National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Navya Ravichandran
Student ID: X22241990

School of Computing
National College of Ireland

Supervisor: Athanasios Staikopoulos

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Navya Ravichandran
Student ID:	X22241990
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Athanasios Staikopoulos
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual
Word Count:	1573
Page Count:	11

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:	Navya Ravichandran
Date:	28th January 2025

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

Navya Ravichandran
X22241990

1 Introduction

The research is about Fake news Prediction using Deep learning and Machine learning on image and text data. In configuration manual detailed steps and details from setting up the environment to final model evaluation are explained. The aim of this manual is to explain in detail about the research study that is being conducted. The primary tools used in this research are Jupyter Notebook, Terminal (Command Prompt), Microsoft Excel and System storage. At end of the document detailed system requirements and specifications of the study from importing libraries to final evaluation is understood.

2 Environment

In the Environment section a detailed hardware and software setup required to perform the study is explained. Here Jupyter notebook is the main software that is used throughout the research project implementation. The installation and setup of jupyter notebook is explained below in an upcoming section named Python and Jupyter notebook setup.The hardware requirements used for the research are also mentioned below.

2.1 Hardware/System Specification

(a) Hardware requirements

(b) Detailed HR

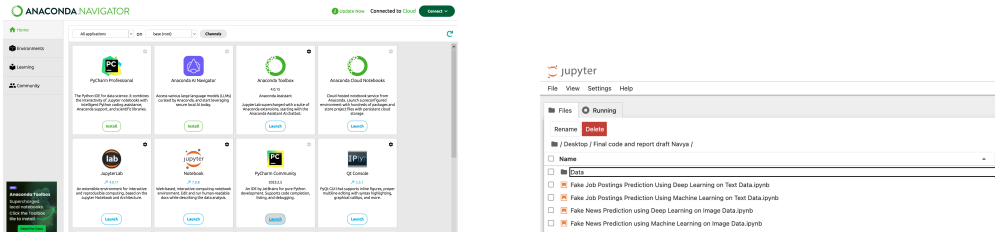
Figure 1: Hardware requirements: (a) Overview of system hardware/System Specification; (b) A detailed data about hardware/System Specification used in the research project.

3 Tools Used and Setup

As this is Data mining research conducted the tools and programming languages used in this research are Jupyter notebook which comes with anaconda navigator, Python and Microsoft excel to store text-based data. For image data storage local storage is used. A detailed explanation about these tools is provided.

3.1 Setting up Jupyter Notebook and Python

The research was conducted using Python Programming Language and Jupyter notebook. To install Jupyter notebook just navigate to your favorite internet browser and install anaconda-navigator that opts your system and hardware requirements. Here anaconda-navigator for mac is installed.



(a) Anaconda Navigator (b) Jupyter Notebook

Figure 2: Tools and Programing Language Used: (a) Jupyter Notebook with preinstalled Python Tools; (b) A detailed Research Project Structure in Jupyter Notebbok.

As shown in Figure 2(a) Anaconda comes with pre-installed python tools that can be used for research where you can Jupyter notebook and launch it your local host for further process. Figure 2(b) is project file's structure setup implemented using jupyter notebook.

The python version used in this research is 3.12.2 make sure the latest version is installed on your machine. Microsoft excel was used to view and store text-based data and image data was downloaded and stored in local system drive and was used for model development and evaluation.

4 Implementation

The implementation section includes the detailed steps and screenshots of code from the importing libraries used,Data preparation,Modeling, and evaluation phase.

4.1 Importing Libraries

The programming language used here is Python. Python has its own libraries that can be used for data visualization, preprocessing, exploratory data analysis, model development and evaluation.The important libraries that have been used in the research are shown in Figure 2.

```
[2]: import os
import numpy as np
import matplotlib.pyplot as plt
import cv2
import numpy as np
import tensorflow as tf
import tensorflow as tf; print(tf.__version__)
from spacy.lang.en import English
import seaborn as sns
import tensorflow as tf
import datetime
import missingno
import pandas as pd
from spacy.lang.en import English
import spacy
import re
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.decomposition import PCA
# Algorithms
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import img_to_array, load_img
from torchvision.datasets import ImageFolder
from torchvision.transforms import Compose, Resize, ToTensor
```

Figure 3: Importing the Necessary Python Libraries.

4.2 Data Preparation

Data preparation include multiple steps for both text and image data.

4.2.1 Preparation of Text data

Here the dataset used for text data is taken from kaggle and it has undergone several data preprocessing steps.The steps involved are:

- Data Cleaning : Removing/Handling null values and eliminating the duplicates from the data.
- Handling Class imbalance: The class imbalance is handled by using Randomundersampler an sampling technique.

(a) Handling missing values

```
#Handling class imbalance using RandomUnderSampler
from imblearn.under_sampling import RandomUnderSampler

under_sampler = RandomUnderSampler()
X_res, y_res = under_sampler.fit_resample(X, Y)

df1 = pd.DataFrame(X_res)

df3 = pd.DataFrame(y_res)

result = pd.concat([df1, df3], axis=1, join='inner')
display(result)
data=result;
```

(b) handling Class imbalance

Figure 4: Code snippets: (a) Handling missing values and eliminating duplicates in text data; (b) Handling class imbalance using sampling technique.

- Handling StopWords: Performed stopwords(”is”, ”the”,”and”, ”in”)removal using wordcloud and NLTK a python toolkit.
- Hyper Parameter Tuning:This is performed based on the algorithms used.

[5]:
```
#  visualize all the words our data using the wordcloud plot
from wordcloud import WordCloud
all_words = ''.join([text for text in data["text"]])
wordcloud = WordCloud(width = 800, height = 500, random_state=21, max_font_size=120).generate(all_words)
plt.figure(figsize=(10,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()

# Common words in real job posting texts
real_post = ''.join([text for text in data["text"][data['fraudulent']==0]])
wordcloud = WordCloud(width = 800, height = 500, random_state=21, max_font_size=120).generate(real_post)
plt.figure(figsize=(10,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
# Common words in fraud job posting texts

fraud_post = ''.join([text for text in data["text"][data['fraudulent'] == 1]])
wordcloud = WordCloud(width = 800, height = 500, random_state=21, max_font_size=120).generate(fraud_post)
plt.figure(figsize=(10,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

(a) Wordcloud

[32]:
```
# NLTK :: Natural Language Toolkit
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/navyaravichandran/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[33]:
```
print(stopwords.words("english"))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'you
rselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 't
heirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'b
een', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'un
til', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'belo
w', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'wher
e', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 's
o', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've',
'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

[34]:
```
#loading the stopwords
stop_words = set(stopwords.words("english"))
```

[35]:
```
#converting all the text to lower case
data['text'] = data['text'].apply(lambda x:x.lower())
```

[36]:
```
#removing the stop words from the corpus
data['text'] = data['text'].apply(lambda x:' '.join([word for word in x.split() if word not in (stop_words)]))
```

[37]:
```
data.reset_index(drop=True, inplace=True)
print(data['text'][0])
```

```
editor hk, , lifehack widely recognized one premier productivity lifestyle content sites web. 10 million readers world, one fastest growing o
nline publishers world. useful practical content tools, want improve every aspect people's lives. we're looking someone passionate create bes
t online content.keep track latest trend viral topics materials across web.create experiment engaging highly shareable content blog social pl
atforms including facebook, twitter pinterest.analyse articles performance based literary style, reports metrics.research topics create high
quality article pitches team writers.guide team writers editors ensure high quality content produced.review articles written edited team writ
ers editors.create attractive copywriting various products featured lifehack. degree english, communication, journalism related fields prefer
redexcellent command english.passionate online content.detail minded high sense content quality control.great communicator driven self-starte
d.ability learn new things quickly.logical.creative. 5 days workflexible working hoursstand-up desks availableenergetic office card games vid
eo game consolereading cornerregular social activities company gatheringsfully-stocked pantry internet
```

(b) NLTK

Figure 5: Stopwords handling codesnippet: (a) Stopwords handling using wordcloud; (b)Handling Stopwords using NLTK Python Toolkit

4.2.2 Preparation of Image data

For image data, all uploaded images are considered only in.jpg format.Image resolution is maintained accordingly and irrelevant images are removed to reduce the noise of the data.Data augmentation is performed but it has not shown more efficient results.Appropriate image dimensions setup is done and image data generator is used to preprocess the images.

Setting appropriate Image dimensions and batch size

[4]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Image dimensions and batch size
IMG_HEIGHT, IMG_WIDTH = 128, 128
BATCH_SIZE = 32
IMG_CHANNELS = 3
LATENT_DIM = 100  # Dimension of the noise vector
BATCH_SIZE = 64
```

Using ImageDatagenerator to process the images

[5]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.3,
    horizontal_flip=True,
    brightness_range=[0.7, 1.3],
    fill_mode='nearest'
)

valid_datagen = ImageDataGenerator(rescale=1.0/255.0)

train_data = train_datagen.flow_from_directory(
    "Version 1 /train",
    target_size=(128, 128),
    batch_size=32,
    class_mode='binary'
)

valid_data = valid_datagen.flow_from_directory(
    "Version 1 /valid",
    target_size=(128, 128),
    batch_size=32,
    class_mode='binary'
)
```

```
Found 1795 images belonging to 2 classes.
Found 175 images belonging to 2 classes.
```

Figure 6: Image Data Preprocessing

5 Model development

This is the important section that includes the model development phase where both the machine and deep lerning algorithms was developed for both text and image data and evaluation was peformed using the evaluation metrics.

5.1 Study 1 : Fake news detection using Machine Learning on Text data

Train Test Split: The first step is splitting the dataset preprocessed into test and train sets.Here it is splitted in 70:30 ratio.

MODEL DEVELOPMENT

```
[39]: from sklearn.model_selection import train_test_split
      # Splitting dataset in train and test
      X_train, X_test, y_train, y_test = train_test_split(data.text, data.fraudulent, test_size=0.3)
```

Figure 7: Train Test Split

Normalization: Next step normalization is performed using CountVectorizer as shown in Figure 8.

Using CountVectorizer for normalization

```
[41]: from sklearn.feature_extraction.text import CountVectorizer

      #  instantiate the vectorizer
      vect = CountVectorizer()

      # learn training data vocabulary, then use it to create a document-term matrix
      # fit
      vect.fit(X_train)

      # transform training data
      X_train_dtm = vect.transform(X_train)

[42]: X_train_dtm
      print(X_train_dtm)
```

```
  (1211, 16937) 1
  (1211, 16990) 3
  (1211, 16992) 2
  (1211, 17027) 1
  (1211, 17079) 1
  (1211, 17935) 1
  (1211, 18199) 1
  (1211, 18444) 2
  (1211, 18683) 1
  (1211, 18891) 1
```

Figure 8: CountVectorizer

5.1.1 Algorithms Implemented.

The machine Learning algorithms used for implementation are SVM,KNN,Decision Trees,Random Forest,Naive-Bayes and Gradient Boosting Machines.A few algorithm which had better performance metrics code snippets have been included.

Here Random Forest code snippet is shown in Figure 9 as it has shown a great performance than all the other machine Learning Algorithms.

RANDOM FOREST

```
[57]: from sklearn.ensemble import RandomForestClassifier

[58]: # Instantiate the Random Forest Classifier
rf = RandomForestClassifier(random_state=42)  # You can set n_estimators and max_depth as needed

[59]: # Train the model using X_train_dtm
%time rf.fit(X_train_dtm, y_train)
```

```
CPU times: user 760 ms, sys: 2.5 ms, total: 762 ms
Wall time: 460 ms
```

```
[59]: RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
[60]: # Make class predictions for X_test_dtm
y_pred_class_rf = rf.predict(X_test_dtm)

[61]: # Model Accuracy
print("Classification Accuracy:", accuracy_score(y_test, y_pred_class_rf))
print("Classification Report\n")
print(classification_report(y_test, y_pred_class_rf))
print("Confusion Matrix\n")
print(confusion_matrix(y_test, y_pred_class_rf))
```

```
Classification Accuracy: 0.925
Classification Report

              precision    recall  f1-score   support

           0       0.91      0.94      0.93       269
           1       0.94      0.90      0.92       251

    accuracy                           0.93       520
   macro avg       0.93      0.92      0.92       520
weighted avg       0.93      0.93      0.92       520

Confusion Matrix

[[254  15]
 [ 24 227]]
```

```
[62]: # Confusion Matrix
cm_rf = confusion_matrix(y_test, y_pred_class_rf)

plt.figure(figsize=(10, 7))
sn.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.title('Confusion Matrix for Random Forest Classifier')
plt.show()
```

Figure 9: Random Forest Code Implementation

5.2 Study 2 : Fake news detection using Deep Learning on Text data

Lemmanization: Lemmanization was performed to measure the time of the text data preprocessing as shown in Figure 10.

LEMMANIZATION

```
[44]: # #Lemmenization
#Time module is just to measure the time it took as i was comparing Spacy, NLTK and Gensim. Spacy was the fastest
sp = spacy.load('en_core_web_sm')
import time
t1=time.time()
output=[]

for sentence in df['text']:
    sentence=sp(str(sentence))
    s=[token.lemma_ for token in sentence]
    output.append(' '.join(s))
df['processed']=pd.Series(output)
t=time.time()-t1
print("Time" + str(t))
```

```
Time1114.1424088478088
```

Figure 10: Lemmanization

Tokenization and Padding: Tokenization is performed before splitting the model into test and train set.

TOKENIZATION AND PADDING

[52]:
```
import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

vocab_size = 100000
embedding_dim = 64
max_length = 250
trunc_type='post'
padding_type='post'
oov_tok = "<OOV>"
training_size = 20000
#Tokenization

tokenizer = Tokenizer(num_words=vocab_size)
tokenizer.fit_on_texts(df['processed'].values)
word_index = tokenizer.word_index
print(len(word_index))
```

```
96956
```

[54]:
```
X = tokenizer.texts_to_sequences(df['processed'].values)                    #Tokenize the dataset
X = pad_sequences(X, maxlen=max_length)     #Padding the dataset
Y=df['fraudulent']                                                          #Assign the value of y
print(Y.shape)
```

```
(17880,)
```

[56]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.20,random_state=41)
```

Figure 11: Tokenization and Padding

5.2.1 Algorithms Implemented.

The Deep Learning algorithms used for implementation are Bert,CNN,and Bi-LSTM .A few algorithm which had better performance metrics code snippets have been included.

Here CNN code snippet is shown in Figure 12 as it has shown a great performance than all the other Deep Learning Algorithms.

CNN

[258…]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
vocab_size = 100000
embedding_dim = 64
max_length = 250
oov_tok = "<OOV>"

# Tokenize and pad the text data
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(df['processed'].values)
X = tokenizer.texts_to_sequences(df['processed'].values)
X = pad_sequences(X, maxlen=max_length)
y = df['fraudulent']

# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[260]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Conv1D(filters=128, kernel_size=5, activation='relu'),
    tf.keras.layers.GlobalMaxPooling1D(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')  # Binary classification
])

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
/opt/anaconda3/lib/python3.12/site-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just r
emove it.
  warnings.warn(
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	?	0 (unbuilt)
conv1d (Conv1D)	?	0 (unbuilt)
global_max_pooling1d (GlobalMaxPooling1D)	?	0
dense_2 (Dense)	?	0 (unbuilt)
dense_3 (Dense)	?	0 (unbuilt)

Total params: 0 (0.00 B)
Trainable params: 0 (0.00 B)
Non-trainable params: 0 (0.00 B)

[262]:
```
from tensorflow.keras.callbacks import EarlyStopping

callbacks = [EarlyStopping(monitor='val_loss', patience=2)]

history = model.fit(
    X_train, y_train,
    epochs=10,
    batch_size=64,
    validation_split=0.1,
    callbacks=callbacks,
    verbose=1
)
```

```
Epoch 1/10
202/202 ━━━━━━━━━━━━━━━━━━━━ 15s 68ms/step - accuracy: 0.9419 - loss: 0.2722 - val_accuracy: 0.9672 - val_loss: 0.1014
Epoch 2/10
202/202 ━━━━━━━━━━━━━━━━━━━━ 14s 68ms/step - accuracy: 0.9766 - loss: 0.0697 - val_accuracy: 0.9881 - val_loss: 0.0453
Epoch 3/10
202/202 ━━━━━━━━━━━━━━━━━━━━ 14s 69ms/step - accuracy: 0.9963 - loss: 0.0134 - val_accuracy: 0.9909 - val_loss: 0.0462
Epoch 4/10
```

Figure 12: CNN Code Implementation

5.3 Study 3 : Fake news detection using Machine Learning on Image data

Data Loading and Splitting: The image data are loaded from local storage and split based on the test,train and validation.

Load the Data

```
[4]: # Set directory paths
train_dir = "Version 2/train"
valid_dir = "Version 2/valid"
test_dir = "Version 2/test"
```

```
[5]: def load_images_from_directory(directory, label):
    images = []
    labels = []
    for file in os.listdir(directory):
        file_path = os.path.join(directory, file)
        if os.path.isfile(file_path):
            img = cv2.imread(file_path, cv2.IMREAD_COLOR)
            img = cv2.resize(img, (128, 128))  # Resize to reduce complexity
            images.append(img)
            labels.append(label)
    return np.array(images), np.array(labels)

# Load training, validation, and testing datasets
fake_train, fake_labels_train = load_images_from_directory(f"{train_dir}/fake", 0)
real_train, real_labels_train = load_images_from_directory(f"{train_dir}/real", 1)

fake_valid, fake_labels_valid = load_images_from_directory(f"{valid_dir}/fake", 0)
real_valid, real_labels_valid = load_images_from_directory(f"{valid_dir}/real", 1)

fake_test, fake_labels_test = load_images_from_directory(f"{test_dir}/fake", 0)
real_test, real_labels_test = load_images_from_directory(f"{test_dir}/real", 1)

# Combine datasets
X_train = np.concatenate((fake_train, real_train), axis=0)
y_train = np.concatenate((fake_labels_train, real_labels_train), axis=0)

X_valid = np.concatenate((fake_valid, real_valid), axis=0)
y_valid = np.concatenate((fake_labels_valid, real_labels_valid), axis=0)

X_test = np.concatenate((fake_test, real_test), axis=0)
y_test = np.concatenate((fake_labels_test, real_labels_test), axis=0)
```

Figure 13: Loading and Splitting image data

5.3.1 Algorithms Implemented.

Random Forest

```
[27]: rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train)

# Evaluate
y_pred_rf = rf_model.predict(X_test_scaled)
print("Random Forest Classification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.90      0.91        41
           1       0.90      0.93      0.91        40

    accuracy                           0.91        81
   macro avg       0.91      0.91      0.91        81
weighted avg       0.91      0.91      0.91        81
```

Naive Bayes

```
[30]: nb_model = GaussianNB()
nb_model.fit(X_train_scaled, y_train)

# Evaluate
y_pred_nb = nb_model.predict(X_test_scaled)
print("Naive Bayes Classification Report:")
print(classification_report(y_test, y_pred_nb))
```

```
Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.54      0.63        41
           1       0.63      0.82      0.72        40

    accuracy                           0.68        81
   macro avg       0.70      0.68      0.67        81
weighted avg       0.70      0.68      0.67        81
```

Figure 14: Random Forest and Naive Bayes Code Implementation

The Machine Learning algorithms used for implementation are SVM,KNN,Decision Trees,Random Forest and Naive-Bayes, .A few algorithm which had better performance metrics code snippets have been included.

Here Random Forest and Naive Bayes code snippet is shown in Figure 14 as it has shown a great performance than all the other Deep Learning Algorithms.

5.4 Study 4 : Fake news detection using Deep Learning on Image data

The data is splitted in test,train and validation sets and the model is evaluated against the data.

5.4.1 Algorithms Implemented

The Deep Learning algorithms used for implementation are Visual-Bert,CNN,RNN,ANN and LSTM .A few algorithm which had better performance metrics code snippets have been included.

Here CNN T code snippet is shown in Figure 12 as it has shown a better performance than all the other Deep Learning Algorithms.

Figure 15: Implementation of CNN Code

6 Evaluation and results

Evaluation is a critical step in the machine and deep learning pipeline that aids in determining the model's efficacy and confirming that it is functioning as intended. Carefully

choosing the pertinent assessment metrics and assessing the performance of the model are essential to predict the false news.

All the model metrics have been compared and tabulated below as shown in Figure 16 which includes the performance of all the Machine Learning and Deep Learning algorithms implemented.

FAKE NEWS PREDICTION ON TEXT DATA USING MACHINE LEARNING			
Algorithm Name	**Precision**	Recall	F1-Score
Random Forest	0.91	0.94	0.93
Decision Tree	0.87	0.88	0.87
Naive Bayes	0.92	0.90	0.91
SVM	0.90	0.92	0.91
KNN			
FAKE NEWS PREDICTION ON TEXT DATA USING DEEP LEARNING			
Bi-LSTM	0.99	0.98	0.98
BERT	0.99	1.00	0.99
CNN	0.98	1.00	0.99
FAKE NEWS PREDICTION ON IMAGE DATA USING MACHINE LEARNING			
Random Forest	0.93	0.90	0.91
SVM	0.77	0.59	0.67
Decision Trees	0.74	0.56	0.64
KNN	0.89	0.20	0.32
Naive Byes	0.76	0.54	0.63
FAKE NEWS PREDICTION ON IMAGE DATA USING DEEP LEARNING			
CNN	Accuracy - 0.86		
RNN	Accuracy - 0.58		
LSTM	Accuracy - 0.74		
ANN	Accuracy - 0.64		
VISUAL BERT	Accuracy - 1.00		

Figure 16: Comparison Table of Performance Metrics