```
# Saudi Soccer News

# Information
# based on below , please to provide full code from begining to model accurecy each part in separate code , classification report , confusion matrix codes , and provide the explanation and comment for each code row as well ,consider the dataset name in code will be 'data' , also I need machine learning model to classify the news.

# About Dataset
# Context : News of MBS league has been more excited in the last years. Since the teams spent more than one billion riyals on contracts, deals and training camps. Also, the competition became more intense due to the small point difference between the top 4 teams. The rank of the league will be affected in the long term if the competition is ongoing like the premier league. The aim of a dataset to do sentiment analysis, prediction and clustering in order to know how the competition is described. The period of news is between December 12, 2020 and January 25, 2021.

# Content Writer: the person or organization who wrote the news. Location: the location of news. Date: the date of news in format yyyy-mm-dd.Time: the time of news in format hh:mm.Title: the title of news.News: the text of news.
# Class: the type of news which are 0 for informative about teams, 1 is related to the match and 2 for the general news not specific on teams.

# Step 1: Import Necessary Libraries

# Import libraries for data handling
import pandas as pd  # For working with dataframes
import matplotlib.pyplot as plt  # For plotting
import seaborn as sns  # For prettier visualizations
from wordcloud import WordCloud  # To visualize frequent words

# For encoding categorical data for numerical analysis
from sklearn.preprocessing import LabelEncoder

# Step 2: Load the Data

# Loading your Saudi Soccer News CSV file into a Pandas dataframe

data = pd.read_csv(r"D:\Projects\Saudi Soccer News\news_dataset1.csv")

# This loads the sales data into a Pandas dataframe so you can easily manipulate and analyze it

# Displaying the first few rows of the data to understand its structure
data.head()  # Prints the first 5 rows of the data
```

```
         writer     location        date   time  \
0      05:32  25-01-2021  بريدة      الرياضية
1      01:41  25-01-2021  مكة المكرمة  الرياضية
2      01:11  25-01-2021  الرياض     الرياضية
3      12:49  25-01-2021  الرياض     الرياضية
4  12:41  25-01-2021  الرياض     حمد الصويلحي

                                                 news  \
0  قاد الكاميروني لياندر تاوامبا مهاجم فريق التعا...
1  يحل فريق الشباب الأول لكرة القدم ضيفا نظيره ال...
2  تتجدد طموحات الهلال بالانفراد بصدارة ترتيب دور...
3  أنقذ البرازيلي رومارينيو لاعب فريق الاتحاد الأ...
4  كشفت الرياضية مصادر خاصة علي البليهي مدافع فري...

                                              title  class
0                 1    التعاون يعبر ضمك هاتريك تاوامبا
1      1    الشباب لتعزيز حظوظه بالمنافسة أمام الوحدة
2  1    الهلال لتجديد الانفراد بالصدارة بوابة الفيصلي
3                   1    الأهلي يضرب والأصفران يتعطلان
4                     0    البليهي يقود الهلال السوبر
```

```
# Step 3: Preprocessing the Data

# Display the shape of the dataset (number of rows and columns)
print(f"Dataset has {data.shape[0]} rows and {data.shape[1]}
columns.")
```

```
Dataset has 1996 rows and 7 columns.
```

```
# Check for missing values in the data
print(data.isnull().sum())  # Counts missing values per column
```

```
writer      0
location    1
date        0
time        0
news        0
title       0
class       0
dtype: int64
```

```
# Display data types of each column
print(data.info())  # Provides info about columns, data types, and
memory usage
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1996 entries, 0 to 1995
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
```

```
 0   writer    1996 non-null   object
 1   location  1995 non-null   object
 2   date      1996 non-null   object
 3   time      1996 non-null   object
 4   news      1996 non-null   object
 5   title     1996 non-null   object
 6   class     1996 non-null   int64
dtypes: int64(1), object(6)
memory usage: 109.3+ KB
None
```

```
# Get summary statistics for numerical columns
print(data.describe())  # Describes statistics such as mean, min, max, etc.
```

```
             class
count  1996.000000
mean      0.259519
std       0.592167
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       2.000000
```

```
# Get summary statistics for all columns
print(data.describe(include='all'))
```

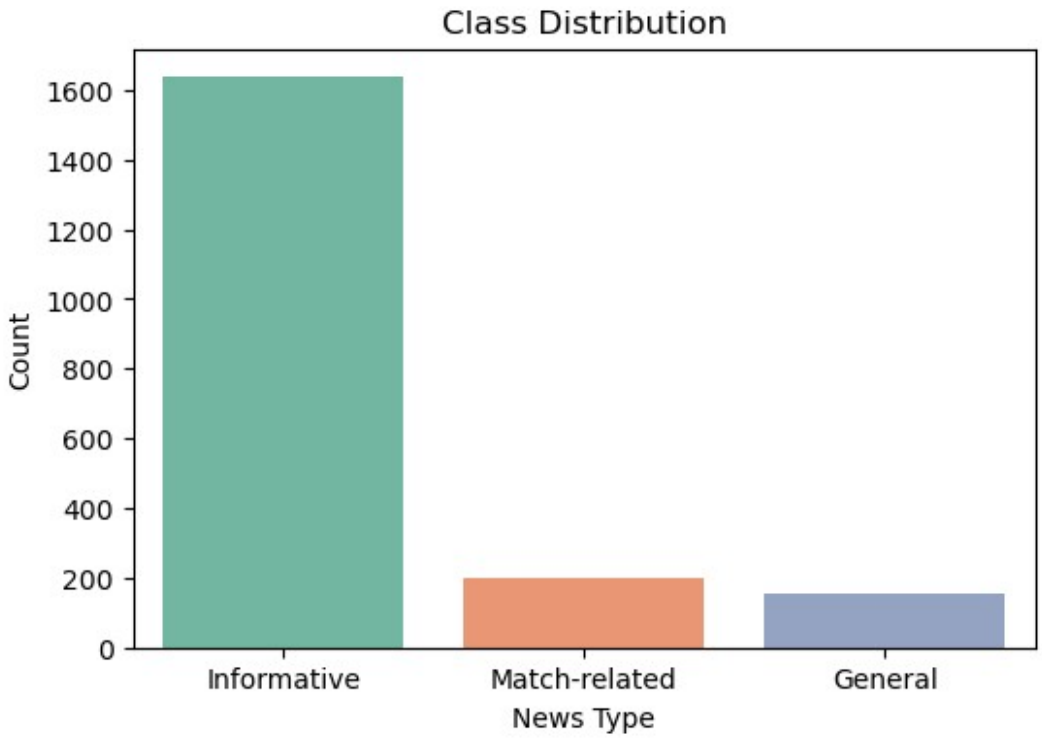
```
          writer location        date   time  \
count       1996     1995        1996   1996
unique        91       27          45    497
top     01:38 24-01-2021 الرياض  الرياضية
freq         668      857          68     22
mean         NaN      NaN         NaN    NaN
std          NaN      NaN         NaN    NaN
min          NaN      NaN         NaN    NaN
25%          NaN      NaN         NaN    NaN
50%          NaN      NaN         NaN    NaN
75%          NaN      NaN         NaN    NaN
max          NaN      NaN         NaN    NaN

                                                    news  \
count                                               1996
unique                                              1992
top     صعد فريق النصر الأول لكرة القدم 3 مراكز الترتي...
freq                                                   2
mean                                                 NaN
std                                                  NaN
min                                                  NaN
25%                                                  NaN
```

```
50%                                                          NaN
75%                                                          NaN
max                                                          NaN
```

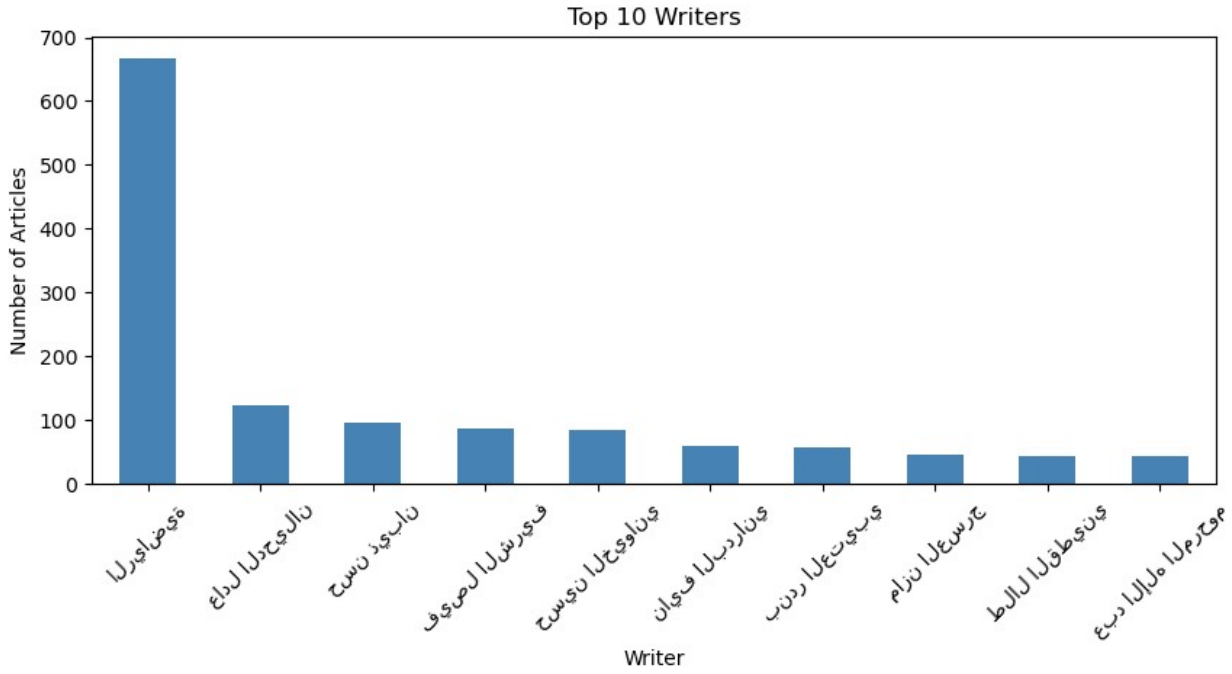
	title	class
count	1996	1996.000000
unique	1979	NaN
top	المكشر يركز اللياقة	NaN
freq	4	NaN
mean	NaN	0.259519
std	NaN	0.592167
min	NaN	0.000000
25%	NaN	0.000000
50%	NaN	0.000000
75%	NaN	0.000000
max	NaN	2.000000

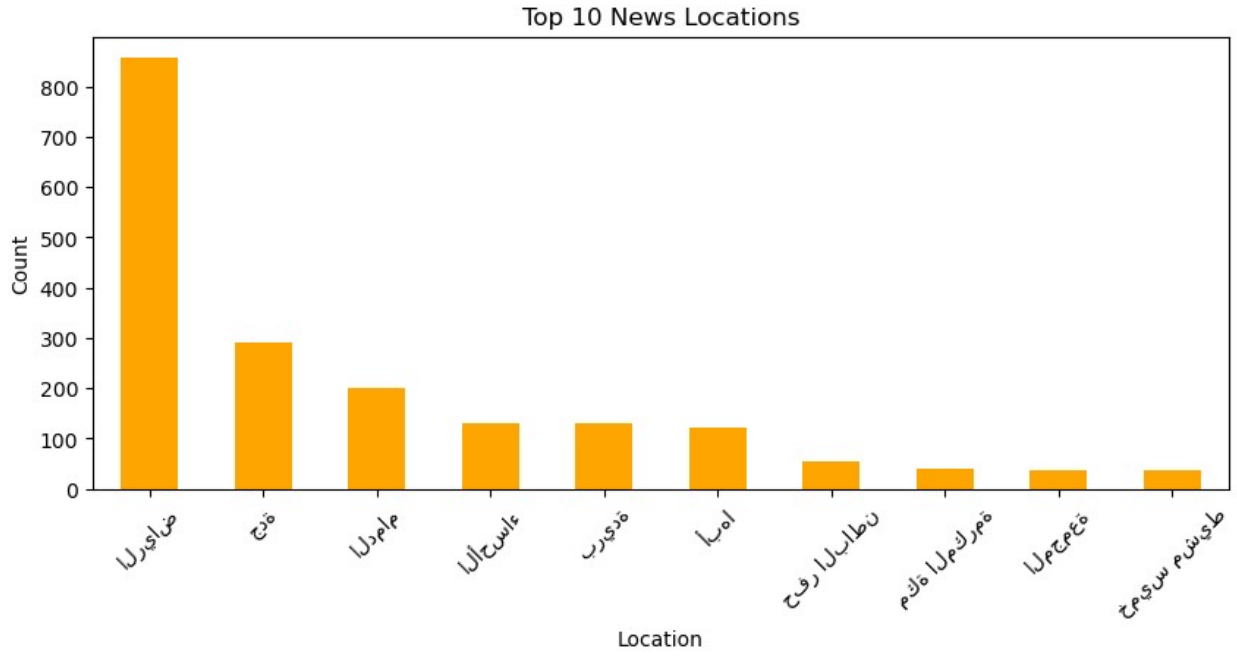
```
# Drop missing values (only one in 'location', so safe to drop)
data.dropna(inplace=True)

# Step 4: Exploratory Data Analysis (EDA)

# Class Distribution

# Visualize the distribution of news classes
plt.figure(figsize=(6,4))
sns.countplot(data=data, x='class', palette='Set2')  # Bar plot of
class counts
plt.title("Class Distribution")  # Add plot title
plt.xticks([0, 1, 2], ['Informative', 'Match-related', 'General'])  #
Rename class labels
plt.xlabel("News Type")
plt.ylabel("Count")
plt.show()
```

```
C:\Users\DELL_7200\AppData\Local\Temp\ipykernel_24316\1819795727.py:3:
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.countplot(data=data, x='class', palette='Set2')  # Bar plot of
class counts
```




```
# Top Writers and Locations

# Plot top 10 most frequent writers
plt.figure(figsize=(10,4))
data['writer'].value_counts().head(10).plot(kind='bar',
color='steelblue')
plt.title("Top 10 Writers")
plt.xlabel("Writer")
plt.ylabel("Number of Articles")
plt.xticks(rotation=45)  # Rotate x-axis labels for clarity
plt.show()

# Plot top 10 most frequent locations
plt.figure(figsize=(10,4))
data['location'].value_counts().head(10).plot(kind='bar',
color='orange')
plt.title("Top 10 News Locations")
plt.xlabel("Location")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```

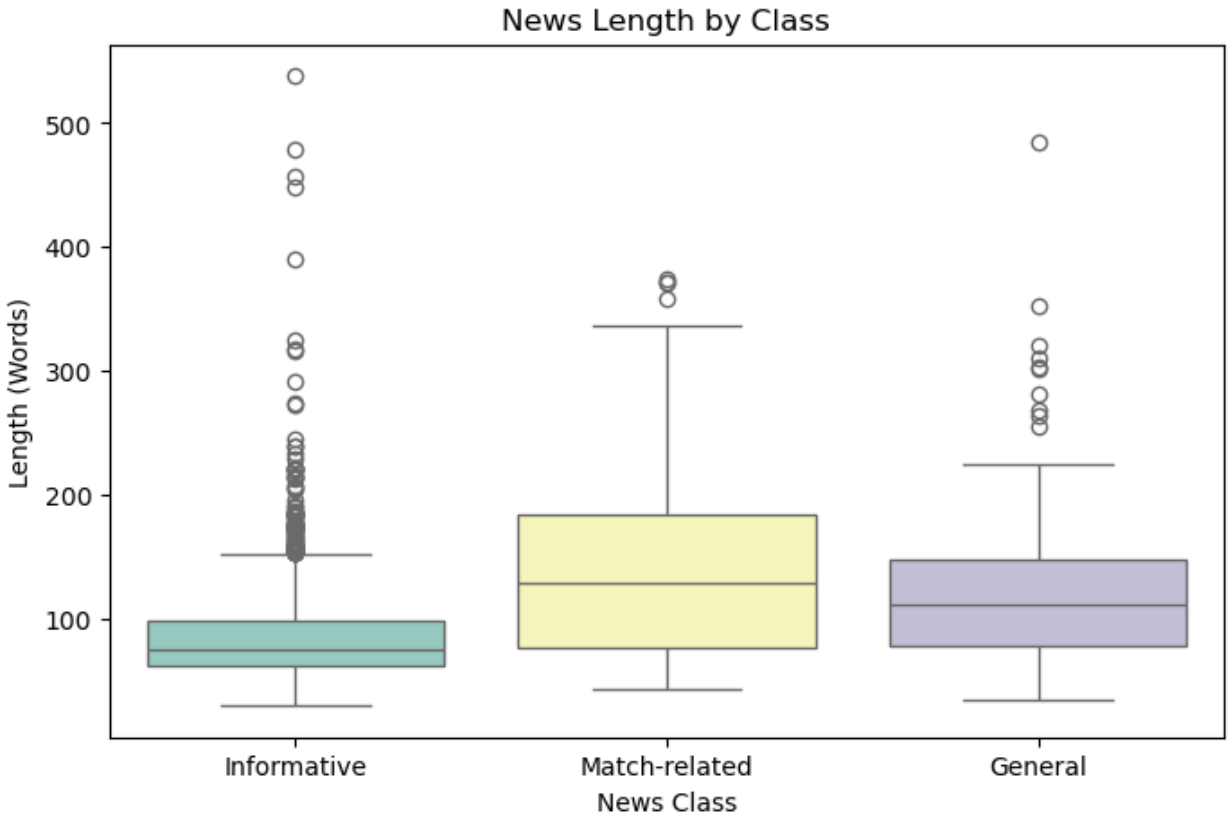





```
# News Length Analysis by Class

# Create a new column for news length (in number of words)
data['news_length'] = data['news'].apply(lambda x: len(x.split()))

# Boxplot: shows how news length varies by class
plt.figure(figsize=(8,5))
sns.boxplot(data=data, x='class', y='news_length', palette='Set3')
```

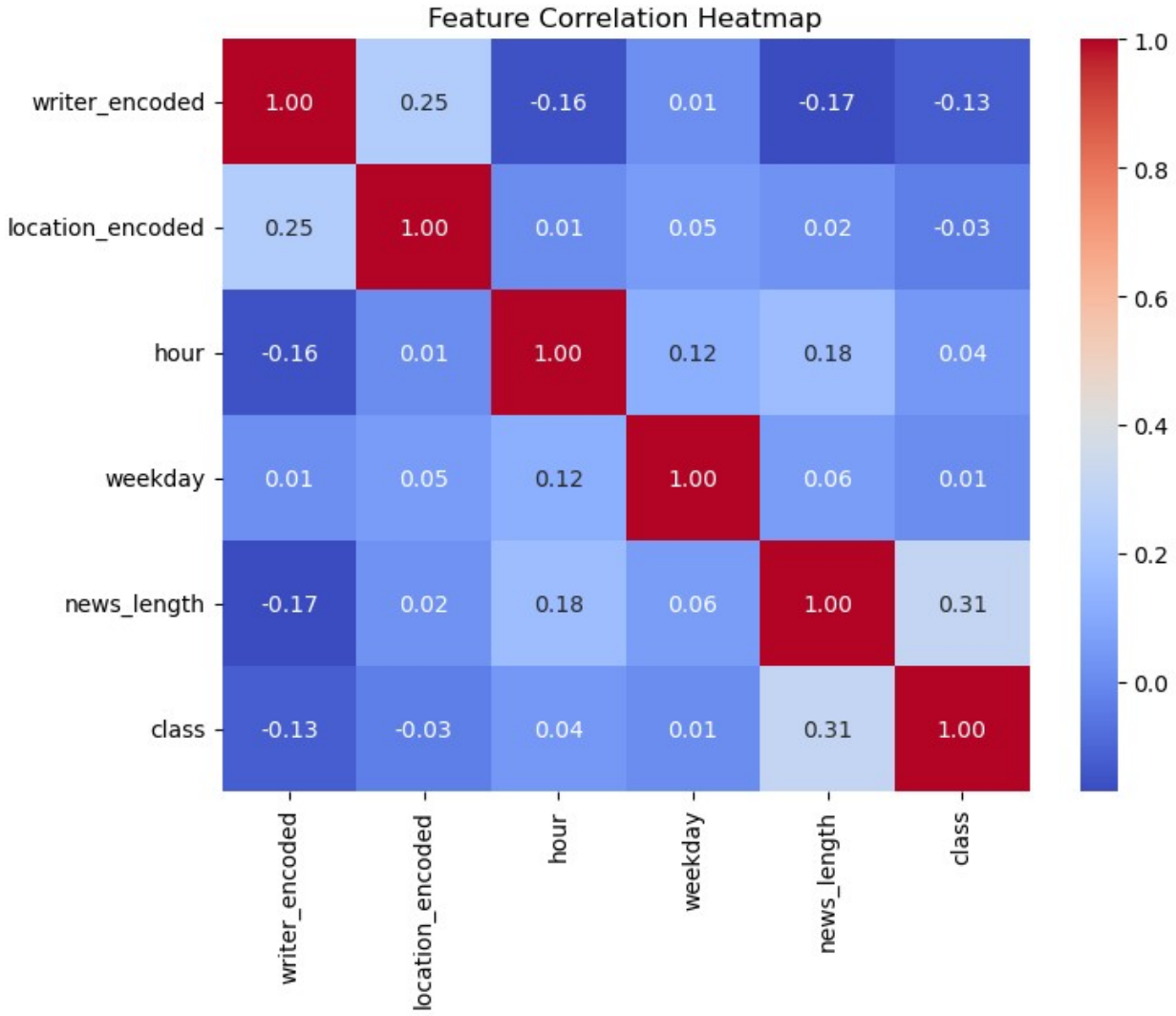
```
plt.title("News Length by Class")
plt.xticks([0,1,2], ['Informative', 'Match-related', 'General'])  # Rename classes
plt.xlabel("News Class")
plt.ylabel("Length (Words)")
plt.show()
```

```
C:\Users\DELL_7200\AppData\Local\Temp\ipykernel_24316\318182082.py:6: 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.boxplot(data=data, x='class', y='news_length', palette='Set3')
```




```
# Encode & Correlate Features

# Encode categorical 'writer' and 'location' columns into numeric values
label_encoder = LabelEncoder()
data['writer_encoded'] = label_encoder.fit_transform(data['writer'])
data['location_encoded'] =
label_encoder.fit_transform(data['location'])
```

```
# Extract hour from time (convert to datetime first)
data['hour'] = pd.to_datetime(data['time'], format='%H:%M').dt.hour

# Convert date to datetime format and extract weekday
data['date'] = pd.to_datetime(data['date'])
data['weekday'] = data['date'].dt.dayofweek  # Monday = 0

# Select relevant numerical features for correlation
correlation_features = ['writer_encoded', 'location_encoded', 'hour',
'weekday', 'news_length', 'class']

# Compute correlation matrix
corr = data[correlation_features].corr()

# Plot heatmap of correlations
plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")  # Show
correlations with color and numbers
plt.title("Feature Correlation Heatmap")
plt.show()
```




```
# Step 5: Machine Learning Pipeline to Classify News

# Preprocess Data and Split

from sklearn.model_selection import train_test_split  # For splitting
dataset
from sklearn.feature_extraction.text import TfidfVectorizer  # For
converting text to numbers

# Drop missing values (only 1 row in 'location')
data.dropna(inplace=True)

# Combine 'title' and 'news' into a single text field for better
context
data['text'] = data['title'] + ' ' + data['news']

# Define input features (X) and target labels (y)
X = data['text']  # News text
```

```
y = data['class']  # News category (0, 1, 2)

# Split data into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# TF-IDF Vectorization

# Example Arabic stop words
arabic_stop_words = [' ','أي','ما','ذلك','هذا','هو','عن','إلى','و','على','من','في',
'أنت','أنا'']  # Extend the list as needed

# Initialize TfidfVectorizer with the stop words
tfidf = TfidfVectorizer(max_features=5000,
stop_words=arabic_stop_words)

# Fit and transform the training text
X_train_tfidf = tfidf.fit_transform(X_train)

# Transform the test text using the same vectorizer
X_test_tfidf = tfidf.transform(X_test)

# Train Logistic Regression Model

from sklearn.linear_model import LogisticRegression  # Import
classifier

# Initialize and train the model
model = LogisticRegression(max_iter=1000)  # Increase iterations to
ensure convergence
model.fit(X_train_tfidf, y_train)  # Fit model on training data

# Predict on test set
y_pred = model.predict(X_test_tfidf)

# Evaluate Accuracy , Classification Report & Confusion Matrix

from sklearn.metrics import accuracy_score  # For calculating accuracy

# Compute and display model accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)

from sklearn.metrics import classification_report

# Display precision, recall, f1-score for each class
print("Classification Report:")
print(classification_report(y_test, y_pred,
target_names=['Informative', 'Match-related', 'General']))

from sklearn.metrics import confusion_matrix
```

```python
# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix as heatmap
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Informative', 'Match-related', 'General'],
            yticklabels=['Informative', 'Match-related', 'General'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

```
Model Accuracy: 0.9122807017543859
Classification Report:
              precision    recall  f1-score   support

  Informative       0.91      0.99      0.95       329
Match-related       0.91      0.64      0.75        45
      General       1.00      0.36      0.53        25

     accuracy                           0.91       399
    macro avg       0.94      0.67      0.74       399
 weighted avg       0.92      0.91      0.90       399
```