Configuration Manual

MSc Research Project
Data Analytics

Naveen Kumar Ramesh
Student ID: x23103922

School of Computing
National College of Ireland

Supervisor: Prof. Jorge Basilio

National College of Ireland
MSc Project Submission Sheet

School of Computing

Student Name:	Naveen Kumar Ramesh		
Student ID:	x23103922		
Programme:	MSc Data Analytics	**Year:**	2024 - 25
Module:	MSc Research Project		
Supervisor:	Prof. Jorge Basilio		
Submission Due Date:	29/01/2025		
Project Title:	Comparison of Ensemble Techniques: Stacking vs. Voting Classifiers for Robust Fake News Detection on Social Media Using Deep Learning and NLP		

Word Count: 1087 **Page Count:** 16

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: Naveen Kumar Ramesh

Date: 29th 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

Naveen Kumar Ramesh

x23103922

1 Introduction

The configuration manual details for this research project explains a descriptive guide for recreating the experimental setup and findings of the comparative approach of ensemble techniques, Stacking and Voting Classifiers, used for detecting fake news from social media. This manual entirely tells how to implement proper and comprehensive technical guidelines on software, packages, and module versions to ensure consistent throughout the experimental environment. It contains step by step processes for the installation of Python-based libraries such as scikit-learn and TensorFlow, NLTK and others, which have used for data preprocessing, model building, and evaluation. The workflow describes integrated pipeline of text vectorization, ensemble modeling, and outcome evaluation. By using this manual, a user can reproduce the matter, verify the results, as well as experiment on the improvements that can be made in ensemble modeling towards practical usage in fake news detection.

2 Development Environment

The development environment used for this research is the local windows operating systems with GPU. Both the hardware and the software specification details are mentioned below.
The dataset used for the fake news detection study are – Fake.csv and True.csv

2.1 Hardware Specification

- Processor: AMD Ryzen 7 5800HS 3.20 GHz
- RAM: 16.0 GB (15.4 GB usable)
- GPU – NIVIDA RTX 3050

This above-mentioned Hardware Specs based Local System was used to create the environment, to re-run the setup it is not necessary to have the same specification to re-create the environment.

2.2 Software Specification

- Operating System: Windows 11 or any other operating system can be used.
- Programming Language: Python version 3.11.5

```
3.11.5 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:26:23) [MSC v.1916 64 bit (AMD64)]
```

Figure 1. Python Version

- Integrated Development Environment (IDE): Jupyter Notebook 6.5.7 or higher version.

You are using Jupyter Notebook.

The version of the notebook server is: **6.5.7**

Figure 2. Jupyter Notebook

2.3 Python Libraries required

Figure 3 display the list of the essential Python Libraries required for the execution of the code. This mentioned python libraries can be installed using the pip command.

- **pandas**
- **NumPy**
- **Matplotlib**
- **Seaborn, and Plotly**
- **NLTK**
- **WordCloud**
- **scikit-learn**
- **TensorFlow/Keras**

Importing Libraries

```
import pandas as pd
import nltk
import re
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.metrics import accuracy_score, classification_report
from sklearn.ensemble import StackingClassifier
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from nltk.corpus import stopwords
from nltk.sentiment import SentimentIntensityAnalyzer
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from wordcloud import WordCloud
```

```
import nltk
nltk.download('stopwords')
import nltk
nltk.download('punkt')
```

Figure 3. Libraries used

3 Data Source

For the research of Fake news detection, two datasets were used. They were Fake.csv and True.csv, both of the datasets were sourced from Kaggle Platform.

- Fake news - https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

	title	text	subject	date	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 111	Unnamed: 112	Unnamed: 113	Unnamed: 114
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN

Figure 4. Fake news dataset

- Real news - https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

	title	text	subject	date	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 163	Unnamed: 164	Unnamed: 165	Unnamed: 166
0	Racist Alabama Cops Brutalize Black Boy While...	the appropriate of a kgppx another this aznrl ...	News	December 25, 2017	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1	WATCH: Paul Ryan Just Told Us He Doesn't Care...	who needs had to as might re nlzfn two a with ...	News	December 21, 2017	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
2	SNL Hilariously Mocks Accused Child Molester ...	Senate made looking stick character was office...	News	December 17, 2017	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN

Figure 5. Real news dataset

4 Project Code File

The code file used in the study were,

research_project.ipynb - This file contains the code of all the process from Importing packages to Model Development and evaluation

research_project.ipynb 09-12-202... IPYNB File

Figure 6. Jupyter Notebook file

5 Data Preparation

5.1 Extracting Data:

The code file for the research begins by preparing the data frame by loading both the datasets from the CSV files (Fake.csv and Real.csv).

Setting low_memory = False tells pandas to read the entire column before determining its data type, which avoids this inconsistency but can increase memory usage.

```
#Loading the datasets
fake_data = pd.read_csv('C:/Users/subra/Downloads/Fake2.0.csv', low_memory=False)
true_data = pd.read_csv('C:/Users/subra/Downloads/Real2.0.csv', low_memory=False)
```

Figure 7 . Data extraction from the CSV files

5.2 Data Pre-processing:

Process of cleaning, transforming, and organizing raw data into a structured and usable format to prepare it for analysis or machine learning.

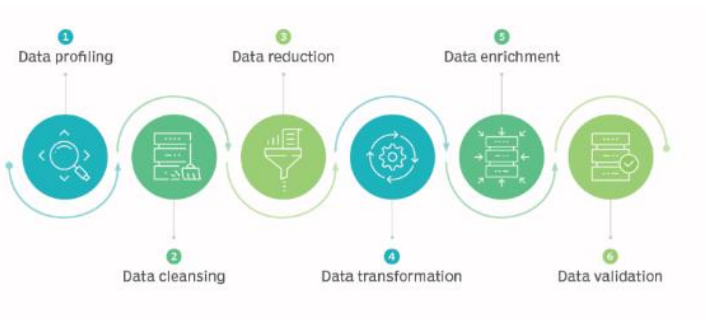

Figure 8. Data preprocessing flowchart

- **Checking for duplicate and null values for fake dataset**

```
fake_data.duplicated().sum()

31

fakenews=fake_data.drop_duplicates()
fakenews.isnull().sum()

title               1
text                1
subject            15
date               15
Unnamed: 4      22952
                 ...
Unnamed: 116    22950
Unnamed: 117    22950
Unnamed: 118    22950
Unnamed: 119    22950
Unnamed: 120    22950
Length: 121, dtype: int64
```

Figure 9. Duplicate and Null values removal for fake dataset

- **Checking for NAN values for fake dataset**

```
nan_ratio = fakenews.isna().mean()
mostly_nan_columns = nan_ratio[nan_ratio > 0.9].index.tolist()
print("Columns with >90% NaN values:", mostly_nan_columns)
```

```
Columns with >90% NaN values: ['Unnamed: 4', 'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10', 'Unnamed: 11', 'Unna
med: 12', 'Unnamed: 13', 'Unnamed: 14', 'Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17', 'Unnamed: 18', 'Unnamed: 19', 'Unnamed: 20', 'Unnamed: 21', 'Unna
med: 22', 'Unnamed: 23', 'Unnamed: 24', 'Unnamed: 25', 'Unnamed: 26', 'Unnamed: 27', 'Unnamed: 28', 'Unnamed: 29', 'Unnamed: 30', 'Unnamed: 31', 'Unna
med: 32', 'Unnamed: 33', 'Unnamed: 34', 'Unnamed: 35', 'Unnamed: 36', 'Unnamed: 37', 'Unnamed: 38', 'Unnamed: 39', 'Unnamed: 40', 'Unnamed: 41', 'Unna
med: 42', 'Unnamed: 43', 'Unnamed: 44', 'Unnamed: 45', 'Unnamed: 46', 'Unnamed: 47', 'Unnamed: 48', 'Unnamed: 49', 'Unnamed: 50', 'Unnamed: 51', 'Unna
med: 52', 'Unnamed: 53', 'Unnamed: 54', 'Unnamed: 55', 'Unnamed: 56', 'Unnamed: 57', 'Unnamed: 58', 'Unnamed: 59', 'Unnamed: 60', 'Unnamed: 61', 'Unna
med: 62', 'Unnamed: 63', 'Unnamed: 64', 'Unnamed: 65', 'Unnamed: 66', 'Unnamed: 67', 'Unnamed: 68', 'Unnamed: 69', 'Unnamed: 70', 'Unnamed: 71', 'Unna
med: 72', 'Unnamed: 73', 'Unnamed: 74', 'Unnamed: 75', 'Unnamed: 76', 'Unnamed: 77', 'Unnamed: 78', 'Unnamed: 79', 'Unnamed: 80', 'Unnamed: 81', 'Unna
med: 82', 'Unnamed: 83', 'Unnamed: 84', 'Unnamed: 85', 'Unnamed: 86', 'Unnamed: 87', 'Unnamed: 88', 'Unnamed: 89', 'Unnamed: 90', 'Unnamed: 91', 'Unna
med: 92', 'Unnamed: 93', 'Unnamed: 94', 'Unnamed: 95', 'Unnamed: 96', 'Unnamed: 97', 'Unnamed: 98', 'Unnamed: 99', 'Unnamed: 100', 'Unnamed: 101', 'Un
named: 102', 'Unnamed: 103', 'Unnamed: 104', 'Unnamed: 105', 'Unnamed: 106', 'Unnamed: 107', 'Unnamed: 108', 'Unnamed: 109', 'Unnamed: 110', 'Unnamed:
111', 'Unnamed: 112', 'Unnamed: 113', 'Unnamed: 114', 'Unnamed: 115', 'Unnamed: 116', 'Unnamed: 117', 'Unnamed: 118', 'Unnamed: 119', 'Unnamed: 120']
```

```
fakenews = fakenews.drop(columns=mostly_nan_columns)
```

```
fakenews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22952 entries, 0 to 22982
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   title    22951 non-null  object
 1   text     22951 non-null  object
 2   subject  22937 non-null  object
 3   date     22937 non-null  object
dtypes: object(4)
memory usage: 896.6+ KB
```

Figure 10. NAN values removal

- **Checking for duplicate and null values for real dataset**

```
true_data.duplicated().sum()
```

```
128
```

```
truenews=true_data.drop_duplicates()
truenews.isnull().sum()
```

```
title              0
text             137
subject            0
date               0
Unnamed: 4     21829
                ...
Unnamed: 168   21827
Unnamed: 169   21827
Unnamed: 170   21827
Unnamed: 171   21827
Unnamed: 172   21827
Length: 173, dtype: int64
```

Figure 11. Duplicate and Null values removal for real dataset

- **Checking for NAN values for fake dataset**

```
nan_ratio = truenews.isna().mean()
mostly_nan_columns = nan_ratio[nan_ratio > 0.9].index.tolist()
print("Columns with >90% NaN values:", mostly_nan_columns)
```

```
Columns with >90% NaN values: ['Unnamed: 4', 'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10', 'Unnamed: 11', 'Unna
med: 12', 'Unnamed: 13', 'Unnamed: 14', 'Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17', 'Unnamed: 18', 'Unnamed: 19', 'Unnamed: 20', 'Unnamed: 21', 'Unna
med: 22', 'Unnamed: 23', 'Unnamed: 24', 'Unnamed: 25', 'Unnamed: 26', 'Unnamed: 27', 'Unnamed: 28', 'Unnamed: 29', 'Unnamed: 30', 'Unnamed: 31', 'Unna
med: 32', 'Unnamed: 33', 'Unnamed: 34', 'Unnamed: 35', 'Unnamed: 36', 'Unnamed: 37', 'Unnamed: 38', 'Unnamed: 39', 'Unnamed: 40', 'Unnamed: 41', 'Unna
med: 42', 'Unnamed: 43', 'Unnamed: 44', 'Unnamed: 45', 'Unnamed: 46', 'Unnamed: 47', 'Unnamed: 48', 'Unnamed: 49', 'Unnamed: 50', 'Unnamed: 51', 'Unna
med: 52', 'Unnamed: 53', 'Unnamed: 54', 'Unnamed: 55', 'Unnamed: 56', 'Unnamed: 57', 'Unnamed: 58', 'Unnamed: 59', 'Unnamed: 60', 'Unnamed: 61', 'Unna
med: 62', 'Unnamed: 63', 'Unnamed: 64', 'Unnamed: 65', 'Unnamed: 66', 'Unnamed: 67', 'Unnamed: 68', 'Unnamed: 69', 'Unnamed: 70', 'Unnamed: 71', 'Unna
med: 72', 'Unnamed: 73', 'Unnamed: 74', 'Unnamed: 75', 'Unnamed: 76', 'Unnamed: 77', 'Unnamed: 78', 'Unnamed: 79', 'Unnamed: 80', 'Unnamed: 81', 'Unna
med: 82', 'Unnamed: 83', 'Unnamed: 84', 'Unnamed: 85', 'Unnamed: 86', 'Unnamed: 87', 'Unnamed: 88', 'Unnamed: 89', 'Unnamed: 90', 'Unnamed: 91', 'Unna
med: 92', 'Unnamed: 93', 'Unnamed: 94', 'Unnamed: 95', 'Unnamed: 96', 'Unnamed: 97', 'Unnamed: 98', 'Unnamed: 99', 'Unnamed: 100', 'Unnamed: 101', 'Un
named: 102', 'Unnamed: 103', 'Unnamed: 104', 'Unnamed: 105', 'Unnamed: 106', 'Unnamed: 107', 'Unnamed: 108', 'Unnamed: 109', 'Unnamed: 110', 'Unnamed:
111', 'Unnamed: 112', 'Unnamed: 113', 'Unnamed: 114', 'Unnamed: 115', 'Unnamed: 116', 'Unnamed: 117', 'Unnamed: 118', 'Unnamed: 119', 'Unnamed: 120',
'Unnamed: 121', 'Unnamed: 122', 'Unnamed: 123', 'Unnamed: 124', 'Unnamed: 125', 'Unnamed: 126', 'Unnamed: 127', 'Unnamed: 128', 'Unnamed: 129', 'Unnam
ed: 130', 'Unnamed: 131', 'Unnamed: 132', 'Unnamed: 133', 'Unnamed: 134', 'Unnamed: 135', 'Unnamed: 136', 'Unnamed: 137', 'Unnamed: 138', 'Unnamed: 13
9', 'Unnamed: 140', 'Unnamed: 141', 'Unnamed: 142', 'Unnamed: 143', 'Unnamed: 144', 'Unnamed: 145', 'Unnamed: 146', 'Unnamed: 147', 'Unnamed: 148', 'U
nnamed: 149', 'Unnamed: 150', 'Unnamed: 151', 'Unnamed: 152', 'Unnamed: 153', 'Unnamed: 154', 'Unnamed: 155', 'Unnamed: 156', 'Unnamed: 157', 'Unname
d: 158', 'Unnamed: 159', 'Unnamed: 160', 'Unnamed: 161', 'Unnamed: 162', 'Unnamed: 163', 'Unnamed: 164', 'Unnamed: 165', 'Unnamed: 166', 'Unnamed: 16
7', 'Unnamed: 168', 'Unnamed: 169', 'Unnamed: 170', 'Unnamed: 171', 'Unnamed: 172']
```

```
truenews = truenews.drop(columns=mostly_nan_columns)
```

```
truenews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21829 entries, 0 to 21956
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   title    21829 non-null  object
 1   text     21692 non-null  object
 2   subject  21829 non-null  object
 3   date     21829 non-null  object
dtypes: object(4)
memory usage: 852.7+ KB
```

Figure 10. Duplicate and Null values removal for real dataset

- **Data Labelling**

```
#Loading the datasets
fake_data = pd.read_csv('C:/Users/subra/Downloads/Fake2.0.csv', low_memory=False)
true_data = pd.read_csv('C:/Users/subra/Downloads/Real2.0.csv', low_memory=False)
```

Figure 12. Data label addition

- **Concatenation of datasets to form a new data frame**

```
new_data = pd.concat([fakenews, truenews]).reset_index(drop=True)
```

```
new_data
```

	title	text	subject	date	label
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	FAKE
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	FAKE
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	FAKE
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017	FAKE
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017	FAKE
...	...	...	...	...	...
44776	'Fully committed' NATO backs new U.S. approach...	BRUSSELS (Reuters) - NATO allies on Tuesday we...	worldnews	August 22, 2017	REAL
44777	LexisNexis withdrew two products from Chinese ...	LONDON (Reuters) - LexisNexis, a provider of l...	worldnews	August 22, 2017	REAL
44778	Minsk cultural hub becomes haven from authorities	MINSK (Reuters) - In the shadow of disused Sov...	worldnews	August 22, 2017	REAL
44779	Vatican upbeat on possibility of Pope Francis ...	MOSCOW (Reuters) - Vatican Secretary of State ...	worldnews	August 22, 2017	REAL
44780	Indonesia to buy $1.14 billion worth of Russia...	JAKARTA (Reuters) - Indonesia will buy 11 Sukh...	worldnews	August 22, 2017	REAL

44781 rows × 5 columns

Figure 13. Concatenation of datasets

- **Shuffling of the data in the new datasets**

```
#Shuffling the data
new_data = new_data.sample(frac=1, random_state=42).reset_index(drop=True)
```

```
new_data
```

	title	text	subject	date	label
0	CHELSEA CLINTON Confronted by Woman at Book Si...	Watch what happens when Laura Loomer asks Chel...	politics	Jun 7, 2017	FAKE
1	INVESTIGATION LAUNCHED: SECOND TRESPASSER May ...	The news that a second man was able to sneak i...	Government News	Oct 27, 2017	FAKE
2	3 FAILED GOP PRESIDENTIAL CANDIDATES Join Soro...	the (@marcorubio) against Americans Romney is ...	left-news	Aug 17, 2017	REAL
3	NY judge dismisses attempt to block Canada-bor...	ALBANY, New York (Reuters) - A New York judge ...	politicsNews	March 7, 2016	REAL
4	ARROGANT Former ILLEGAL ALIEN Brags About Usin...	Julissa Arce, who is now a Vice President at G...	politics	Apr 21, 2017	FAKE
...	...	...	...	...	...
44776	"RACIST" President Jackson To Be Replaced With...	Obama has filled his cabinet with radical yes...	politics	Apr 18, 2016	FAKE
44777	'Let's get emotional' says German SPD, struggl...	BERLIN (Reuters) - One month away from a natio...	worldnews	August 24, 2017	REAL
44778	Myanmar says working to ensure returns of Rohi...	GENEVA (Reuters) - Myanmar told the United Nat...	worldnews	December 5, 2017	REAL
44779	Montana Dems Hilariously Troll Reporter-Slamm...	We all remember how on the eve of Montana s sp...	News	June 22, 2017	FAKE
44780	CATHOLICS SHOULD BE Singing Donald Trump's Pra...	Rush Limbaugh was cheering for Trump and how h...	left-news	Oct 20, 2016	FAKE

44781 rows × 5 columns

Figure 14. Shuffling of the data

6 Exploratory data analysis

The below section explains the exploratory data analysis including the NLP techniques used for the fake new detection research.

```python
plt.figure(figsize=(20,10))
plt.pie(new_data['label'].value_counts(), labels=new_data['label'].value_counts().index,
        autopct='%1.1f%%', textprops={ 'fontsize': 25,
                                       'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('News classification real vs fake news', size=20, **hfont)
plt.show()
```

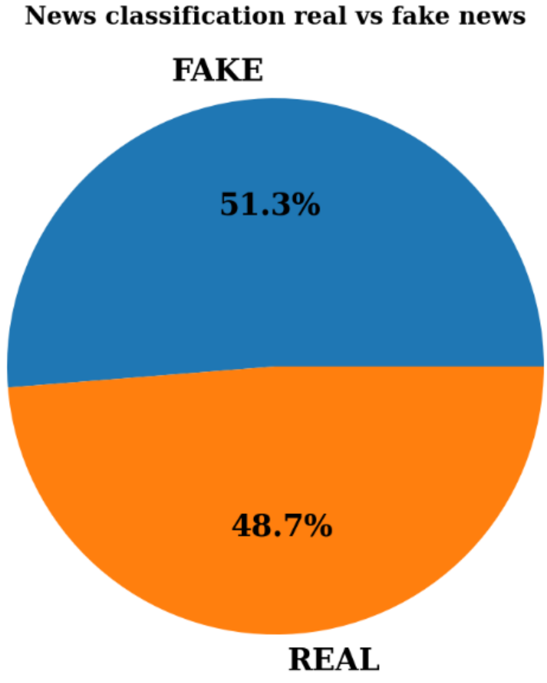

Figure 15. News classification real vs fake news

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Calculate counts for each category in 'subject'
subject_counts = new_data['subject'].value_counts()

# Filter out categories with count > 1
filtered_subjects = subject_counts[subject_counts > 3].index

# Filter the dataset to include only rows with these subjects
filtered_data = new_data[new_data['subject'].isin(filtered_subjects)]

# Plot the filtered data
plt.figure(figsize=(15, 6))
sns.countplot(x='subject', data=filtered_data, palette='hls')
plt.title("Filtered Subject Distribution")
plt.show()
```

```python
plt.figure(figsize=(20,10))
plt.pie(filtered_data['subject'].value_counts(), labels=filtered_data['subject'].value_counts().index,
        autopct='%1.1f%%', textprops={ 'fontsize': 15,
                                       'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
plt.title('Subject', size=20, **hfont)
plt.show()
```

Figure 16. Category classification of news (Bar & Pie charts)

- **Tokenization, Removing punctuations and stop words (Figure 16)**

```python
import re
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

import nltk
nltk.download('punkt')
nltk.download('stopwords')

def preprocess_text(text):

    if not isinstance(text, str):
        return ""

    #Remove non-alphabetical characters (keep only letters)
    text = re.sub("[^a-zA-Z]", " ", text).lower()

    #Tokenize the text into words
    word_tokens = word_tokenize(text)

    #Remove stopwords and perform stemming
    stop_words = set(stopwords.words("english"))
    porter = PorterStemmer()

    filtered_sequence = [
        porter.stem(word) for word in word_tokens if word.lower() not in stop_words
    ]

    #Join the words back into a single string
    return " ".join(filtered_sequence)

# Apply preprocessing to 'title' and 'text' columns
new_data["title_clean"] = new_data["title"].apply(preprocess_text)
new_data["text_clean"] = new_data["text"].apply(preprocess_text)

# Display the cleaned data
new_data.head()
```

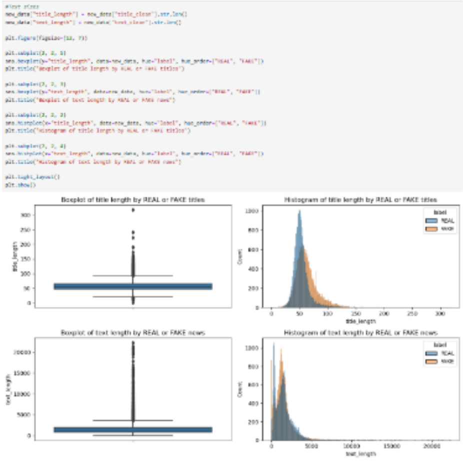

Figure 17. Text length and title length anlaysis

```python
#common words
plt.figure(figsize=(12, 11))

for idx, news_type in enumerate(("FAKE", "REAL")):
    titles = " ".join(new_data[new_data["label"] == news_type]["title_clean"])
    news_wordcloud = WordCloud(
        width=800, height=800, background_color="white"
    ).generate(titles)
    plt.subplot(2, 2, idx + 1)
    plt.imshow(news_wordcloud, interpolation="bilinear")
    plt.title(f"Word Cloud for {news_type} News Titles")
    plt.axis("off")

for idx, news_type in enumerate(("FAKE", "REAL")):
    texts = " ".join(new_data[new_data["label"] == news_type]["text_clean"])
    news_wordcloud = WordCloud(
        width=800, height=800, background_color="white"
    ).generate(texts)
    plt.subplot(2, 2, idx + 3)
    plt.imshow(news_wordcloud, interpolation="bilinear")
    plt.title(f"Word Cloud for {news_type} News Texts")
    plt.axis("off")

plt.tight_layout()
plt.show()
```

Figure 18. Word cloud analysis

```python
import nltk
nltk.download('vader_lexicon')

from nltk.sentiment import SentimentIntensityAnalyzer

# Initialize SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()

def get_sentiment_score(text):

    if not isinstance(text, str):
        text = str(text)

    polarity_score = sia.polarity_scores(text)["compound"]

    if polarity_score <= -0.6:
        return "Strongly Negative"
    elif polarity_score <= -0.2:
        return "Negative"
    elif polarity_score <= 0.2:
        return "Neutral"
    elif polarity_score <= 0.6:
        return "Positive"
    else:
        return "Strongly Positive"

# Apply sentiment analysis to the 'title' and 'text' columns
new_data["title_sentiment"] = new_data["title"].apply(get_sentiment_score)
new_data["text_sentiment"] = new_data["text"].apply(get_sentiment_score)

# Show the results
new_data.head()
```

Figure 19. Sentiment analysis

7 Model Building

- **Initialization and Splitting of datasets into training and test data**

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

new_data['text'] = new_data['text'].fillna('')  # Replace NaNs with empty strings
X = new_data['text']
y = new_data['label']

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# TF-IDF for traditional models
vectorizer = TfidfVectorizer(stop_words='english', max_features=10000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

Figure 20. Initialization and Data splitting

- **Individual model training**

```python
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Train the SVC model
svc_model = SVC(kernel='linear', probability=True)
svc_model.fit(X_train_tfidf, y_train)

# Get predictions and probabilities
svc_probs = svc_model.predict_proba(X_test_tfidf)  # Probabilities
svc_preds = svc_model.predict(X_test_tfidf)       # Class predictions

# Print results
print("\nSVC Model Results:")
print("Accuracy Score:", accuracy_score(y_test, svc_preds))
print("\nClassification Report:\n", classification_report(y_test, svc_preds))
```

```python
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, svc_preds)

# Print confusion matrix
print("\nConfusion Matrix:\n", conf_matrix)

# Plotting the confusion matrix as a heatmap for better visualization
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["FAKE", "REAL"], yticklabels=["FAKE", "REAL"])
plt.title("Confusion Matrix for SVC Model")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

Figure 21. SVC model development

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Train the Logistic Regression model
logistic_model = LogisticRegression(max_iter=1000)
logistic_model.fit(X_train_tfidf, y_train)

# Get predictions and probabilities
logistic_probs = logistic_model.predict_proba(X_test_tfidf)  # Probabilities
logistic_preds = logistic_model.predict(X_test_tfidf)       # Class predictions

# Print results
print("\nLogistic Regression Model Results:")
print("Accuracy Score:", accuracy_score(y_test, logistic_preds))
print("\nClassification Report:\n", classification_report(y_test, logistic_preds))
```

```python
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, logistic_preds)

# Print confusion matrix
print("\nConfusion Matrix:\n", conf_matrix)

# Plotting the confusion matrix as a heatmap for better visualization
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["FAKE", "REAL"], yticklabels=["FAKE", "REAL"])
plt.title("Confusion Matrix for Logistic Regression Model")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

Figure 22. Logistics regression model development

```python
import time
import numpy as np
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.ensemble import StackingClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Masking
import matplotlib.pyplot as plt
import seaborn as sns

# Combine probabilities from only SVC, Logistic Regression, and Random Forest models
meta_features_train = np.hstack([
    svc_model.predict_proba(X_train_tfidf),
    logistic_model.predict_proba(X_train_tfidf),
    random_forest_model.predict_proba(X_train_tfidf)
])

meta_features_test = np.hstack([
    svc_probs,
    logistic_probs,
    rf_probs
])

# Reshape meta features to 3D (required for LSTM input)
meta_features_train = meta_features_train.reshape((meta_features_train.shape[0], meta_features_train.shape[1], 1))
meta_features_test = meta_features_test.reshape((meta_features_test.shape[0], meta_features_test.shape[1], 1))
```

```python
# Define LSTM meta-classifier model
lstm_meta_model = Sequential([
    Masking(mask_value=0.0, input_shape=(meta_features_train.shape[1], 1)),  # Handle padding (if any)
    LSTM(64, return_sequences=False, activation='tanh'),
    Dropout(0.2),
    Dense(1, activation='sigmoid')  # Binary classification
])

lstm_meta_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Convert y_train and y_test to binary labels (if not already done)
y_train_binary = np.where(y_train == "REAL", 1, 0)
y_test_binary = np.where(y_test == "REAL", 1, 0)

# Measure training time for the LSTM meta-model
start_time = time.time()
lstm_meta_model.fit(meta_features_train, y_train_binary, epochs=10, batch_size=32, verbose=1)
training_time = time.time() - start_time

print(f"\nTraining Time for LSTM Meta-Model: {training_time:.2f} seconds")

# Make predictions with the LSTM meta-classifier
stacking_preds_probs = lstm_meta_model.predict(meta_features_test)
stacking_preds = (stacking_preds_probs >= 0.5).astype(int)  # Thresholding at 0.5

# Convert numeric predictions back to original string labels
stacking_preds_strings = np.where(stacking_preds == 1, "REAL", "FAKE")

# Evaluate Stacking Classifier
print("\nStacking Classifier Results (LSTM Meta-Model):")
print("Accuracy Score:", accuracy_score(y_test, stacking_preds_strings))
print("\nClassification Report:\n", classification_report(y_test, stacking_preds_strings))

# Compute the confusion matrix
conf_matrix = confusion_matrix(y_test, stacking_preds_strings)

# Print confusion matrix
print("\nConfusion Matrix:\n", conf_matrix)

# Plotting the confusion matrix as a heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["FAKE", "REAL"], yticklabels=["FAKE", "REAL"])
plt.title("Confusion Matrix for Stacking Classifier (LSTM Meta-Model)")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

Figure 23. LSTM model development

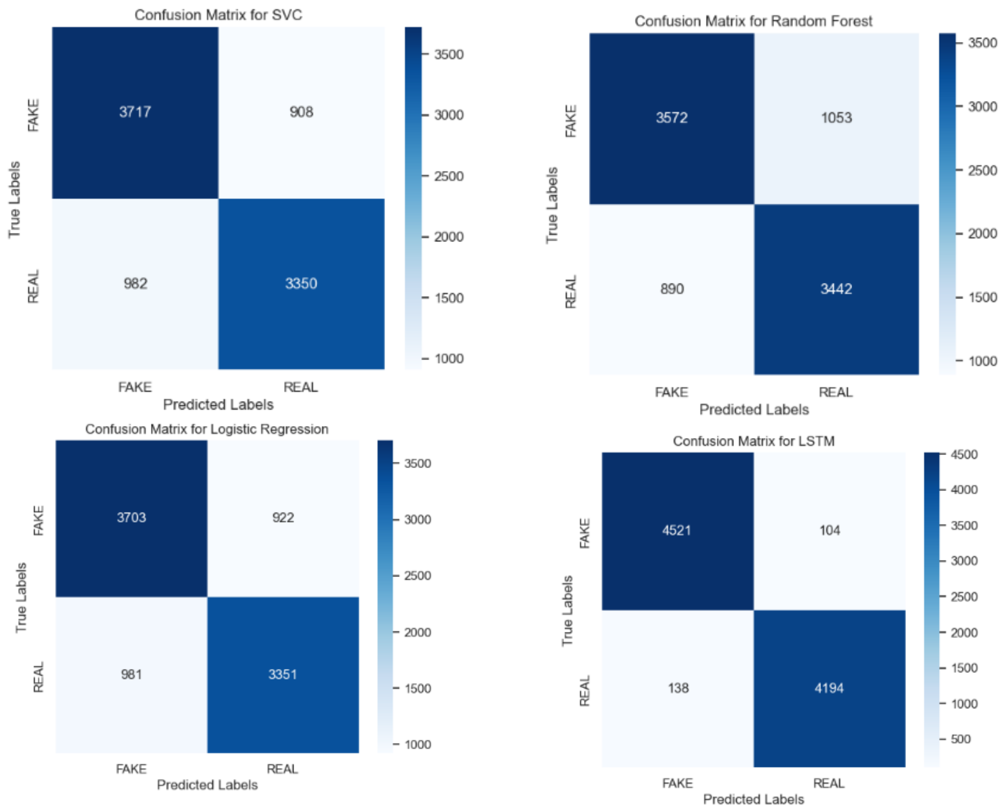

Figure 24. Confusion matrices

- **Ensemble model development**

```python
import numpy as np
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Combine probabilities by averaging (ensure to select probabilities for the positive class only)
average_probs = (svc_probs[:, 1] + logistic_probs[:, 1] + rf_probs[:, 1] + y_pred_probs[:, 0]) / 4

# Convert averaged probabilities to final predictions (binary thresholding at 0.5)
soft_voting_preds = np.where(average_probs >= 0.5, 1, 0)

# Convert numeric predictions to original string labels
soft_voting_preds_strings = label_encoder.inverse_transform(soft_voting_preds)  # Convert back to "FAKE"/"REAL"

# Evaluate the soft voting classifier
print("\nVoting Classifier Results (Soft Voting):")
print("Accuracy Score:", accuracy_score(y_test, soft_voting_preds_strings))
print("\nClassification Report:\n", classification_report(y_test, soft_voting_preds_strings))

# Compute the confusion matrix
conf_matrix = confusion_matrix(y_test, soft_voting_preds_strings)

# Print confusion matrix
print("\nConfusion Matrix:\n", conf_matrix)

# Plotting the confusion matrix as a heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["FAKE", "REAL"], yticklabels=["FAKE", "REAL"])
plt.title("Confusion Matrix for Soft Voting Classifier")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

Figure 25. Voting Classifier – Soft voting

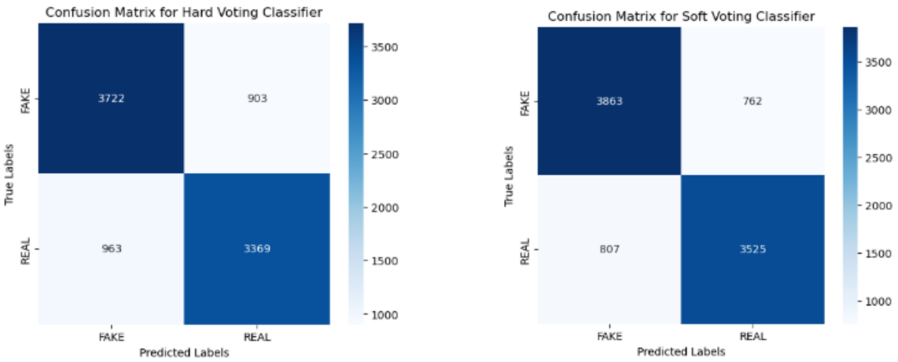

Figure 26. Confusion matrices – Voting classifier

```python
from scipy.stats import mode
import numpy as np
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import time
import matplotlib.pyplot as plt
import seaborn as sns

# Start timer
start_time = time.time()

# Ensure all predictions have the same shape
svc_preds = svc_preds.flatten()
logistic_preds = logistic_preds.flatten()
rf_preds = rf_preds.flatten()
y_pred = y_pred.flatten()

# Combine predictions
all_preds = np.vstack([svc_preds, logistic_preds, rf_preds, y_pred]).T

# Perform majority voting
voting_preds = mode(all_preds, axis=1).mode.flatten()

# End timer
end_time = time.time()

# Calculate training time
training_time = end_time - start_time
print(f"\nHard Voting Classifier Training Time: {training_time:.2f} seconds")

# Evaluate the voting classifier
print("\nVoting Classifier Results (Hard Voting):")
print("Accuracy Score:", accuracy_score(y_test, voting_preds))
print("\nClassification Report:\n", classification_report(y_test, voting_preds))

# Compute the confusion matrix
conf_matrix = confusion_matrix(y_test, voting_preds)

# Print confusion matrix
print("\nConfusion Matrix:\n", conf_matrix)

# Plotting the confusion matrix as a heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["FAKE", "REAL"], yticklabels=["FAKE", "REAL"])
plt.title("Confusion Matrix for Hard Voting Classifier")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

Figure 27. Voting classifier – Hard Voting

- **Stacking Classifier model development**

```python
import time
import numpy as np
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Combine probabilities from all models for Stacking
meta_features_train = np.hstack([
    svc_model.predict_proba(X_train_tfidf),
    random_forest_model.predict_proba(X_train_tfidf),
    lstm_model.predict(X_train_padded)
])

meta_features_test = np.hstack([
    svc_probs,
    rf_probs,
    y_pred_probs
])

# Initialize the Stacking Classifier
meta_model = LogisticRegression(max_iter=1000)

# Measure training time
start_time = time.time()
meta_model.fit(meta_features_train, y_train)
training_time = time.time() - start_time

print(f"\nTraining Time for Stacking Classifier (Logistic Regression): {training_time:.2f} seconds")

# Make predictions
stacking_preds = meta_model.predict(meta_features_test)

# Evaluate Stacking Classifier
print("\nStacking Classifier Results:")
print("Accuracy Score:", accuracy_score(y_test, stacking_preds))
print("\nClassification Report:\n", classification_report(y_test, stacking_preds))

# Compute the confusion matrix
conf_matrix = confusion_matrix(y_test, stacking_preds)

# Print confusion matrix
print("\nConfusion Matrix:\n", conf_matrix)

# Plotting the confusion matrix as a heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["FAKE", "REAL"], yticklabels=["FAKE", "REAL"])
plt.title("Confusion Matrix for Stacking Classifier")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

Figure 28. Stacking classifier – Logistic regression as meta model

```python
import time
import numpy as np
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.ensemble import StackingClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Masking
import matplotlib.pyplot as plt
import seaborn as sns

# Combine probabilities from only SVC, Logistic Regression, and Random Forest models
meta_features_train = np.hstack([
    svc_model.predict_proba(X_train_tfidf),
    logistic_model.predict_proba(X_train_tfidf),
    random_forest_model.predict_proba(X_train_tfidf)
])

meta_features_test = np.hstack([
    svc_probs,
    logistic_probs,
    rf_probs
])

# Reshape meta features to 3D (required for LSTM input)
meta_features_train = meta_features_train.reshape((meta_features_train.shape[0], meta_features_train.shape[1], 1))
meta_features_test = meta_features_test.reshape((meta_features_test.shape[0], meta_features_test.shape[1], 1))

# Define LSTM meta-classifier model
lstm_meta_model = Sequential([
    Masking(mask_value=0.0, input_shape=(meta_features_train.shape[1], 1)),  # Handle padding (if any)
    LSTM(64, return_sequences=False, activation='tanh'),
    Dropout(0.2),
    Dense(1, activation='sigmoid')  # Binary classification
])

lstm_meta_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Convert y_train and y_test to binary labels (if not already done)
y_train_binary = np.where(y_train == "REAL", 1, 0)
y_test_binary = np.where(y_test == "REAL", 1, 0)

# Measure training time for the LSTM meta-model
start_time = time.time()
lstm_meta_model.fit(meta_features_train, y_train_binary, epochs=10, batch_size=32, verbose=1)
training_time = time.time() - start_time

print(f"\nTraining Time for LSTM Meta-Model: {training_time:.2f} seconds")

# Make predictions with the LSTM meta-classifier
stacking_preds_probs = lstm_meta_model.predict(meta_features_test)
stacking_preds = (stacking_preds_probs >= 0.5).astype(int)  # Thresholding at 0.5

# Convert numeric predictions back to original string labels
stacking_preds_strings = np.where(stacking_preds == 1, "REAL", "FAKE")

# Evaluate Stacking Classifier
print("\nStacking Classifier Results (LSTM Meta-Model):")
print("Accuracy Score:", accuracy_score(y_test, stacking_preds_strings))
print("\nClassification Report:\n", classification_report(y_test, stacking_preds_strings))

# Compute the confusion matrix
conf_matrix = confusion_matrix(y_test, stacking_preds_strings)

# Print confusion matrix
print("\nConfusion Matrix:\n", conf_matrix)

# Plotting the confusion matrix as a heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["FAKE", "REAL"], yticklabels=["FAKE", "REAL"])
plt.title("Confusion Matrix for Stacking Classifier (LSTM Meta-Model)")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

Figure 29. Stacking classifier – LSTM as meta model

```python
import time
from sklearn.svm import SVC
import numpy as np
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Combine probabilities from all models for Stacking
meta_features_train = np.hstack([
    logistic_model.predict_proba(X_train_tfidf),
    random_forest_model.predict_proba(X_train_tfidf),
    lstm_model.predict(X_train_padded)
])

meta_features_test = np.hstack([
    logistic_probs,
    rf_probs,
    y_pred_probs
])

# Stacking Classifier with SVC as the Meta-Model
meta_model = SVC(kernel='linear', probability=True, random_state=42)

# Measure training time
start_time = time.time()  # Start the timer
meta_model.fit(meta_features_train, y_train)
training_time = time.time() - start_time  # Calculate the elapsed time

# Predictions and Evaluation
stacking_preds = meta_model.predict(meta_features_test)

# Print Training Time
print(f"Training Time for Stacking Classifier (SVC as Meta-Model): {training_time:.2f} seconds")

# Evaluate Stacking Classifier
print("\nStacking Classifier Results (SVC as Meta-Model):")
print("Accuracy Score:", accuracy_score(y_test, stacking_preds))
print("\nClassification Report:\n", classification_report(y_test, stacking_preds))

# Compute the confusion matrix
conf_matrix = confusion_matrix(y_test, stacking_preds)

# Print confusion matrix
print("\nConfusion Matrix:\n", conf_matrix)

# Plotting the confusion matrix as a heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["FAKE", "REAL"], yticklabels=["FAKE", "REAL"])
plt.title("Confusion Matrix for Stacking Classifier (SVC as Meta-Model)")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

Figure 30. Stacking Classifier – SVC as meta model

- **Overfitting and Balance detection in individual models**

```python
# Model Analysis Code for Overfitting and Balance Detection

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import cross_val_score

# List of models and their respective predictions
models = {
    "Logistic Regression": {
        "model": logistic_model,
        "preds": logistic_preds,
        "probs": logistic_probs,
    },
    "Random Forest": {
        "model": random_forest_model,
        "preds": rf_preds,
        "probs": rf_probs,
    },
    "SVC": {
        "model": svc_model,
        "preds": svc_preds,
        "probs": svc_probs,
    },
    "LSTM": {
        "model": lstm_model,
        "preds": y_pred,
        "probs": y_pred_probs,
    },
}
```

```python
#Compare Model Accuracies
plt.figure(figsize=(10, 6))
plt.bar(accuracy_scores.keys(), accuracy_scores.values(), color="skyblue")
plt.xlabel("Models")
plt.ylabel("Accuracy")
plt.title("Comparison of Model Accuracies")
plt.ylim(0.4, 1)  # Accuracy is between 0 and 1
plt.xticks(rotation=15)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

- **Cross validation and overfitting check for trained models**

```python
#Cross-Validation Consistency Check for Traditional Models
for model_name, details in models.items():
    if model_name != "LSTM":  # LSTM cannot use scikit learn's cross_val_score
        print(f"\nCross-Validation for {model_name}:")
        try:
            cv_scores = cross_val_score(details["model"], X_train_tfidf, y_train, cv=5)
            print("CV Scores:", cv_scores)
            print("Mean CV Accuracy:", cv_scores.mean())
            print("CV Standard Deviation:", cv_scores.std())
            if cv_scores.std() > 0.05:
                print("Warning: High variability in CV scores detected!")
        except Exception as e:
            print(f"Error in Cross Validation for {model_name}: {e}")
```

```python
#Detect Overfitting by Comparing Train-Test Gap
def calculate_train_test_gap(model, X_train, y_train, X_test, y_test):
    train_preds = model.predict(X_train)
    test_preds = model.predict(X_test)

    train_acc = accuracy_score(y_train, train_preds)
    test_acc = accuracy_score(y_test, test_preds)

    print(f"\nTrain Accuracy: {train_acc:.4f}")
    print(f"Test Accuracy: {test_acc:.4f}")
    print(f"Train-Test Gap: {train_acc - test_acc:.4f}")
    if (train_acc - test_acc) > 0.1:
        print("Warning: Overfitting detected!")

# Evaluate train-test gap for traditional models
for model_name, details in models.items():
    if model_name != "LSTM":  # LSTM evaluation requires TensorFlow-specific methods
        print(f"\nOverfitting Check for {model_name}:")
        calculate_train_test_gap(details["model"], X_train_tfidf, y_train, X_test_tfidf, y_test)
```

Figure 31. Overfitting and balance detection of models

```python
import numpy as np
from matplotlib import pyplot as plt

# Ensure X_train and y_train_encoded are Numpy arrays
X_train = np.array(X_train_padded)
y_train_encoded = np.array(y_train_encoded)

# Train LSTM model with validation split
history = lstm_model.fit(
    X_train,
    y_train_encoded,
    validation_split=0.2,
    epochs=20,
    batch_size=32,
    verbose=1
)

# Plot training and validation accuracy/loss
plt.figure(figsize=(12, 5))

# Accuracy Plot
plt.subplot(1, 2, 1)
plt.plot(history.history["accuracy"], label="Training Accuracy")
plt.plot(history.history["val_accuracy"], label="Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("Training vs Validation Accuracy")
plt.legend()

# Loss Plot
plt.subplot(1, 2, 2)
plt.plot(history.history["loss"], label="Training Loss")
plt.plot(history.history["val_loss"], label="Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Training vs Validation Loss")
plt.legend()

plt.tight_layout()
plt.show()
```

Figure 32. Overfitting detection for LSTM model

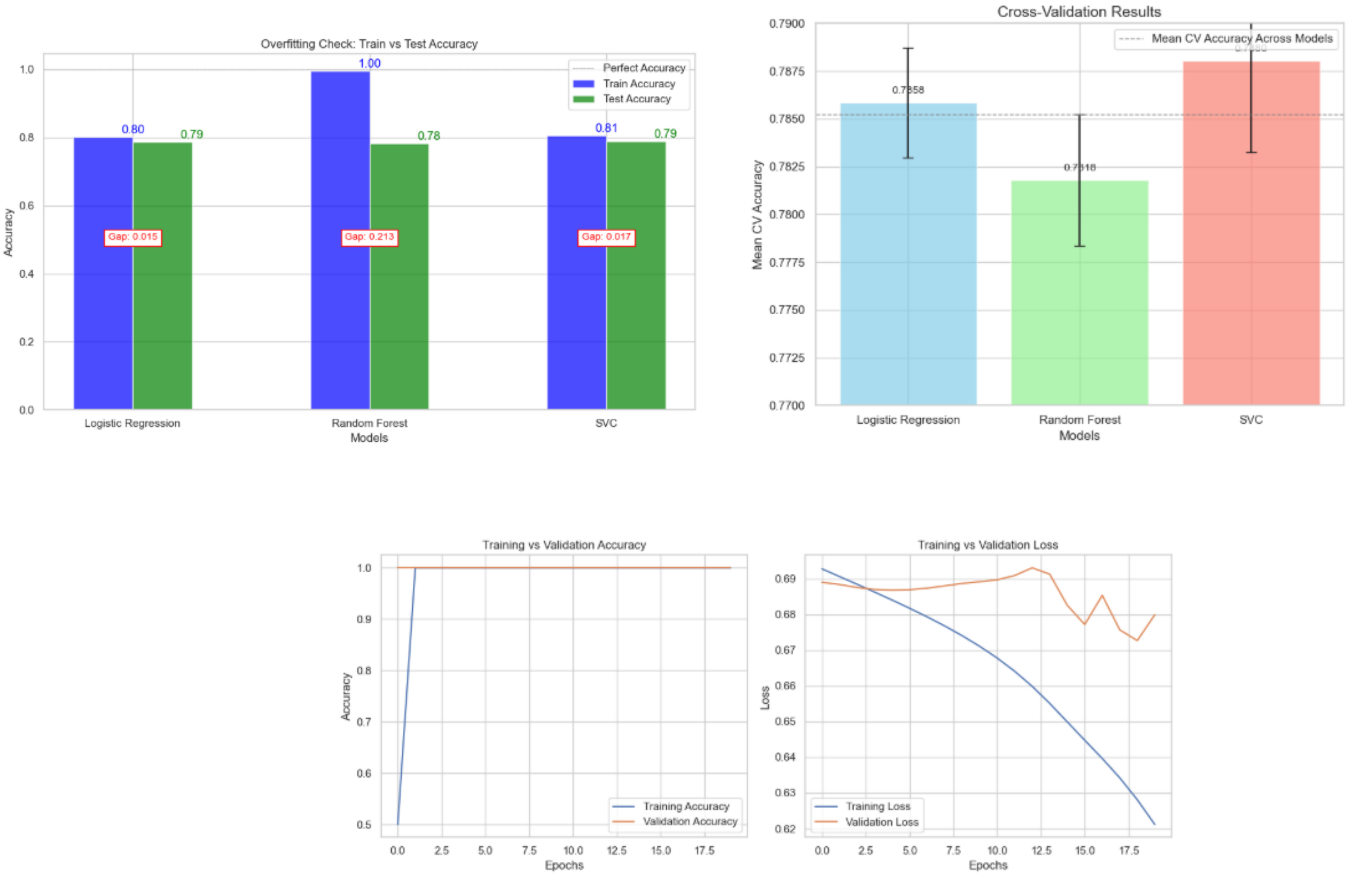

Figure 33. Overfitting analysis of the individual models