Fake News Detection Project

This project implements a machine learning system to detect fake news using GPU-accelerated libraries (RAPIDS) for improved performance. The system processes text data, extracts features using TF-IDF vectorization, and trains various machine learning models to classify news articles as real or fake.

Data Analysis and Visualization

Data Loading and Initial Setup

The project begins by importing necessary libraries and loading the datasets. We use RAPIDS cuDF for GPU-accelerated data processing, which significantly speeds up operations on large datasets. The datasets consist of real and fake news articles.

```
import cudf  # RAPIDS cuDF for GPU-accelerated dataframes
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import re
from cuml.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
import cupy as cp
from cuml.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, roc_curve, auc
import cuml
from cuml.model_selection import train_test_split
from cuml.metrics import accuracy_score, roc_auc_score
from cuml.linear_model import LogisticRegression
from cuml.ensemble import RandomForestClassifier
import numpy as np
from cuml.svm import SVC
import xgboost as xgb
import joblib


# Load the datasets
df_real = cudf.read_csv("True.csv")
df_fake = cudf.read_csv("Fake.csv")
```

Data Exploration and Visualization

This section performs initial data exploration, including:

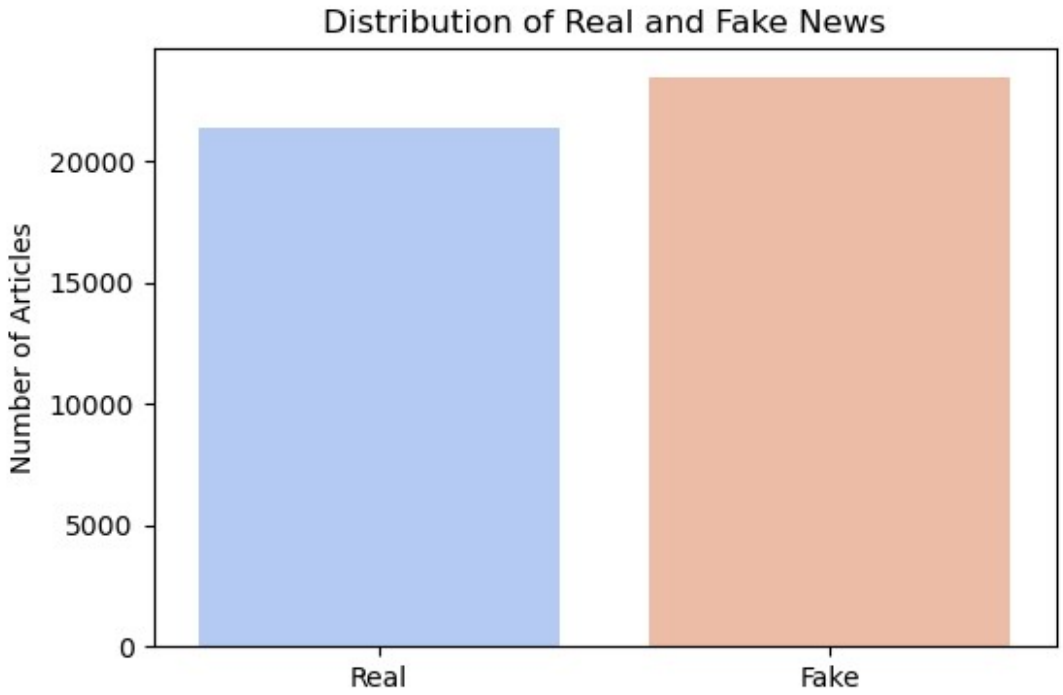
- Displaying sample data
- Checking for missing values
- Analyzing dataset shapes
- Creating visualizations to understand the data distribution

```
# Display the first few rows
print("Real News Dataset:")
print(df_real.head())

print("\nFake News Dataset:")
print(df_fake.head())

# Check for missing values
print("\nMissing values in Real News:")
print(df_real.isnull().sum())

print("\nMissing values in Fake News:")
print(df_fake.isnull().sum())

# Check dataset shapes
print(f"\nShape of Real News: {df_real.shape}")
print(f"Shape of Fake News: {df_fake.shape}")
```

```
Real News Dataset:
                                               title  \
0  As U.S. budget fight looms, Republicans flip t...
1  U.S. military to accept transgender recruits o...
2  Senior U.S. Republican senator: 'Let Mr. Muell...
3  FBI Russia probe helped by Australian diplomat...
4  Trump wants Postal Service to charge 'much mor...

                                                text       subject  \
0  WASHINGTON (Reuters) - The head of a conservat...  politicsNews
1  WASHINGTON (Reuters) - Transgender people will...  politicsNews
2  WASHINGTON (Reuters) - The special counsel inv...  politicsNews
3  WASHINGTON (Reuters) - Trump campaign adviser ...  politicsNews
4  SEATTLE/WASHINGTON (Reuters) - President Donal...  politicsNews

                 date
0  December 31, 2017
1  December 29, 2017
2  December 31, 2017
3  December 30, 2017
4  December 29, 2017

Fake News Dataset:
                                               title  \
0   Donald Trump Sends Out Embarrassing New Year’...
1   Drunk Bragging Trump Staffer Started Russian ...
2   Sheriff David Clarke Becomes An Internet Joke...
```

```
3   Trump Is So Obsessed He Even Has Obama’s Name...
4   Pope Francis Just Called Out Donald Trump Dur...

                                                text subject  \
0  Donald Trump just couldn t wish all Americans ...    News
1  House Intelligence Committee Chairman Devin Nu...    News
2  On Friday, it was revealed that former Milwauk...    News
3  On Christmas day, Donald Trump announced that ...    News
4  Pope Francis used his annual Christmas Day mes...    News

                date
0  December 31, 2017
1  December 31, 2017
2  December 30, 2017
3  December 29, 2017
4  December 25, 2017

Missing values in Real News:
title      0
text       0
subject    0
date       0
dtype: int64

Missing values in Fake News:
title      0
text       0
subject    0
date       0
dtype: int64

Shape of Real News: (21417, 4)
Shape of Fake News: (23481, 4)


# Convert cuDF to pandas for visualization
df_real_pandas = df_real.to_pandas()
df_fake_pandas = df_fake.to_pandas()

# Plot distribution of Real vs. Fake news
plt.figure(figsize=(6,4))
sns.barplot(x=["Real", "Fake"], y=[len(df_real), len(df_fake)],
palette="coolwarm")
plt.title("Distribution of Real and Fake News")
plt.ylabel("Number of Articles")
plt.show()


/tmp/ipykernel_4089/2393645841.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
```

```
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=["Real", "Fake"], y=[len(df_real), len(df_fake)],
palette="coolwarm")
```

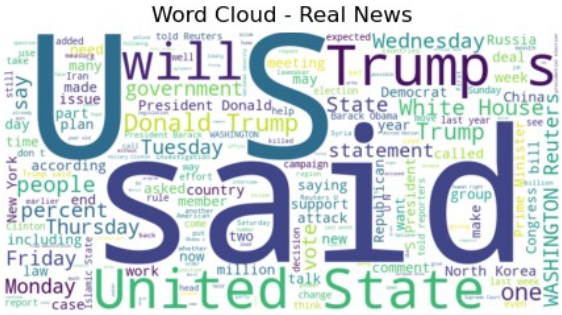


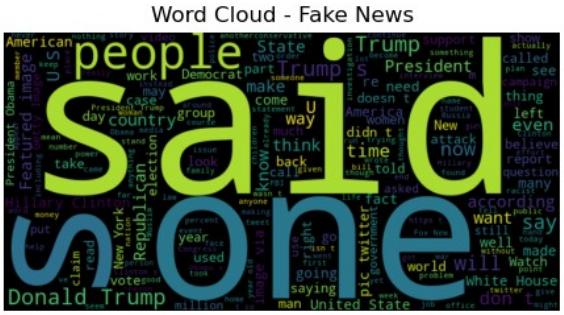
Word Cloud Analysis

Word clouds are generated to visualize the most frequent words in both real and fake news articles. This helps in understanding the language patterns and common terms used in each category.

```
# Generate Word Clouds
real_text = " ".join(df_real_pandas["text"].dropna())
fake_text = " ".join(df_fake_pandas["text"].dropna())

wordcloud_real = WordCloud(width=800, height=400,
background_color='white').generate(real_text)
wordcloud_fake = WordCloud(width=800, height=400,
background_color='black').generate(fake_text)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(wordcloud_real, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud - Real News")
```

```
plt.subplot(1, 2, 2)
plt.imshow(wordcloud_fake, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud - Fake News")

plt.show()
```

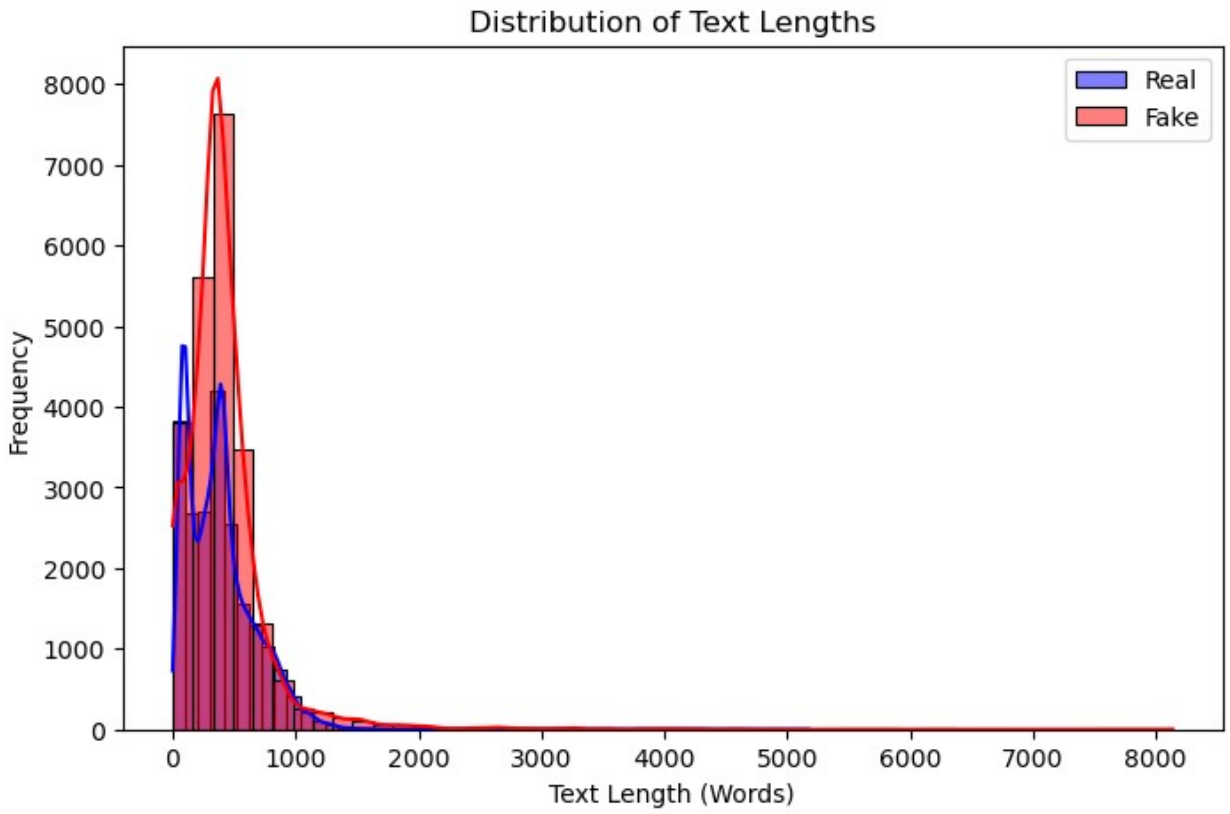




Text Length Analysis

This section analyzes the distribution of text lengths in both real and fake news articles to identify any patterns or differences in article length between the two categories.

```
# Text Length Distribution
df_real_pandas['text_length'] = df_real_pandas['text'].apply(lambda x:
len(x.split()))
df_fake_pandas['text_length'] = df_fake_pandas['text'].apply(lambda x:
len(x.split()))

plt.figure(figsize=(8,5))
sns.histplot(df_real_pandas['text_length'], bins=50, color='blue',
label='Real', kde=True)
sns.histplot(df_fake_pandas['text_length'], bins=50, color='red',
label='Fake', kde=True)
plt.xlabel("Text Length (Words)")
plt.ylabel("Frequency")
plt.legend()
plt.title("Distribution of Text Lengths")
plt.show()
```



Data Preprocessing

This section prepares the data for machine learning by:

- Combining real and fake news datasets
- Adding labels (1 for real, 0 for fake)
- Cleaning text data
- Performing TF-IDF vectorization for feature extraction

```
# Label datasets: 1 for Real, 0 for Fake
df_real["label"] = 1
df_fake["label"] = 0

# Combine datasets
df = cudf.concat([df_real, df_fake]).reset_index(drop=True)

# Drop unnecessary columns safely
df = df.drop(columns=["subject", "date"], errors="ignore")

print("Columns in df after dropping:", df.columns)
```

```
Columns in df after dropping: Index(['title', 'text', 'label'],
dtype='object')
```

```
# Function to clean text
def clean_text(text):
    text = text.str.lower()  # Convert to lowercase
    text = text.str.replace(r'\W', ' ', regex=True)  # Remove special
characters
    text = text.str.replace(r'\s+', ' ', regex=True)  # Remove extra
spaces
    return text.str.strip()

df["text"] = clean_text(df["text"])  # No need for `.apply()`

# TF-IDF Vectorization
vectorizer = TfidfVectorizer(stop_words="english", max_features=5000)
X = vectorizer.fit_transform(df["text"])

# Convert sparse matrix to CuPy array (or cuDF DataFrame)
X = cp.asarray(X.todense())  # Ensure compatibility with cuML

y = df["label"]

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

print(f"Train size: {X_train.shape}, Test size: {X_test.shape}")
```

```
Train size: (35919, 5000), Test size: (8979, 5000)
```

Model Training and Evaluation

This section implements the Random Forest Classifier using cuML for GPU-accelerated training and evaluation. The model's performance is measured using accuracy and AUC metrics.

```
import cuml
from cuml.ensemble import RandomForestClassifier as cuRF
from cuml.metrics import accuracy_score, roc_auc_score

# Define the model using cuML
model = cuRF(n_estimators=100)

# Store results
results = {}

# Training the model
print("Training Random Forest Classifier using cuML...")
```

```
model.fit(X_train, y_train)

# After training, print a confirmation
print("Training completed.")

# Predictions
y_pred = model.predict(X_test)

# Calculate accuracy and AUC
accuracy = accuracy_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred)

# Store the results
results["Random Forest"] = {"Accuracy": accuracy, "AUC": auc}

# Print Results
for model_name, scores in results.items():
    print(f"{model_name} - Accuracy: {scores['Accuracy']:.4f}, AUC:
{scores['AUC']:.4f}")
```

```
Training Random Forest Classifier using cuML...
Training completed.
Random Forest - Accuracy: 0.9939, AUC: 0.9939
```

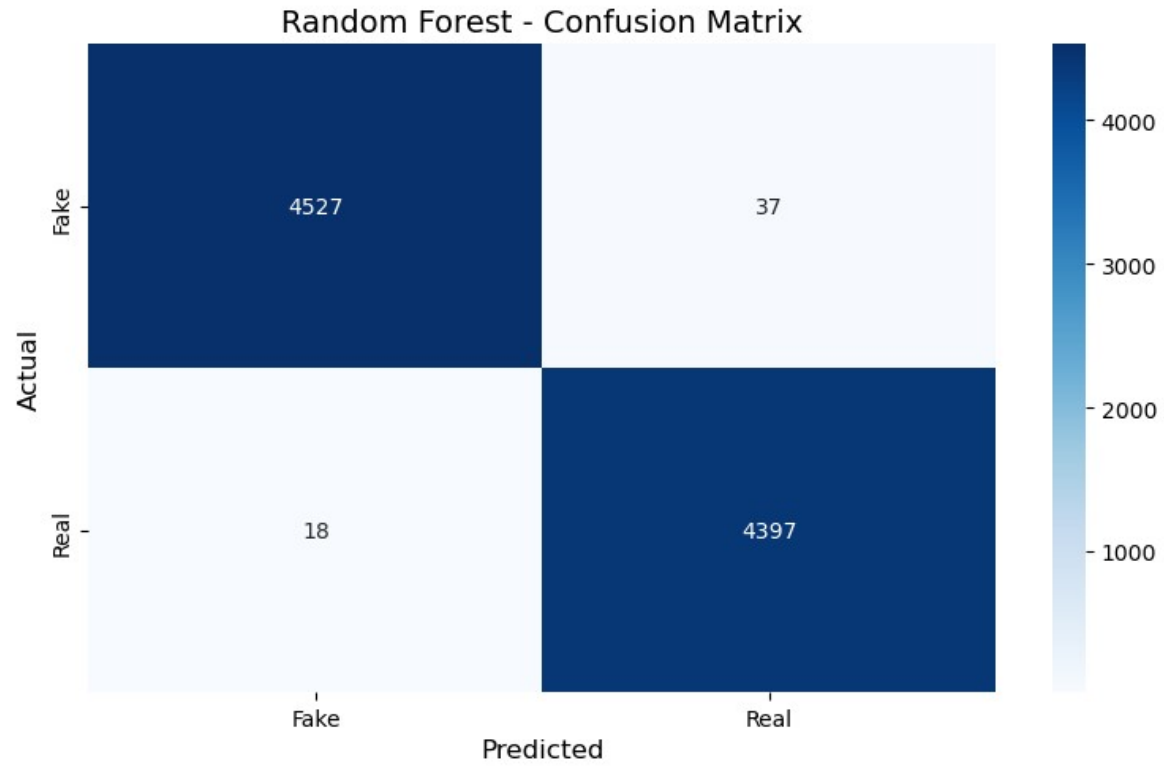
Model Performance Visualization

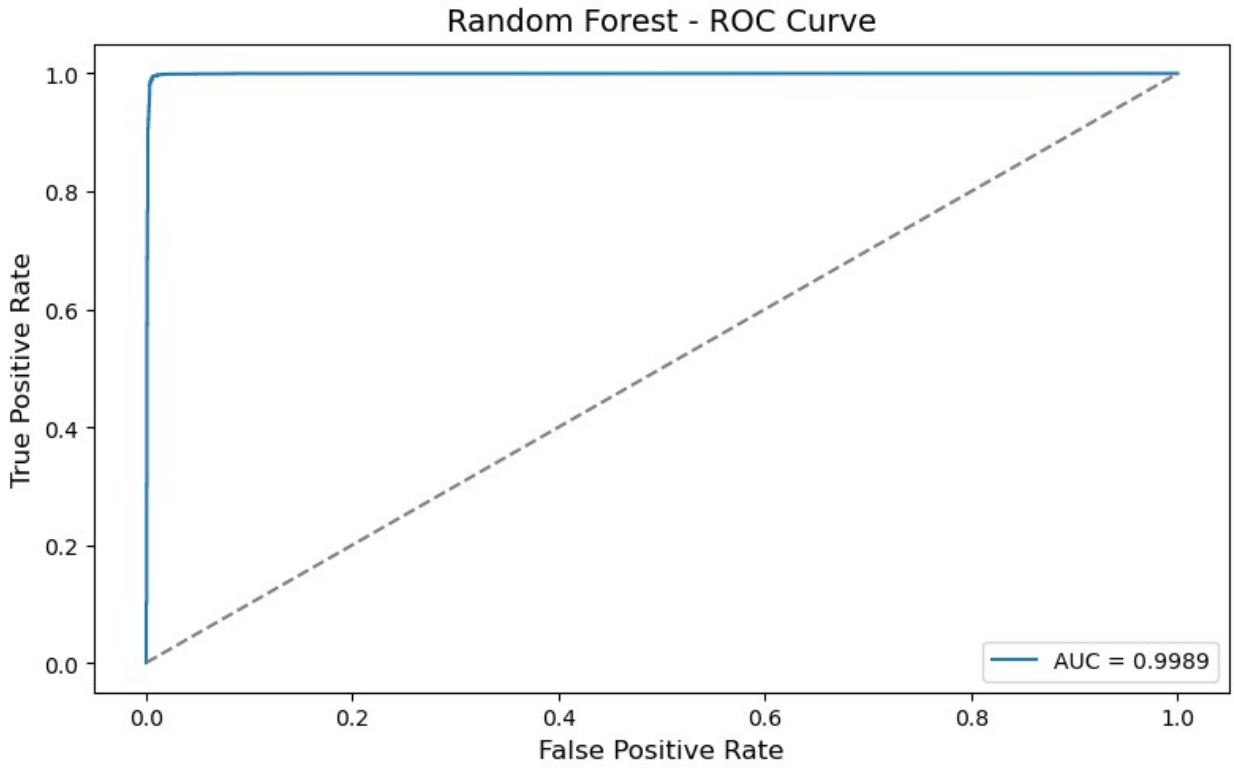
This section creates visualizations to better understand the model's performance:

- **Confusion Matrix**: Shows true positives, true negatives, false positives, and false negatives
- **ROC Curve**: Illustrates the trade-off between true positive rate and false positive rate

```
from sklearn.metrics import roc_curve, auc
# Predictions
y_pred = model.predict(X_test)
y_pred = y_pred.get() if isinstance(y_pred, cp.ndarray) else y_pred  #
Convert CuPy to NumPy
y_test_np = y_test.get() if isinstance(y_test, cp.ndarray) else
y_test.to_numpy()  # Convert CuPy/cuDF to NumPy

# Create plots
plt.figure(figsize=(8, 10))  # Adjusted for better spacing

# Confusion Matrix
cm = confusion_matrix(y_test_np, y_pred)
cm = cm.astype(int)  # Ensure matrix is integer type

# ROC Curve
if hasattr(model, "predict_proba"):
```

```
    y_pred_proba = model.predict_proba(X_test)[:, 1]
    y_pred_proba = y_pred_proba.get() if isinstance(y_pred_proba,
cp.ndarray) else y_pred_proba
else:
    y_pred_proba = y_pred  # Ensure consistency

fpr, tpr, _ = roc_curve(y_test_np, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Plot Confusion Matrix
plt.subplot(2, 1, 1)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=["Fake", "Real"], yticklabels=["Fake", "Real"])
plt.title(f"Random Forest - Confusion Matrix", fontsize=14)
plt.xlabel("Predicted", fontsize=12)
plt.ylabel("Actual", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)

# Plot ROC Curve
plt.subplot(2, 1, 2)
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.4f}")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray")
plt.title(f"Random Forest - ROC Curve", fontsize=14)
plt.xlabel("False Positive Rate", fontsize=12)
plt.ylabel("True Positive Rate", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.legend()

# Adjust spacing
plt.subplots_adjust(hspace=0.4)  # Increase spacing between subplots
plt.tight_layout()
plt.show()
```





Detailed Performance Metrics

This section calculates and displays comprehensive performance metrics including:

- **Accuracy**
- **Precision**
- **Recall**
- **F1 Score**

```
from cuml.metrics import accuracy_score  # GPU-accelerated accuracy
from sklearn.metrics import precision_score, recall_score, f1_score  # CPU-based metrics
import cupy as cp

# Predictions
y_pred = model.predict(X_test)
y_pred = y_pred.get() if isinstance(y_pred, cp.ndarray) else y_pred  # Convert to NumPy
y_test_np = y_test.get() if isinstance(y_test, cp.ndarray) else y_test.to_numpy()  # Convert to NumPy

# Compute Metrics
accuracy = accuracy_score(y_test_np, y_pred)
precision = precision_score(y_test_np, y_pred)
recall = recall_score(y_test_np, y_pred)
f1 = f1_score(y_test_np, y_pred)

# Print Results
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```

```
Accuracy: 0.9939
Precision: 0.9917
Recall: 0.9959
F1 Score: 0.9938
```

Prediction Function

This section implements a utility function to predict whether a given news article is real or fake. The function:

- Preprocesses the input text
- Converts it to feature vectors
- Makes predictions using the trained model

```
import cupy as cp
import cudf
```

```
import numpy as np

def preprocess_text(text, vectorizer):
    """Preprocess text and convert it into a dense feature vector."""
    text_series = cudf.Series([text])  # Convert to cuDF Series for
GPU processing
    text_vector = vectorizer.transform(text_series)  # Transform using
trained vectorizer

    # Convert sparse matrix to dense DataFrame
    if hasattr(text_vector, "todense"):
        text_vector = text_vector.todense()  # Convert sparse to dense
if needed

    return cudf.DataFrame(text_vector)  # Convert to cuDF DataFrame

def predict_news(text, model, vectorizer):
    """Predict whether the given news is Fake or Real."""
    # Preprocess and convert text to feature vector
    text_vector = preprocess_text(text, vectorizer)

    # Predict
    y_pred = model.predict(text_vector)

    # Convert CuPy array to NumPy if needed
    y_pred = y_pred.get() if isinstance(y_pred, cp.ndarray) else
y_pred

    # Interpret prediction
    return "Fake News" if y_pred[0] == 0 else "Real News"

# Example usage
news_article = "Breaking news: Scientists discover a new planet in our
solar system!"
prediction = predict_news(news_article, model, vectorizer)
print(f"Prediction: {prediction}")
```

```
Prediction: Fake News
```

Project Summary

This project successfully implements a fake news detection system using GPU-accelerated machine learning. Key features include:

- Efficient data processing using RAPIDS libraries
- Comprehensive data analysis and visualization
- Advanced text preprocessing and feature extraction

- High-performance model training and evaluation
- Detailed performance metrics and visualizations
- Practical prediction function for real-world use