Sarcasm Detection from News Headline Report

Submitted By:

- **Minhaj Asghar**

---

1. Introduction

This project focuses on building a **Sarcasm Detection System** using deep learning. The goal is to determine whether a given headline expresses sarcasm or not. Detecting sarcasm is a challenging task because sarcastic sentences can often resemble genuine ones in structure and tone. This model helps solve that problem by training a deep learning algorithm on labeled sarcastic and non-sarcastic data.

---

2. Dataset Used

We used a dataset of news headlines in JSON format. Each entry in the dataset contains:

- A **headline** (text)
- A **label** (1 for sarcastic, 0 for not sarcastic)

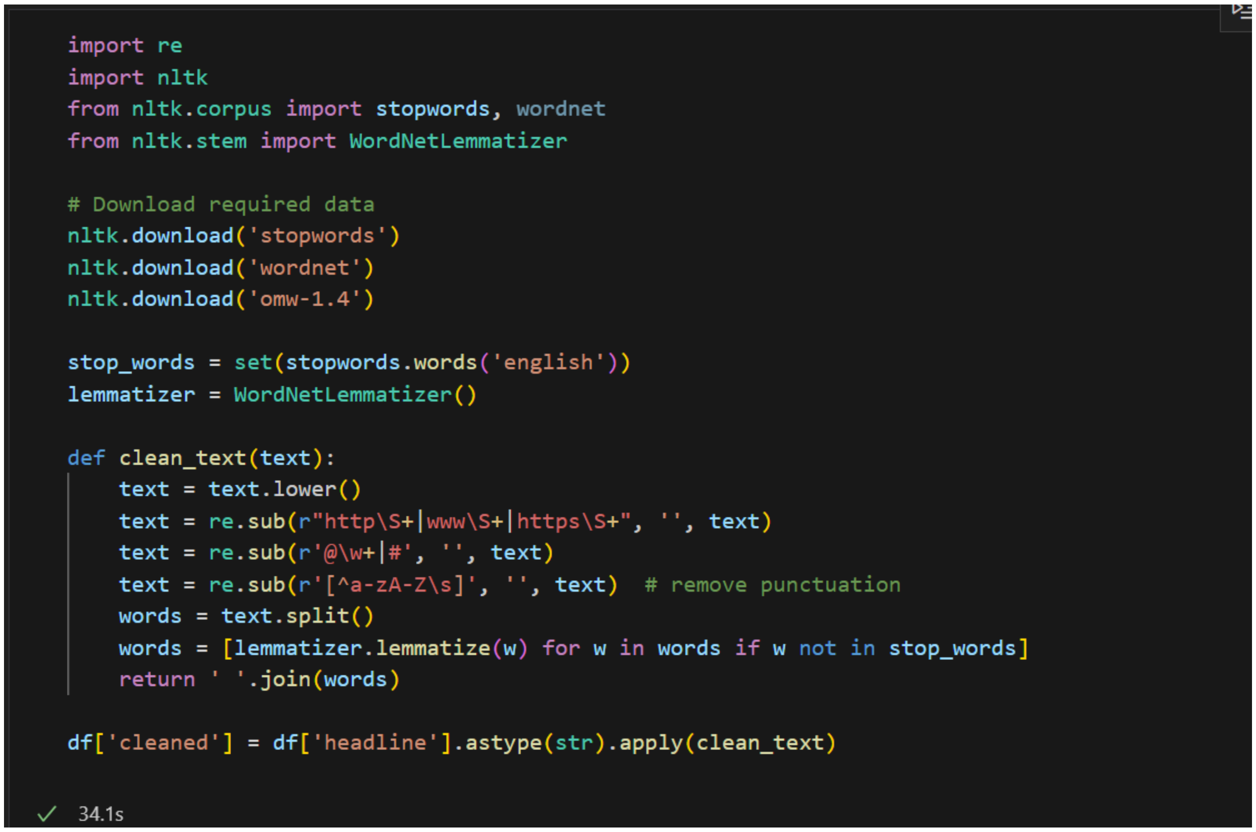
Source:

Kaggle — Sarcasm Detection in News Headlines

3. Preprocessing

We cleaned the text data using the following steps:

- Converted all text to lowercase.
- Removed URLs, mentions, punctuation.
- Removed stopwords using NLTK.
- Final cleaned text was stored in a new column for model training.



```python
import re
import nltk
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer

# Download required data
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def clean_text(text):
    text = text.lower()
    text = re.sub(r"http\S+|www\S+|https\S+", '', text)
    text = re.sub(r'@\w+|#', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # remove punctuation
    words = text.split()
    words = [lemmatizer.lemmatize(w) for w in words if w not in stop_words]
    return ' '.join(words)

df['cleaned'] = df['headline'].astype(str).apply(clean_text)
```

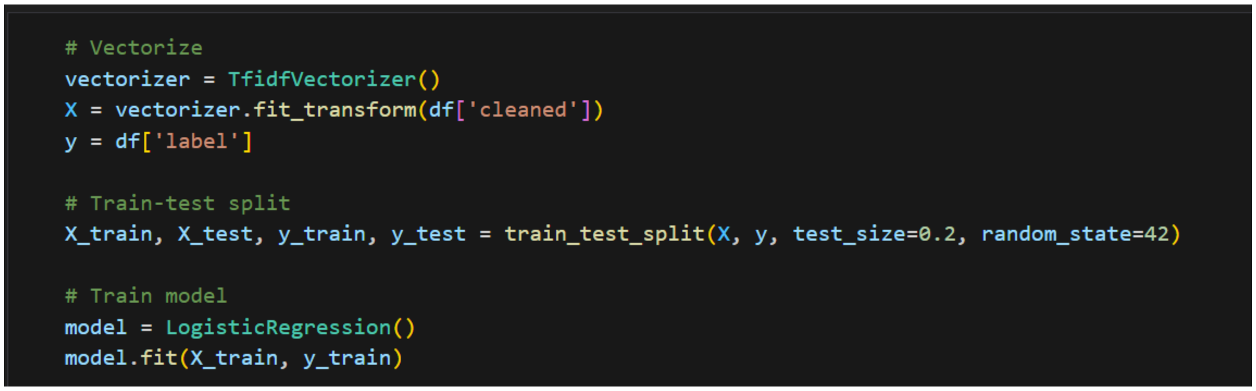
✓ 34.1s

4. Model Training

We trained our model using the **Logistic Regression algorithm** along with **TF-IDF vectorization** to convert text into numeric format.

Steps:

- Split data into training and testing sets (80-20).
- Applied TfidfVectorizer for feature extraction.
- Trained Logistic Regression model.
- Achieved good accuracy with balanced results.



```python
# Vectorize
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['cleaned'])
y = df['label']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = LogisticRegression()
model.fit(X_train, y_train)
```

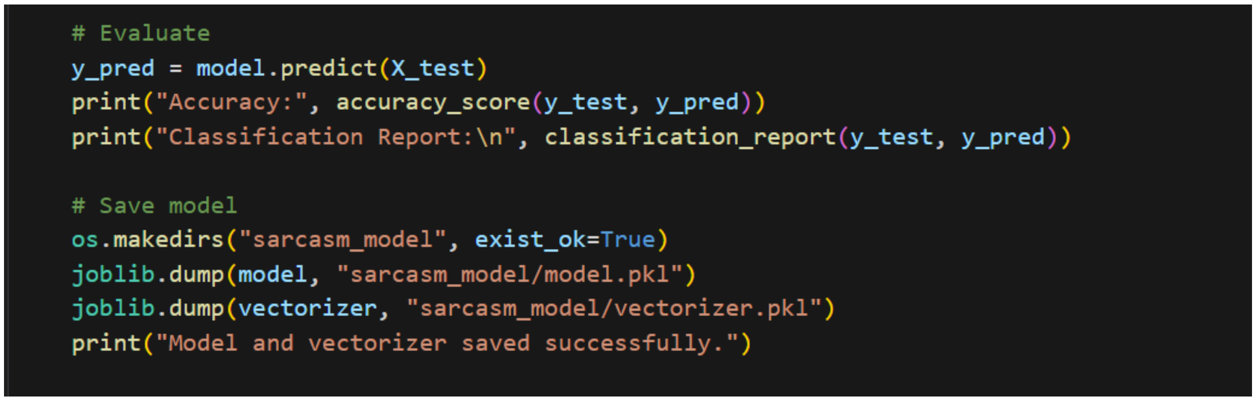
---

5. Evaluation

We evaluated the model using:

- **Accuracy Score**
- **Classification Report (Precision, Recall, F1-Score)**

The model was tested on several sarcastic and non-sarcastic headlines and gave satisfactory results.



```python
# Evaluate
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Save model
os.makedirs("sarcasm_model", exist_ok=True)
joblib.dump(model, "sarcasm_model/model.pkl")
joblib.dump(vectorizer, "sarcasm_model/vectorizer.pkl")
print("Model and vectorizer saved successfully.")
```

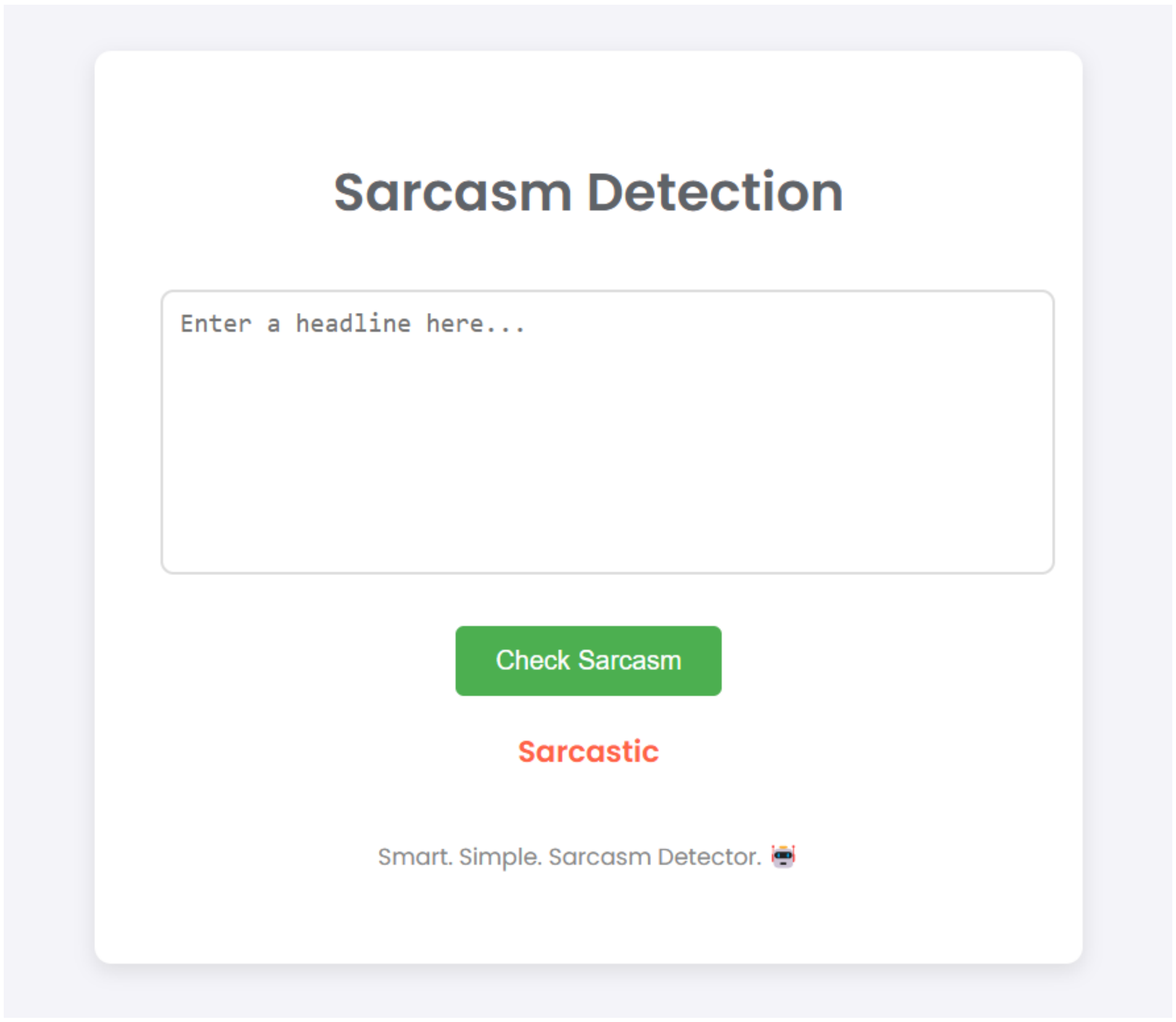
6. Web Application (Flask)

To make the model interactive, we created a simple **Flask web app** with the following features:

- A textbox for entering any headline.
- A "Check" button to detect sarcasm.
- Displays prediction: "Sarcastic" or "Not Sarcastic".

Frontend Features:

- Responsive and modern design using HTML and CSS.
- Clear heading and smooth user experience.
- Classy tagline: *"Smart. Simple. Sarcasm Detector. 🤖"*



Sarcasm Detection

Enter a headline here...

Sarcastic

Smart. Simple. Sarcasm Detector. 🤖

---

7. Directory Structure

```
sarcasm_app/
│
├── sarcasm_model/
│   ├── model.pkl
│   └── vectorizer.pkl
│
```

```
├── app.py
├── train_model.ipynb
└── templates/
    └── index.html
```

---

8. Conclusion

This project successfully demonstrates sarcasm detection using deep learning and nlp presents it through a functional web interface. We learned about text preprocessing, model training, and deploying DL models with Flask. The project was a great learning experience, especially in handling real-world NLP data and delivering a complete solution.

---

9. Future Work

- Use more advanced models like LSTM, BERT for better accuracy.
- Handle multi-lingual sarcasm.
- Extend dataset size for improved predictions.