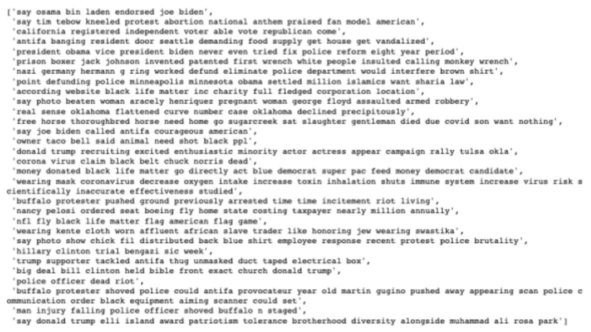
The lemmatization process I adopted is as follows:

1. Converting all the news headline strings to lowercase

2. Substituting all special characters and punctuation found in the news headline string values

3. Extracting all the meaningful words in the news headline string corpus by ignoring the stop-words present in the news title

This allows the linguistical integrity of the news headline to not be compromised. The screenshot below shows some sample results of the lemmatization process.

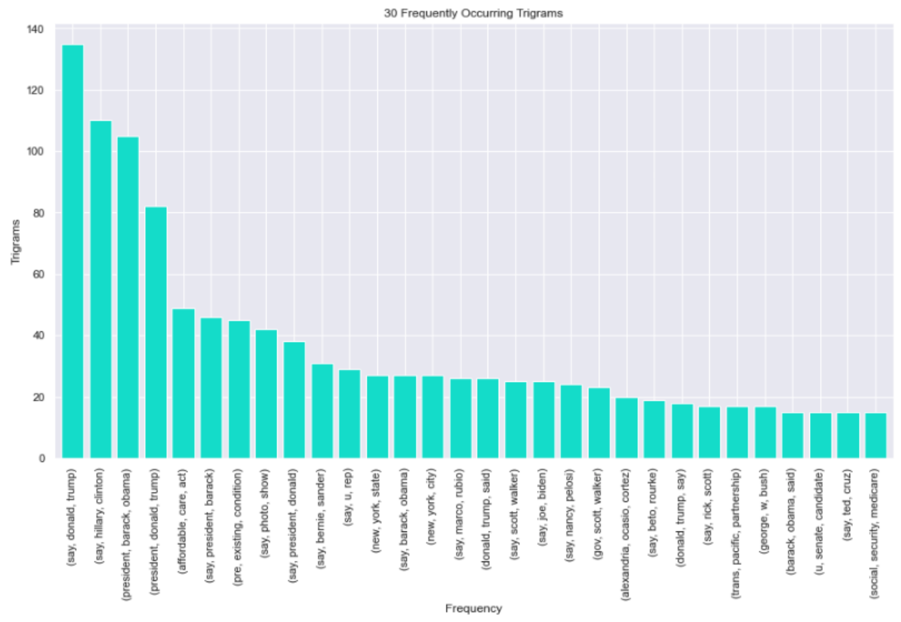


```
['say osama bin laden endorsed joe biden',
 'say tim tebow kneeled protest abortion national anthem praised fan model american',
 'california registered independent voter able vote republican come',
 'antifa banging resident door seattle demanding food supply get house get vandalized',
 'president obama vice president biden never even tried fix police reform eight year period',
 'prison boxer jack johnson invented patented first wrench white people insulted calling monkey wrench',
 'nazi germany hermann g ring worked defund eliminate police department would interfere brown shirt',
 'point defunding police minneapolis minnesota obama settled million islamics want sharia law',
 'according website black life matter inc charity full fledged corporation location',
 'say photo beaten woman aracely henriquez pregnant woman george floyd assaulted armed robbery',
 'real sense oklahoma flattened curve number case oklahoma declined precipitously',
 'free horse thoroughbred horse need home go sugarcreek sat slaughter gentleman died due covid son want nothing',
 'say joe biden called antifa courageous american',
 'owner taco bell said animal need shot black ppl',
 'donald trump recruiting excited enthusiastic minority actor actress appear campaign rally tulsa okla',
 'corona virus claim black belt chuck norris dead',
 'money donated black life matter go directly act blue democrat super pac feed money democrat candidate',
 'wearing mask coronavirus decrease oxygen intake increase toxin inhalation shuts immune system increase virus risk s
cientifically inaccurate effectiveness studied',
 'buffalo protester pushed ground previously arrested time time incitement riot living',
 'nancy pelosi ordered seat boeing fly home state costing taxpayer nearly million annually',
 'nfl fly black life matter flag american flag game',
 'wearing kente cloth worn affluent african slave trader like honoring jew wearing swastika',
 'say photo show chick fil distributed back blue shirt employee response recent protest police brutality',
 'hillary clinton trial bengazi sic week',
 'trump supporter tackled antifa thug unmasked duct taped electrical box',
 'big deal bill clinton held bible front exact church donald trump',
 'police officer dead riot',
 'buffalo protester shoved police could antifa provocateur year old martin gugino pushed away appearing scan police c
ommunication order black equipment aiming scanner could set',
 'man injury falling police officer shoved buffalo n staged',
 'say donald trump elli island award patriotism tolerance brotherhood diversity alongside muhammad ali rosa park']
```

n-gram Analysis:

I also conducted n-gram analysis of the resultant lemmatized text corpus to gain a better understanding of the common bigrams, trigrams and 4-grams which are present in the news headline that is being textually classified.

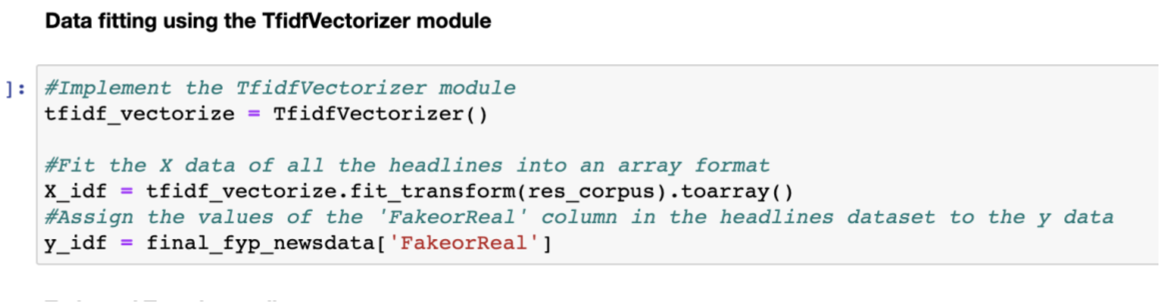
The screenshot below shows some sample n-gram analysis results.



4.4 SimpleNLP Workflows + Classification Approaches

4.4.1 Baseline Performance Approach: SimpleNLP + Tf-Idf + Multinomial Naïve Bayes Classification Model

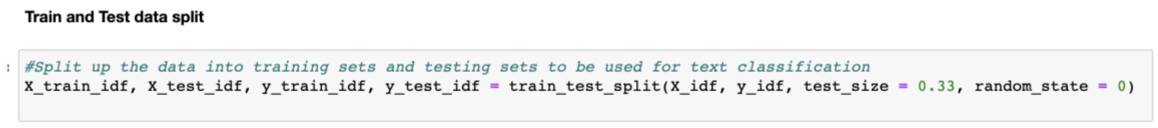
I have implemented the Term Frequency – Inverse Document Frequency (TF-IDF) measure approach to retrieve the textual information present in the news headlines found in the dataset. This approach aids in providing a quantified representation of linguistical information found in the text corpus. Using the TfidfVectorizer module, I was able to convert the textual data into a vector representation to retrieve the information presented in the dataset as a classifiable entity. The full code for this can be found in Jupyter Notebook Chapter 6.



Data fitting using the TfidfVectorizer module

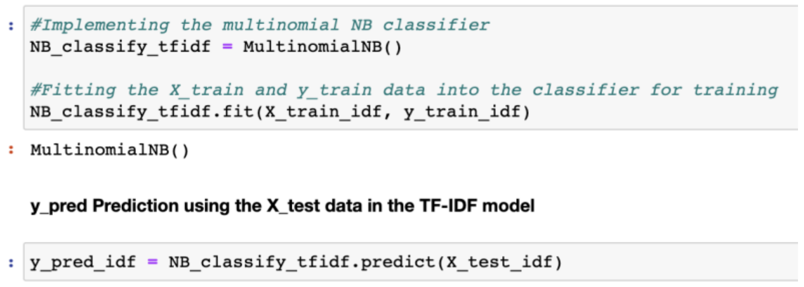
```
]: #Implement the TfidfVectorizer module
   tfidf_vectorize = TfidfVectorizer()

   #Fit the X data of all the headlines into an array format
   X_idf = tfidf_vectorize.fit_transform(res_corpus).toarray()
   #Assign the values of the 'FakeorReal' column in the headlines dataset to the y data
   y_idf = final_fyp_newsdata['FakeorReal']
```

The standardized train-test data split was done to prepare the x, y training data and x, y testing data.


Train and Test data split

```
#Split up the data into training sets and testing sets to be used for text classification
X_train_idf, X_test_idf, y_train_idf, y_test_idf = train_test_split(X_idf, y_idf, test_size = 0.33, random_state = 0)
```

The Multinomial Naïve Bayes (NB) classifier was used in this text classification pipeline to evaluate the baseline performance of the model I implemented.


```
#Implementing the multinomial NB classifier
NB_classify_tfidf = MultinomialNB()

#Fitting the X_train and y_train data into the classifier for training
NB_classify_tfidf.fit(X_train_idf, y_train_idf)
```

```
MultinomialNB()
```

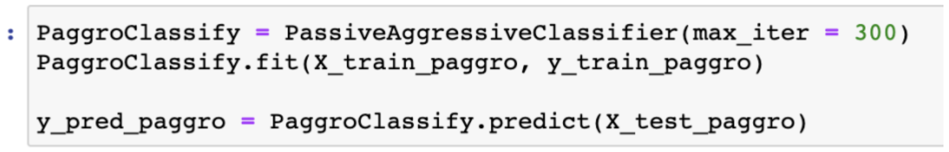
y_pred Prediction using the X_test data in the TF-IDF model

```
y_pred_idf = NB_classify_tfidf.predict(X_test_idf)
```

The evaluation results for the above baseline performance will be discussed in Chapter 5: Evaluation.

4.4.2 Alternative Performance Approach: SimpleNLP + Tf-Idf + Passive Aggressive Classification Model

An alternative approach I chose to experiment with was using the TF-IDF vectorized data to input into the Passive Aggressive Classifier found in the sklearn library. The full code for this can be found in Jupyter Notebook Chapter 7.


```
PaggroClassify = PassiveAggressiveClassifier(max_iter = 300)
PaggroClassify.fit(X_train_paggro, y_train_paggro)

y_pred_paggro = PaggroClassify.predict(X_test_paggro)
```

The evaluation results for the above alternative model performance will be discussed in Chapter 5: Evaluation.

4.5 Extensive Data Cleanse NLP (EDC-NLP)

At this stage, we can now commence on a more sophisticated and comprehensive data cleaning pipeline which will be able to format the textual information found in the web-scraped dataset for fake news detection uses.

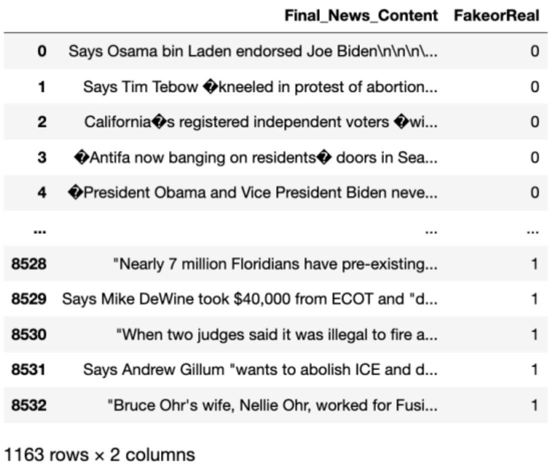
The full code for the EDC-NLP pipeline can be found in the Jupyter Notebook Chapter 8.

I have implemented the following functions as shown in Appendix A.6 to effectively clean the text corpus of common linguistic hindrances which may cause errors and pose as inaccurate contextual indicators on the news headlines being classified. These functions will be collectively implemented into 2 driver functions representing an improved NLP pipeline which I have dubbed the EDC-NLP pipeline as part of this project.

The EDC-NLP pipeline handles the following text cleaning, data pre-processing operations:

1. Removing newlines and tabs
2. Removing HTML tags
3. Removing links
4. Removing accented characters
5. Removing extra whitespaces
6. Removing special characters
7. Converting the text to lowercase

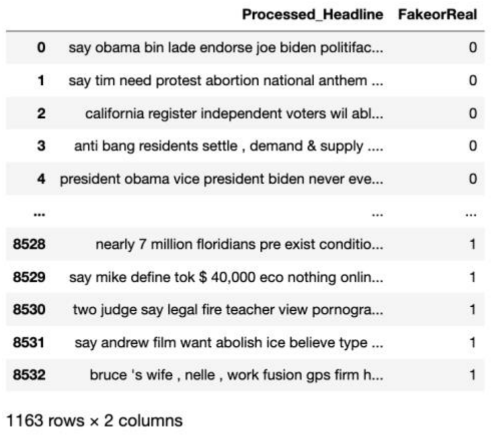
This is the EDC-NLP, web-scraped dataset before the EDC-NLP pre-processing is executed.



	Final_News_Content	FakeorReal
0	Says Osama bin Laden endorsed Joe Biden\n\n\n\...	0
1	Says Tim Tebow �kneeled in protest of abortion...	0
2	California�s registered independent voters �wi...	0
3	�Antifa now banging on residents� doors in Sea...	0
4	�President Obama and Vice President Biden neve...	0
...	...	...
8528	"Nearly 7 million Floridians have pre-existing...	1
8529	Says Mike DeWine took $40,000 from ECOT and "d...	1
8530	"When two judges said it was illegal to fire a...	1
8531	Says Andrew Gillum "wants to abolish ICE and d...	1
8532	"Bruce Ohr's wife, Nellie Ohr, worked for Fusi...	1

1163 rows × 2 columns

This is the resultant dataset, which will be used as an input dataframe for the corresponding BERT, Deep Learning and PRO text classification workflows as part of this Fake News Detection project implementation.



	Processed_Headline	FakeorReal
0	say obama bin lade endorse joe biden politifac...	0
1	say tim need protest abortion national anthem ...	0
2	california register independent voters wil abl...	0
3	anti bang residents settle , demand & supply	0
4	president obama vice president biden never eve...	0
...	...	...
8528	nearly 7 million floridians pre exist conditio...	1
8529	say mike define tok $ 40,000 eco nothing onlin...	1
8530	two judge say legal fire teacher view pornogra...	1
8531	say andrew film want abolish ice believe type ...	1
8532	bruce 's wife , nelle , work fusion gps firm h...	1

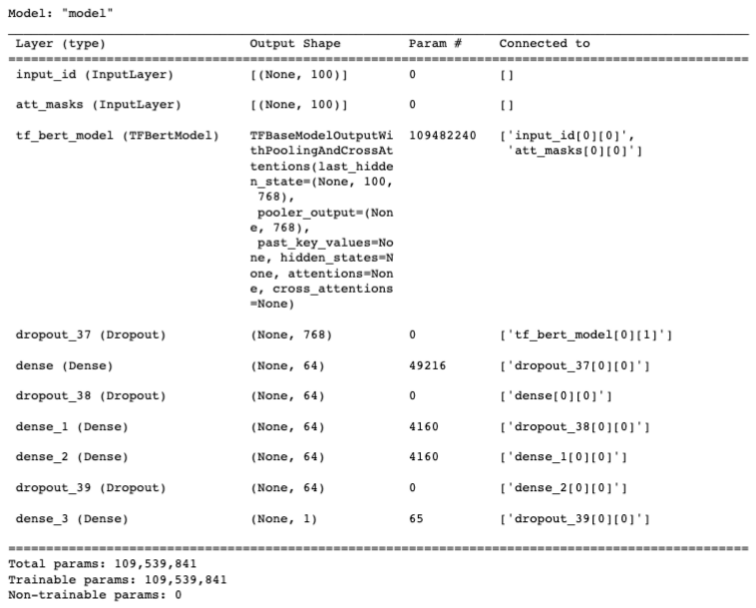
1163 rows × 2 columns

4.6 BERT Approach

Let us now scope into another unique approach that is commonly used in text classification workflows. I implemented this approach in my project so that the Bidirectional Encoder Representations from Transformers (BERT) framework will serve as a benchmark for the much more sophisticated Deep Learning routes we are going to take.

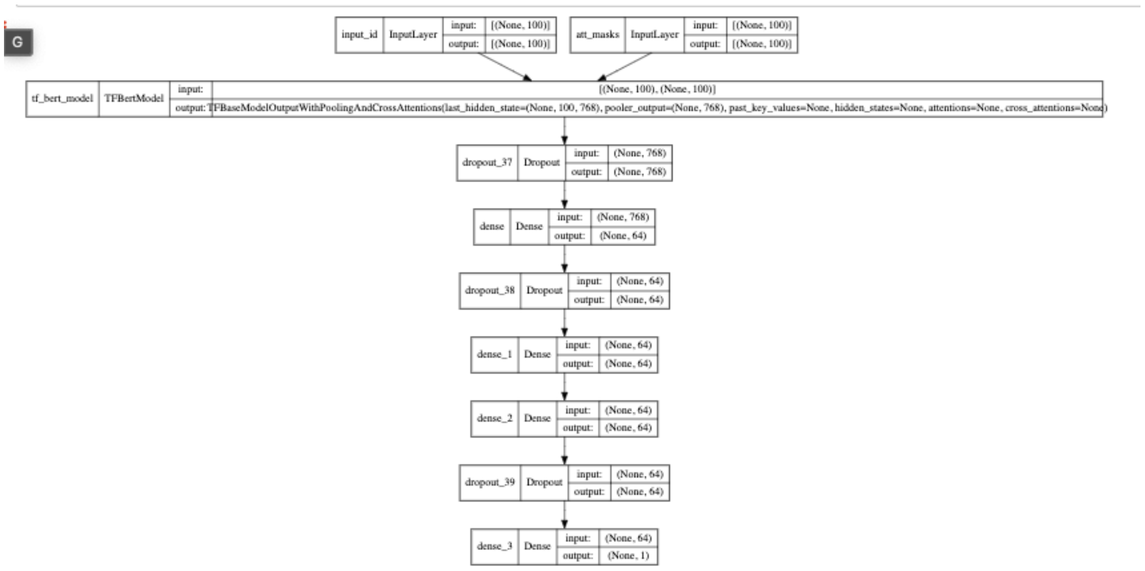
The full code for the BERT Approach can be found in the Jupyter Notebook Chapter 9. The model compilation, fitting and training code can be found in Appendix A.7.

The regimental train-test data splitting, auto-tokenization processes also apply here. Once the input data is prepared, we can then construct the model architecture using the 'bert-base-uncased' base model from the transformers library to compile a model. The model architecture summary is as shown below:



```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_id (InputLayer)          [(None, 100)]        0           []

 att_masks (InputLayer)         [(None, 100)]        0           []

 tf_bert_model (TFBertModel)    TFBaseModelOutputWi  109482240   ['input_id[0][0]',
                                thPoolingAndCrossAt               'att_masks[0][0]']
                                tentions(last_hidde
                                n_state=(None, 100,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 dropout_37 (Dropout)           (None, 768)          0           ['tf_bert_model[0][1]']

 dense (Dense)                  (None, 64)           49216       ['dropout_37[0][0]']

 dropout_38 (Dropout)           (None, 64)           0           ['dense[0][0]']

 dense_1 (Dense)                (None, 64)           4160        ['dropout_38[0][0]']

 dense_2 (Dense)                (None, 64)           4160        ['dense_1[0][0]']

 dropout_39 (Dropout)           (None, 64)           0           ['dense_2[0][0]']

 dense_3 (Dense)                (None, 1)            65          ['dropout_39[0][0]']

==================================================================================================
Total params: 109,539,841
Trainable params: 109,539,841
Non-trainable params: 0
```

The data flow within the architectural design of the BERT model is as shown below:


input_id | InputLayer | input: [(None, 100)] | output: [(None, 100)]
att_masks | InputLayer | input: [(None, 100)] | output: [(None, 100)]
tf_bert_model | TFBertModel | input: [(None, 100), (None, 100)] | output: TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 100, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)
dropout_37 | Dropout | input: (None, 768) | output: (None, 768)
dense | Dense | input: (None, 768) | output: (None, 64)
dropout_38 | Dropout | input: (None, 64) | output: (None, 64)
dense_1 | Dense | input: (None, 64) | output: (None, 64)
dense_2 | Dense | input: (None, 64) | output: (None, 64)
dropout_39 | Dropout | input: (None, 64) | output: (None, 64)
dense_3 | Dense | input: (None, 64) | output: (None, 1)

The evaluation results for the BERT approach model performance will be discussed in Chapter 5: Evaluation.

4.7 Deep Learning Approaches

Now that we have gathered sufficient evaluation metrics and benchmarks from the 3 initial approaches I have taken to optimize the detection of fake news, we can take a look at further Deep Learning approaches as mentioned in Chapter 3: Design.

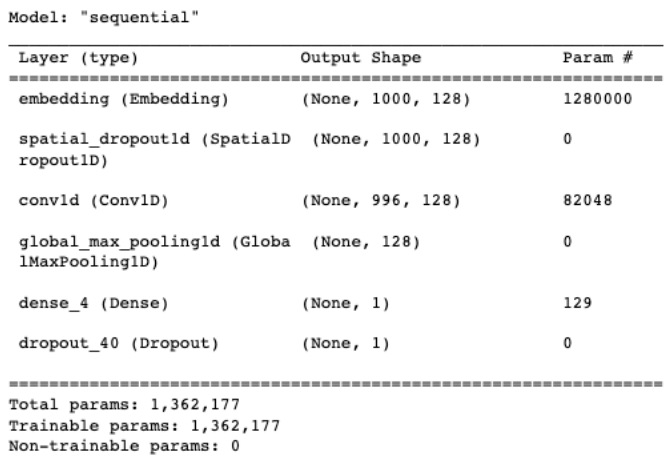
I have taken 2 important Deep Learning approaches, as follows. The full code for the Deep Learning approaches can be found in the Jupyter Notebook Chapter 10.

4.7.1 EDC-NLP + CNN Deep Learning Approach

Accompanying code for the following explanations can be found in Appendix A.8

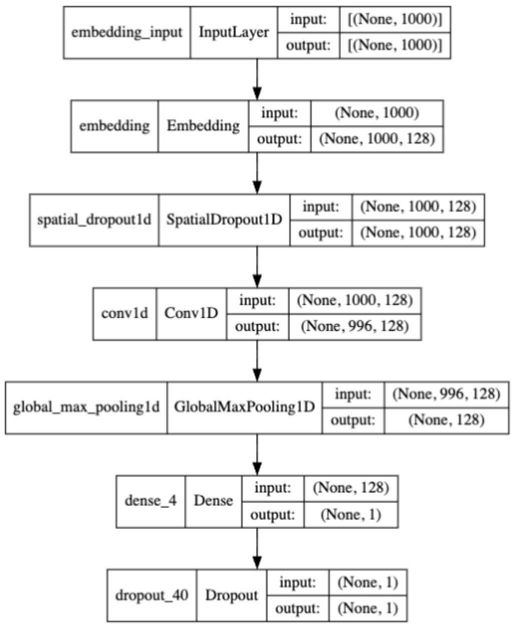
To commence the EDC-NLP + CNN approach, I first split the EDC-NLP processed dataset with training and testing data, using Train-Test Split. Next, tokenization is performed to convert the text data to sequences which can padded or truncated according to the maximum sequence length.

The compiled model summary for the EDC-NLP + CNN Approach is as shown below:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 1000, 128)         1280000

 spatial_dropout1d (SpatialD  (None, 1000, 128)        0
 ropout1D)

 conv1d (Conv1D)             (None, 996, 128)          82048

 global_max_pooling1d (Globa  (None, 128)              0
 lMaxPooling1D)

 dense_4 (Dense)             (None, 1)                 129

 dropout_40 (Dropout)        (None, 1)                 0

=================================================================
Total params: 1,362,177
Trainable params: 1,362,177
Non-trainable params: 0
```

The data flow within the architectural design of the EDC-NLP + CNN approach model is as shown below:


embedding_input | InputLayer | input: [(None, 1000)] | output: [(None, 1000)]
embedding | Embedding | input: (None, 1000) | output: (None, 1000, 128)
spatial_dropout1d | SpatialDropout1D | input: (None, 1000, 128) | output: (None, 1000, 128)
conv1d | Conv1D | input: (None, 1000, 128) | output: (None, 996, 128)
global_max_pooling1d | GlobalMaxPooling1D | input: (None, 996, 128) | output: (None, 128)
dense_4 | Dense | input: (None, 128) | output: (None, 1)
dropout_40 | Dropout | input: (None, 1) | output: (None, 1)

Rationale behind this model architecture:

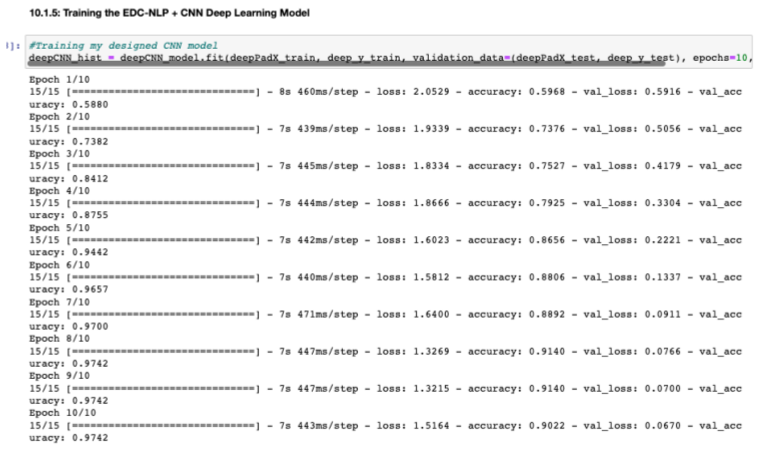
1. Sequential Model Layout
    a. Enables the input and output layers to be connected in a modular format
2. Embedding Layer
    a. Converts the text news data into a dense, vector representation of words, sentences or important contextual sequences
3. Spatial Dropout Layer
    a. Prevents overfitting and generalizes the CNN model
    b. Typically with news articles data, since there are a lot of commonly used phrases and text in journalistic language, this is useful in training on closely related text data, where overfitting may occur
    c. Reduces the usage of computational resources during testing and training
4. Convolutional Layer
    a. Basic 1-Dimensional convolutional layer
    b. Applies filters on input text news data to distinguish and extract important linguistic features
    c. Detects relationships and linguistic norms between words, phrases and sentences to accurately identify the characteristics of fake or real news headlines and article content
5. Global Max Pooling Layer
    a. Reduces every channel in the text feature maps to a single element
    b. Useful in condensing the dimensionality of the outputs from the previous layers
6. Dense Layer

        a. Uses the 'sigmoid' activation function to add complexity to the model
        b. The activation function mentioned consolidates feature vector output values into a 0 to 1 range
7. Dropout Layer
        a. Reduces overfitting of the general model design

I have used a 'binary cross entropy' loss function for the CNN model, which allows us to calculate the amount of offset carried by the predicted probability of the fake or real news data points and obtain a revised, corrected probability score.

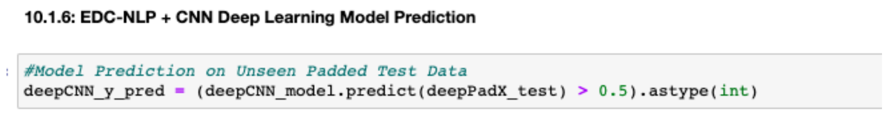
The 'adam' optimizer is an algorithm whereby the gradient descent of the news data classification is exponentially weighted as a mean value and optimizes the rate at which the data is being trained and classified. This optimization algorithm is especially useful for my project's use case due to it being efficient with large datasets like what is used here.

Once the model is compiled, I fitted the model on the input training and validation data.


10.1.5: Training the EDC-NLP + CNN Deep Learning Model

```
#Training my designed CNN model
deepCNN_hist = deepCNN_model.fit(deepPadX_train, deep_y_train, validation_data=(deepPadX_test, deep_y_test), epochs=10,
Epoch 1/10
15/15 [==============================] - 8s 460ms/step - loss: 2.0529 - accuracy: 0.5968 - val_loss: 0.5916 - val_acc
uracy: 0.5880
Epoch 2/10
15/15 [==============================] - 7s 439ms/step - loss: 1.9339 - accuracy: 0.7376 - val_loss: 0.5056 - val_acc
uracy: 0.7382
Epoch 3/10
15/15 [==============================] - 7s 445ms/step - loss: 1.8334 - accuracy: 0.7527 - val_loss: 0.4179 - val_acc
uracy: 0.8412
Epoch 4/10
15/15 [==============================] - 7s 444ms/step - loss: 1.8666 - accuracy: 0.7925 - val_loss: 0.3304 - val_acc
uracy: 0.8755
Epoch 5/10
15/15 [==============================] - 7s 442ms/step - loss: 1.6023 - accuracy: 0.8656 - val_loss: 0.2221 - val_acc
uracy: 0.9442
Epoch 6/10
15/15 [==============================] - 7s 440ms/step - loss: 1.5812 - accuracy: 0.8806 - val_loss: 0.1337 - val_acc
uracy: 0.9657
Epoch 7/10
15/15 [==============================] - 7s 471ms/step - loss: 1.6400 - accuracy: 0.8892 - val_loss: 0.0911 - val_acc
uracy: 0.9700
Epoch 8/10
15/15 [==============================] - 7s 447ms/step - loss: 1.3269 - accuracy: 0.9140 - val_loss: 0.0766 - val_acc
uracy: 0.9742
Epoch 9/10
15/15 [==============================] - 7s 447ms/step - loss: 1.3215 - accuracy: 0.9140 - val_loss: 0.0700 - val_acc
uracy: 0.9742
Epoch 10/10
15/15 [==============================] - 7s 443ms/step - loss: 1.5164 - accuracy: 0.9022 - val_loss: 0.0670 - val_acc
uracy: 0.9742
```

Which would lead us to the final step, model prediction using our trained and optimized EDC-NLP + CNN Deep Learning model.


10.1.6: EDC-NLP + CNN Deep Learning Model Prediction

```
#Model Prediction on Unseen Padded Test Data
deepCNN_y_pred = (deepCNN_model.predict(deepPadX_test) > 0.5).astype(int)
```

The evaluation results for the EDC-NLP + CNN approach model performance will be discussed in Chapter 5: Evaluation.

4.7.2 EDC-NLP + RCNN + LSTM Deep Learning Approach

Accompanying code for the following explanations can be found in Appendix A.9.

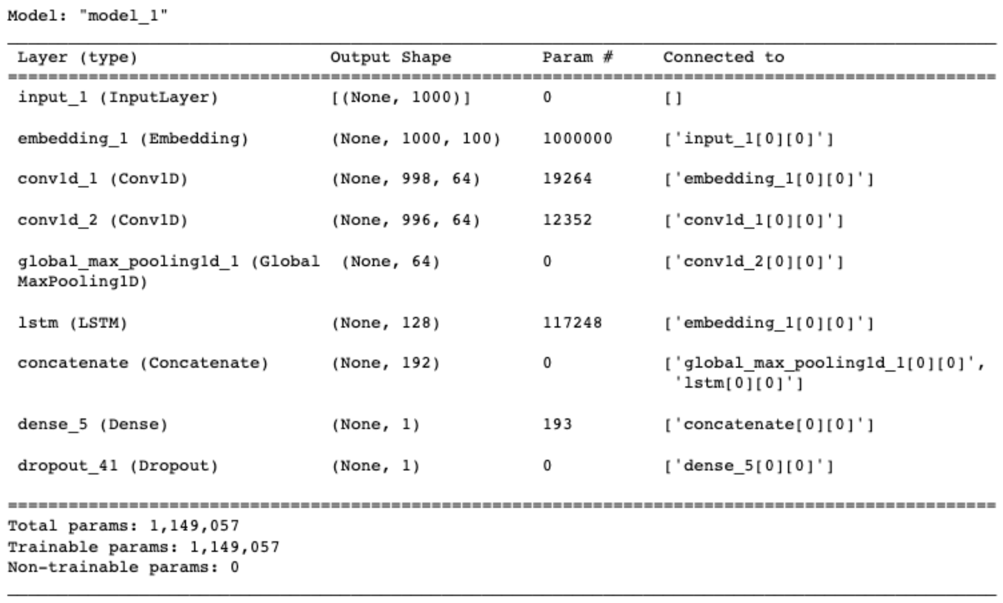
Now that we have explored the implementation of the EDC-NLP + CNN Deep Learning Approach, it is time to explore a more layered and optimized approach I have taken to improve on the accuracies of detecting fake news.

The RCNN (Recurrent Convolutional Neural Network) and the Long Short Term Memory (LSTM) will collectively be able to:

1. Learn contextual information important to the journalistic language and situation it is used in to identify the text representations within several news article data points
2. Be effective in handling long term dependencies with data points of a sequential nature

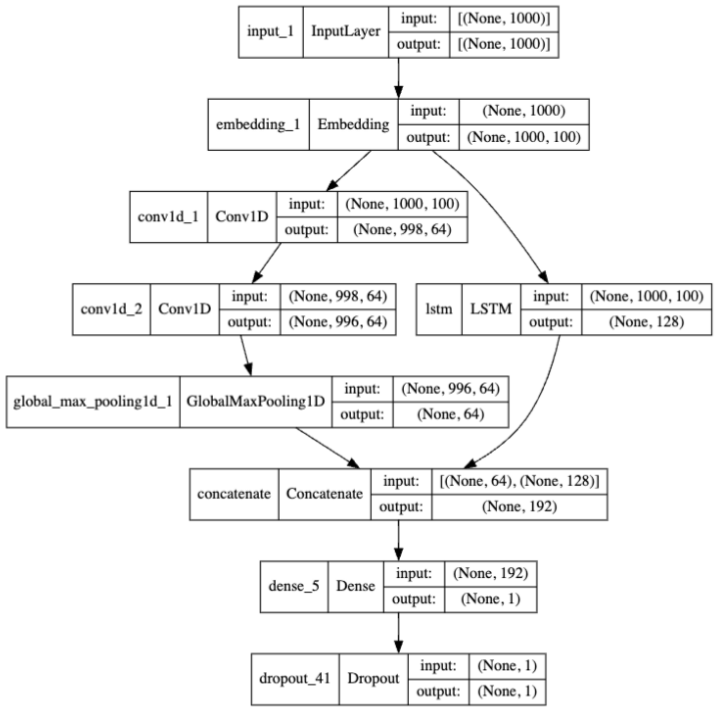
Hence for the implementation proper, I first split the EDC-NLP processed dataset with training and testing data, using Train-Test Split. Next, tokenization is performed to convert the text data to sequences which can padded or truncated according to the maximum sequence length.

The compiled model summary for the EDC-NLP + RCNN _LSTM Approach is as shown below:



```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 1000)]       0           []

 embedding_1 (Embedding)        (None, 1000, 100)    1000000     ['input_1[0][0]']

 conv1d_1 (Conv1D)              (None, 998, 64)      19264       ['embedding_1[0][0]']

 conv1d_2 (Conv1D)              (None, 996, 64)      12352       ['conv1d_1[0][0]']

 global_max_pooling1d_1 (Global  (None, 64)          0           ['conv1d_2[0][0]']
 MaxPooling1D)

 lstm (LSTM)                    (None, 128)          117248      ['embedding_1[0][0]']

 concatenate (Concatenate)      (None, 192)          0           ['global_max_pooling1d_1[0][0]',
                                                                  'lstm[0][0]']

 dense_5 (Dense)                (None, 1)            193         ['concatenate[0][0]']

 dropout_41 (Dropout)           (None, 1)            0           ['dense_5[0][0]']

==================================================================================================
Total params: 1,149,057
Trainable params: 1,149,057
Non-trainable params: 0
__________________________________________________________________________________________________
```

The data flow within the architectural design of the EDC-NLP + RCNN + LSTM approach model is as shown below:


input_1 InputLayer input: [(None, 1000)] output: [(None, 1000)]
embedding_1 Embedding input: (None, 1000) output: (None, 1000, 100)
conv1d_1 Conv1D input: (None, 1000, 100) output: (None, 998, 64)
conv1d_2 Conv1D input: (None, 998, 64) output: (None, 996, 64)
lstm LSTM input: (None, 1000, 100) output: (None, 128)
global_max_pooling1d_1 GlobalMaxPooling1D input: (None, 996, 64) output: (None, 64)
concatenate Concatenate input: [(None, 64), (None, 128)] output: (None, 192)
dense_5 Dense input: (None, 192) output: (None, 1)
dropout_41 Dropout input: (None, 1) output: (None, 1)

Rationale behind this model architecture:

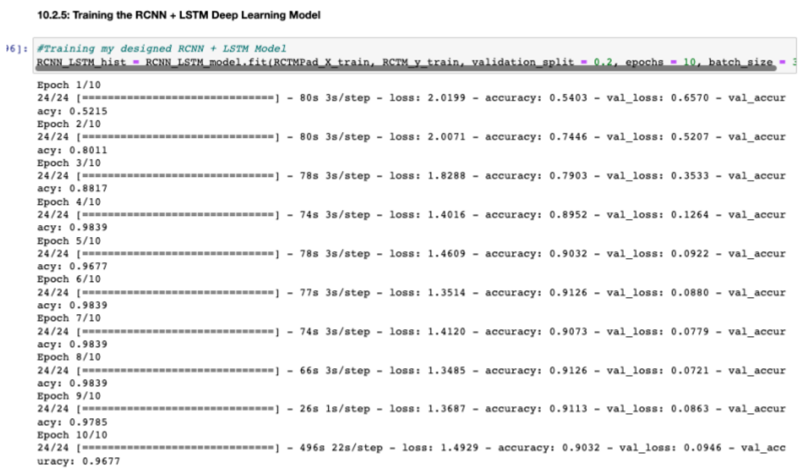
1. Sequential Model Layout
    a. Enables the input and output layers to be connected in a modular format
2. Embedding Layer
    a. Converts the text news data into a dense, vector representation of words, sentences or important contextual sequences
3. Convolutional Layer X 2 (RCNN)
    a. 2 Basic 1-Dimensional convolutional layers
    b. Kernel size of 3 with a filter size of 64
    c. Applies filters on input text news data to distinguish and extract important linguistic features
    d. Uses the 'relu' activation function: The rectified linear unit function is adept at converting negative probability values to usable integers according to the binary node outputs
    e. Detects patterns between words, phrases and sentences to accurately identify the characteristics of fake or real news headlines and article content
4. Global Max Pooling Layer
    a. Reduces every channel in the text feature maps to a single element
    b. Useful in condensing the dimensionality of the outputs from the previous layers
5. LSTM Layer
    a. Used to overcome dependency issues caused by RCNNs

   b. Provides ‘feedback’ node design as opposed to conventional ‘feed forward’ in deep learning neural networks
   c. Works well on time series data, which also applies here with time sensitive news headline + content sequential data
6. Dense Layer
   a. Uses the ‘sigmoid’ activation function to add complexity to the model
   b. The activation function mentioned consolidates feature vector output values into a 0 to 1 range
7. Dropout Layer
   a. Prevents overfitting and generalizes the RCNN + LSTM model
   b. Typically with news articles data, since there are a lot of commonly used phrases and text in journalistic language, this is useful in training on closely related text data, where overfitting may occur
   c. Reduces the usage of computational resources during testing and training

It is important to note that the Global Max Pooling Layer and the LSTM Layer are concatenated within the model architecture to provide a symbiotic relationship between condensing vector representation output data and reducing dependency issues when training the model.

Similar optimizers and loss functions are used for the EDC-NLP + RCNN + LSTM Deep Learning model.

Once the model is compiled, I fitted the model on the input training and validation data.

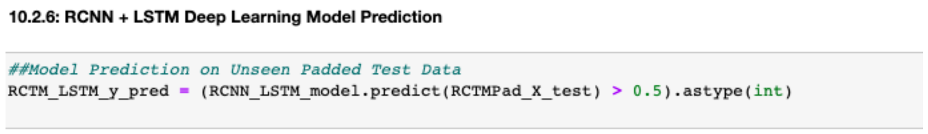


10.2.5: Training the RCNN + LSTM Deep Learning Model

```
)6]: #Training my designed RCNN + LSTM Model
RCNN_LSTM_hist = RCNN_LSTM_model.fit(RCTMPad_X_train, RCTM_y_train, validation_split = 0.2, epochs = 10, batch_size = 3
```

```
Epoch 1/10
24/24 [==============================] - 80s 3s/step - loss: 2.0199 - accuracy: 0.5403 - val_loss: 0.6570 - val_accur
acy: 0.5215
Epoch 2/10
24/24 [==============================] - 80s 3s/step - loss: 2.0071 - accuracy: 0.7446 - val_loss: 0.5207 - val_accur
acy: 0.8011
Epoch 3/10
24/24 [==============================] - 78s 3s/step - loss: 1.8288 - accuracy: 0.7903 - val_loss: 0.3533 - val_accur
acy: 0.8817
Epoch 4/10
24/24 [==============================] - 74s 3s/step - loss: 1.4016 - accuracy: 0.8952 - val_loss: 0.1264 - val_accur
acy: 0.9839
Epoch 5/10
24/24 [==============================] - 78s 3s/step - loss: 1.4609 - accuracy: 0.9032 - val_loss: 0.0922 - val_accur
acy: 0.9677
Epoch 6/10
24/24 [==============================] - 77s 3s/step - loss: 1.3514 - accuracy: 0.9126 - val_loss: 0.0880 - val_accur
acy: 0.9839
Epoch 7/10
24/24 [==============================] - 74s 3s/step - loss: 1.4120 - accuracy: 0.9073 - val_loss: 0.0779 - val_accur
acy: 0.9839
Epoch 8/10
24/24 [==============================] - 66s 3s/step - loss: 1.3485 - accuracy: 0.9126 - val_loss: 0.0721 - val_accur
acy: 0.9839
Epoch 9/10
24/24 [==============================] - 26s 1s/step - loss: 1.3687 - accuracy: 0.9113 - val_loss: 0.0863 - val_accur
acy: 0.9785
Epoch 10/10
24/24 [==============================] - 496s 22s/step - loss: 1.4929 - accuracy: 0.9032 - val_loss: 0.0946 - val_acc
uracy: 0.9677
```

Which would lead us to the final step, model prediction using our trained and optimized EDC-NLP + RCNN + LSTM Deep Learning model.


10.2.6: RCNN + LSTM Deep Learning Model Prediction

```
##Model Prediction on Unseen Padded Test Data
RCTM_LSTM_y_pred = (RCNN_LSTM_model.predict(RCTMPad_X_test) > 0.5).astype(int)
```

The evaluation results for the EDC-NLP + RCNN + LSTM approach model performance will be discussed in Chapter 5: Evaluation.

4.8 PRO Approaches (EDC-NLP)

Once I was done with developing my own Deep Learning model architectures and evaluating the metrics (accuracy/recall/precision) scores; I was also intrigued to find out how the EDC-NLP pre-process, combined with the web-scraped dataset which now contains comprehensive data on news headline and news content will affect the metrics on the initial Baseline and Alternative approaches.

4.8.1 Baseline PRO Approach

The overall implementation for the Baseline PRO approach is similar to the original baseline approach with only 1 difference.

It was trained on the EDC-NLP Web Scraped dataset.

The full code can be found in Appendix A10 / Jupyter Notebook Chapter 11.

The evaluation results for the EDC-NLP + TF-IDF + Multinomial Naïve Bayes Classification approach model performance will be discussed in Chapter 5: Evaluation.

4.8.2 Alternative PRO Approach

The overall implementation for the Alternative PRO approach is similar to the original alternative approach with only 1 difference.

It was trained on the EDC-NLP Web Scraped dataset.

The full code can be found in Appendix A11 / Jupyter Notebook Chapter 12.

The evaluation results for the EDC-NLP + TF-IDF + Passive Aggressive Classification approach model performance will be discussed in Chapter 5: Evaluation.

CHAPTER 5: EVALUATION

5.1 Evaluation Scope and Rationale

Evaluation metrics are an essential component of any machine learning, or classification workflow. It enables us to understand how effective the model or approach has been in learning, understanding, and analyzing the input data of the news articles and predicting the legitimacy of an unknown test news data point.

The evaluation metrics used for this project has been mainly accuracy scores. However, this metric in itself is not sufficient to make concrete conclusions on the efficacy of the fake news detection workflows. Hence, I have also included precision and recall scores as part of the evaluation process for this project.

As has been discussed in the Chapter 1: Introduction, in Section 1.5, under Aims; this project aims to understand the significance and impact of building new NLP workflows minted with unique Deep Learning or machine learning classification approaches on the Fake News Detection problem.

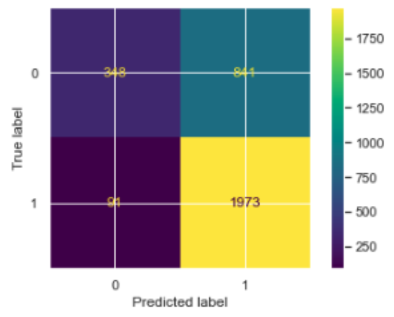
Since I have developed such workflows, doing the necessary model performance evaluation steps are key to determining the benefits of developing new NLP pipelines and text classification workflows.

Let us now look at the corresponding confusion matrix results and evaluation metric scores for each of the approaches I have implemented in this project.

5.2 Model Performance Evaluation

5.2.1 Baseline Model Evaluation (SimpleNLP + TF-IDF + Multinomial Naïve Bayes Classifier)

The evaluation results, which have been represented as a confusion matrix are as follows:



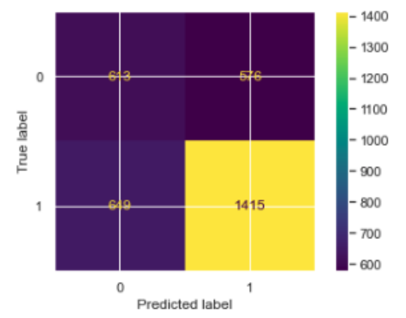
The below table shows the relevant metric scores which have been calculated in the evaluation stage of this workflow, to better assess the performance of the baseline model.

Table 5.2.1 Evaluation Metrics for SimpleNLP + TF-IDF + Multinomial NB classification (Baseline)

Evaluation Metric	**ACCURACY**	**PRECISION**	**RECALL**
Score	71.35%	70.11%	95.59%

5.2.2 Alternative Model Evaluation (SimpleNLP + TF-IDF + Passive Aggressive Classifier)

The evaluation results, which have been represented as a confusion matrix are as follows:



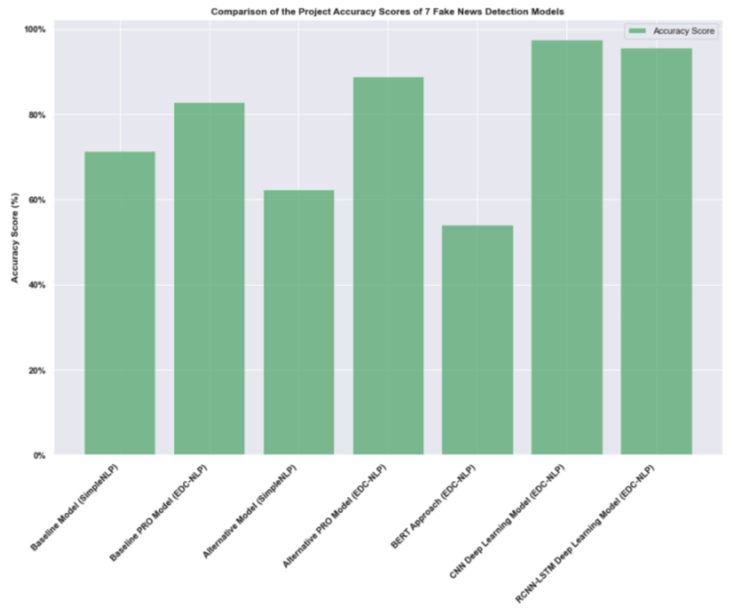
The below table shows the relevant metric scores which have been calculated in the evaluation stage of this workflow, to better assess the performance of the alternative model.

Table 5.2.2 Evaluation Metrics for SimpleNLP + TF-IDF + Passive Aggressive classification (Alternative)

Evaluation Metric	**ACCURACY**	**PRECISION**	**RECALL**
Score	62.34%	71.07%	68.56%

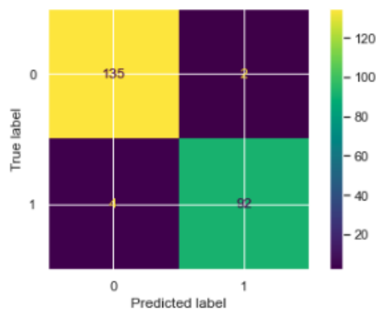
5.2.3 BERT Model Evaluation (EDC-NLP + BERT Approach)

The bar chart below shows how the BERT model's accuracy score compares with the rest of the models.



5.2.4 Deep Learning Model 1 Evaluation (EDC-NLP + CNN Deep Learning Approach)

The evaluation results, which have been represented as a confusion matrix are as follows:



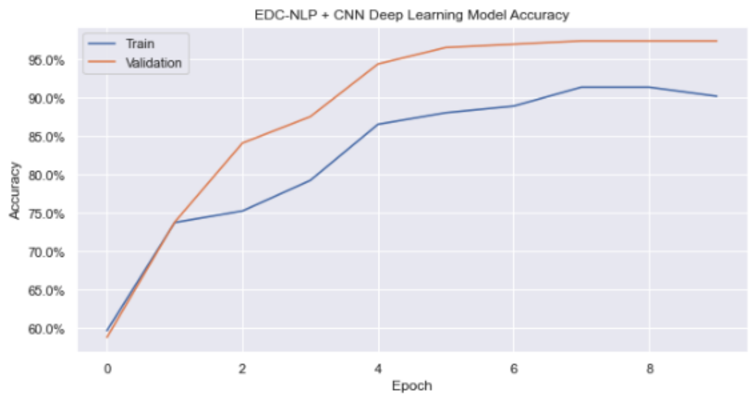
The below table shows the relevant metric scores which have been calculated in the evaluation stage of this workflow, to better assess the performance of the deep learning model.

Table 5.2.4 Evaluation Metrics for EDC-NLP + CNN Deep Learning Model

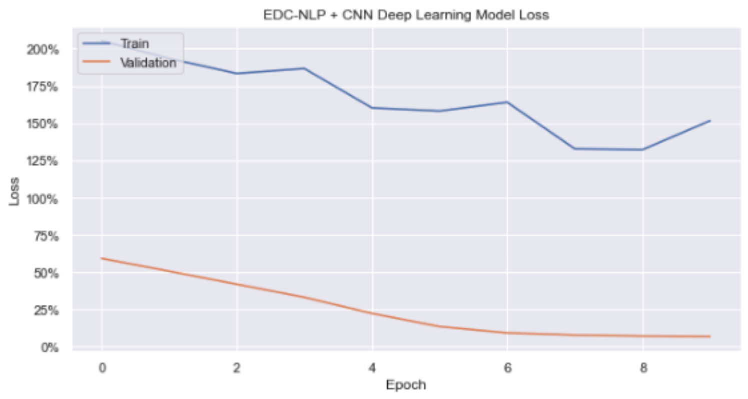
Evaluation Metric	**ACCURACY**	**PRECISION**	**RECALL**
Score	97.42%	97.87%	95.83%

Since Deep Learning neural networks also train on validation data, it is important for us to assess the model loss performance (indicates how unfavorably the model has predicted the right value for the news data point) and accuracy performance for each training Epoch using the following graph plots as show below:

EDC-NLP + CNN Deep Learning Model Accuracy Performance History



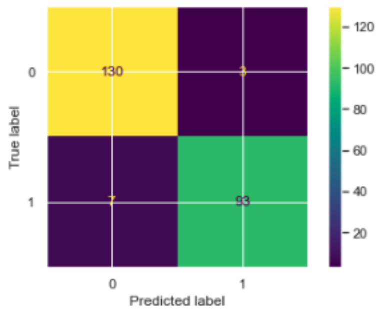
EDC-NLP + CNN Deep Learning Model Loss Performance History



The full code for the above plots can be found in Appendix A12 / Jupyter Notebook Chapter 13.

5.2.5 Deep Learning Model 2 Evaluation (EDC-NLP + RCNN + LSTM Deep Learning Approach)

The evaluation results, which have been represented as a confusion matrix are as follows:



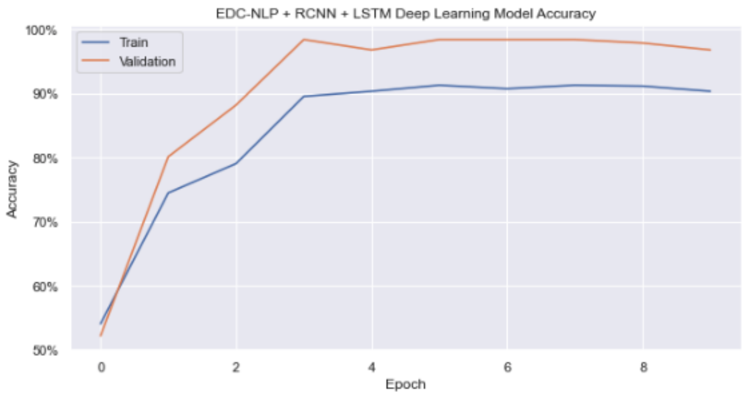
The below table shows the relevant metric scores which have been calculated in the evaluation stage of this workflow, to better assess the performance of the deep learning model.

Table 5.2.5 Evaluation Metrics for EDC-NLP + RCNN + LSTM Deep Learning Model

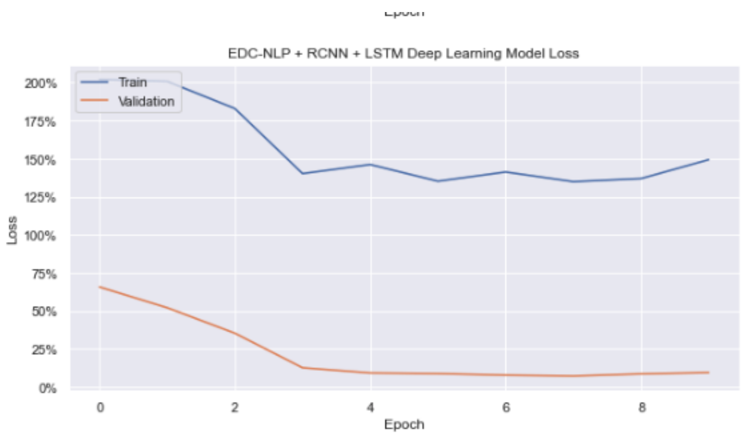
Evaluation Metric	**ACCURACY**	**PRECISION**	**RECALL**
Score	95.71%	96.88%	93.00%

Since Deep Learning neural networks also train on validation data, it is important for us to assess the model loss performance (indicates how unfavorably the model has predicted the right value for the news data point) and accuracy performance for each training Epoch using the following graph plots as show below:

EDC-NLP + RCNN + LSTM Deep Learning Model Accuracy Performance History



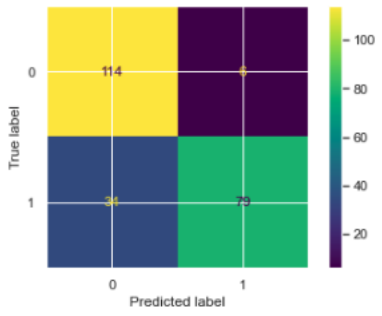
EDC-NLP + RCNN + LSTM Deep Learning Model Loss Performance History



The full code for the above plots can be found in Appendix A13 / Jupyter Notebook Chapter 13.

5.2.6 Baseline PRO Model Evaluation (EDC-NLP + TF-IDF + Multinomial Naïve Bayes Classifier)

The evaluation results, which have been represented as a confusion matrix are as follows:



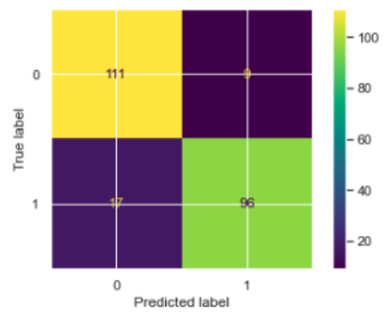
The below table shows the relevant metric scores which have been calculated in the evaluation stage of this workflow, to better assess the performance of the baseline PRO model.

Table 5.2.6 Evaluation Metrics for EDC-NLP + TF-IDF + Multinomial NB classification (Baseline PRO)

Evaluation Metric	**ACCURACY**	**PRECISION**	**RECALL**
Score	82.83%	92.94%	69.91%

5.2.7 Alternative PRO Model Evaluation (EDC-NLP + TF-IDF + Passive Aggressive Classifier)

The evaluation results, which have been represented as a confusion matrix are as follows:



The below table shows the relevant metric scores which have been calculated in the evaluation stage of this workflow, to better assess the performance of the alternative PRO model.

Table 5.2.7 Evaluation Metrics for EDC-NLP + TF-IDF + Passive Aggressive classification (Alternative PRO)

Evaluation Metric	**ACCURACY**	**PRECISION**	**RECALL**
Score	88.84%	91.43%	84.96%

5.3 Initial Prototype Evaluation

As can be seen from the table 4.2.1 above, it is evident that while the accuracy score (71.35%) for the TF-IDF + Multinomial NB classification workflow is not high at this stage, there is room for significant improvement to be made from this baseline performance.

Additionally, the accuracy score from the table 4.3.1 also shows that the alternative approach of TF-IDF + Passive Aggressive classification poses significant leeway for development into a much more accurate model for fake news detection.

5.4 Project Results Overview

The table below summarizes the evaluation metrics scores of all the Fake News Detection approaches taken in this project.

Table 5.4.1 Overall Evaluation Metric Score Summary

Fake News Detection Approach	Pre-Processing/ Dataset Used	Accuracy Score (%)	Precision Score (%)	Recall Score (%)
Baseline (tf-idf + multNB classifier)	SimpleNLP	71.35	70.11	95.59
Alternative (tf-idf + passive aggressive classifier)	SimpleNLP	62.34	71.07	68.56
CNN Deep Learning	EDC-NLP	97.42	97.87	95.83
RCNN + LSTM Deep Learning	EDC-NLP	95.71	96.88	93.00
Baseline PRO	EDC-NLP	82.83	92.94	69.91
Alternative PRO	EDC-NLP	88.84	91.43	84.96

To evaluate the models according to the proposed project aims, let us analyze the statistics associated with the accuracy scores of the models.

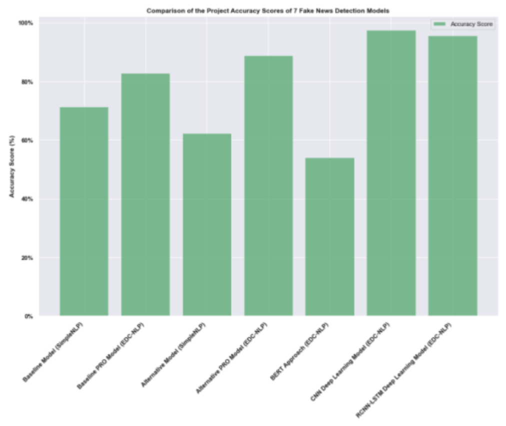
As can be observed from table 5.4.1, the Fake News Detection approach with the highest accuracy score poses to be the Convolutional Neural Networks (CNN) Deep Learning model, at 97.42%. The increasingly developed more sophisticated RCNN + LSTM Deep Learning model comes in at a close second at 95.71%. The lowest accuracy score stands with the Alternative model, where a Passive Aggressive classification is used at 62.34%. The difference between the highest and lowest accuracy scores stands at 56.22%. Hence it can be concluded that to observe a ~50% jump in accuracy rates, it is better to deploy deep learning models in Fake News Detection applications.

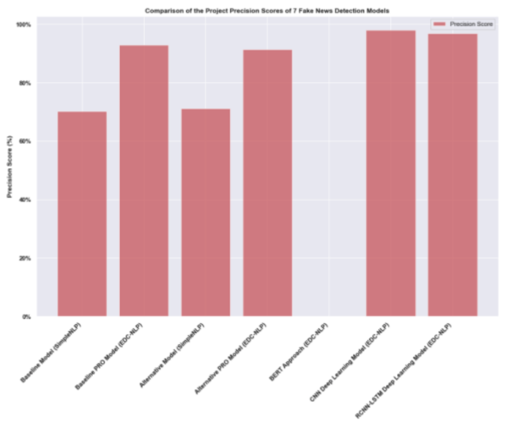
This shows that the Deep Learning models, which use data cleansed by properly implemented NLP pipelines such as the EDC-NLP I have implemented, generally fair better at predicting the correct labels for fake news data points out of the overall prediction outputs than conventional machine learning workflows which also use the EDC-NLP pipeline for pre-processing news data.

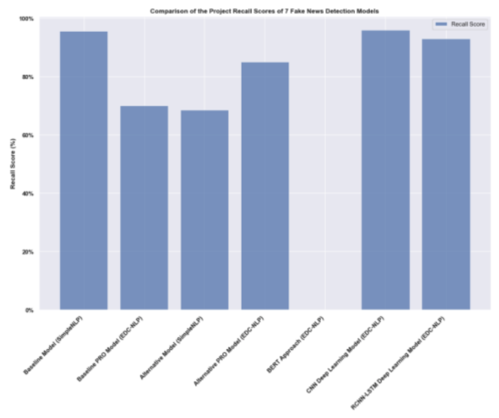
The precision scores of the models indicate the rate at which the model is able to identify real news as real amongst its overall prediction outputs. A general trend in the summary depicts that models which have used the EDC-NLP pipeline have higher precision scores.

The Deep Learning models also show increased precision scores at 97.87% and 96.88% respectively. The lowest precision score stands at 70.11% for the baseline model. The difference between the highest and lowest precision score is 39.52%. Hence, we can deduce that to observe a ~40% increase in precision scores, it is better to utilize sophisticated NLP pipeline with thorough data cleansing operations such as the EDC-NLP pipeline combined with Deep Learning classification workflows to optimize both accuracy and precision scores in detecting fake news amongst large datasets or data sources.

I have also included bar charts comparing the evaluation metrics of each of the 7 approaches as shown below. The code for these can be found in Appendix A14 / Jupyter Notebook Chapter 13:







5.5 Feature Evaluation:

The main feature I have used throughout my project implementation would be linguistical and contextual text data.

Overall, analyzing and processing the text data required some development using NLP tools. However, it is worth noting that this development and data analysis on the linguistical patterns and contextual evaluation paid off when developing the appropriate text classification models and training the input data. This was observed when the models were tested on unseen fake news data, especially in the Deep Learning approaches.

This project helped me understand that the importance of obtaining and formatting data into a rich, comprehensive input using feature extraction techniques, be it using Count Vectorization, Tokenization, or other NLP techniques.

From this project, it can be observed that it is also vital to consider that fake news articles usually contain some common characteristics. Fake information and articles are usually disseminated based on piggybacking on the reader's emotional response. Hence, when extracting features, it is important to flag out indicators / linguistical features such as:

1. Taunting / provocative language
2. Emotive language
3. Images with obscenities
4. Persuasive language

5.6 Critical Evaluation

At this stage of the project, it is vital to assess how the topography of my Final Year Project changed and evolved over the past 6 months. Since we have looked at the evaluation of the models I have implemented, we can now explore how some phases of the project be better implemented to address the Fake News Detection problem.

5.6.1 What went well in this project?

Overall, I would say my project implementation was able to sufficiently adhere to the project design I initially set out to work on. I was able to achieve most of the aims and objectives I have mentioned in Chapter 1 of this report.

I was able to establish 1 basic NLP pipeline and 2 starter Fake News Detection models as a baseline to develop further from, which were the baseline and alternative models.

Next, I was able to fine tune the NLP pipeline and apply it on to further optimize the input dataset used in the other 5 Fake News Detection models, which include Deep Learning workflows too.

I was able to compare and obtain deductions based on the evaluation metrics which were output by the models successfully, and this helped me to gauge if I was able to deploy a Fake News Detection model which would perform better than the baseline models I first implemented in the preliminary stages.

5.6.2 What could have been done better in this project?

However, it is also worth noting that there is always room for improvement. Project aims and objectives always have scope to evolve and be optimized to better serve the respective user domains of the project.

I could have developed a better implementation of my BERT model. I was unable to obtain some of the relevant evaluation metrics for analysis.

As for dataset wise, I could have found a workaround to be able to incorporate the entire 10K+ articles dataset into all the approaches, for a more comprehensive and accurate text classification process. However, this was not able to be achieved as the dataset was too large and occupied rlong durations of the timeframe allotted to the implementation of each of the approaches. Since I was working under a timeframe set by me as can be referenced by the Gantt Chart in this report, I had to progress across subsequent phases of the project which served greater weightage to the aims and objectives of the bigger scope.

CHAPTER 6: CONCLUSION

To conclude the Misinformation Mitigation project, I would like to first remind you of the preliminary research questions that I initially mentioned in Chapter 1.8 of this report.

Let us see if these questions have been addressed, and hence have the aims of this project been accomplished.

1. Is it necessary to build an increasingly accurate NLP pipeline and model system to detect fake news when considering the capabilities of currently available options?

   Yes, it is very necessary to develop and implement a robust and reliable pre-processing system, such as EDC-NLP for the text data in your dataset. Well formatted input data goes a long way in how the classification model learns and predicts the test data. It is also worth noting that not all NLP pipelines apply to all datasets, and it is good to have a pipeline that is tailored just right for the dataset you are using. This also helps in the model architecture design process of the classification models to be accurate and efficient in prediction workflows.

2. How can the content of the fake news be analyzed using feature extraction methods to improve model performance in detecting the presence of fake news identifiers?

   This is addressed in this report under Chapter 5.5: Feature Evaluation.

3. Are deep learning neural network models better in comparison to classification models for the chosen dataset and to achieve better accuracies when it comes to evaluating the improved models?

   The results I have presented in the previous chapter falls in line with this question. It can be well concluded that Deep Learning neural network models are better in terms of text classification as compared to conventional machine learning techniques. However, it is also important to have the right input data to train the models.

Hence in conclusion, Deep Learning models such as convolutional neural networks (CNN) produce high accuracies in fake news detection applications, and ought to be supplemented with optimized model architectures, and well formatted input data with the appropriate NLP pipelines. The RCNN and LSTM learning models are much more suitable for time series data, as we are using the LSTM layers which deal with sequential data. Hence for such models, it would be apt if the news or text corpus dataset had information such as date of publication etc.

6.1 Now, this brings us to another question. How else may I evolve this project in the future?

To evolve this project, I would first pose this question as a starter point to work towards:

At which point does using NLP and machine learning workflows to classify data points as fake or real compromise the representation of marginalized groups, especially in the journalistic, reporting context?

What if a Fake News Detection model wrongly classifies a data point as fake, when the article is in fact real, and the reason this happened is because it was not able to consider the linguistic semantics which accompany marginalized communities, or other ethical contexts?

Important news articles which present authentic and urgent information may be mistakenly censored and the implications that may arise if this happens may sometimes be catastrophical even.

How can we further develop these models to extract these relevant features to identify these situations and avoid misclassification of news articles as legitimate or illegitimate?

6.2 Other dataset features I can analyze to achieve better metrics

Another dataset feature I could have used to further improve my metrics could have been using when the news articles were published from a singular source. Thinking back, I would go to say that the RCNN + LSTM Deep Learning model would have worked better on time series data too, as the LSTM layer is suitable for the training of sequential time data.

6.3 Impact of my Final Year Project

It is with no question, the number of times I have re-iterated the importance of developing improved models to enable better fake news detection applications in today’s digital landscape. Fake information travels like wildfire through social media and websites due to its emotive and provocative linguistical characteristics, and it is vital to curb the spread by having firewalls which serve as a double checker before the information reaches a reader’s eyes.

However, what other uses do these text classification models I have developed have in the real world, but not limited to?

1. Sentiment Analysis (Product reviews)
2. PDF Report → Database Generation
3. Spam emails/SMS detection

CHAPTER 7: PROJECT SOURCE CODE LINK

Source Code Link:

https://github.com/mnp1223/NP-NLP-FinalYearProject.git

I have published my Jupyter Notebook Project on the public repository on the Github platform. The repository contains the following items:

1. FYP Jupyter Notebook (.ipynb file)
2. fyp_requirements.txt (.txt file)
3. Fake-Real News Dataset (.csv file)
4. READ ME file

CHAPTER 8: REFERENCES

ABHIJNAN CHAKRABORTY, BHARGAVI PARANJAPE, SOURYA KAKARLA, AND NILOY GANGULY. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. pages 9–16

ALEXANDRE BOVET & HERNÁN A. MAKSE. 2019. Influence of fake news in twitter during the 2016 us presidential election. Nature Communications, 10(7).

DAVIS, R., PROCTOR, C. 2017. Fake News, Real Consequences: Recruiting Neural Networks for the Fight against Fake News. https://web.stanford.edu/class/cs224n/reports/2761239.pdf

FENG S, BANERJEE R, CHOI Y. 2012 Syntactic stylometry for deception detection. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers, vol 2. Association for Computational Linguistics, pp 171–175
https://www.statista.com/statistics/1112026/fake-news-prevalence-attitudes-worldwide/

I. TRAORE ET AL. 2017 “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques.” International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments (pp. 127–138)

K LUDWIG, M CREATION. 2020 “Dissemination and uptake of fake-quotes in lay political discourse on Facebook and Twitter” J. Pragmat, 157, 101–118.

KAJAL YADAV. 2020 Fake-Real News. Kaggle Dataset Source. https://www.kaggle.com/datasets/c5c1decebd99f153eaf807468727d1df374afd6873d6f49cc72eab1c9080e9dd

KALIYAR, R.K., GOSWAMI, A., & NARANG, P. 2020. DeepFakE: improving fake news detection using tensor decomposition-based deep neural network. *The Journal of Supercomputing*, 1-23.

M. CHANG ET AL J. DEVLIN. BERT. 2017: Pre-training of deep bidirectional transformers for language understanding

M. MAIMAITI, A. WUMAIER, K. ABIDEREXITI, AND T. YIBULAYIN. 2017 “Bidirectional long short-term memory network with a conditional random field layer for Uyghur part-of-speech tagging,” Information, vol. 8, no. 4, article no. 157.

NATEKIN A, KNOLL A. 2013 Gradient boosting machines, a tutorial. Front Neurorobotics 7:21

P. M. SOSA, "TWITTER SENTIMENT ANALYSIS USING COMBINED LSTM-CNN MODELS," 2018 [Online]. Available: http://konukoii.com/blog/2018/02/19/twitter-sentiment-analysis-using-combined-lstm-cnn-models/.

PAPANASTASIOU F, KATSIMPRAS G, PALIOURAS G. 2019 Tensor factorization with label information for fake news detection
PAVLESKA T, ŠKOLKAY A, ZANKOVA B, ET AL. 2018 Performance analysis of fact-checking organizations and initiatives in Europe: A critical overview of online platforms fighting fake news. In: *EIDS6 (European Integration and Democracy Series)*, pp. 1–29.

PÉREZ-ROSAS V AND MIHALCEA R. 2015 Experiments in open domain deception detection. In: *Proceedings of the conference on empirical methods in natural language processing,* pp. 1120–1125.

RABANSER S, SHCHUR O, GNNEMANN S. 2017 Introduction to tensor decompositions and their applications in machine learning.

RASHKI HANNAH AND CHOI EUNSOL ET AL. 2017 Truth of varying shades: Analyzing language in fake news and political fact-checking. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2931–2937, Copenhagen, Denmark. Association for Computational Linguistics.

SAMMUT C, WEBB, GI (eds). 2011 Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_832

SHU K, WANG S, LIU H. 2019 Beyond news contents: the role of social context for fake news detection. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. ACM, pp 312–320

SWIRE B, ECKER UK, LEWANDOWSKY S. 2017 The role of familiarity in correcting inaccurate information. *Journal of Experimental Psychology*. *Learning, Memory, and Cognition* 43(12): 1948. Crossref. PubMed.

T. ZHAI AND H. WANG. 2022 "Online Passive-Aggressive Multilabel Classification Algorithms," in IEEE Transactions on Neural Networks and Learning Systems

TOM YOUNG, DEVAMANYU HAZARIKA, SOUJANYA PORIA, AND ERIK CAMBRIA. 2017. Recent trends in deep learning based natural language processing

TORABI ASR, F., & TABOADA, M. 2019. Big Data and quality data for fake news and misinformation detection. Big Data & Society, 6(1). https://doi.org/10.1177/2053951719843310

WANG WY. 2017 ‘Liar, liar pants on fire’: A new benchmark dataset for fake news detection. In: *Proceedings of the 55th annual meeting of the Association for Computational Linguistics*, Vol. 2, Vancouver, Canada, pp. 422–426.

WARDLE C. 2016 Six types of misinformation circulated this election season. *Columbia Journalism Review,* 18 November.
WATSON, A. 2021. Share of adults worldwide who believe fake news is prevalent in selected media sources as of February 2019. *Perceived prevalence of fake news in media sources worldwide 2019.*

WEI, W., ZHANG, X., LIU, X., CHEN, W., WANG, T. 2016 A Specific Convolutional Neural Network System for Effective Stance Detection. http://www.aclweb.org/anthology/S16-1062

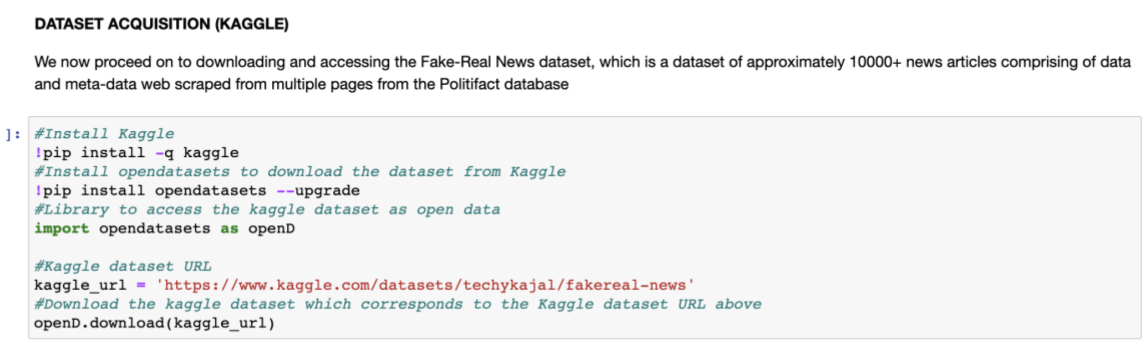
Y. KIM. 2014 “Convolutional neural networks for sentence classification,” Available (Online): https://arxiv.org/abs/1408.5882.

ZHANG X, ZHAO J AND LECUN Y. 2015 Character-level convolutional networks for text classification. In: *Proceedings of the 28th international conference on neural information processing systems,* Montréal, Canada, pp. 649–657.

CHAPTER 9: APPENDIX

Appendix A

A.1

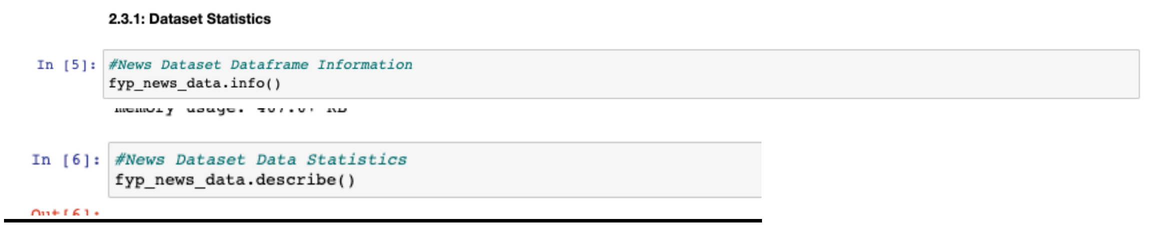


DATASET ACQUISITION (KAGGLE)

We now proceed on to downloading and accessing the Fake-Real News dataset, which is a dataset of approximately 10000+ news articles comprising of data and meta-data web scraped from multiple pages from the Politifact database

```
]: #Install Kaggle
   !pip install -q kaggle
   #Install opendatasets to download the dataset from Kaggle
   !pip install opendatasets --upgrade
   #Library to access the kaggle dataset as open data
   import opendatasets as openD

   #Kaggle dataset URL
   kaggle_url = 'https://www.kaggle.com/datasets/techykajal/fakereal-news'
   #Download the kaggle dataset which corresponds to the Kaggle dataset URL above
   openD.download(kaggle_url)
```

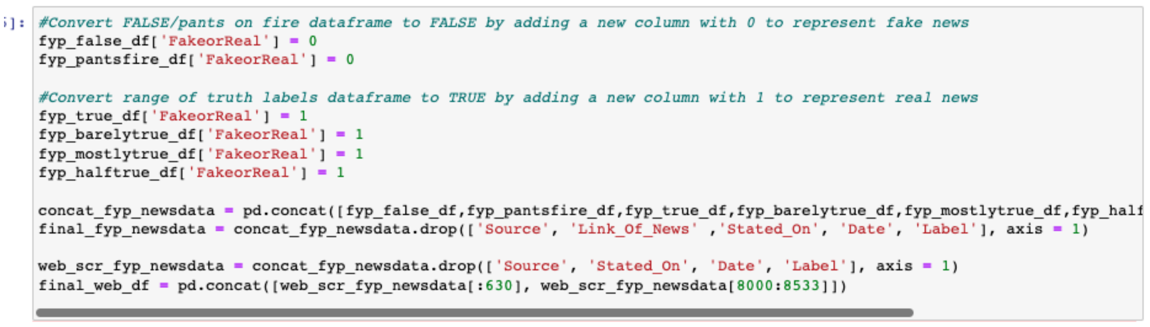
A.2



2.3.1: Dataset Statistics

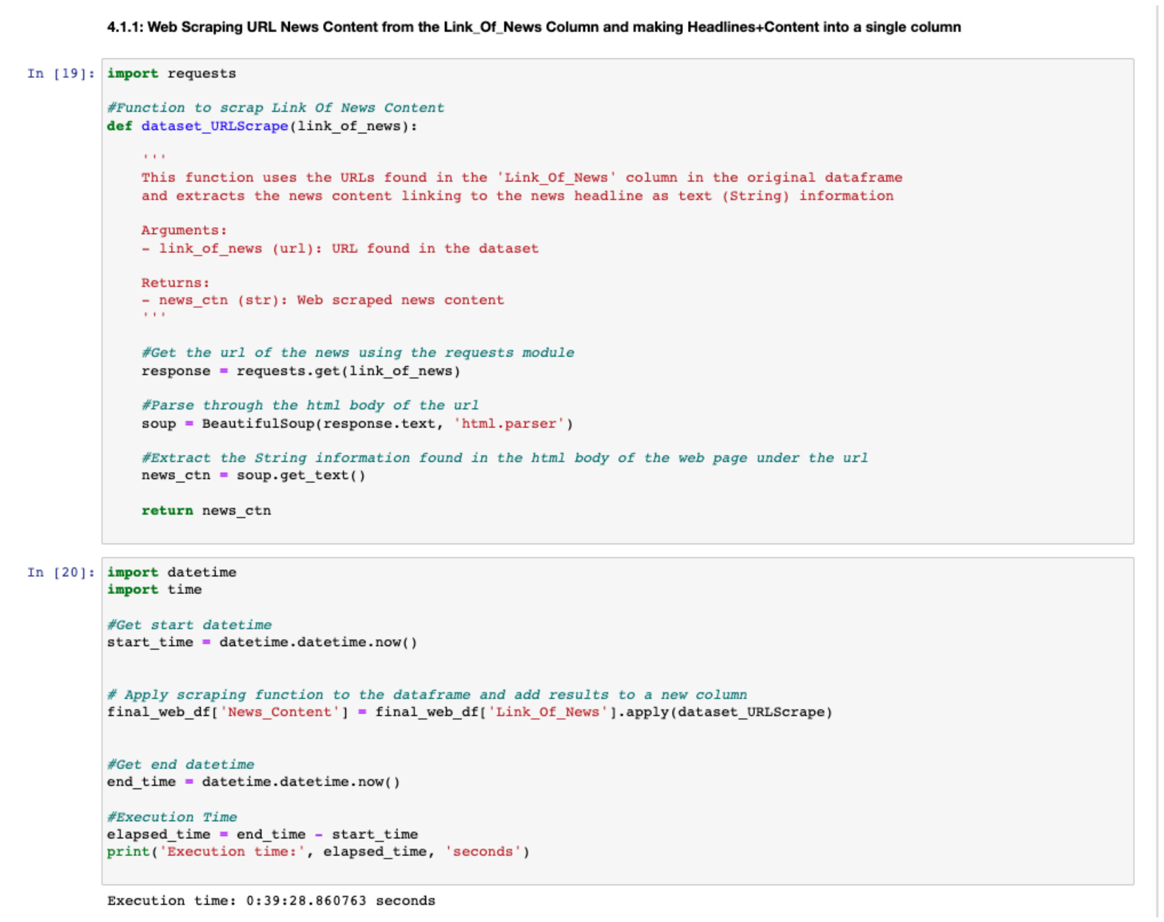
```
In [5]: #News Dataset Dataframe Information
        fyp_news_data.info()
        memory usage: 407.0+ KB

In [6]: #News Dataset Data Statistics
        fyp_news_data.describe()
Out[6]:
```

A.3



```
]: #Convert FALSE/pants on fire dataframe to FALSE by adding a new column with 0 to represent fake news
   fyp_false_df['FakeorReal'] = 0
   fyp_pantsfire_df['FakeorReal'] = 0

   #Convert range of truth labels dataframe to TRUE by adding a new column with 1 to represent real news
   fyp_true_df['FakeorReal'] = 1
   fyp_barelytrue_df['FakeorReal'] = 1
   fyp_mostlytrue_df['FakeorReal'] = 1
   fyp_halftrue_df['FakeorReal'] = 1

   concat_fyp_newsdata = pd.concat([fyp_false_df,fyp_pantsfire_df,fyp_true_df,fyp_barelytrue_df,fyp_mostlytrue_df,fyp_half
   final_fyp_newsdata = concat_fyp_newsdata.drop(['Source', 'Link_Of_News' ,'Stated_On', 'Date', 'Label'], axis = 1)

   web_scr_fyp_newsdata = concat_fyp_newsdata.drop(['Source', 'Stated_On', 'Date', 'Label'], axis = 1)
   final_web_df = pd.concat([web_scr_fyp_newsdata[:630], web_scr_fyp_newsdata[8000:8533]])
```

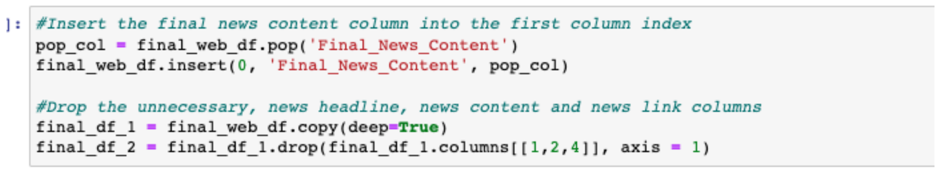
A.4



4.1.1: Web Scraping URL News Content from the Link_Of_News Column and making Headlines+Content into a single column

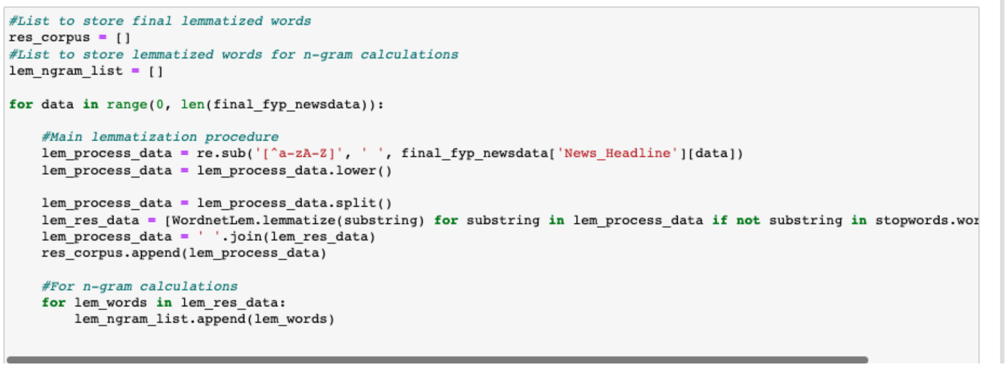
```
In [19]: import requests

         #Function to scrap Link Of News Content
         def dataset_URLScrape(link_of_news):

             '''
             This function uses the URLs found in the 'Link_Of_News' column in the original dataframe
             and extracts the news content linking to the news headline as text (String) information

             Arguments:
             - link_of_news (url): URL found in the dataset

             Returns:
             - news_ctn (str): Web scraped news content
             '''

             #Get the url of the news using the requests module
             response = requests.get(link_of_news)

             #Parse through the html body of the url
             soup = BeautifulSoup(response.text, 'html.parser')

             #Extract the String information found in the html body of the web page under the url
             news_ctn = soup.get_text()

             return news_ctn
```

```
In [20]: import datetime
         import time

         #Get start datetime
         start_time = datetime.datetime.now()


         # Apply scraping function to the dataframe and add results to a new column
         final_web_df['News_Content'] = final_web_df['Link_Of_News'].apply(dataset_URLScrape)


         #Get end datetime
         end_time = datetime.datetime.now()

         #Execution Time
         elapsed_time = end_time - start_time
         print('Execution time:', elapsed_time, 'seconds')

         Execution time: 0:39:28.860763 seconds
```



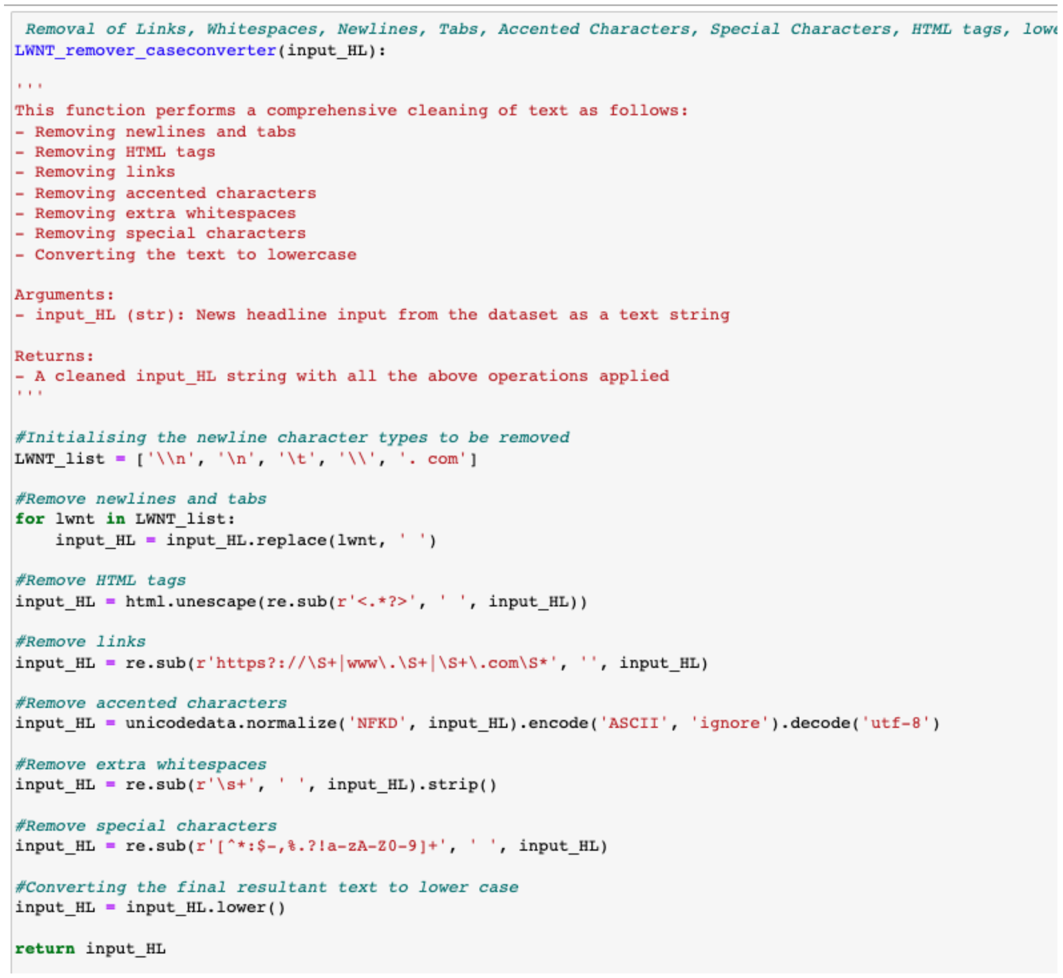
```
]: #Insert the final news content column into the first column index
   pop_col = final_web_df.pop('Final_News_Content')
   final_web_df.insert(0, 'Final_News_Content', pop_col)

   #Drop the unnecessary, news headline, news content and news link columns
   final_df_1 = final_web_df.copy(deep=True)
   final_df_2 = final_df_1.drop(final_df_1.columns[[1,2,4]], axis = 1)
```

A.5

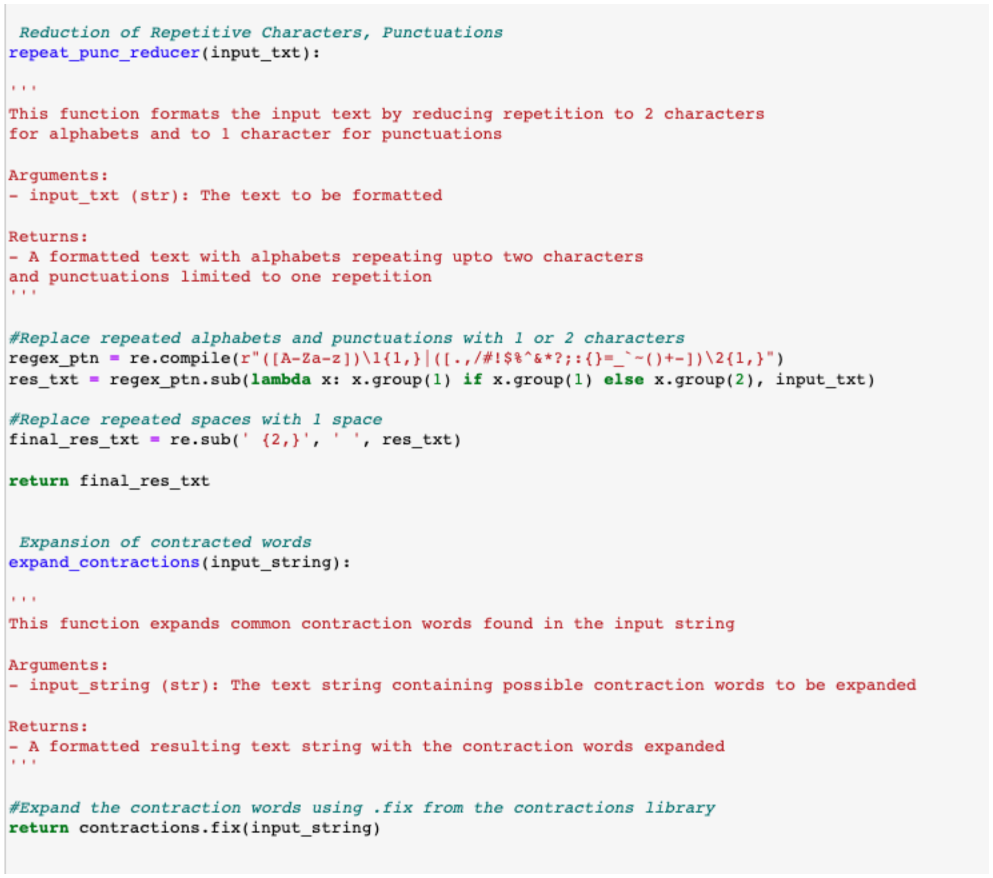


```
#List to store final lemmatized words
res_corpus = []
#List to store lemmatized words for n-gram calculations
lem_ngram_list = []

for data in range(0, len(final_fyp_newsdata)):

    #Main lemmatization procedure
    lem_process_data = re.sub('[^a-zA-Z]', ' ', final_fyp_newsdata['News_Headline'][data])
    lem_process_data = lem_process_data.lower()

    lem_process_data = lem_process_data.split()
    lem_res_data = [WordnetLem.lemmatize(substring) for substring in lem_process_data if not substring in stopwords.wor
    lem_process_data = ' '.join(lem_res_data)
    res_corpus.append(lem_process_data)

    #For n-gram calculations
    for lem_words in lem_res_data:
        lem_ngram_list.append(lem_words)
```

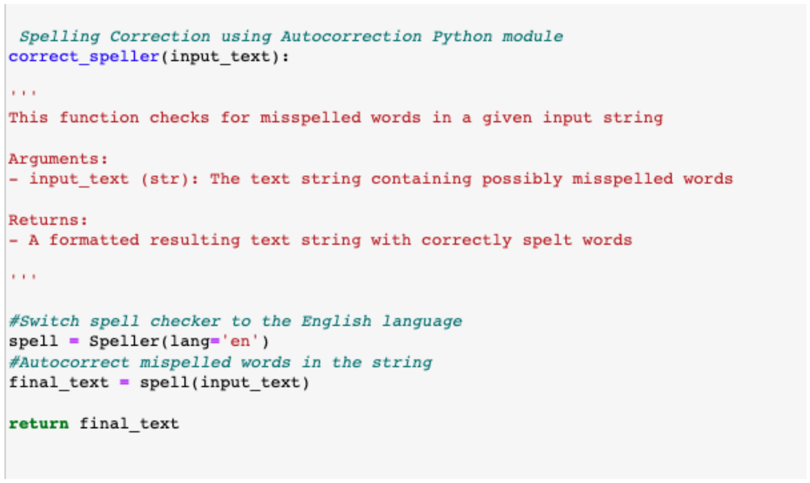
A.6



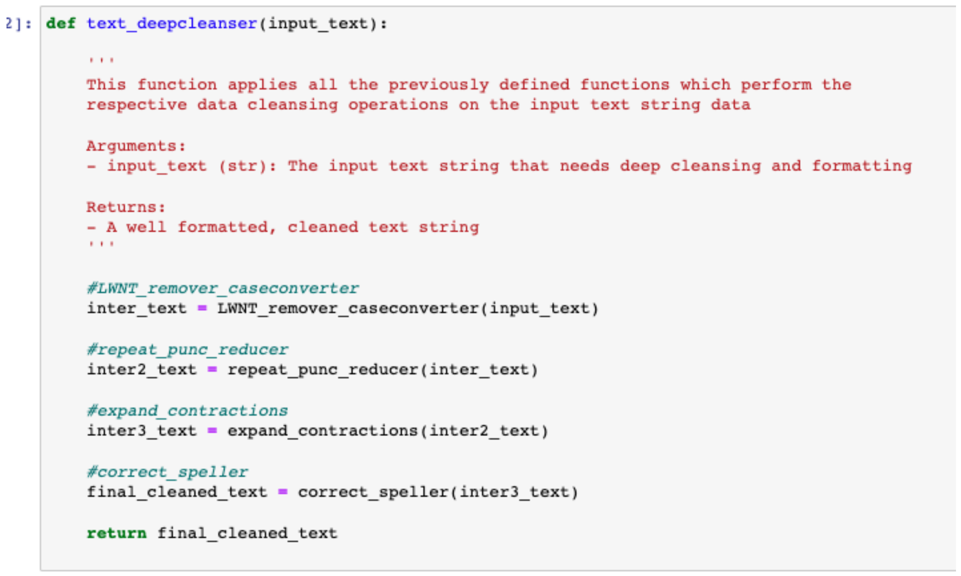
```
 Removal of Links, Whitespaces, Newlines, Tabs, Accented Characters, Special Characters, HTML tags, lowe
LWNT_remover_caseconverter(input_HL):

'''
This function performs a comprehensive cleaning of text as follows:
- Removing newlines and tabs
- Removing HTML tags
- Removing links
- Removing accented characters
- Removing extra whitespaces
- Removing special characters
- Converting the text to lowercase

Arguments:
- input_HL (str): News headline input from the dataset as a text string

Returns:
- A cleaned input_HL string with all the above operations applied
'''

#Initialising the newline character types to be removed
LWNT_list = ['\\n', '\n', '\t', '\\', '. com']

#Remove newlines and tabs
for lwnt in LWNT_list:
    input_HL = input_HL.replace(lwnt, ' ')

#Remove HTML tags
input_HL = html.unescape(re.sub(r'<.*?>', ' ', input_HL))

#Remove links
input_HL = re.sub(r'https?://\S+|www\.\S+|\S+\.com\S*', '', input_HL)

#Remove accented characters
input_HL = unicodedata.normalize('NFKD', input_HL).encode('ASCII', 'ignore').decode('utf-8')

#Remove extra whitespaces
input_HL = re.sub(r'\s+', ' ', input_HL).strip()

#Remove special characters
input_HL = re.sub(r'[^*:$-,%.?!a-zA-Z0-9]+', ' ', input_HL)

#Converting the final resultant text to lower case
input_HL = input_HL.lower()

return input_HL
```



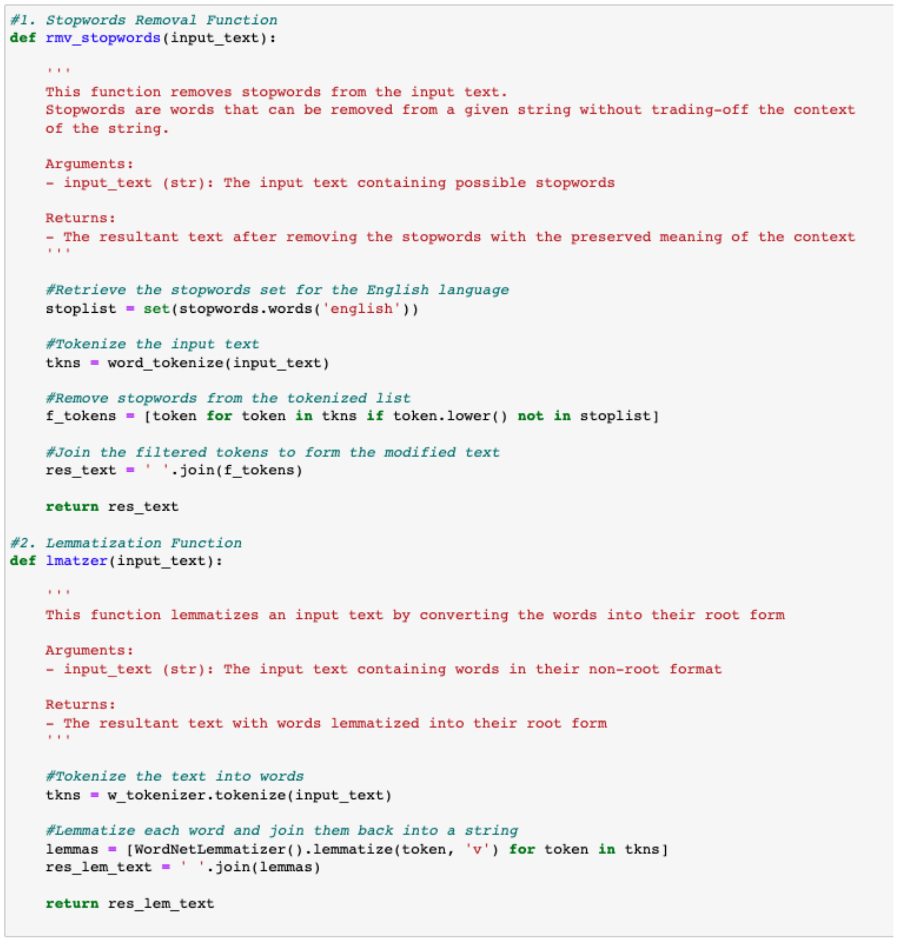
```
 Reduction of Repetitive Characters, Punctuations
repeat_punc_reducer(input_txt):

'''
This function formats the input text by reducing repetition to 2 characters
for alphabets and to 1 character for punctuations

Arguments:
- input_txt (str): The text to be formatted

Returns:
- A formatted text with alphabets repeating upto two characters
and punctuations limited to one repetition
'''

#Replace repeated alphabets and punctuations with 1 or 2 characters
regex_ptn = re.compile(r"([A-Za-z])\1{1,}|([.,/#!$%^&*?;:{}=_`~()+-])\2{1,}")
res_txt = regex_ptn.sub(lambda x: x.group(1) if x.group(1) else x.group(2), input_txt)

#Replace repeated spaces with 1 space
final_res_txt = re.sub(' {2,}', ' ', res_txt)

return final_res_txt


 Expansion of contracted words
expand_contractions(input_string):

'''
This function expands common contraction words found in the input string

Arguments:
- input_string (str): The text string containing possible contraction words to be expanded

Returns:
- A formatted resulting text string with the contraction words expanded
'''

#Expand the contraction words using .fix from the contractions library
return contractions.fix(input_string)
```



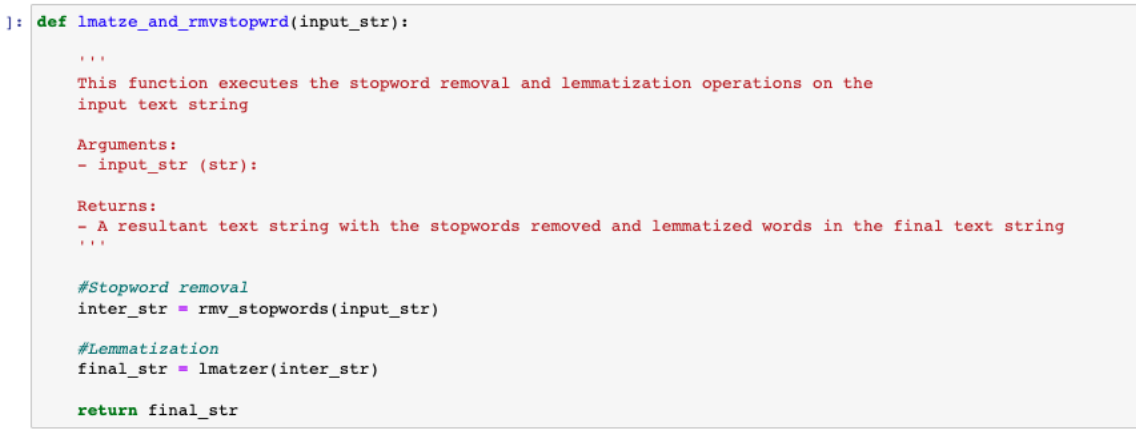
```
 Spelling Correction using Autocorrection Python module
correct_speller(input_text):

'''
This function checks for misspelled words in a given input string

Arguments:
- input_text (str): The text string containing possibly misspelled words

Returns:
- A formatted resulting text string with correctly spelt words

'''

#Switch spell checker to the English language
spell = Speller(lang='en')
#Autocorrect mispelled words in the string
final_text = spell(input_text)

return final_text
```



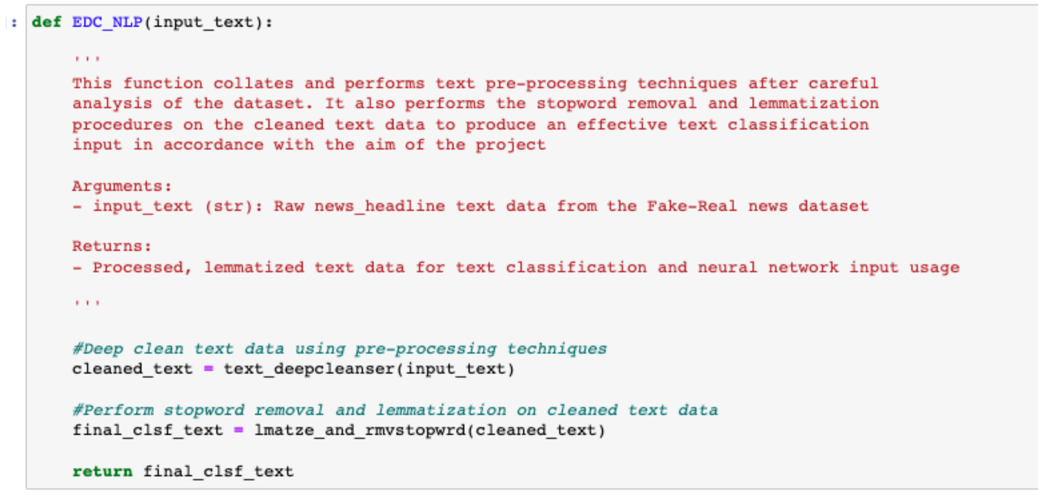
```
2]: def text_deepcleanser(input_text):

        '''
        This function applies all the previously defined functions which perform the
        respective data cleansing operations on the input text string data

        Arguments:
        - input_text (str): The input text string that needs deep cleansing and formatting

        Returns:
        - A well formatted, cleaned text string
        '''

        #LWNT_remover_caseconverter
        inter_text = LWNT_remover_caseconverter(input_text)

        #repeat_punc_reducer
        inter2_text = repeat_punc_reducer(inter_text)

        #expand_contractions
        inter3_text = expand_contractions(inter2_text)

        #correct_speller
        final_cleaned_text = correct_speller(inter3_text)

        return final_cleaned_text
```



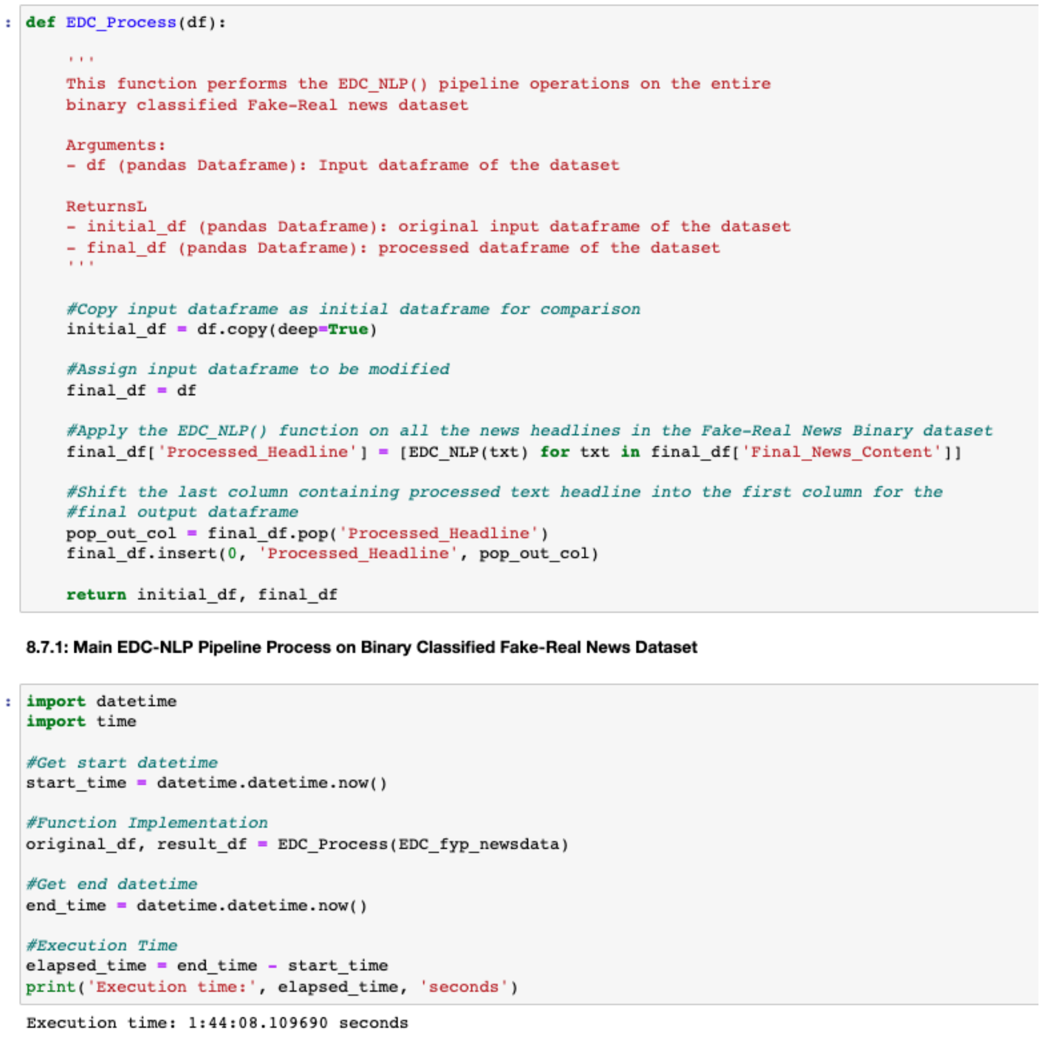
```
#1. Stopwords Removal Function
def rmv_stopwords(input_text):

    '''
    This function removes stopwords from the input text.
    Stopwords are words that can be removed from a given string without trading-off the context
    of the string.

    Arguments:
    - input_text (str): The input text containing possible stopwords

    Returns:
    - The resultant text after removing the stopwords with the preserved meaning of the context
    '''

    #Retrieve the stopwords set for the English language
    stoplist = set(stopwords.words('english'))

    #Tokenize the input text
    tkns = word_tokenize(input_text)

    #Remove stopwords from the tokenized list
    f_tokens = [token for token in tkns if token.lower() not in stoplist]

    #Join the filtered tokens to form the modified text
    res_text = ' '.join(f_tokens)

    return res_text

#2. Lemmatization Function
def lmatzer(input_text):

    '''
    This function lemmatizes an input text by converting the words into their root form

    Arguments:
    - input_text (str): The input text containing words in their non-root format

    Returns:
    - The resultant text with words lemmatized into their root form
    '''

    #Tokenize the text into words
    tkns = w_tokenizer.tokenize(input_text)

    #Lemmatize each word and join them back into a string
    lemmas = [WordNetLemmatizer().lemmatize(token, 'v') for token in tkns]
    res_lem_text = ' '.join(lemmas)

    return res_lem_text
```



```
]: def lmatze_and_rmvstopwrd(input_str):

       '''
       This function executes the stopword removal and lemmatization operations on the
       input text string

       Arguments:
       - input_str (str):

       Returns:
       - A resultant text string with the stopwords removed and lemmatized words in the final text string
       '''

       #Stopword removal
       inter_str = rmv_stopwords(input_str)

       #Lemmatization
       final_str = lmatzer(inter_str)

       return final_str
```



```
: def EDC_NLP(input_text):

      '''
      This function collates and performs text pre-processing techniques after careful
      analysis of the dataset. It also performs the stopword removal and lemmatization
      procedures on the cleaned text data to produce an effective text classification
      input in accordance with the aim of the project

      Arguments:
      - input_text (str): Raw news_headline text data from the Fake-Real news dataset

      Returns:
      - Processed, lemmatized text data for text classification and neural network input usage

      '''

      #Deep clean text data using pre-processing techniques
      cleaned_text = text_deepcleanser(input_text)

      #Perform stopword removal and lemmatization on cleaned text data
      final_clsf_text = lmatze_and_rmvstopwrd(cleaned_text)

      return final_clsf_text
```



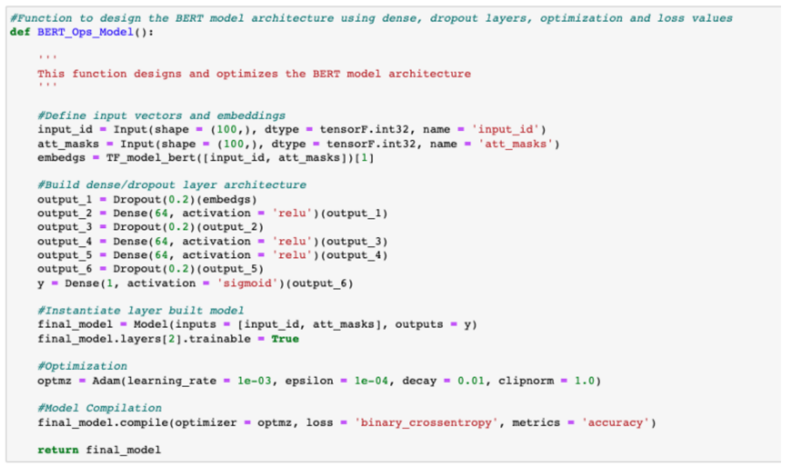
```
: def EDC_Process(df):

      '''
      This function performs the EDC_NLP() pipeline operations on the entire
      binary classified Fake-Real news dataset

      Arguments:
      - df (pandas Dataframe): Input dataframe of the dataset

      ReturnsL
      - initial_df (pandas Dataframe): original input dataframe of the dataset
      - final_df (pandas Dataframe): processed dataframe of the dataset
      '''

      #Copy input dataframe as initial dataframe for comparison
      initial_df = df.copy(deep=True)

      #Assign input dataframe to be modified
      final_df = df

      #Apply the EDC_NLP() function on all the news headlines in the Fake-Real News Binary dataset
      final_df['Processed_Headline'] = [EDC_NLP(txt) for txt in final_df['Final_News_Content']]

      #Shift the last column containing processed text headline into the first column for the
      #final output dataframe
      pop_out_col = final_df.pop('Processed_Headline')
      final_df.insert(0, 'Processed_Headline', pop_out_col)

      return initial_df, final_df
```

8.7.1: Main EDC-NLP Pipeline Process on Binary Classified Fake-Real News Dataset

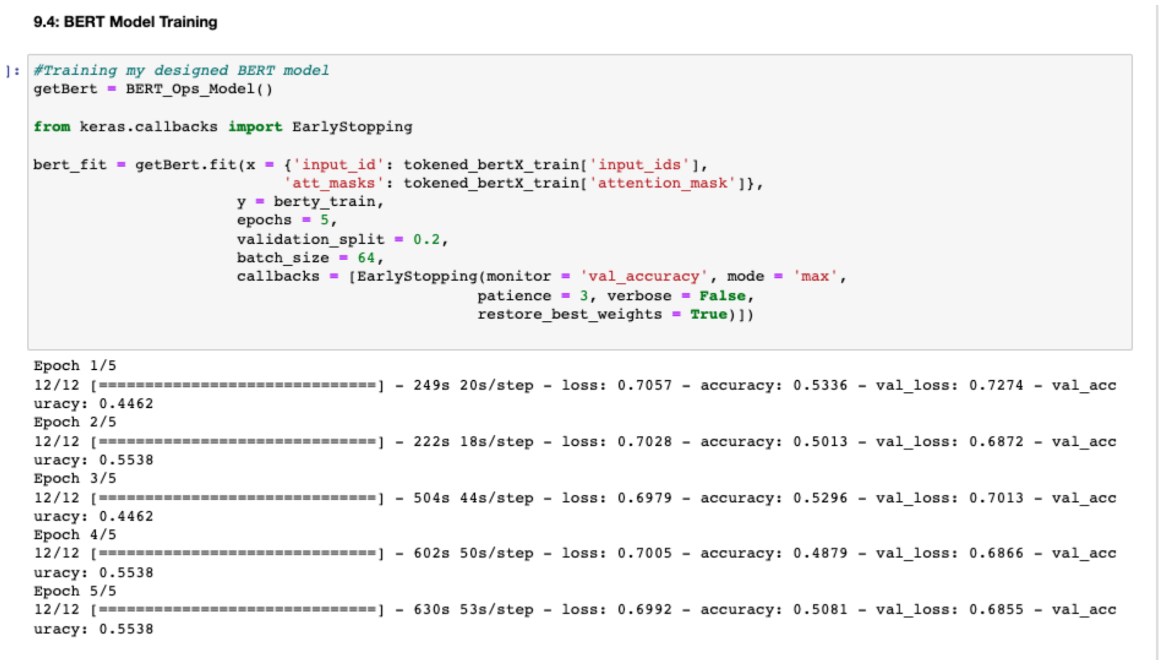
```
: import datetime
  import time

  #Get start datetime
  start_time = datetime.datetime.now()

  #Function Implementation
  original_df, result_df = EDC_Process(EDC_fyp_newsdata)

  #Get end datetime
  end_time = datetime.datetime.now()

  #Execution Time
  elapsed_time = end_time - start_time
  print('Execution time:', elapsed_time, 'seconds')
```

```
Execution time: 1:44:08.109690 seconds
```

A.7



```
#Function to design the BERT model architecture using dense, dropout layers, optimization and loss values
def BERT_Ops_Model():

    '''
    This function designs and optimizes the BERT model architecture
    '''

    #Define input vectors and embeddings
    input_id = Input(shape = (100,), dtype = tensorF.int32, name = 'input_id')
    att_masks = Input(shape = (100,), dtype = tensorF.int32, name = 'att_masks')
    embedgs = TF_model_bert([input_id, att_masks])[1]

    #Build dense/dropout layer architecture
    output_1 = Dropout(0.2)(embedgs)
    output_2 = Dense(64, activation = 'relu')(output_1)
    output_3 = Dropout(0.2)(output_2)
    output_4 = Dense(64, activation = 'relu')(output_3)
    output_5 = Dense(64, activation = 'relu')(output_4)
    output_6 = Dropout(0.2)(output_5)
    y = Dense(1, activation = 'sigmoid')(output_6)

    #Instantiate layer built model
    final_model = Model(inputs = [input_id, att_masks], outputs = y)
    final_model.layers[2].trainable = True

    #Optimization
    optmz = Adam(learning_rate = 1e-03, epsilon = 1e-04, decay = 0.01, clipnorm = 1.0)

    #Model Compilation
    final_model.compile(optimizer = optmz, loss = 'binary_crossentropy', metrics = 'accuracy')

    return final_model
```

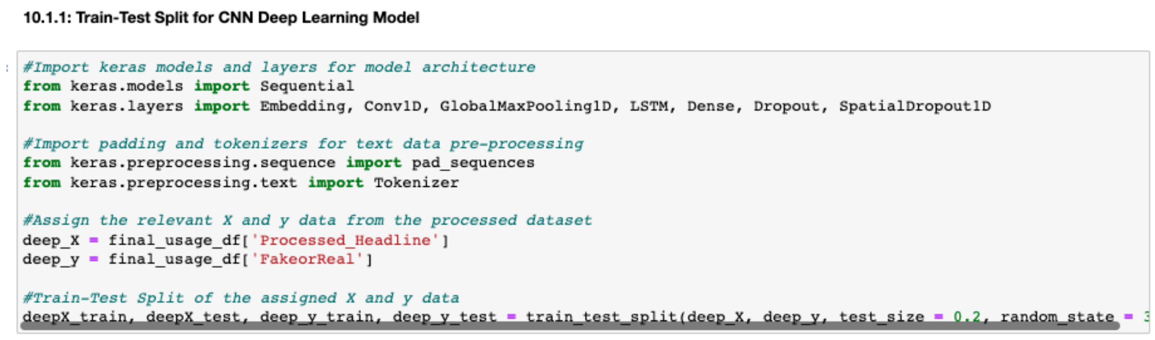


9.4: BERT Model Training

```
]: #Training my designed BERT model
getBert = BERT_Ops_Model()

from keras.callbacks import EarlyStopping

bert_fit = getBert.fit(x = {'input_id': tokened_bertX_train['input_ids'],
                            'att_masks': tokened_bertX_train['attention_mask']},
                       y = berty_train,
                       epochs = 5,
                       validation_split = 0.2,
                       batch_size = 64,
                       callbacks = [EarlyStopping(monitor = 'val_accuracy', mode = 'max',
                                                  patience = 3, verbose = False,
                                                  restore_best_weights = True)])
```

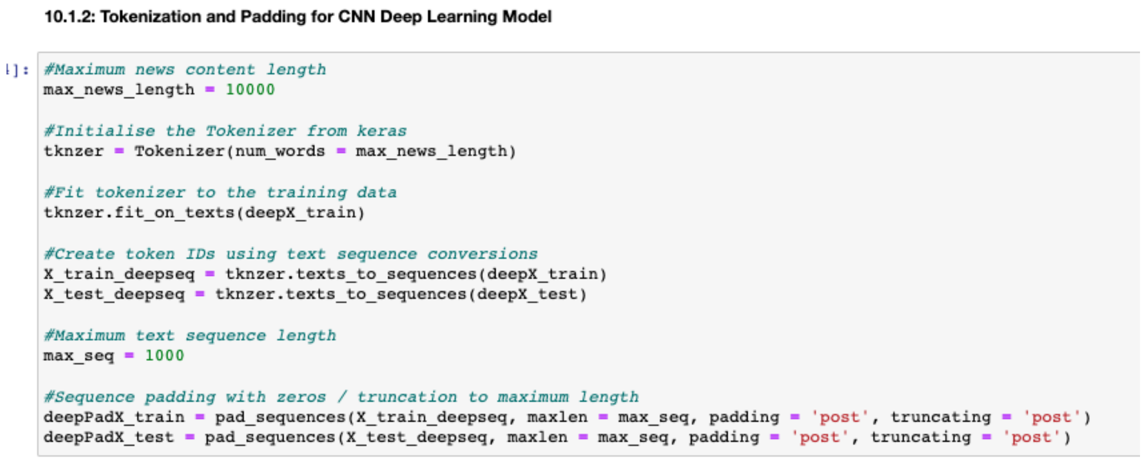
```
Epoch 1/5
12/12 [==============================] - 249s 20s/step - loss: 0.7057 - accuracy: 0.5336 - val_loss: 0.7274 - val_acc
uracy: 0.4462
Epoch 2/5
12/12 [==============================] - 222s 18s/step - loss: 0.7028 - accuracy: 0.5013 - val_loss: 0.6872 - val_acc
uracy: 0.5538
Epoch 3/5
12/12 [==============================] - 504s 44s/step - loss: 0.6979 - accuracy: 0.5296 - val_loss: 0.7013 - val_acc
uracy: 0.4462
Epoch 4/5
12/12 [==============================] - 602s 50s/step - loss: 0.7005 - accuracy: 0.4879 - val_loss: 0.6866 - val_acc
uracy: 0.5538
Epoch 5/5
12/12 [==============================] - 630s 53s/step - loss: 0.6992 - accuracy: 0.5081 - val_loss: 0.6855 - val_acc
uracy: 0.5538
```

A.8



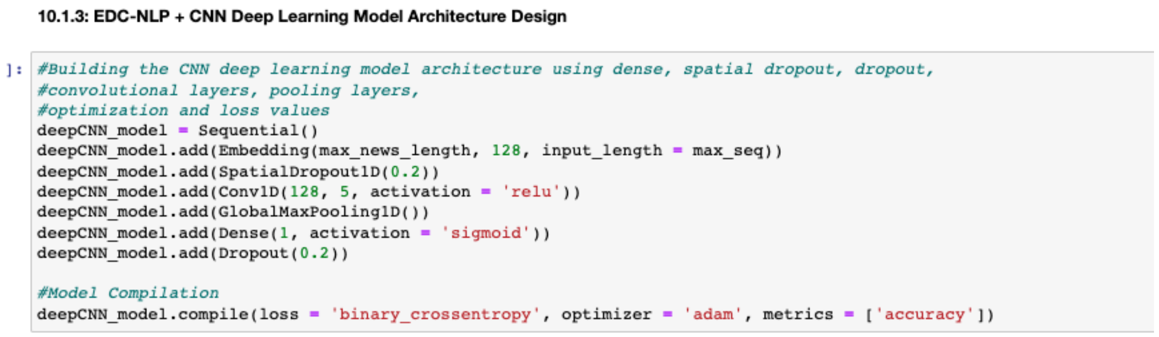
10.1.1: Train-Test Split for CNN Deep Learning Model

```
#Import keras models and layers for model architecture
from keras.models import Sequential
from keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, LSTM, Dense, Dropout, SpatialDropout1D

#Import padding and tokenizers for text data pre-processing
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer

#Assign the relevant X and y data from the processed dataset
deep_X = final_usage_df['Processed_Headline']
deep_y = final_usage_df['FakeorReal']

#Train-Test Split of the assigned X and y data
deepX_train, deepX_test, deep_y_train, deep_y_test = train_test_split(deep_X, deep_y, test_size = 0.2, random_state = 3
```



10.1.2: Tokenization and Padding for CNN Deep Learning Model

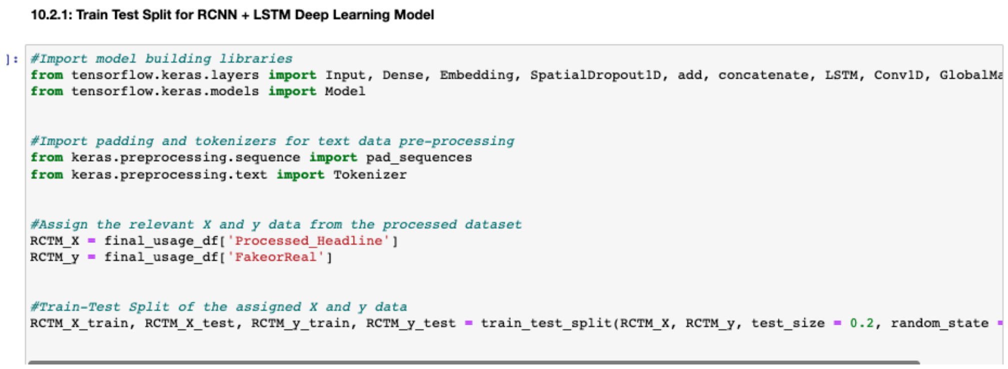
```
]: #Maximum news content length
max_news_length = 10000

#Initialise the Tokenizer from keras
tknzer = Tokenizer(num_words = max_news_length)

#Fit tokenizer to the training data
tknzer.fit_on_texts(deepX_train)

#Create token IDs using text sequence conversions
X_train_deepseq = tknzer.texts_to_sequences(deepX_train)
X_test_deepseq = tknzer.texts_to_sequences(deepX_test)

#Maximum text sequence length
max_seq = 1000

#Sequence padding with zeros / truncation to maximum length
deepPadX_train = pad_sequences(X_train_deepseq, maxlen = max_seq, padding = 'post', truncating = 'post')
deepPadX_test = pad_sequences(X_test_deepseq, maxlen = max_seq, padding = 'post', truncating = 'post')
```



10.1.3: EDC-NLP + CNN Deep Learning Model Architecture Design

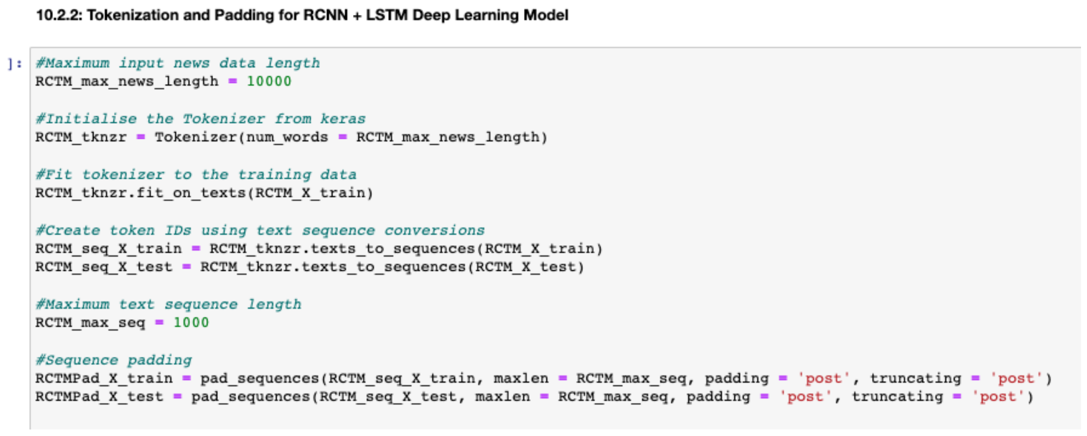
```
]: #Building the CNN deep learning model architecture using dense, spatial dropout, dropout,
#convolutional layers, pooling layers,
#optimization and loss values
deepCNN_model = Sequential()
deepCNN_model.add(Embedding(max_news_length, 128, input_length = max_seq))
deepCNN_model.add(SpatialDropout1D(0.2))
deepCNN_model.add(Conv1D(128, 5, activation = 'relu'))
deepCNN_model.add(GlobalMaxPooling1D())
deepCNN_model.add(Dense(1, activation = 'sigmoid'))
deepCNN_model.add(Dropout(0.2))

#Model Compilation
deepCNN_model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

A.9



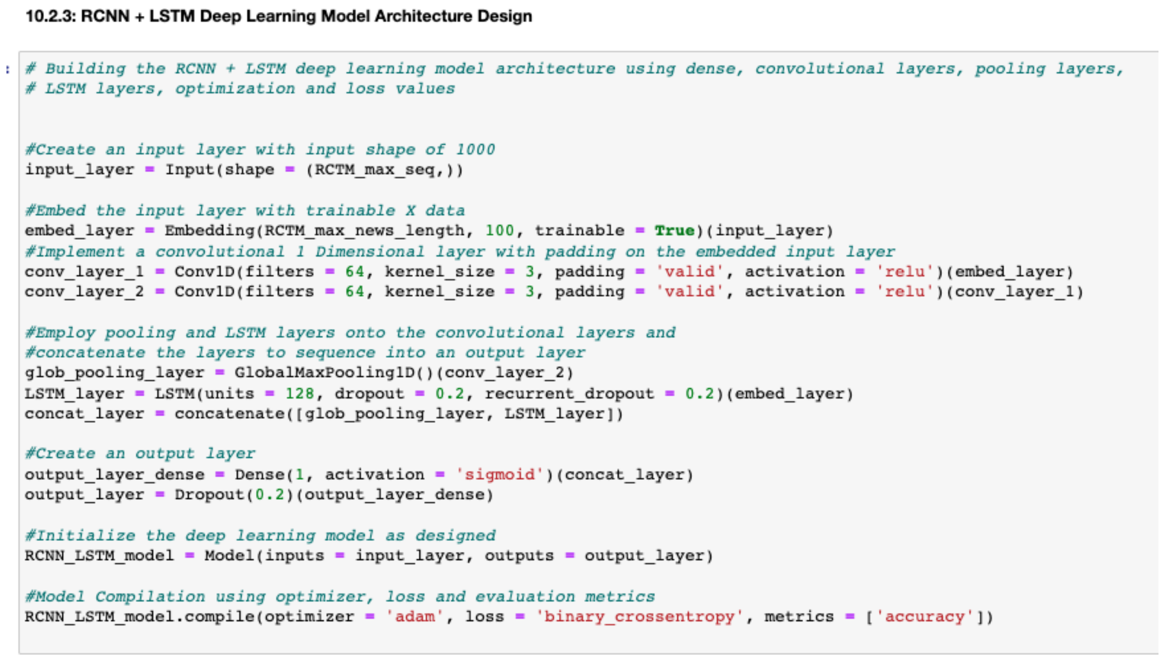
10.2.1: Train Test Split for RCNN + LSTM Deep Learning Model

```
]: #Import model building libraries
from tensorflow.keras.layers import Input, Dense, Embedding, SpatialDropout1D, add, concatenate, LSTM, Conv1D, GlobalMa
from tensorflow.keras.models import Model


#Import padding and tokenizers for text data pre-processing
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer


#Assign the relevant X and y data from the processed dataset
RCTM_X = final_usage_df['Processed_Headline']
RCTM_y = final_usage_df['FakeorReal']


#Train-Test Split of the assigned X and y data
RCTM_X_train, RCTM_X_test, RCTM_y_train, RCTM_y_test = train_test_split(RCTM_X, RCTM_y, test_size = 0.2, random_state =
```



10.2.2: Tokenization and Padding for RCNN + LSTM Deep Learning Model

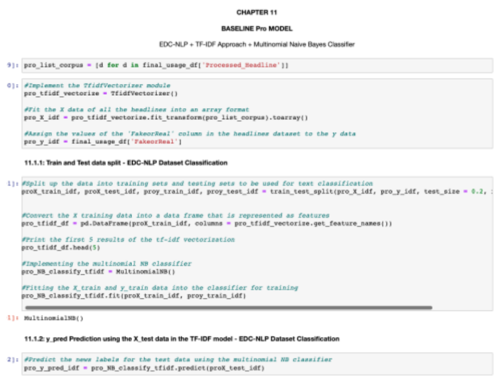
```
]: #Maximum input news data length
RCTM_max_news_length = 10000

#Initialise the Tokenizer from keras
RCTM_tknzr = Tokenizer(num_words = RCTM_max_news_length)

#Fit tokenizer to the training data
RCTM_tknzr.fit_on_texts(RCTM_X_train)

#Create token IDs using text sequence conversions
RCTM_seq_X_train = RCTM_tknzr.texts_to_sequences(RCTM_X_train)
RCTM_seq_X_test = RCTM_tknzr.texts_to_sequences(RCTM_X_test)

#Maximum text sequence length
RCTM_max_seq = 1000

#Sequence padding
RCTMPad_X_train = pad_sequences(RCTM_seq_X_train, maxlen = RCTM_max_seq, padding = 'post', truncating = 'post')
RCTMPad_X_test = pad_sequences(RCTM_seq_X_test, maxlen = RCTM_max_seq, padding = 'post', truncating = 'post')
```



10.2.3: RCNN + LSTM Deep Learning Model Architecture Design

```
: # Building the RCNN + LSTM deep learning model architecture using dense, convolutional layers, pooling layers,
# LSTM layers, optimization and loss values


#Create an input layer with input shape of 1000
input_layer = Input(shape = (RCTM_max_seq,))

#Embed the input layer with trainable X data
embed_layer = Embedding(RCTM_max_news_length, 100, trainable = True)(input_layer)
#Implement a convolutional 1 Dimensional layer with padding on the embedded input layer
conv_layer_1 = Conv1D(filters = 64, kernel_size = 3, padding = 'valid', activation = 'relu')(embed_layer)
conv_layer_2 = Conv1D(filters = 64, kernel_size = 3, padding = 'valid', activation = 'relu')(conv_layer_1)

#Employ pooling and LSTM layers onto the convolutional layers and
#concatenate the layers to sequence into an output layer
glob_pooling_layer = GlobalMaxPooling1D()(conv_layer_2)
LSTM_layer = LSTM(units = 128, dropout = 0.2, recurrent_dropout = 0.2)(embed_layer)
concat_layer = concatenate([glob_pooling_layer, LSTM_layer])

#Create an output layer
output_layer_dense = Dense(1, activation = 'sigmoid')(concat_layer)
output_layer = Dropout(0.2)(output_layer_dense)

#Initialize the deep learning model as designed
RCNN_LSTM_model = Model(inputs = input_layer, outputs = output_layer)

#Model Compilation using optimizer, loss and evaluation metrics
RCNN_LSTM_model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

A10



CHAPTER 11

BASELINE Pro MODEL

EDC-NLP + TF-IDF Approach + Multinomial Naive Bayes Classifier

```
9]: pro_list_corpus = [d for d in final_usage_df['Processed_Headline']]
```

```
0]: #Implement the TfidfVectorizer module
pro_tfidf_vectorize = TfidfVectorizer()

#Fit the X data of all the headlines into an array format
pro_X_idf = pro_tfidf_vectorize.fit_transform(pro_list_corpus).toarray()

#Assign the values of the 'FakeorReal' column in the headlines dataset to the y data
pro_y_idf = final_usage_df['FakeorReal']
```

11.1.1: Train and Test data split - EDC-NLP Dataset Classification

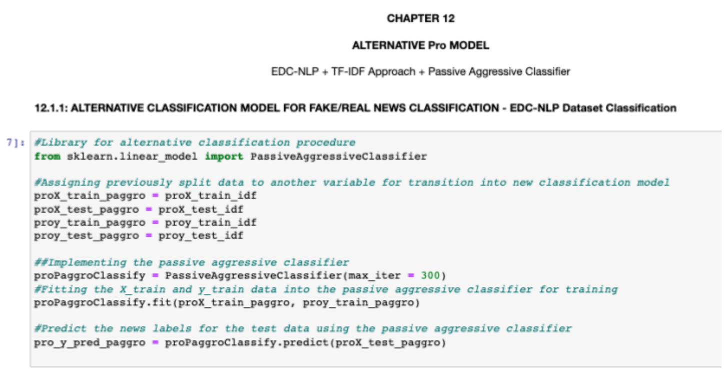
```
1]: #Split up the data into training sets and testing sets to be used for text classification
proX_train_idf, proX_test_idf, proy_train_idf, proy_test_idf = train_test_split(pro_X_idf, pro_y_idf, test_size = 0.2, :


#Convert the X training data into a data frame that is represented as features
pro_tfidf_df = pd.DataFrame(proX_train_idf, columns = pro_tfidf_vectorize.get_feature_names())

#Print the first 5 results of the tf-idf vectorization
pro_tfidf_df.head(5)

#Implementing the multinomial NB classifier
pro_NB_classify_tfidf = MultinomialNB()

#Fitting the X_train and y_train data into the classifier for training
pro_NB_classify_tfidf.fit(proX_train_idf, proy_train_idf)
```

```
1]: MultinomialNB()
```

11.1.2: y_pred Prediction using the X_test data in the TF-IDF model - EDC-NLP Dataset Classification

```
2]: #Predict the news labels for the test data using the multinomial NB classifier
pro_y_pred_idf = pro_NB_classify_tfidf.predict(proX_test_idf)
```

A11



CHAPTER 12

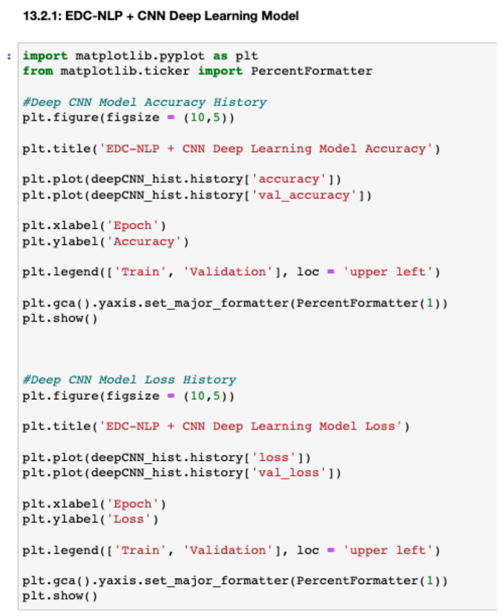
ALTERNATIVE Pro MODEL

EDC-NLP + TF-IDF Approach + Passive Aggressive Classifier

12.1.1: ALTERNATIVE CLASSIFICATION MODEL FOR FAKE/REAL NEWS CLASSIFICATION - EDC-NLP Dataset Classification

```
7]: #Library for alternative classification procedure
    from sklearn.linear_model import PassiveAggressiveClassifier

    #Assigning previously split data to another variable for transition into new classification model
    proX_train_paggro = proX_train_idf
    proX_test_paggro = proX_test_idf
    proy_train_paggro = proy_train_idf
    proy_test_paggro = proy_test_idf

    ##Implementing the passive aggressive classifier
    proPaggroClassify = PassiveAggressiveClassifier(max_iter = 300)
    #Fitting the X_train and y_train data into the passive aggressive classifier for training
    proPaggroClassify.fit(proX_train_paggro, proy_train_paggro)

    #Predict the news labels for the test data using the passive aggressive classifier
    pro_y_pred_paggro = proPaggroClassify.predict(proX_test_paggro)
```

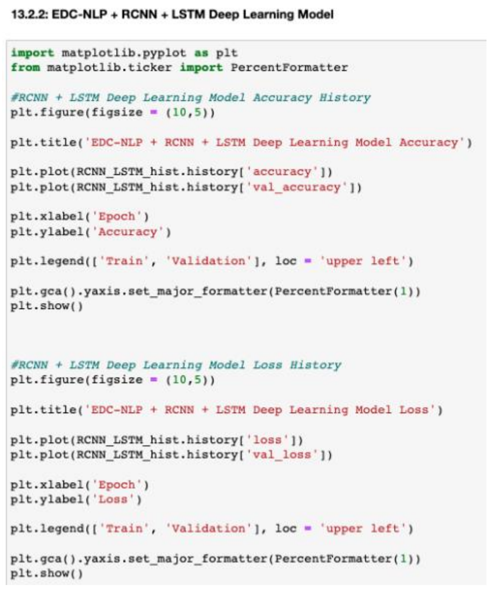
A12



13.2.1: EDC-NLP + CNN Deep Learning Model

```
: import matplotlib.pyplot as plt
  from matplotlib.ticker import PercentFormatter

  #Deep CNN Model Accuracy History
  plt.figure(figsize = (10,5))

  plt.title('EDC-NLP + CNN Deep Learning Model Accuracy')

  plt.plot(deepCNN_hist.history['accuracy'])
  plt.plot(deepCNN_hist.history['val_accuracy'])

  plt.xlabel('Epoch')
  plt.ylabel('Accuracy')

  plt.legend(['Train', 'Validation'], loc = 'upper left')

  plt.gca().yaxis.set_major_formatter(PercentFormatter(1))
  plt.show()


  #Deep CNN Model Loss History
  plt.figure(figsize = (10,5))

  plt.title('EDC-NLP + CNN Deep Learning Model Loss')

  plt.plot(deepCNN_hist.history['loss'])
  plt.plot(deepCNN_hist.history['val_loss'])

  plt.xlabel('Epoch')
  plt.ylabel('Loss')

  plt.legend(['Train', 'Validation'], loc = 'upper left')

  plt.gca().yaxis.set_major_formatter(PercentFormatter(1))
  plt.show()
```

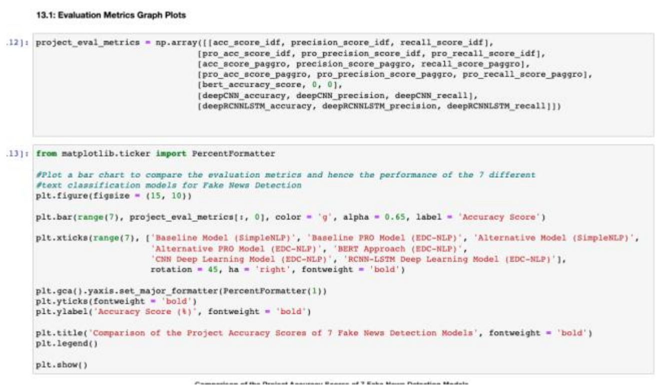
A13



13.2.2: EDC-NLP + RCNN + LSTM Deep Learning Model

```
import matplotlib.pyplot as plt
from matplotlib.ticker import PercentFormatter

#RCNN + LSTM Deep Learning Model Accuracy History
plt.figure(figsize = (10,5))

plt.title('EDC-NLP + RCNN + LSTM Deep Learning Model Accuracy')

plt.plot(RCNN_LSTM_hist.history['accuracy'])
plt.plot(RCNN_LSTM_hist.history['val_accuracy'])

plt.xlabel('Epoch')
plt.ylabel('Accuracy')

plt.legend(['Train', 'Validation'], loc = 'upper left')

plt.gca().yaxis.set_major_formatter(PercentFormatter(1))
plt.show()


#RCNN + LSTM Deep Learning Model Loss History
plt.figure(figsize = (10,5))

plt.title('EDC-NLP + RCNN + LSTM Deep Learning Model Loss')

plt.plot(RCNN_LSTM_hist.history['loss'])
plt.plot(RCNN_LSTM_hist.history['val_loss'])

plt.xlabel('Epoch')
plt.ylabel('Loss')

plt.legend(['Train', 'Validation'], loc = 'upper left')

plt.gca().yaxis.set_major_formatter(PercentFormatter(1))
plt.show()
```

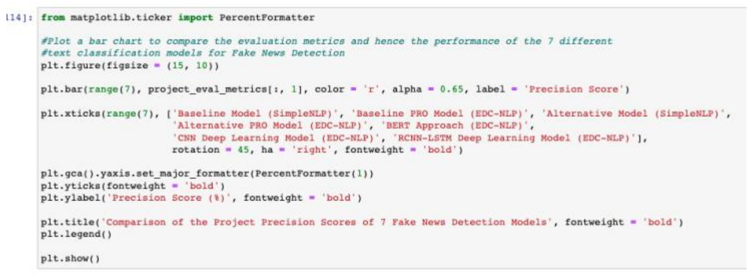
A14



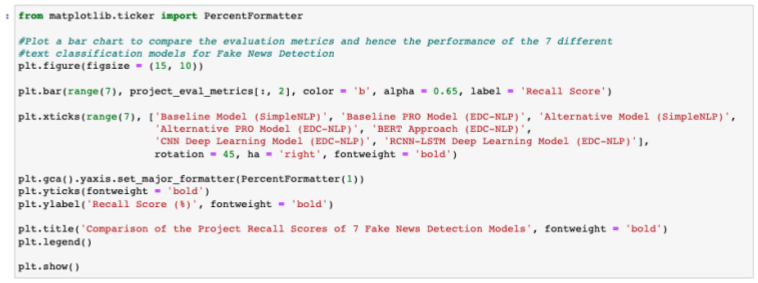
13.1: Evaluation Metrics Graph Plots

```
.12]: project_eval_metrics = np.array([[acc_score_idf, precision_score_idf, recall_score_idf],
                                      [pro_acc_score_idf, pro_precision_score_idf, pro_recall_score_idf],
                                      [acc_score_paggro, precision_score_paggro, recall_score_paggro],
                                      [pro_acc_score_paggro, pro_precision_score_paggro, pro_recall_score_paggro],
                                      [bert_accuracy_score, 0, 0],
                                      [deepCNN_accuracy, deepCNN_precision, deepCNN_recall],
                                      [deepRCNNLSTM_accuracy, deepRCNNLSTM_precision, deepRCNNLSTM_recall]])
```

```
.13]: from matplotlib.ticker import PercentFormatter

      #Plot a bar chart to compare the evaluation metrics and hence the performance of the 7 different
      #text classification models for Fake News Detection
      plt.figure(figsize = (15, 10))

      plt.bar(range(7), project_eval_metrics[:, 0], color = 'g', alpha = 0.65, label = 'Accuracy Score')

      plt.xticks(range(7), ['Baseline Model (SimpleNLP)', 'Baseline PRO Model (EDC-NLP)', 'Alternative Model (SimpleNLP)',
                            'Alternative PRO Model (EDC-NLP)', 'BERT Approach (EDC-NLP)',
                            'CNN Deep Learning Model (EDC-NLP)', 'RCNN-LSTM Deep Learning Model (EDC-NLP)'],
                            rotation = 45, ha = 'right', fontweight = 'bold')

      plt.gca().yaxis.set_major_formatter(PercentFormatter(1))
      plt.yticks(fontweight = 'bold')
      plt.ylabel('Accuracy Score (%)', fontweight = 'bold')

      plt.title('Comparison of the Project Accuracy Scores of 7 Fake News Detection Models', fontweight = 'bold')
      plt.legend()

      plt.show()
```



```
114]: from matplotlib.ticker import PercentFormatter

      #Plot a bar chart to compare the evaluation metrics and hence the performance of the 7 different
      #text classification models for Fake News Detection
      plt.figure(figsize = (15, 10))

      plt.bar(range(7), project_eval_metrics[:, 1], color = 'r', alpha = 0.65, label = 'Precision Score')

      plt.xticks(range(7), ['Baseline Model (SimpleNLP)', 'Baseline PRO Model (EDC-NLP)', 'Alternative Model (SimpleNLP)',
                            'Alternative PRO Model (EDC-NLP)', 'BERT Approach (EDC-NLP)',
                            'CNN Deep Learning Model (EDC-NLP)', 'RCNN-LSTM Deep Learning Model (EDC-NLP)'],
                            rotation = 45, ha = 'right', fontweight = 'bold')

      plt.gca().yaxis.set_major_formatter(PercentFormatter(1))
      plt.yticks(fontweight = 'bold')
      plt.ylabel('Precision Score (%)', fontweight = 'bold')

      plt.title('Comparison of the Project Precision Scores of 7 Fake News Detection Models', fontweight = 'bold')
      plt.legend()

      plt.show()
```



```
: from matplotlib.ticker import PercentFormatter

#Plot a bar chart to compare the evaluation metrics and hence the performance of the 7 different
#text classification models for Fake News Detection
plt.figure(figsize = (15, 10))

plt.bar(range(7), project_eval_metrics[:, 2], color = 'b', alpha = 0.65, label = 'Recall Score')

plt.xticks(range(7), ['Baseline Model (SimpleNLP)', 'Baseline PRO Model (EDC-NLP)', 'Alternative Model (SimpleNLP)',
                      'Alternative PRO Model (EDC-NLP)', 'BERT Approach (EDC-NLP)',
                      'CNN Deep Learning Model (EDC-NLP)', 'RCNN-LSTM Deep Learning Model (EDC-NLP)'],
                      rotation = 45, ha = 'right', fontweight = 'bold')

plt.gca().yaxis.set_major_formatter(PercentFormatter(1))
plt.yticks(fontweight = 'bold')
plt.ylabel('Recall Score (%)', fontweight = 'bold')

plt.title('Comparison of the Project Recall Scores of 7 Fake News Detection Models', fontweight = 'bold')
plt.legend()

plt.show()
```