Fraudulent News Detection on Social Media

MSc Research Project
Data Analytics

Archana Uday Mahajan
Student ID: x20198825

School of Computing
National College of Ireland

Supervisor: Prof. Taimur Hafeez

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Archana Uday Mahajan
Student ID:	X20198825
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Prof. Taimur Hameez
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual
Word Count:	713
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Archana Uday Mahajan
Date:	15th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Archana Uday Mahajan - x20198825

1 Introduction

Many experiments were performed for this research, and this configuration manual states and explains all the device specifications like hardware and software requirements to fulfill those experiments. It also specifies the programming language, libraries, and required packages. It then explains how the dataset was loaded, the data exploration, preprocessing, and how the models were implemented.

2 System Configuration

This section describes the Hardware and Software requirements to run this project.

2.1 Hardware Specification

Figure 1 shows the hardware requirements for this project, local machine was used for the implementation:

Hardware	Build
System	HP Laptop 14s - LAPTOP-3LJMQK9R
Processor	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
RAM	8.00 GB
System Type	64-bit operating system, x64-based processor

Figure 1: Hardware Specification

2.2 Software Specification

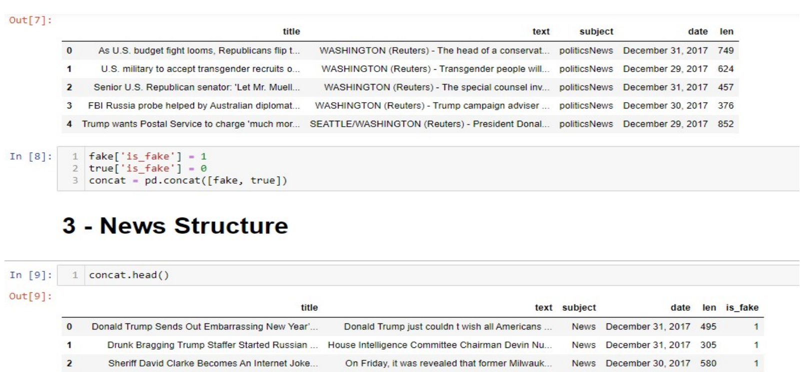
Windows 10 OS was used, all required libraries were installed, and Jupyter Notebook was used to implement the algorithms. Libraries like tqdm, sklearn, pyenchant, enchant, TensorFlow was used to run the algorithm. Matplotlib and Plotly were used for visualizations. Numpy, pandas, nltk, and re were used for preprocessing.

Libraries	Version
Python	3.8.8
numpy	1.20.1
pandas	1.2.4
plotly	5.5.0
tqdm	4.59.0
pyenchant	3.2.2
enchant	3.2.2
nltk	3.6.1
re	2021.4.4
sklearn	0.24.1
matplotlib	3.3.4
tensorflow	2.9.1
seaborn	0.11.1
transformers	4.21.0

Figure 2: Software Specification

3 Data Set Preparation

The data were selected and combined from various datasets from Kaggle.com; it was a total of two datasets, Fake and Real news, which were imported into the Jupyter Notebook and then concatenated into one single data frame. The final dataset had six features title, text, subject, date, len, and is_fake, where is_fake = 1 meant news is_fake and 0 meant that the news is real.



Out[7]:

	title	text	subject	date	len
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017	749
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017	624
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017	457
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017	376
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017	852

In [8]:

```
fake['is_fake'] = 1
true['is_fake'] = 0
concat = pd.concat([fake, true])
```

3 - News Structure

In [9]:

```
concat.head()
```

Out[9]:

	title	text	subject	date	len	is_fake
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	495	1
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	305	1
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	580	1
3	Trump Is So Obsessed He Even Has Obama's Name	On Christmas day, Donald Trump announced that	News	December 29, 2017	444	1

Figure 3: Data Set Preparation

4 Project Implementation

After the dataset concatenation, the data was explored barplots and pyplots, where it was seen that the data was biased toward fake news as seen in Figure 4.

```python
In [10]:
fake_ = concat[concat['is_fake']==1]
true_ = concat[concat['is_fake']==0]
fig = go.Figure()
fig.add_trace(go.Box(y=list(fake_['len']), name='Fake',
                marker_color = 'indianred'))
fig.add_trace(go.Box(y=list(true_['len']), name = 'Real',
                marker_color = 'lightseagreen'))

fig.update_layout({
        'plot_bgcolor': 'rgba(0, 0, 0, 0)',
        'paper_bgcolor': 'rgba(0, 0, 0, 0)',
        'title': 'Box plot',
})
fig.show()
```

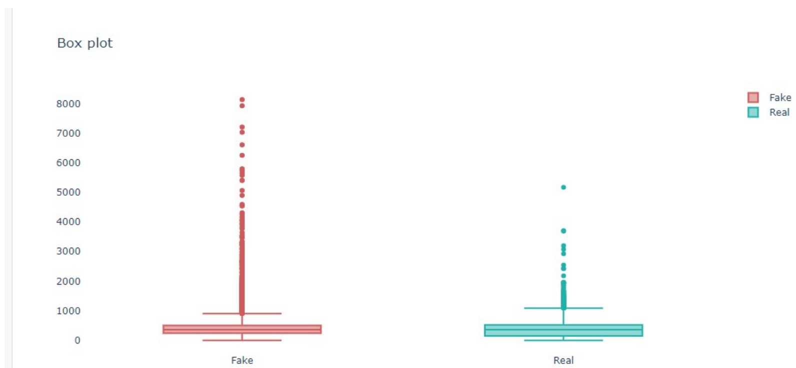


Figure 4: Data Set Bar Plot

4.1 Dataset Preprocessing

To reduce the bias of the dataset, below Figure 5 preprocessing steps were applied to it, using the nltk and re packages, the results of which are shown in Figure 6:

```python
In [17]:
import nltk
import re
tqdm.pandas()
def preprocess(df):
    stopwords = nltk.corpus.stopwords.words('english')
    df['text_pre'] = df['text']
    df['text_pre'] = df['text_pre'].progress_apply(lambda x : x.lower())
    df['text_pre'] = df['text_pre'].progress_apply(lambda x : x.split(" "))
    df['text_pre'] = df['text_pre'].progress_apply(lambda x : [item for item in x if item not in stopwords])
    df['text_pre'] = df['text_pre'].progress_apply(lambda x : " ".join(x))
#    df['text_pre'] = df['text_pre'].str.replace('@[^\s]+', "")
    df['text_pre'] = df['text_pre'].str.replace('https?:\/\/.*[\r\n]*', '')
    df['text_pre'] = df['text_pre'].str.normalize('NFKD').str.encode('ascii', errors='ignore').str.decode('utf-8')
    df['text_pre'] = df['text_pre'].str.replace('\d+', '')
    df['text_pre'] = df['text_pre'].str.replace('[^\w\s]', '')
    return df

fake = preprocess(fake)
true = preprocess(true)

100%|██████████| 23481/23481 [00:00<00:00, 87388.85it/s]
100%|██████████| 23481/23481 [00:02<00:00, 9092.05it/s]
100%|██████████| 23481/23481 [00:52<00:00, 447.12it/s]
100%|██████████| 23481/23481 [00:00<00:00, 40655.32it/s]
<ipython-input-17-66a3129cf2a0>:12: FutureWarning:
```

```
In [18]:
def unique_tokens2(df):
    unique_tokens = set()
    for text in tqdm(df['text_pre']):
        splited = text.split()
        for token in splited:
            unique_tokens.add(token)
    return unique_tokens

unique_tokens_fake2 = unique_tokens2(fake)
unique_tokens_true2 = unique_tokens2(true)

100%| | 23481/23481 [00:03<00:00, 6229.86it/s]
100%| | 21417/21417 [00:03<00:00, 6769.94it/s]

In [19]:
fig = go.Figure()
fig.add_trace(go.Bar(y=[len(unique_tokens_fake2), len(unique_tokens_true2)],
                     x=['Fake', 'True'],
                    marker_color='lightsalmon'
))
fig.update_layout({
        'plot_bgcolor': 'rgba(0, 0, 0, 0)',
        'paper_bgcolor': 'rgba(0, 0, 0, 0)',
})
fig.show()
```

Figure 5: Data Preprocessing

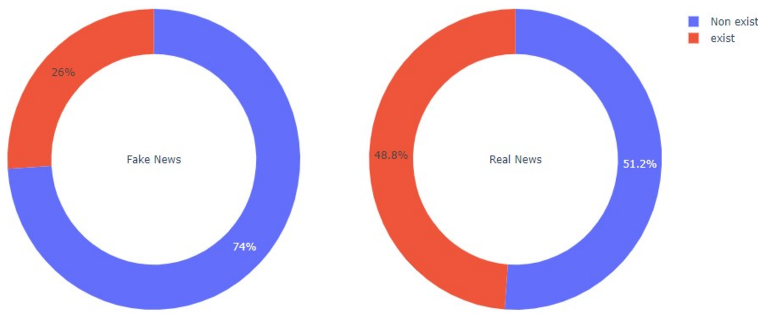


Figure 6: Words after Preprocessing

4.2 Feature Extraction

For creating an accurate model, relevant words were extracted using the chi2 hypothesis and used for the modeling shown in Figure 7, for which SelectKBest was used :

```
In [24]:
def get_wrong_tokens(list_):
    d = enchant.DictWithPWL("en_US", "vocab.txt")
    tokens = set()
    for token in tqdm(list_):
        if not d.check(token) and not d.check(token.capitalize()):
            tokens.add(token)
    return tokens

def get_top_n_words2(corpus, n=None, vocabulary=None):
    vec = CountVectorizer(vocabulary=vocabulary).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in     vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

wrong = get_wrong_tokens(unique_tokens_true2)
wrong_true = get_top_n_words2(true['text_pre'], n=100, vocabulary=wrong)
wrong = get_wrong_tokens(unique_tokens_fake2)
wrong_fake = get_top_n_words2(fake['text_pre'], n=100, vocabulary=wrong)
```

The next step was to model the word by topic, which was done using the LatentDirichletAllocation, a popular topic modelling technique, as shown in Figure 8:

```
new_list_words = [ seq[0] for seq in wrong_true ]
new_list_values = [ seq[1] for seq in wrong_true ]

fig = go.Figure()
fig.add_trace(go.Bar(y=new_list_values,
                          x=new_list_words,
                         marker_color='lightsalmon'
))
fig.update_layout({
        'plot_bgcolor': 'rgba(0, 0, 0, 0)',
        'paper_bgcolor': 'rgba(0, 0, 0, 0)',
        'title': 'Real chi2'
})
fig.show()
```

```
100%|| 78044/78044 [01:07<00:00, 1154.03it/s]
100%|| 160734/160734 [02:42<00:00, 989.90it/s]
```

```
In [25]:
new_list_words = [ seq[0] for seq in wrong_fake ]
new_list_values = [ seq[1] for seq in wrong_fake ]
fig = go.Figure()
fig.add_trace(go.Bar(y=new_list_values,
                          x=new_list_words,
                         marker_color='lightsalmon'
))
fig.update_layout({
        'plot_bgcolor': 'rgba(0, 0, 0, 0)',
        'paper_bgcolor': 'rgba(0, 0, 0, 0)',
        'title': 'Fake chi2'
})
fig.show()
```

Figure 7: Feature Extraction using chi2 hypothesis

```
In [30]:
from sklearn.decomposition import NMF, LatentDirichletAllocation
def topics(model, feature_names, no_top_words):
    dict_ = {}
    for topic_idx, topic in enumerate(model.components_):
        dict_[topic_idx] = [feature_names[i] for i in topic.argsort()[:-no_top_words - 1:-1]]
    return dict_
lda = LatentDirichletAllocation(random_state=42).fit(X)
topic_all = topics(lda, vectorizer.get_feature_names(), 15)
```

```
In [31]:
vectorizer_fake = CountVectorizer()
vectorizer_true = CountVectorizer()

X_fake = vectorizer_fake.fit_transform(fake['text_pre'])
X_true = vectorizer_true.fit_transform(true['text_pre'])

lda_fake = LatentDirichletAllocation(random_state=42, n_components=5).fit(X_fake)
lda_true = LatentDirichletAllocation(random_state=42, n_components=5).fit(X_true)

topic_true = topics(lda_true, vectorizer_true.get_feature_names(), 15)
topic_fake = topics(lda_fake, vectorizer_fake.get_feature_names(), 15)
```

Figure 8: LDA Topic Modelling

5 Modelling

In this section, the first step was to vectorize the words for which purpose a TF-IDF vectorizer was used (Jalilifard et al., 2021), the code of which can be seen in Figure 9. For this the sklearn.feature_extraction.text package was used:

```
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


vect = TfidfVectorizer()
X = vect.fit_transform(concat2['text_pre'])
y = concat2['is_fake']
```

Figure 9: TF-IDF Vectorizer

The above vectorized words are then given as input to three models that are Naive Bayes, Bi-directional LSTM and BERT, which is explained in below subsections:

5.1 Naive Bayes

Naive Bayes is a Machine learning Model based on the Bayes theorem, which assumes that predictors are independent, therefore the name naive. Below Figure 9 shows the implementation of the code for which the BernoulliNB from sklearn.naive_bayes package was used:

```
In [87]:
def select(X, y):
    dict_ = {}
    for i in tqdm(range(1, 11)):
        value = X.shape[1] * i * 0.1
        X_new = SelectKBest(chi2, k=int(value)).fit_transform(X, y)
        X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.33, random_state=42)
        from sklearn.naive_bayes import BernoulliNB
        from sklearn.metrics import confusion_matrix, classification_report
        bnb = BernoulliNB().fit(X_train, y_train)
        print('\nAccuracy score :',bnb.score(X_test, y_test))
        predict = bnb.predict(X_test)
        score = accuracy_score(y_test, predict)
        dict_[str(int(value))] = score
        from sklearn.metrics import confusion_matrix, classification_report

        print('Confusion Metrics : \n\n', confusion_matrix(y_test, predict),'\n\n')
        print('Classification Report :\n\n', classification_report(y_test, predict))
    return dict_

dict_ = select(X, y)
```

Figure 10: Naive Bayes Implementation

5.2 Bidirectional LSTM

The Bidirectional LSTM was implemented using the Keras API, for which the Tokenizer from tensorflow.keras.preprocessing.text and sklearn.model_selectio package was used (Jain et al., 2022), the code implementation for which is given in Figure 11:

```
In [105]:
tokenizer = text.Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
X_train = tf.keras.preprocessing.sequence.pad_sequences(X_train, padding='post', maxlen=256)
X_test = tf.keras.preprocessing.sequence.pad_sequences(X_test, padding='post', maxlen=256)
```

```
In [109]:
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(max_vocab, 32),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,  return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1)
])

model.summary()
```

Figure 11: Bidirectional LSTM Implementation

5.3 BERT

The BERT Model was implemented using transformers and param packages, which after training was saved in the model_after_train.pt model, as shown in Figure 12.

```
In [129]:
total = len(test_data)
number_right = 0
model.eval()
with torch.no_grad():
    for idx, row in test_data.iterrows():
        text_parts = preprocess_text(str(row['text']))
        label = torch.tensor([row['is_fake']]).float().to(device)

        overall_output = torch.zeros((1,2)).to(device)
        try:
            for part in text_parts:
                if len(part) > 0:
                    overall_output += model(part.reshape(1, -1))[0]
        except RuntimeError:
            print("GPU out of memory, skipping this entry.")
            continue

        overall_output = F.softmax(overall_output[0], dim=-1)

        result = overall_output.max(0)[1].float().item()

        if result == label.item():
            number_right += 1

        if idx % print_every == 0 and idx > 0:
            print("{}/{}. Current accuracy: {}".format(idx, total, number_right / idx))

print("Accuracy on test data: {}".format(number_right / total))
```

Figure 12: BERT Implementation

The BERT model was further validated on two random posts from the internet which received a accuracy of fake at 95.24649381637573% real at 88.04030418395996%, and was thus concluded to be the best fit model.

References

Agrawal, C., Pandey, A. and Goyal, S. (2021). A Survey on Role of Machine Learning and NLP in fraud News Detection on Social Media.2021 IEEE 4th International

Conference on Computing, Power and Communication
Technologies (GUCON).

Jain, P., Sharma, S., Monica and Aggarwal, P.K. (2022). Classifying fraud News Detection Using SVM, Naive Bayes and LSTM. [online] IEEE Xplore.
Available at: https://ieeexplore.ieee.org/document/9734129 [Accessed 9 Apr. 2022].

Jalilifard, A. and Caridá, V.F. (2021). Semantic Sensitive TF-IDF to Determine Word Relevance in Documents. [online] Available at: https://arxiv.org/abs/2001.09896.