Twitter Topic Classification

Business Understanding

The goal of this task is to develop a Natural Language Processing (NLP) pipeline to classify tweets into six categories. The categories are as follows:

1. Sports and gaming
2. Pop culture and entertainment
3. Daily life
4. Science and Technology
5. Business and Entrepreneurship
6. Arts and culture

By classifying tweets into these categories, we aim to gain insights into the different topics and themes discussed on social media. This can help in various applications such as understanding user preferences, targeted advertising, sentiment analysis, and trend analysis.

The NLP pipeline will involve various steps such as data preprocessing, text cleaning, feature extraction, model training, and evaluation. By building an accurate classification model, we can automate the process of categorizing tweets and gain valuable insights from large volumes of social media data.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

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

```
# loading the data to a dataframe

data = pd.read_json('/content/drive/MyDrive/NLP_data/CETM47-22_23-AS2-Data.json')
data.head(10)
```

	text	date	label	id	label_name
0	The {@Clinton LumberKings@} beat the {@Cedar R...	2019-09-08	4	1170516324419866624	sports_&_gaming
1	I would rather hear Eli Gold announce this Aub...	2019-09-08	4	1170516440690176006	sports_&_gaming
2	Someone take my phone away, I'm trying to not ...	2019-09-08	4	1170516543387709440	sports_&_gaming
3	A year ago, Louisville struggled to beat an FC...	2019-09-08	4	1170516620466429953	sports_&_gaming
4	Anyone know why the #Dodgers #Orioles game nex...	2019-09-08	4	1170516711411310592	sports_&_gaming
5	I don't care. you gave him a shot, he is strug...	2019-09-08	4	1170516891053580288	sports_&_gaming
6	Okay how can I watch the {@Arkansas State Foot...	2019-09-08	4	1170516916554936322	sports_&_gaming
7	Check out largest crowds ever for a basketball...	2019-09-08	4	1170516940902805504	sports_&_gaming
8	I voted #WeWantNCAAFootball on {{USERNAME}}	2019-09-08	4	1170517092489187328	sports_&_gaming
9	Streaming a new game #minionmasters come stop ...	2019-09-08	4	1170546366566846464	sports_&_gaming

```
data.sample(10)
```

	text	date	label	id	label_name
5910	Check out ONE MILLION STRONG - THE ALBUM - PA ...	2021-05-09	2	1391350257871736832	pop_culture

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6443 entries, 0 to 6442
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   text        6443 non-null   object
 1   date        6443 non-null   datetime64[ns]
 2   label       6443 non-null   int64
 3   id          6443 non-null   int64
 4   label_name  6443 non-null   object
dtypes: datetime64[ns](1), int64(2), object(2)
memory usage: 251.8+ KB
```

```
data['label_name'].value_counts()
```

```
pop_culture                2512
sports_&_gaming            2291
daily_life                  883
science_&_technology        326
business_&_entrepreneurs    287
arts_&_culture              144
Name: label_name, dtype: int64
```

```
data['label'].value_counts()
```

```
2    2512
4    2291
3     883
5     326
1     287
0     144
Name: label, dtype: int64
```

Data Preprocessing and Cleaning

In this task, we perform the following steps to preprocess and clean the data:

1. **Remove Stopwords:** We eliminate common words that do not carry significant meaning in the context of the data.
2. **Remove Punctuations, Numbers, and Special Characters:** We get rid of non-alphabetic characters, digits, and symbols.
3. **Tokenization:** We split the text into individual words or tokens.
4. **Lemmatization:** We reduce words to their base or root form to normalize the text.
5. **Vectorization:** We convert the text data into numerical representations to be used in machine learning algorithms.
6. **Splitting the Data into Train and Test Sets:** We divide the dataset into two parts, one for training the model and the other for evaluating its performance.

These steps help prepare the data for further analysis or modeling by reducing noise, standardizing the text, and enabling numerical representation for machine learning algorithms.

```
import nltk
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
# removing stopwords and punctuation
stopwords_list  = stopwords.words('english')
stopwords_list += list(string.punctuation)
```

```
# creating a function to clean the text
def clean_text(text):
    # Convert text to lowercase
    text = text.lower()
```

```
    # Tokenize the text into individual words
    word_tokens = word_tokenize(text)

    # Remove stopwords from the text
    text = [word for word in word_tokens if word not in stopwords_list]

    # Remove non-alphabetic characters
    text = [word for word in text if word.isalpha()]

    # Lemmatize the words to their base form
    lemmatizer = WordNetLemmatizer()
    text = [lemmatizer.lemmatize(word) for word in text]

    # Join the cleaned words back into a single string
    text = ' '.join(text)

    # Return the cleaned text
    return text
```

```
import nltk
nltk.download('punkt')
```

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

```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
# applying the clean_text function to the text column
data['clean_text'] = data['text'].apply(clean_text)
data.head()
```

	text	date	label	id	label_name	
0	The {@Clinton LumberKings@} beat the {@Cedar R...	2019-09-08	4	1170516324419866624	sports_&_gaming	clinton lumberkings beat cedar
1	I would rather hear Eli Gold announce this Aub...	2019-09-08	4	1170516440690176006	sports_&_gaming	would rather hear eli gold announ
2	Someone take my phone away, I'm trying to not ...	2019-09-08	4	1170516543387709440	sports_&_gaming	someone take phone away trying
3	A year ago, Louisville struggled to beat an FC...	2019-09-08	4	1170516620466429953	sports_&_gaming	year ago louisville struggled be
4	Anyone know why the #Dodgers #Orioles game nex...	2019-09-08	4	1170516711411310592	sports_&_gaming	anyone know dodger oriole game

Wordcloud Visualization

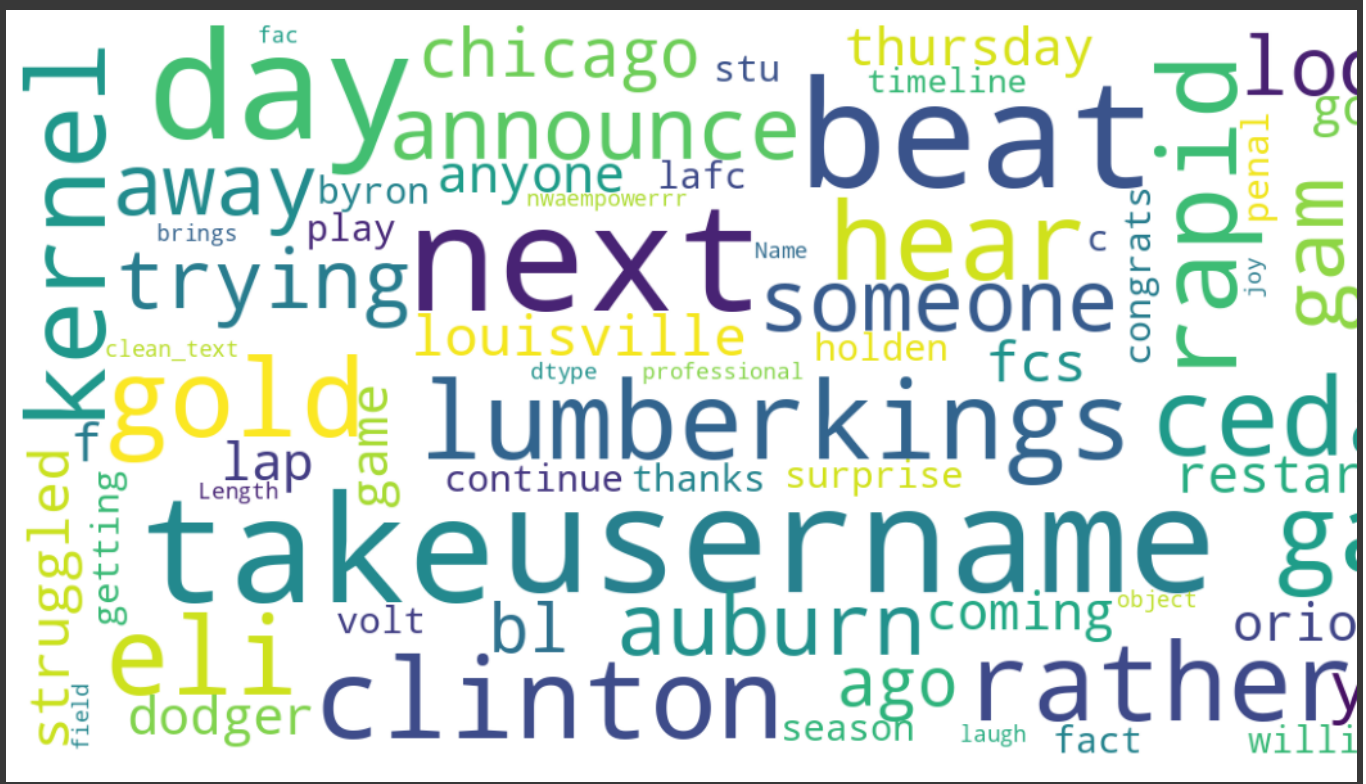
```
# function to create a wordcloud
from wordcloud import WordCloud

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color='white',
        max_words=400,
        max_font_size=40,
        scale=3,
        random_state=1
    ).generate(str(data))

    fig = plt.figure(1, figsize=(15, 15))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize=20)
        fig.subplots_adjust(top=2.3)

    plt.imshow(wordcloud)
    plt.show()

# creating a wordcloud for the text column
show_wordcloud(data['clean_text'])
```


```
data.label_name.value_counts()
```

```
pop_culture                2512
sports_&_gaming            2291
daily_life                  883
science_&_technology        326
business_&_entrepreneurs    287
arts_&_culture              144
Name: label_name, dtype: int64
```

Word cloud for Sports_&_gaming tweets

```
# wordcloud for the text column for the label sports
show_wordcloud(data[data['label_name'] == 'sports_&_gaming']['clean_text'])
```

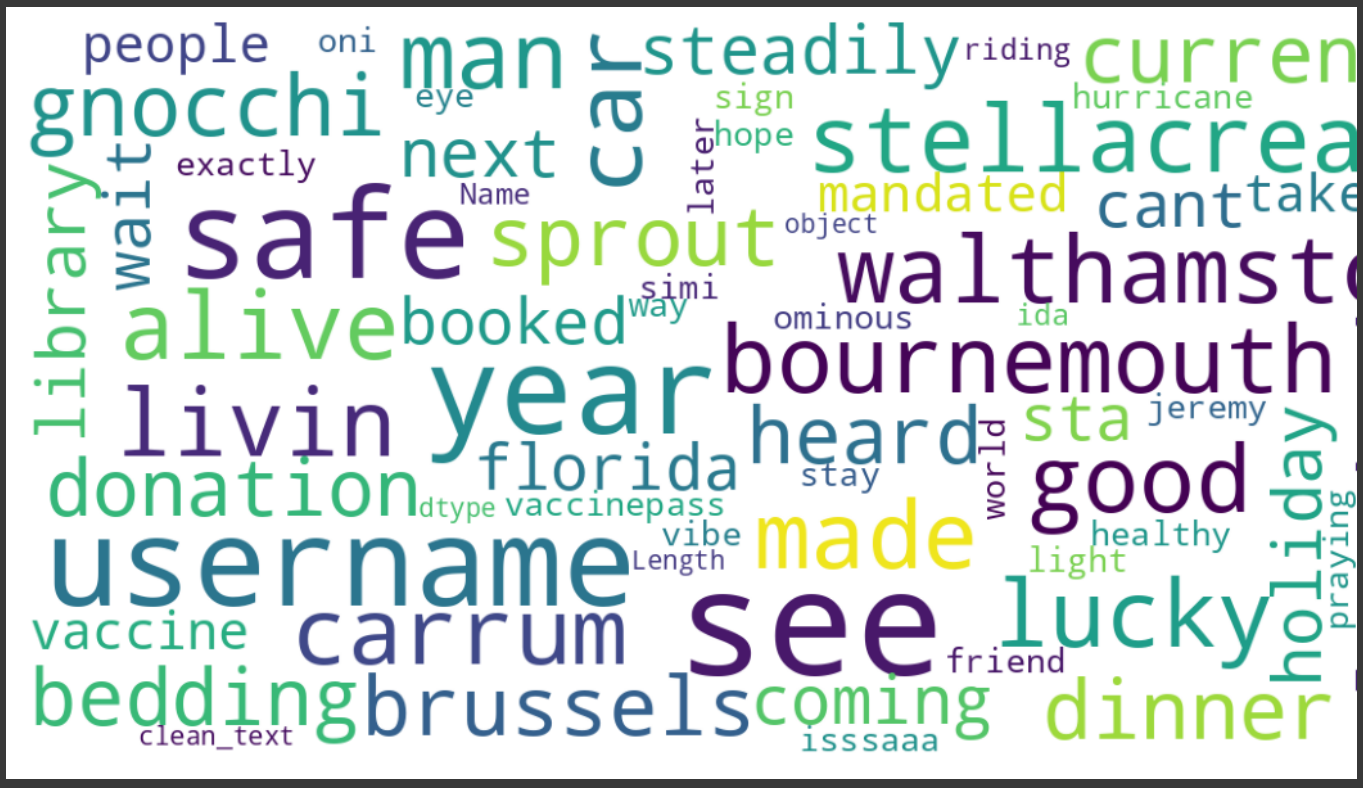


word cloud for Pop_culture tweets

```
show_wordcloud(data[data['label_name'] == 'pop_culture']['clean_text'])
```

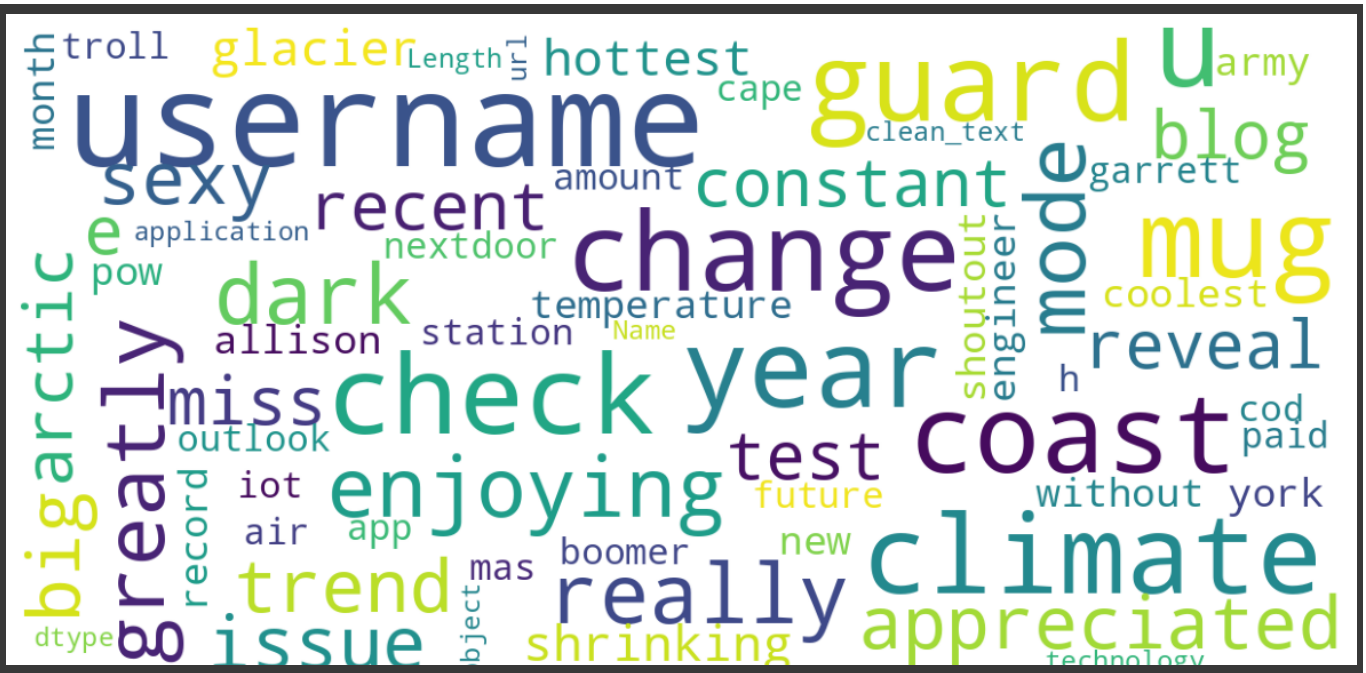
word cloud for Daily tweets

```
show_wordcloud(data[data['label_name'] == 'daily_life']['clean_text'])
```



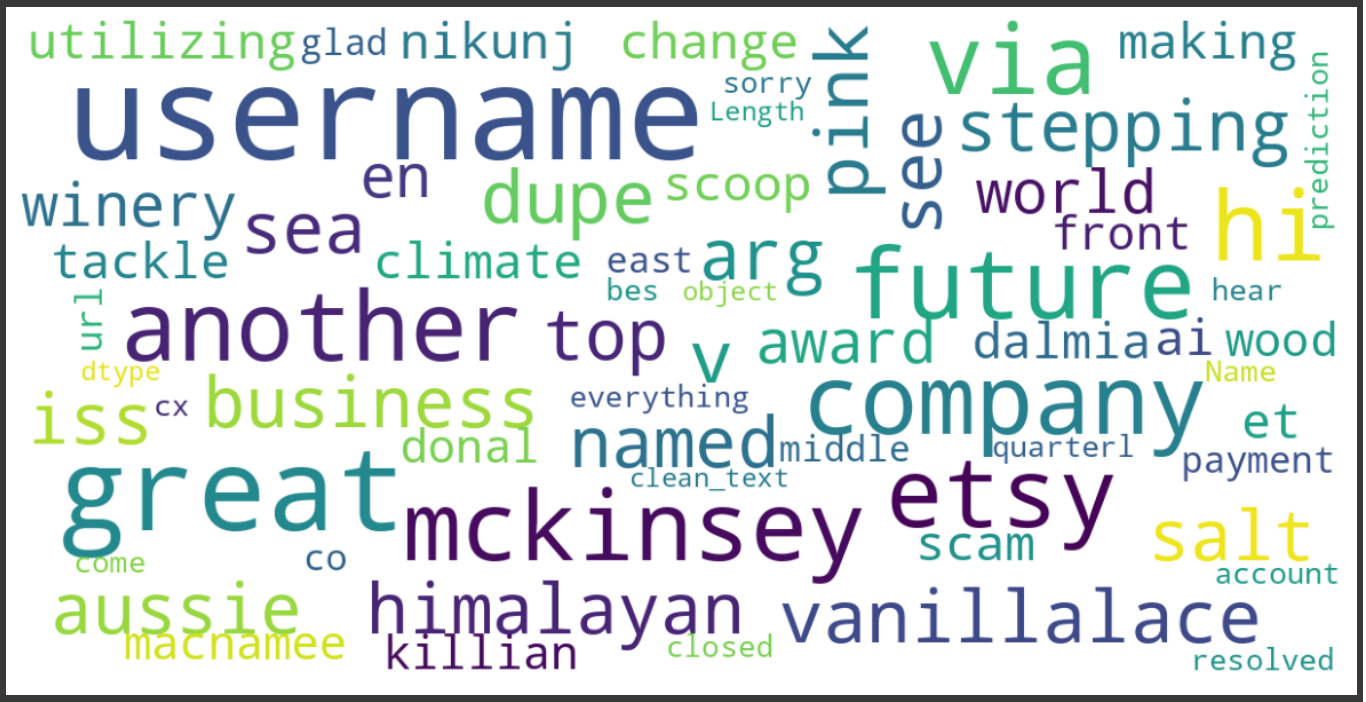
word cloud for science_&_technology tweets

```
show_wordcloud(data[data['label_name'] == 'science_&_technology']['clean_text'])
```



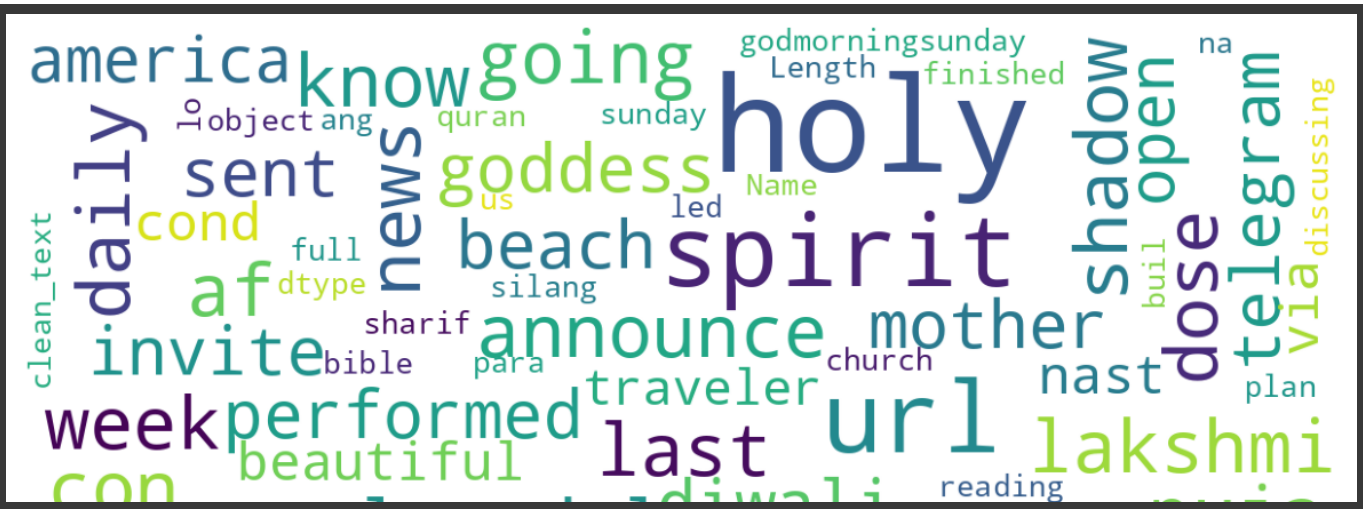
word cloud for business_&_entrepreneurs tweets

```
show_wordcloud(data[data['label_name'] == 'business_&_entrepreneurs']['clean_text'])
```



word cloud for arts_&_culture tweets

```
show_wordcloud(data[data['label_name'] == 'arts_&_culture']['clean_text'])
```



Vectorization

```
# function to vectorize the text
from sklearn.feature_extraction.text import TfidfVectorizer

def vectorize_text(text):
    vectorizer = TfidfVectorizer()
    text_vectorized = vectorizer.fit_transform(text)
    return pd.DataFrame(text_vectorized.toarray(), columns=vectorizer.get_feature_names_out())

# vectorizing the text column
text_vectorized = vectorize_text(data['clean_text'])
```

```
# one-hot encoding the label_name column
label_name_onehot = pd.get_dummies(data['label_name'])
label_name_onehot.head()
```

```
data.rename(columns={'label': 'Target'}, inplace=True)
```

```
data.head()
```

	text	date	Target	id	label_name	clean_text
0	The {@Clinton LumberKings@} beat the {@Cedar R...	2019-09-08	4	1170516324419866624	sports_&_gaming	clinton lumberkings beat cedar rapid kernel ga...
1	I would rather hear Eli Gold announce this Aub...	2019-09-08	4	1170516440690176006	sports_&_gaming	would rather hear eli gold announce auburn gam...
2	Someone take my phone away, I'm trying to not ...	2019-09-08	4	1170516543387709440	sports_&_gaming	someone take phone away trying look chicago bl...
	A year ago, Louisville struggled to beat an	2019-09-				year ago louisville struggled beat fcs

```
# creating a dataframe with the vectorized text
data_vectorized = pd.concat([data['Target'], text_vectorized], axis=1)
data_vectorized.sample(10)
```

	Target	aa	aaa	aaaaa	aaaaaaaa	aaaand	aaand	aadar	aafp	aag	...	제이비	지훈	찬열	크래비티	태민	투모로우바이투게더	트레저	파이팅	행복	헹가래
4398	5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
274	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1190	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3881	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2385	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2901	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2298	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6298	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5530	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3021	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
# splitting the data into train ,test sets and validation set
from sklearn.model_selection import train_test_split

y = data_vectorized['Target']
X = data_vectorized.drop('Target', axis=1)

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Splitting the training set into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.5, random_state=42)
```

```
print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
print('X_val shape:', X_val.shape)
```

```
X_train shape: (2255, 18573)
X_test shape: (1933, 18573)
X_val shape: (2255, 18573)
```

Modelling

In this project, we explore various machine learning algorithms to classify tweets into the six predefined categories. The following algorithms are used:

1. **Logistic Regression**: Logistic regression is a linear classification algorithm that models the probability of each category. It works well for binary classification tasks and can be extended to multi-class classification using techniques like one-vs-rest or softmax regression.
2. **Naive Bayes**: Naive Bayes is a probabilistic classifier based on Bayes' theorem with the assumption of independence between features. It is known for its simplicity and efficiency, making it suitable for text classification tasks.
3. **Support Vector Machine (SVM)**: SVM is a powerful algorithm for both binary and multi-class classification. It finds a hyperplane that maximally separates the data points of different classes. SVM can handle high-dimensional data and is effective in dealing with complex decision boundaries.
4. **Random Forest**: Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It is robust against overfitting and capable of handling both numerical and categorical features. Random Forest can provide insights into feature importance and handle high-dimensional data effectively.

These algorithms are trained on the preprocessed and vectorized text data to learn the patterns and relationships between the features and the corresponding categories. The models will be evaluated based on various metrics such as accuracy, precision, recall, and F1-score to assess their performance.

Base Logistic Regression Model

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
```

```
# base model
logreg = LogisticRegression(penalty='l2', C=1.0, solver='lbfgs', max_iter=1000)
logreg.fit(X_train, y_train)

# evaluating the model
y_pred = logreg.predict(X_test)
print('Accuracy Score:\n ', accuracy_score(y_test, y_pred))
print('\nConfusion Matrix: \n', confusion_matrix(y_test, y_pred))
print('\nClassification Report: \n', classification_report(y_test, y_pred))
```

```
Accuracy Score:
  0.7221934816347646
Confusion Matrix:
 [[  0   0  30   1   8   0]
 [  0   3  44   3  36   1]
 [  0   0 683  11  52   0]
 [  0   0 140  62  70   0]
 [  0   0  59   1 642   0]
 [  0   0  59   1  21   6]]
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00        39
           1       1.00      0.03      0.07        87
           2       0.67      0.92      0.78       746
```

```
                3       0.78      0.23      0.35       272
                4       0.77      0.91      0.84       702
                5       0.86      0.07      0.13        87

         accuracy                           0.72      1933
        macro avg       0.68      0.36      0.36      1933
     weighted avg       0.73      0.72      0.66      1933

     /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
       _warn_prf(average, modifier, msg_start, len(result))
     /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
       _warn_prf(average, modifier, msg_start, len(result))
     /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
       _warn_prf(average, modifier, msg_start, len(result))
```

```
# cross validation
from sklearn.model_selection import cross_val_score

scores = cross_val_score(logreg, X, y, cv=5)

print('The Models scores are :', scores)
print('Cross Validation Score: ', scores.mean())
```

```
The Models scores are : [0.73235066 0.75717611 0.74709077 0.74223602 0.74378882]
Cross Validation Score:  0.7445284755383585
```

Base Naive Bayes Model

```
# Base Naive Bayes Model
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)

# evaluating the model
y_pred = nb_model.predict(X_test)

print('Accuracy Score:\n ', accuracy_score(y_test, y_pred))
print('\nConfusion Matrix: \n', confusion_matrix(y_test, y_pred))
print('\nClassification Report: \n', classification_report(y_test, y_pred))
```

```
Accuracy Score:
  0.6968442834971547
Confusion Matrix:
 [[  0   0  32   0   7   0]
 [  0   0  56   0  31   0]
 [  0   0 694   0  52   0]
 [  0   0 196   6  70   0]
 [  0   0  55   0 647   0]
 [  0   0  69   0  18   0]]
Classification Report:
               precision    recall  f1-score   support

            0       0.00      0.00      0.00        39
            1       0.00      0.00      0.00        87
            2       0.63      0.93      0.75       746
            3       1.00      0.02      0.04       272
            4       0.78      0.92      0.85       702
            5       0.00      0.00      0.00        87

     accuracy                           0.70      1933
    macro avg       0.40      0.31      0.27      1933
 weighted avg       0.67      0.70      0.60      1933

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
  _warn_prf(average, modifier, msg_start, len(result))
```

```
# cross validation
from sklearn.model_selection import cross_val_score

scores = cross_val_score(nb_model, X, y, cv=5)

print('The Models scores are :', scores)
print('Cross Validation Score: ', scores.mean())
```

```
The Models scores are : [0.69821567 0.6943367  0.70364624 0.70263975 0.69720497]
Cross Validation Score:  0.6992086648131105
```

Random Forest Classifier

```
# Random Forest Classifier
forest_model = RandomForestClassifier()
forest_model.fit(X_train, y_train)

# evaluating the model
y_pred = forest_model.predict(X_test)

print('Accuracy Score:\n ', accuracy_score(y_test, y_pred))
print('\nConfusion Matrix: \n', confusion_matrix(y_test, y_pred))
print('\nClassification Report: \n', classification_report(y_test, y_pred))
```

```
Accuracy Score:
  0.7097775478530781

Confusion Matrix:
 [[  0   0  33   0   5   1]
 [  0   9  57   4  15   2]
 [  0   0 691  10  41   4]
 [  0   1 154  73  44   0]
 [  0   0 119   5 577   1]
 [  0   0  50   5  10  22]]

Classification Report:
               precision    recall  f1-score   support

            0       0.00      0.00      0.00        39
            1       0.90      0.10      0.19        87
            2       0.63      0.93      0.75       746
            3       0.75      0.27      0.40       272
            4       0.83      0.82      0.83       702
            5       0.73      0.25      0.38        87

    accuracy                           0.71      1933
   macro avg       0.64      0.40      0.42      1933
weighted avg       0.72      0.71      0.67      1933

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-
  _warn_prf(average, modifier, msg_start, len(result))
```

```
# cross validation
from sklearn.model_selection import cross_val_score

scores = cross_val_score(forest_model, X, y, cv=5)

print('The Models scores are :', scores)
print('Cross Validation Score: ', scores.mean())
```

```
The Models scores are : [0.72847168 0.72924748 0.73157486 0.73136646 0.72670807]
Cross Validation Score:  0.7294737121077054
```

Evaluation

```
## Evaluation of the model on the test set

# Creating a function that will evaluate the model on the validation set
def evaluate_model(model):
    y_pred = model.predict(X_val)
    print('Accuracy Score:\n ', accuracy_score(y_val, y_pred))
    print('Confusion Matrix: \n', confusion_matrix(y_val, y_pred))
    print('Classification Report: \n', classification_report(y_val, y_pred))

# Evaluating the Logistic Regression model
evaluate_model(forest_model)
```

```
Accuracy Score:
  0.70509977827051
Confusion Matrix:
 [[  2   0  35   6   8   0]
 [  0   8  78  11  11   1]
 [  0   1 826   6  51   1]
 [  1   2 162  94  48   4]
 [  0   1 134   6 634   0]
 [  0   0  73   3  22  26]]
Classification Report:
               precision    recall  f1-score   support

            0       0.67      0.04      0.07        51
            1       0.67      0.07      0.13       109
```

```
              2       0.63      0.93      0.75       885
              3       0.75      0.30      0.43       311
              4       0.82      0.82      0.82       775
              5       0.81      0.21      0.33       124

       accuracy                           0.71      2255
      macro avg       0.72      0.40      0.42      2255
   weighted avg       0.72      0.71      0.66      2255
```

Conclusion

The model achieved moderate accuracy scores ranging from 69% to 71% .However further improvements can be made by exploring other algorithmns and fine-tuning the hyperparameters of the existing models.

✓ 0s completed at 1:21 PM