Classification of Text Documents

Text classification enables us to extract valuable insights from unstructured data.

The dataset we will be using will be of text data categorized into four labels: Technology, Sports, Politics and Entertainment. Each entry contains a short sentence or statement related to a specific topic with the label indicating the category it belongs to.

Import Libraries and Load the data

Import the pandas library, read the CSV file into a DataFrame, display the first few rows, and print the column names and data types.

In [79]:
```python
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv("/content/synthetic_text_data.csv")

display(df.head())
display(df.info())
```

	text	label
0	Artificial intelligence is advancing in health...	Technology
1	Football fans are excited about the upcoming W...	Sports
2	New policies regarding climate change have spa...	Politics
3	The latest blockbuster movie has shattered box...	Entertainment
4	Quantum computing promises to revolutionize in...	Technology

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 85 entries, 0 to 84
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    85 non-null     object
 1   label   85 non-null     object
dtypes: object(2)
memory usage: 1.5+ KB
None
```

Preprocess the data

Create a function to preprocess the text data by converting to lowercase, removing punctuation and special characters, tokenizing, removing stop words, and joining the tokens back into a string. Then, apply this function to the 'text' column and store the result in a new column 'preprocessed_text'. Finally, use TF-IDF Vectorizer to convert the 'preprocessed_text' into numerical features and store them in a variable.

In [80]:
```
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer

# Download NLTK data
nltk.download('punkt', quiet=True)
nltk.download('stopwords', quiet=True)
nltk.download('punkt_tab', quiet=True)


def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove punctuation and special characters
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Tokenize
    tokens = word_tokenize(text)
    # Remove stop words
    tokens = [word for word in tokens if word not in stopwords.words('english'
    # Join tokens back into a string
    return ' '.join(tokens)

# Apply preprocessing to the 'text' column
df['preprocessed_text'] = df['text'].apply(preprocess_text)

# Use TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000) # Limit features to avoi
tfidf_features = tfidf_vectorizer.fit_transform(df['preprocessed_text'])

display(df.head())
display(tfidf_features.shape)
```

	text	label	preprocessed_text
0	Artificial intelligence is advancing in health...	Technology	artificial intelligence advancing healthcare n...
1	Football fans are excited about the upcoming W...	Sports	football fans excited upcoming world cup teams...
2	New policies regarding climate change have spa...	Politics	new policies regarding climate change sparked ...
3	The latest blockbuster movie has shattered box...	Entertainment	latest blockbuster movie shattered box office ...
4	Quantum computing promises to revolutionize in...	Technology	quantum computing promises revolutionize indus...

```
(85, 423)
```

Split the data

Split the data into training and testing sets.

In [81]:
```python
from sklearn.model_selection import train_test_split

X = tfidf_features
y = df['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ranc

display(X_train.shape)
display(X_test.shape)
display(y_train.shape)
display(y_test.shape)
```

```
(63, 423)
(22, 423)
(63,)
(22,)
```

Select and train models

We will build classification models, Naive Bayes, SVM, Logistic Regression, or a simple neural network and train them on the training data. Instantiate and train at least four different classification model objects on the training data.

In [82]:
```python
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
```

```
# Instantiate models
mnb_model = MultinomialNB()
svm_model = SVC()
lr_model = LogisticRegression(max_iter=1000) # Increase max_iter for convergen
mlp_model = MLPClassifier(max_iter=1000) # Increase max_iter for convergence

# Train models
mnb_model.fit(X_train, y_train)
svm_model.fit(X_train, y_train)
lr_model.fit(X_train, y_train)
mlp_model.fit(X_train, y_train)

print("Models trained successfully.")
```

```
Models trained successfully.
```

Evaluate the models

Evaluate the performance of each trained model using appropriate metrics (e.g., accuracy, precision, recall, F1-score) on the testing data.

Evaluate the performance of each trained model using accuracy and classification report.

In [83]:
```
from sklearn.metrics import accuracy_score, classification_report

# Evaluate Multinomial Naive Bayes
mnb_predictions = mnb_model.predict(X_test)
mnb_accuracy = accuracy_score(y_test, mnb_predictions)
mnb_report = classification_report(y_test, mnb_predictions)

print("Multinomial Naive Bayes Performance:")
print(f"Accuracy: {mnb_accuracy:.4f}")
print("Classification Report:")
print(mnb_report)

# Evaluate SVM
svm_predictions = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)
svm_report = classification_report(y_test, svm_predictions)

print("\nSVM Performance:")
print(f"Accuracy: {svm_accuracy:.4f}")
print("Classification Report:")
print(svm_report)

# Evaluate Logistic Regression
lr_predictions = lr_model.predict(X_test)
lr_accuracy = accuracy_score(y_test, lr_predictions)
lr_report = classification_report(y_test, lr_predictions)
```

```python
print("\nLogistic Regression Performance:")
print(f"Accuracy: {lr_accuracy:.4f}")
print("Classification Report:")
print(lr_report)

# Evaluate MLP Classifier
mlp_predictions = mlp_model.predict(X_test)
mlp_accuracy = accuracy_score(y_test, mlp_predictions)
mlp_report = classification_report(y_test, mlp_predictions)

print("\nMLP Classifier Performance:")
print(f"Accuracy: {mlp_accuracy:.4f}")
print("Classification Report:")
print(mlp_report)
```

```
Multinomial Naive Bayes Performance:
Accuracy: 0.6818
Classification Report:
               precision    recall  f1-score   support

Entertainment       0.33      0.50      0.40         2
     Politics       1.00      0.83      0.91         6
       Sports       0.00      0.00      0.00         4
   Technology       0.64      0.90      0.75        10

     accuracy                           0.68        22
    macro avg       0.49      0.56      0.51        22
 weighted avg       0.60      0.68      0.63        22


SVM Performance:
Accuracy: 0.6364
Classification Report:
               precision    recall  f1-score   support

Entertainment       0.50      0.50      0.50         2
     Politics       1.00      0.67      0.80         6
       Sports       0.00      0.00      0.00         4
   Technology       0.56      0.90      0.69        10

     accuracy                           0.64        22
    macro avg       0.52      0.52      0.50        22
 weighted avg       0.57      0.64      0.58        22


Logistic Regression Performance:
Accuracy: 0.6818
Classification Report:
               precision    recall  f1-score   support

Entertainment       0.50      0.50      0.50         2
     Politics       1.00      0.83      0.91         6
       Sports       0.00      0.00      0.00         4
   Technology       0.60      0.90      0.72        10

     accuracy                           0.68        22
    macro avg       0.53      0.56      0.53        22
 weighted avg       0.59      0.68      0.62        22


MLP Classifier Performance:
Accuracy: 0.8636
Classification Report:
               precision    recall  f1-score   support

Entertainment       0.50      1.00      0.67         2
     Politics       1.00      1.00      1.00         6
       Sports       1.00      0.75      0.86         4
   Technology       0.89      0.80      0.84        10
```

```
    accuracy                           0.86        22
   macro avg       0.85      0.89      0.84        22
weighted avg       0.90      0.86      0.87        22
```

Compare and select the best model

Based on the accuracy scores and classification reports from the previous step, I will compare the models and write a conclusion selecting the best-performing one.

In [84]:
```
# The accuracy scores from the previous step are:
# mnb_accuracy = 0.6818
# svm_accuracy = 0.6364
# lr_accuracy = 0.6818
# mlp_accuracy = 0.8636

# Comparing the accuracy scores
best_accuracy = max(mnb_accuracy, svm_accuracy, lr_accuracy, mlp_accuracy)

if best_accuracy == mnb_accuracy:
    best_model = "Multinomial Naive Bayes"
elif best_accuracy == svm_accuracy:
    best_model = "SVM"
elif best_accuracy == lr_accuracy:
    best_model = "Logistic Regression"
else:
    best_model = "MLP Classifier"

print(f"Based on accuracy, the best performing model is: {best_model} with an
```

```
Based on accuracy, the best performing model is: MLP Classifier with an accurac
y of 0.8636
```

Conclusion

The MLP Classifier achieved the highest accuracy (0.8636) on the test set compared to Multinomial Naive Bayes (0.6818), Logistic Regression (0.6818), and SVM (0.6364). While Multinomial Naive Bayes and Logistic Regression performed similarly, and SVM had the lowest accuracy, the MLP Classifier demonstrated significantly better overall performance. Therefore, based on the evaluation metrics, the MLP Classifier is the best-performing model for this text classification task.

Summary:

- The dataset contains text data and corresponding labels for classification.
- Preprocessing involved converting text to lowercase, removing punctuation and stop words, and tokenization.
- Text data was transformed into numerical features using TF-IDF vectorization, resulting in 85 samples and 423 features.
- The data was split into training (63 samples) and testing (22 samples) sets.
- Four classification models were trained: Multinomial Naive Bayes, SVM, Logistic Regression, and MLP Classifier.
- Model evaluation on the test set showed the following accuracies:
    - Multinomial Naive Bayes: 0.6818
    - SVM: 0.6364
    - Logistic Regression: 0.6818
    - MLP Classifier: 0.8636
- The MLP Classifier achieved the highest accuracy among the evaluated models.

Next Steps

- The MLP (Multi-Layer Perceptron) Classifier is the best-performing model for this text classification task based on the evaluation metrics.
- Further tuning of the MLP Classifier's hyperparameters or exploring other advanced techniques could potentially improve performance further.