The dataset is available Here

Problem Identification

You are required to build an emotion classification model that can identify the emotion expressed in a given text. The target has 8 labels (sadness, joy, anticipation, optimism, anger, fear, disgust, surprise).

You will be evaluated based on the following criteria:

- Data preprocessing
- Feature representation
- Model building
- Model evaluation
- Results demonstration

Good luck! ;)

Steps

1. Data Preprocessing

- Load necessary libraries and dataset.
- Perform basic data exploration.
- Handle missing values if any.
- Tokenize, normalize, and remove stopwords from the text data if needed.
- Apply text preprocessing function to clean the text data.
- Split the dataset into training and testing sets.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

```
df = pd.read_csv("Movies_Reviews_modified_version1.csv")
```

```
df.head()
```

	Unnamed: 0	Ratings	Reviews	movie_name	Resenhas	genres	Description
0	0	3.0	It had some laughs, but overall the motivation...	Waiting to Exhale	Riu algumas risadas, mas no geral a motivação ...	['Comedy', 'Drama', 'Romance']	Based on Terry McMillan's novel, this film fol...
1	1	4.0	WAITING TO EXHALE Waiting, and waiting, and wa...	Waiting to Exhale	ESPERANDO PARA EXALAR Esperando, e esperando, ...	['Comedy', 'Drama', 'Romance']	Based on Terry McMillan's novel, this film fol...
2	2	4.0	Angela Basset was good as expected, but Whitne...	Waiting to Exhale	Angela Basset foi boa como o esperado, mas Whi...	['Comedy', 'Drama', 'Romance']	Based on Terry McMillan's novel, this film fol...
3	3	5.0	The movie is okay, mediocre might even be the ...	Waiting to Exhale	O filme é bom, medíocre pode até ser a palavra...	['Comedy', 'Drama', 'Romance']	Based on Terry McMillan's novel, this film fol...
4	4	5.0	I got an opportunity to see Waiting To Exhale ...	Waiting to Exhale	Tive a oportunidade de ver Waiting To Exhale p...	['Comedy', 'Drama', 'Romance']	Based on Terry McMillan's novel, this film fol...

Next steps: Generate code with df View recommended plots

```
df.shape
```

```
(46173, 8)
```

```
df.dtypes
```

```
Unnamed: 0       int64
Ratings        float64
Reviews         object
movie_name      object
Resenhas        object
genres          object
Description     object
emotion         object
dtype: object
```

```
df.isnull().sum()
```

```
Unnamed: 0     0
Ratings        0
Reviews        0
movie_name     0
Resenhas       0
genres         0
Description    0
emotion        0
dtype: int64
```

```
df.describe()
```

	Unnamed: 0	Ratings
count	46173.000000	46173.000000
mean	23086.000000	5.983735
std	13329.141326	2.893144
min	0.000000	1.000000
25%	11543.000000	3.000000
50%	23086.000000	6.000000
75%	34629.000000	9.000000
max	46172.000000	10.000000

```
# Drop rows with missing values
df.dropna(inplace=True)
```

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

nltk.download('punkt')
nltk.download('stopwords')

# Tokenize and remove stopwords
stop_words = set(stopwords.words('english'))
df['text_tokens'] = df['Reviews'].apply(word_tokenize)
df['text_tokens'] = df['text_tokens'].apply(lambda x: [word.lower() for word in x
df['text_tokens'] = df['text_tokens'].apply(lambda x: [word for word in x if word
```

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

2. Feature Representation

You can use various methods for feature representation, such as TF-IDF or Bag of Words. Here is how to implement TF-IDF feature representation:

- Import `TfidfVectorizer` from `sklearn`.
- Initialize `TfidfVectorizer` with desired parameters.
- Fit and transform the training data to obtain TF-IDF features.
- Transform the testing data using the same vectorizer.

```
# Split the dataset into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df['Reviews'], df['emotion'],

from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)  # You can adjust max_featu

# Fit and transform the training data
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

# Transform the testing data
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

3. Model Building

- Import necessary libraries for model building (e.g., Naive Bayes, SVM, Random Forest).
- Initialize the model.
- Train the model to predict the emotion.

```
from sklearn.naive_bayes import MultinomialNB

# Initialize the model
nb_model = MultinomialNB()

# Train the model
nb_model.fit(X_train_tfidf, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

4. Model Evaluation

- Import evaluation metrics.

- Predict the labels for the test data using the trained model.
- Evaluate the model using classification report and accuracy score.

```
from sklearn.metrics import classification_report, accuracy_score

# Predict the labels for the test data
y_pred = nb_model.predict(X_test_tfidf)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.4220898754737412
Classification Report:
              precision    recall  f1-score   support

       anger       0.54      0.19      0.28       737
anticipation       0.51      0.10      0.17      1478
     disgust       0.49      0.05      0.10       339
        fear       0.34      0.14      0.20       701
         joy       0.45      0.14      0.21      1563
    optimism       0.62      0.09      0.15       986
     sadness       0.41      0.93      0.57      3417
    surprise       0.00      0.00      0.00        14

    accuracy                           0.42      9235
   macro avg       0.42      0.21      0.21      9235
weighted avg       0.46      0.42      0.33      9235
```

5. Results Demonstration

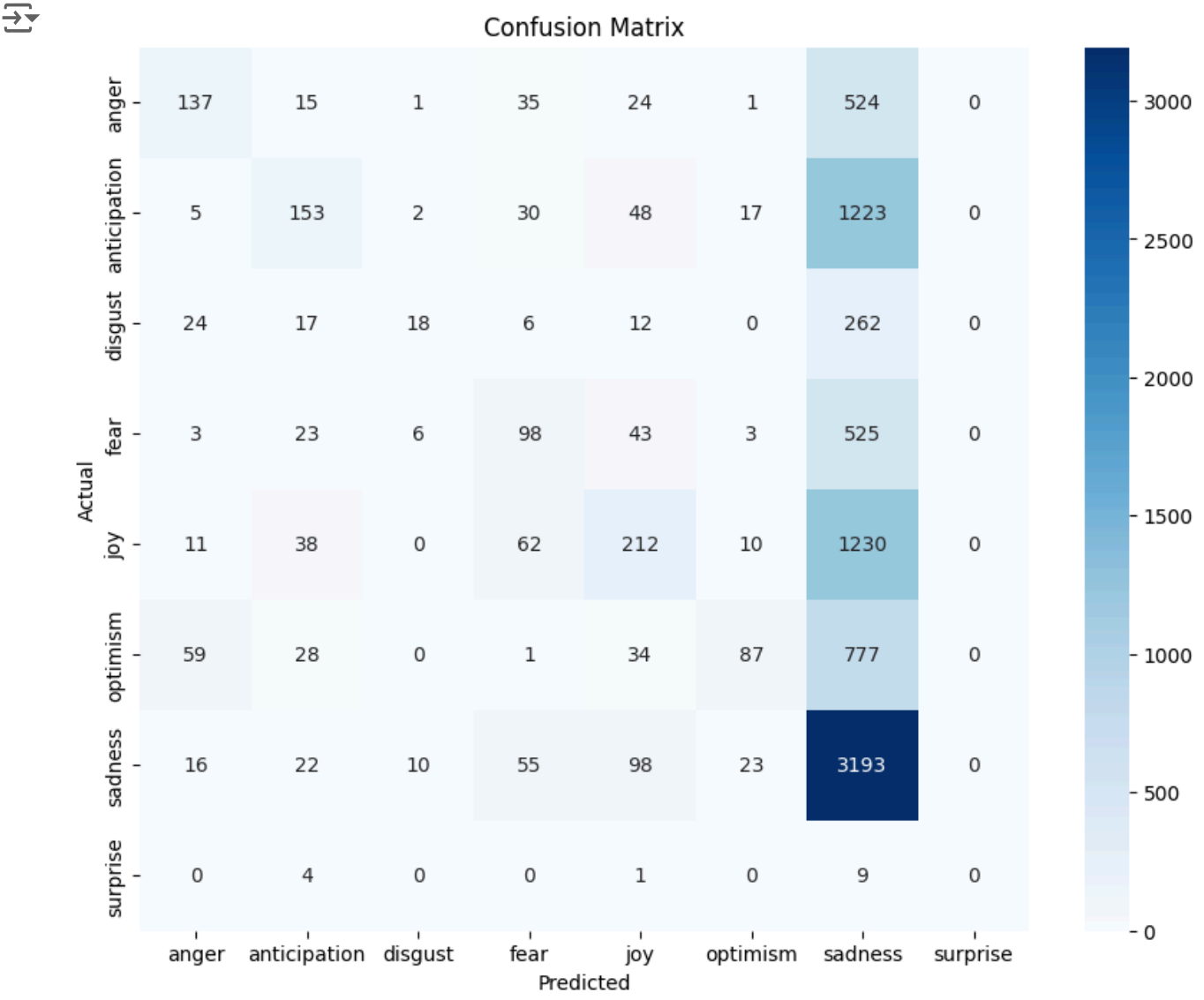
- Visualize the results using appropriate plots (e.g., confusion matrix).
- Providing example predictions on new/unseen data

```
# Visualizing results with a confusion matrix
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=nb_model.classes_,
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Example predictions on new/unseen data
new_texts = ["This movie made me so happy!", "I'm really scared of horror movies."
new_texts_tfidf = tfidf_vectorizer.transform(new_texts)
new_predictions = nb_model.predict(new_texts_tfidf)
```

```
for text, prediction in zip(new_texts, new_predictions):
    print(f"Text: {text} | Predicted Emotion: {prediction}")
```




```
Text: This movie made me so happy! | Predicted Emotion: sadness
Text: I'm really scared of horror movies. | Predicted Emotion: sadness
```