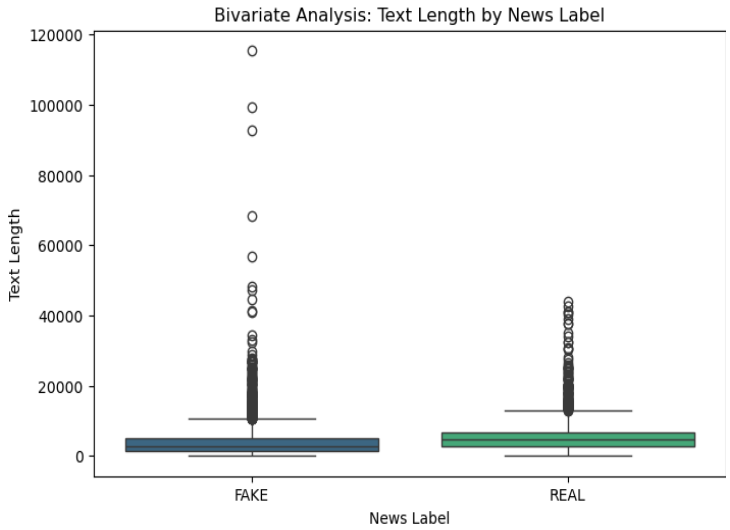
Data Collection and Preprocessing Phase

Date	4 November 2024
Team ID	SWTID1726834817
Project Title	Fake News Analysis in social media using NLP
Maximum Marks	6 Marks

Data Exploration and Preprocessing Report

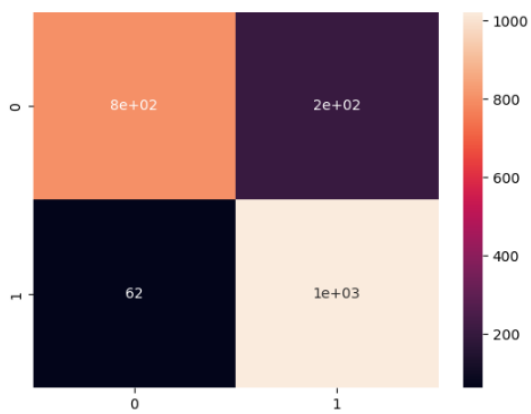
Dataset variables will be statistically analyzed to identify patterns and outliers, with Python employed for preprocessing tasks like normalization and feature engineering. Data cleaning will address missing values and outliers, ensuring quality for subsequent analysis and modeling, and forming a strong foundation for insights and predictions.

Section	**Description**
Data Overview	<u>Dimension:</u> 7796rows × 4 columns <u>Descriptive statistics:</u> Unnamed: 0 / text_length / avg_word_length count: 6335.000000 / 6335.000000 / 6299.000000 mean: 5280.415627 / 4707.250355 / 5.083970 std: 3038.503953 / 5090.956446 / 0.714039 min: 2.000000 / 1.000000 / 3.680000 25%: 2674.500000 / 1741.500000 / 4.881325 50%: 5271.000000 / 3642.000000 / 5.049355 75%: 7901.000000 / 6192.000000 / 5.226158 max: 10557.000000 / 115372.000000 / 48.496000
Univariate Analysis	

	
Bivariate Analysis	

Multivariate Analysis	

Outliers and Anomalies	-
Data Preprocessing Code Screenshots	
Loading Data	`df = pd.read_csv('news.csv')` `df.head()` |	Unnamed: 0	title	text	label | |---|---|---|---|---| | 0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE | | 1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE | | 2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL | | 3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE | | 4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL |

Handling Missing Data	[41]:
	``python
	# Step 1: Identifying missing values
	print("Missing values per column:")
	print(df.isnull().sum())
	
	# Step 2: Handling missing values
	df.fillna({'text': ' '}, inplace=True)
	
	# Drop any rows where 'label' is missing (if it exists)
	df.dropna(subset=['label'], inplace=True)
	
	# Step 3: Proceed with the rest of the analysis as before
	X = df['text']
	y = df['label']
	
	# Split dataset
	X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=53)
	
	# Vectorize text data
	count_vectorizer = CountVectorizer(stop_words='english')
	count_train = count_vectorizer.fit_transform(X_train)
	count_test = count_vectorizer.transform(X_test)
	
	tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
	tfidf_train = tfidf_vectorizer.fit_transform(X_train)
	tfidf_test = tfidf_vectorizer.transform(X_test)
	
	print("CountVectorizer sample features:", count_vectorizer.get_feature_names_out()[:10])
	print("TfidfVectorizer sample features:", tfidf_vectorizer.get_feature_names_out()[:10])
	``
	``
	Missing values per column:
	Unnamed: 0 0
	title 0
	text 0
	label 0
	dtype: int64
	``
	[71]:
	``python
	# Predict and evaluate
	pred = nb_classifier.predict(tfidf_test_scaled)
	score = accuracy_score(y_test, pred)
	print("Accuracy:", score)
	
	# Confusion matrix
	cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
	sns.heatmap(cm, annot=True)
	``
	``
	Accuracy: 0.8723098995695839
	``
	[71]: <Axes: >
	 

Data Transformation	

Feature Engineering	

```python
# 1. Text length feature
df['text_length'] = df['text'].apply(len)

# 2. Word count feature
df['word_count'] = df['text'].apply(lambda x: len(x.split()))

# 3. Average word length feature
df['avg_word_length'] = df['text'].apply(lambda x: sum(len(word) for word in x.split()) / max(len(x.split()), 1))

# Split dataset
X = df['text']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=53)

# Vectorize text data using Tfidf
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
tfidf_train = tfidf_vectorizer.fit_transform(X_train)
tfidf_test = tfidf_vectorizer.transform(X_test)

# Add new features to the Tfidf vectors (dense conversion)
X_train_dense = pd.DataFrame(tfidf_train.toarray(), columns=tfidf_vectorizer.get_feature_names_out())
X_train_dense['text_length'] = X_train.apply(len).values
X_train_dense['word_count'] = X_train.apply(lambda x: len(x.split())).values
X_train_dense['avg_word_length'] = X_train.apply(lambda x: sum(len(word) for word in x.split()) / max(len(x.split()), 1)).values

X_test_dense = pd.DataFrame(tfidf_test.toarray(), columns=tfidf_vectorizer.get_feature_names_out())
X_test_dense['text_length'] = X_test.apply(len).values
X_test_dense['word_count'] = X_test.apply(lambda x: len(x.split())).values
X_test_dense['avg_word_length'] = X_test.apply(lambda x: sum(len(word) for word in x.split()) / max(len(x.split()), 1)).values

# Save processed data and vectorizer for reuse
processed_data = {
    'X_train': X_train_dense,
    'X_test': X_test_dense,
    'y_train': y_train,
    'y_test': y_test,
    'vectorizer': tfidf_vectorizer
}
```

Save Processed Data	

```python
# Save with pickle
with open('processed_data.pkl', 'wb') as file:
    pickle.dump(processed_data, file)

print("Processed data saved successfully.")
```

```
Processed data saved successfully.
```