SENTIMENT DATA ANALYSIS

Exploring dataset

```
import chardet
import pandas as pd
with open('/content/training.1600000.processed.noemoticon.csv', 'rb') as f:
    result = chardet.detect(f.read(1000000))  # read only 1 million bytes

df = pd.read_csv('/content/training.1600000.processed.noemoticon.csv', encoding=result['encoding'])
```

```
df.shape
```

```
(1048572, 6)
```

```
import pandas as pd

# Load the dataset


# Display basic information about the dataset
print(df.info())

# Check the first few rows of the dataset
print('\n\n\n')
print('The fist five columns of the dataset is displayed')
print(df.head())
```

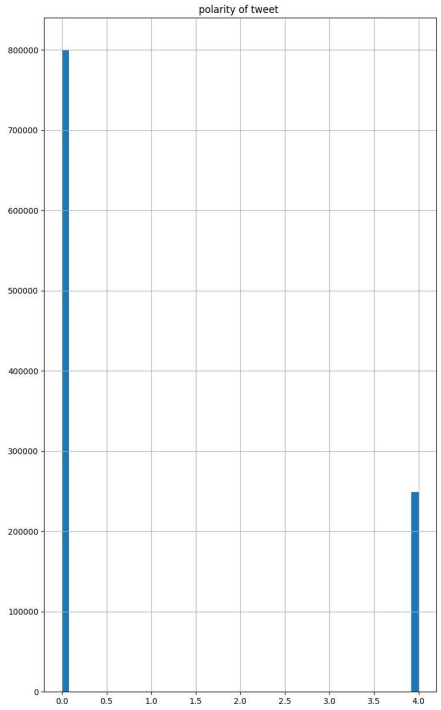
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048572 entries, 0 to 1048571
Data columns (total 6 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   polarity of tweet   1048572 non-null  int64
 1   id of the tweet     1048572 non-null  int64
 2   date of the tweet   1048572 non-null  object
 3   query               1048572 non-null  object
 4   user                1048572 non-null  object
 5   text of the tweet   1048572 non-null  object
dtypes: int64(2), object(4)
memory usage: 48.0+ MB
None



The fist five columns of the dataset is displayed
   polarity of tweet   id of the tweet             date of the tweet  \
0                   0       1467810672  Mon Apr 06 22:19:49 PDT 2009
1                   0       1467810917  Mon Apr 06 22:19:53 PDT 2009
2                   0       1467811184  Mon Apr 06 22:19:57 PDT 2009
3                   0       1467811193  Mon Apr 06 22:19:57 PDT 2009
4                   0       1467811372  Mon Apr 06 22:20:00 PDT 2009

      query           user                                text of the tweet
0  NO_QUERY  scotthamilton  is upset that he can't update his Facebook by ...
1  NO_QUERY       mattycus  @Kenichan I dived many times for the ball. Man...
2  NO_QUERY        ElleCTF    my whole body feels itchy and like its on fire
3  NO_QUERY         Karoli  @nationwideclass no, it's not behaving at all....
4  NO_QUERY       joy_wolf                      @Kwesidei not the whole crew
```

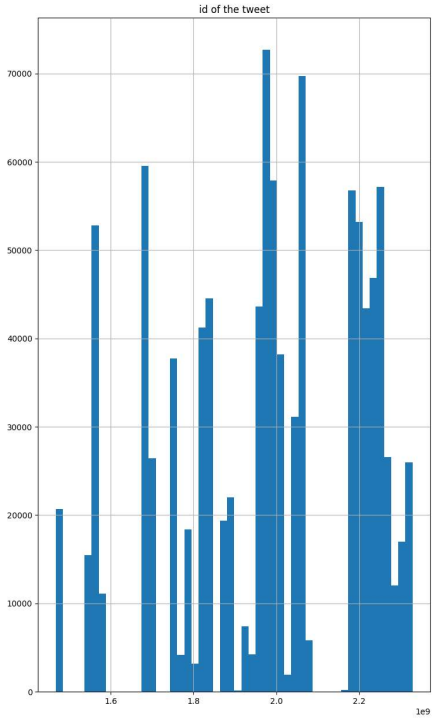
Exploratory data analysis

```
import pandas as pd
import matplotlib.pyplot as plt

# Display the first few rows of the DataFrame
print(df.head())

# Display the summary statistics of the
df.describe()

# Display the number of missing values in each column
print(df.isnull().sum())

# Plot a histogram for each numerical attribute
df.hist(bins=50, figsize=(20,15))
```

```
plt.show()
```

```
   polarity of tweet   id of the tweet            date of the tweet  \
0                  0       1467810672  Mon Apr 06 22:19:49 PDT 2009
1                  0       1467810917  Mon Apr 06 22:19:53 PDT 2009
2                  0       1467811184  Mon Apr 06 22:19:57 PDT 2009
3                  0       1467811193  Mon Apr 06 22:19:57 PDT 2009
4                  0       1467811372  Mon Apr 06 22:20:00 PDT 2009

      query           user                                text of the tweet
0  NO_QUERY  scotthamilton  is upset that he can't update his Facebook by ...
1  NO_QUERY       mattycus  @Kenichan I dived many times for the ball. Man...
2  NO_QUERY        ElleCTF    my whole body feels itchy and like its on fire
3  NO_QUERY         Karoli  @nationwideclass no, it's not behaving at all....
4  NO_QUERY       joy_wolf                      @Kwesidei not the whole crew
polarity of tweet     0
id of the tweet       0
date of the tweet     0
query                 0
user                  0
text of the tweet     0
dtype: int64
```






```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt



# Display the first few rows of the data
print(df.head())

# Display the data types of each column
print(df.dtypes)

# Check for missing values
print(df.isnull().sum())
```

```
   polarity of tweet   id of the tweet             date of the tweet  \
0                  0       1467810672  Mon Apr 06 22:19:49 PDT 2009
1                  0       1467810917  Mon Apr 06 22:19:53 PDT 2009
2                  0       1467811184  Mon Apr 06 22:19:57 PDT 2009
3                  0       1467811193  Mon Apr 06 22:19:57 PDT 2009
4                  0       1467811372  Mon Apr 06 22:20:00 PDT 2009

      query           user                                text of the tweet
0  NO_QUERY  scotthamilton  is upset that he can't update his Facebook by ...
1  NO_QUERY       mattycus  @Kenichan I dived many times for the ball. Man...
2  NO_QUERY        ElleCTF    my whole body feels itchy and like its on fire
3  NO_QUERY         Karoli  @nationwideclass no, it's not behaving at all....
4  NO_QUERY       joy_wolf                    @Kwesidei not the whole crew
polarity of tweet      int64
id of the tweet        int64
date of the tweet     object
query                 object
user                  object
text of the tweet     object
dtype: object
polarity of tweet     0
id of the tweet       0
date of the tweet     0
query                 0
user                  0
text of the tweet     0
dtype: int64


df.nunique()

polarity of tweet           2
id of the tweet       1048041
date of the tweet      662450
query                       1
user                   511364
text of the tweet     1036132
dtype: int64


#Displaying the unique values in the dataset
print('\n\nUnique values of query ')
df['query'].unique()



Unique values of query
array(['NO_QUERY'], dtype=object)


df.columns=df.columns.str.strip()


df['polarity of tweet'].unique()

array([0, 4])
```

Perform text preprocessing tasks, such as lowercasing, removing stop words, and handling special characters

```
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
import string

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

def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove punctuation
    text = ''.join([c for c in text if c not in string.punctuation])
    # Remove stopwords and lemmatize
    text = [lemmatizer.lemmatize(word) for word in word_tokenize(text) if word not in stop_words]
    return ' '.join(text)


 [nltk_data] Downloading package stopwords to /root/nltk_data...
 [nltk_data]   Package stopwords is already up-to-date!
 [nltk_data] Downloading package wordnet to /root/nltk_data...
 [nltk_data]   Package wordnet is already up-to-date!
 [nltk_data] Downloading package punkt to /root/nltk_data...
 [nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
df['preprocessed_text'] = df['text of the tweet'].apply(preprocess_text)
```

Convert the preprocessed text into numerical vectors using techniques like TF-IDF or word embeddings

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['text of the tweet'])
y = df['polarity of tweet']
```

Split the data into training and testing sets using the train_test_split() function from sklearn

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7832725365376821
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Assuming that 'y_test' are your true labels and 'predictions' are the predictions made by your model
# y_test = ...
predictions = model.predict(X_test)

# Calculate metrics
accuracy = accuracy_score(y_test, predictions)
precision = precision_score(y_test, predictions, average='weighted')
recall = recall_score(y_test, predictions, average='weighted')
f1 = f1_score(y_test, predictions, average='weighted')

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

```
Accuracy: 0.7832725365376821
Precision: 0.809377300332663
Recall: 0.7832725365376821
F1 Score: 0.7084173085763821
```

Implement cross-validation techniques to assess the generalization performance of the model and prevent overfitting

```
from sklearn.model_selection import GridSearchCV
parameters = {'alpha': [0.01, 0.1, 0.5, 1, 10]}
grid_search = GridSearchCV(model, parameters, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
print('Best parameters:', grid_search.best_params_)
print('Best score:', grid_search.best_score_)
```

```
Best parameters: {'alpha': 0.1}
Best score: 0.8094108994918333
```

```
from sklearn.model_selection import cross_val_score
best_model = MultinomialNB(alpha=0.1)
scores = cross_val_score(best_model, X_train, y_train, cv=5, scoring='accuracy')
print('Cross-validation scores:', scores)
print('Mean cross-validation score:', scores.mean())
```

```
Cross-validation scores: [0.80956894 0.81033784 0.80996716 0.80844723 0.80873333]
Mean cross-validation score: 0.8094108994918333
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

# some example text
corpus = [
    'This is the first document.',
    'This document is the second document.',
    'And this is the third one.',
    'Is this the first document?',
]

# initialize the vectorizer
vectorizer = TfidfVectorizer()

# fit the vectorizer to the text
X = vectorizer.fit_transform(corpus)

# now you can get the feature names
feature_names = vectorizer.get_feature_names()

print(feature_names)

coefs = model.coef_[0]
sorted_coefs = np.argsort(coefs)
print('Top 10 most positive words:')
for i in sorted_coefs[-10:]:
    print(feature_names[i], coefs[i])
print('Top 10 most negative words:')
for i in sorted_coefs[:10]:
    print(feature_names[i], coefs[i])
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-58-2bf6c1d4fa53> in <cell line: 18>()
     16
     17 # now you can get the feature names
---> 18 feature_names = vectorizer.get_feature_names()
     19
     20 print(feature_names)

AttributeError: 'TfidfVectorizer' object has no attribute 'get_feature_names'
```

EXPLAIN ERROR

```
print(type(vectorizer))
```

```
<class 'sklearn.feature_extraction.text.TfidfVectorizer'>
```

Evaluate the model's performance using relevant evaluation metrics for sentiment analysis, such as confusion matrix, precision-recall curves, and ROC-AUC

```
import sklearn
print(sklearn.__version__)
```

```
1.2.2
```

```
from sklearn.metrics import confusion_matrix

y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix:\n', cm)
```

```
Confusion matrix:
 [[159635    495]
 [ 44956   4629]]
```

```
from sklearn.metrics import roc_auc_score

y_pred_prob = model.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, y_pred_prob)
print('ROC-AUC score:', roc_auc)
```

```
ROC-AUC score: 0.8264902349653249
```

Deploy the trained model for real-time sentiment analysis, creating an API or integrating it into a web application

```
import streamlit as st
st.title('Sentiment Analysis on Movie Reviews')
review = st.text_input('Enter a movie review:')
if st.button('Predict'):
    review = preprocess_text(review)
    review = vectorizer.transform([review])
    prediction = model.predict(review)
    st.write('The sentiment is:', prediction[0])
```