You are tasked with developing a Python code for sentiment extraction utilizing a provided sample dataset. The dataset consists of textual data annotated with labels categorizing sentiments into four categories: "rude," "normal," "insult," and "sarcasm." Dataset: ● Real News: https://drive.google.com/file/d/1FL2HqgLDAP5550nd1h_8iBhAVISTnzr/view?usp=sharing ● Outline the key steps involved in developing a sentiment extraction algorithm using Python. 2. Describe the structure and format of the sample dataset required for sentiment extraction. 3. Implement the Python code to read and preprocess the sample dataset for sentiment analysis. Ensure that the code correctly handles text data and labels. 4. Discuss the process of classifying sentiments into the specified categories: "rude," "normal," "insult," and "sarcasm." Explain any techniques or algorithms employed for this classification task. 5. Evaluate the effectiveness of the sentiment extraction algorithm on the provided sample dataset. Consider metrics such as accuracy, precision, recall, and F1-score. 6. Propose potential enhancements or modifications to improve the performance of the sentiment extraction algorithm. Justify your recommendations. 7. Reflect on the ethical considerations associated with sentiment analysis, particularly regarding privacy, bias, and potential misuse of extracted sentiments. 8. Write a complete code for this assignment.

**1. Key Steps:**

1. **Data Preprocessing:** Read data, clean text (remove punctuation, stop words, lowercase), and convert labels to numerical format.
2. **Feature Engineering:** Extract features like word embeddings or n-grams.
3. **Model Training:** Train a machine learning model (e.g., Naive Bayes, Support Vector Machine) on features and labels.
4. **Evaluation:** Assess model performance using metrics like accuracy, precision, recall, and F1-score.

**2. Sample Dataset Format:**

The ideal format is a tab-separated file with two columns:

- **Text:** The actual sentence or phrase for sentiment analysis.
- **Label:** The sentiment category ("rude," "normal," "insult," or "sarcasm").

**3. Python Code for Reading and Preprocessing:**

```
Python
import pandas as pd

# Define file paths
real_news_file = "path/to/real_news.txt"
fake_news_file = "path/to/fake_news.txt"

# Define function to read and preprocess data
def read_preprocess_data(filepath):
  data = pd.read_csv(filepath, sep="\t", header=None, names=["text", "label"])

  # Text cleaning (replace with desired cleaning steps)
  data["text"] = data["text"].str.lower()
  data["text"] = data["text"].str.replace("[^\w\s]", "", regex=True)  # Remove non-
alphanumeric characters

  # Label encoding (replace with appropriate mapping)
  label_map = {"rude": 0, "normal": 1, "insult": 2, "sarcasm": 3}
  data["label"] = data["label"].map(label_map)

  return data

# Read and preprocess real and fake news data
real_data = read_preprocess_data(real_news_file)
fake_data = read_preprocess_data(fake_news_file)

# Combine data (optional)
data = pd.concat([real_data, fake_data])
```

Use code with caution.
content_copy

**4. Sentiment Classification:**

This task requires a supervised learning model. Here's an example using Naive Bayes:

- Train the model on the preprocessed features (e.g., word embeddings) and labels.
- For new text data:

    - Preprocess the text data similarly.
    - Extract features.
    - Use the trained model to predict the sentiment category.

**5. Evaluation:**

Split the data into training and testing sets. Train the model on the training set and evaluate its performance on the testing set using metrics like:

- **Accuracy:** Proportion of correctly classified sentiments.
- **Precision:** Ratio of true positives (correctly identified category) to all predicted instances of that category.
- **Recall:** Ratio of true positives to all actual instances of that category in the testing set.
- **F1-Score:** Harmonic mean of precision and recall, combining both metrics.

**6. Performance Enhancements:**

- **Feature Selection:** Choose the most informative features for classification.
- **Model Tuning:** Hyperparameter tuning can improve model performance.
- **Advanced Models:** Explore deep learning models like Recurrent Neural Networks (RNNs) for potentially better handling of sequential text data.

**7. Ethical Considerations:**

- **Privacy:** Ensure data anonymization and user consent for sentiment analysis.
- **Bias:** Training data bias can lead to biased sentiment classification. Mitigate bias by using diverse datasets.
- **Misuse:** Sentiment analysis can be misused for manipulation or social engineering. Utilize the technology responsibly.

**8. Complete Code (Naive Bayes Example):**

```
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.model_selection import train_test_split


vectorizer = TfidfVectorizer(max_features=1000)
```

```
features = vectorizer.fit_transform(data["text"])

 X_train, X_test, y_train, y_test = train_test_split(features, data["label"], test_size=0.2)

 model = MultinomialNB()

model.fit(X_train, y_train)

 new_text = "This movie is awful!"
```