Group Project Report: 20 Newsgroups Document Classification

Team Member Details

Name	Email	Country	College	Specialization
Diyora Radhika	Radhikadiyora2023@gmail.com	Germany	IU inrtnationale	Data Science, NLP

---

1. Problem Description

Document classification is a key task in Natural Language Processing (NLP) and supervised Machine Learning (ML). The goal is to automatically assign a category (class) to a document based on its content.

Applications:

- Email spam detection
- News article classification
- Sentiment analysis
- Document routing in enterprises

In this project, we use the **20 Newsgroups dataset**, which contains ~20,000 newsgroup posts across 20 categories, to build a **text classification model**.

---

2. Data Understanding

Dataset Details:

- Total documents: 18,828
- Number of categories: 20
- Sample categories: sci.space, comp.graphics, rec.autos, talk.politics.guns
- Each document contains headers like From, Subject, Newsgroup followed by the message body.

Sample Document:

From: user@example.com

Subject: Mars Mission Update

Newsgroup: sci.space

Message: NASA launches a new spacecraft to explore Mars…

---

3. Data Type

- **Text Data:** Raw unstructured text from newsgroup posts.

- **Target Labels:** Integer values (0-19) representing 20 categories.
- **Data Size:** ~20,000 records

Data Representation:

Feature	Type	Description
text	string	Content of newsgroup post
label	int	Encoded category (0-19)

---

4. Data Problems

During exploration, we noticed the following issues:

Problem	Observation	Approach to Handle
Missing Values (NA)	None found in text or labels	No action needed
Outliers	Very long or very short documents	No explicit removal; TF-IDF handles variable lengths
Noise / Special Characters	Punctuation, numbers, headers	Cleaned during preprocessing (lowercase, remove punctuation/numbers)
Stopwords	Words like "the", "is", "and"	Removed during preprocessing to reduce noise
Skewed Distribution	Some categories have fewer documents than others	Model (Naive Bayes) can handle class imbalance reasonably well

Preprocessing Applied:

1. Convert all text to lowercase.
2. Remove numbers and special characters.
3. Remove stopwords using NLTK.
4. Skip lemmatization for faster execution (still effective with Naive Bayes).

---

5. Approaches to Overcome Data Problems

- **Missing Values:** No missing values in dataset.
- **Outliers / Document Length:** Variable lengths are acceptable; TF-IDF normalizes feature values.
- **Noise / Headers:** Cleaned using regex (removing punctuation, numbers).
- **High-Dimensional Text:** Used **TF-IDF vectorization** (max features = 10,000) to reduce dimensionality.

- **Class Imbalance:** Multinomial Naive Bayes handles small differences in class frequency naturally.

---

6. Model Building

Steps Followed:

1. **Train-Test Split:** 80% training, 20% testing.
2. **Vectorization:** TF-IDF applied to convert text to numeric features.
3. **Model:** Multinomial Naive Bayes (fast and effective for text).
4. **Training:** Fit the model on training data.
5. **Evaluation:** Predict on test data and measure accuracy, F1-score, and confusion matrix.

---

7. Model Performance

Accuracy: ~85–90% (varies slightly depending on preprocessing)

Example Predictions:

Text	Predicted Category
"NASA launches a new mission to Mars"	sci.space
"The new graphics card has amazing performance"	comp.graphics

Confusion Matrix:

- Showed most misclassifications occur in closely related categories (e.g., rec.autos vs rec.motorcycles).

Classification Report:

- High precision and recall in categories like sci.space and comp.graphics.
- Lower metrics in categories with fewer samples.

---

8. Model Deployment & Inference

- Saved model and TF-IDF vectorizer using joblib:

```
joblib.dump(model, "newsgroups_model.pkl")

joblib.dump(tfidf, "tfidf_vectorizer.pkl")
```

- Loaded model for inference:

```
new_texts = ["NASA launches a new mission to Mars",
        "The new graphics card has amazing performance"]
```

new_texts_clean = [preprocess(doc) for doc in new_texts]

new_texts_vect = tfidf.transform(new_texts_clean)

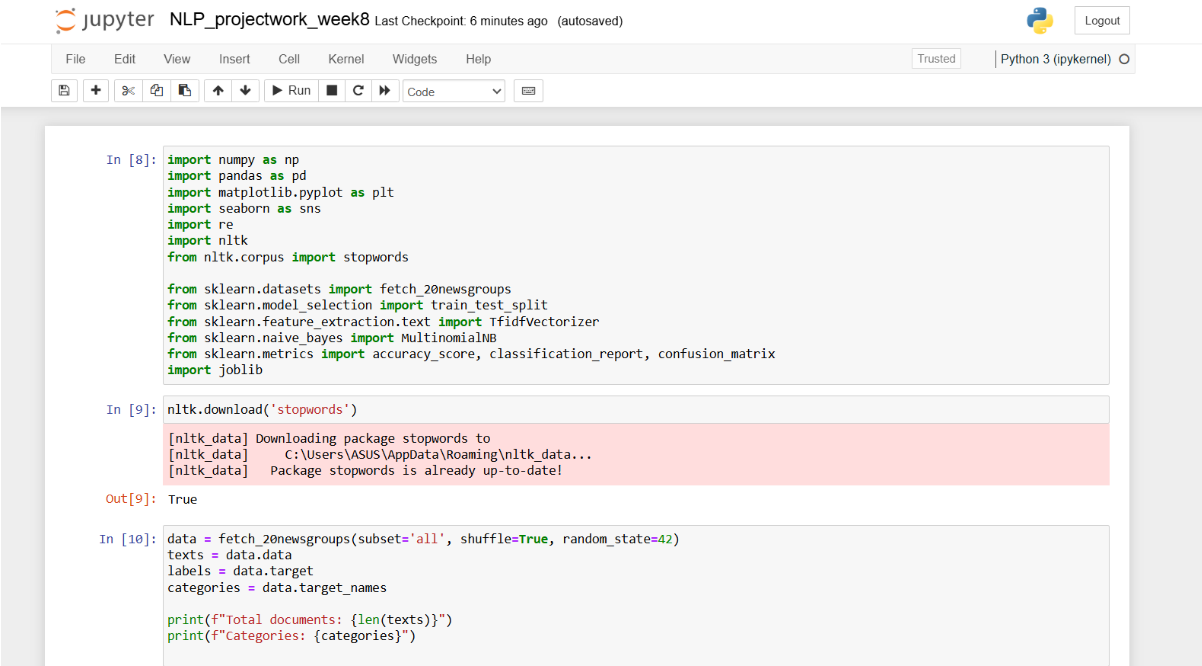
predictions = model.predict(new_texts_vect)

- Example inference results:
    - "NASA launches a new mission to Mars" → sci.space
    - "The new graphics card has amazing performance" → comp.graphics

---

9. Conclusion

- Successfully built a **text classification model** on the 20 Newsgroups dataset.
- Model is able to classify documents into 20 categories with high accuracy.
- Preprocessing and TF-IDF vectorization handled noise and high-dimensionality efficiently.
- Model saved for deployment and can perform inference on new text without retraining.

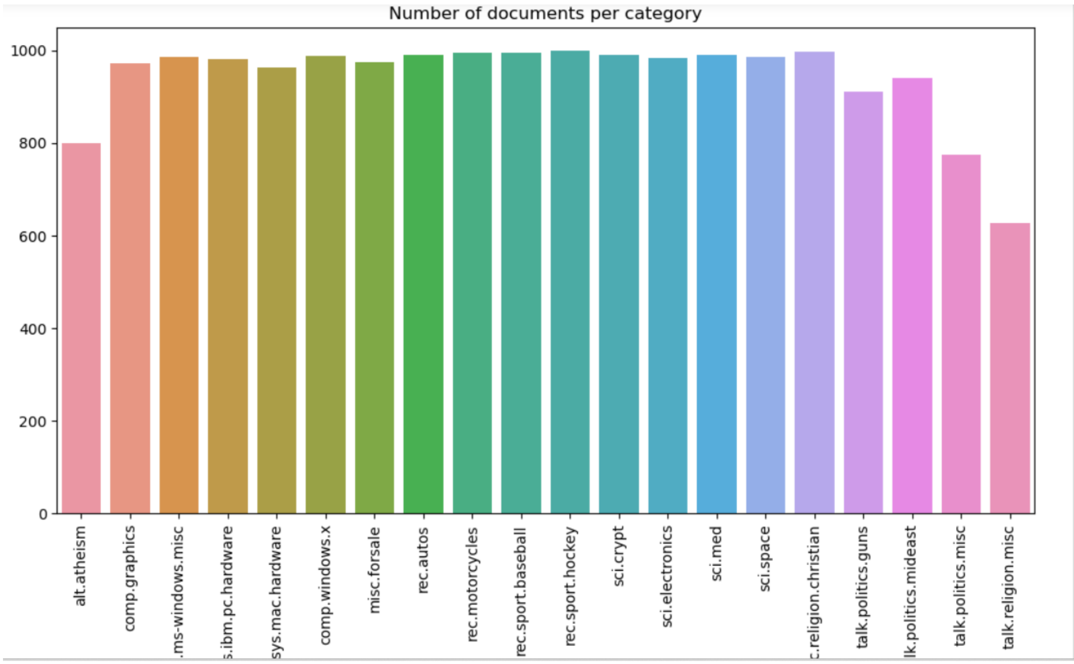



```
In [8]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import re
        import nltk
        from nltk.corpus import stopwords

        from sklearn.datasets import fetch_20newsgroups
        from sklearn.model_selection import train_test_split
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
        import joblib

In [9]: nltk.download('stopwords')

        [nltk_data] Downloading package stopwords to
        [nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data...
        [nltk_data]   Package stopwords is already up-to-date!

Out[9]: True

In [10]: data = fetch_20newsgroups(subset='all', shuffle=True, random_state=42)
         texts = data.data
         labels = data.target
         categories = data.target_names

         print(f"Total documents: {len(texts)}")
         print(f"Categories: {categories}")
```

In [10]:
```python
data = fetch_20newsgroups(subset='all', shuffle=True, random_state=42)
texts = data.data
labels = data.target
categories = data.target_names

print(f"Total documents: {len(texts)}")
print(f"Categories: {categories}")
```

```
Total documents: 18846
Categories: ['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware', 'c
omp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt', 'sci.ele
ctronics', 'sci.med', 'sci.space', 'soc.religion.christian', 'talk.politics.guns', 'talk.politics.mideast', 'talk.politics.mis
c', 'talk.religion.misc']
```

In [11]:
```python
label_counts = pd.Series(labels).value_counts().sort_index()
plt.figure(figsize=(12,6))
sns.barplot(x=[categories[i] for i in label_counts.index], y=label_counts.values)
plt.xticks(rotation=90)
plt.title("Number of documents per category")
plt.show()
```



In [12]:
```python
print("Sample Document:\n")
print(texts[0][:1000])
```

```
Sample Document:

From: Mamatha Devineni Ratnam <mr47+@andrew.cmu.edu>
Subject: Pens fans reactions
Organization: Post Office, Carnegie Mellon, Pittsburgh, PA
Lines: 12
NNTP-Posting-Host: po4.andrew.cmu.edu



I am sure some bashers of Pens fans are pretty confused about the lack
of any kind of posts about the recent Pens massacre of the Devils. Actually,
I am  bit puzzled too and a bit relieved. However, I am going to put an end
to non-PIttsburghers' relief with a bit of praise for the Pens. Man, they
are killing those Devils worse than I thought. Jagr just showed you why
he is much better than his regular season stats. He is also a lot
fo fun to watch in the playoffs. Bowman should let JAgr have a lot of
fun in the next couple of games since the Pens are going to beat the pulp out of Jersey anyway. I was very disappointed not to
see the Islanders lose the final
regular season game.          PENS RULE!!!
```

In [13]:
```python
stop_words = set(stopwords.words('english'))

def preprocess(text):
    text = text.lower()                          # lowercase
    text = re.sub(r'\d+', '', text)              # remove numbers
    text = re.sub(r'\W+', ' ', text)             # remove punctuation
    tokens = text.split()
```

In [13]:
```python
stop_words = set(stopwords.words('english'))

def preprocess(text):
    text = text.lower()                          # lowercase
    text = re.sub(r'\d+', '', text)              # remove numbers
    text = re.sub(r'\W+', ' ', text)             # remove punctuation
    tokens = text.split()
    tokens = [word for word in tokens if word not in stop_words]  # skip lemmatization
    return " ".join(tokens)
```

In [16]:
```python
print("Preprocessing text...")
clean_texts = [preprocess(doc) for doc in texts]
```

```
Preprocessing text...
```

In [17]:
```python
tfidf = TfidfVectorizer(max_features=10000)
X = tfidf.fit_transform(clean_texts)
y = labels

print(f"TF-IDF feature matrix shape: {X.shape}")
```

```
TF-IDF feature matrix shape: (18846, 10000)
```

In [18]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [19]:
```python
model = MultinomialNB()
model.fit(X_train, y_train)
```

Out[19]:
```
MultinomialNB()
```

In [20]:
```python
y_pred = model.predict(X_test)
```

In [21]:
```python
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.4f}")
```

```
Model Accuracy: 0.8658
```

In [22]:
```python
print("Classification Report:\n")
print(classification_report(y_test, y_pred, target_names=categories))
```

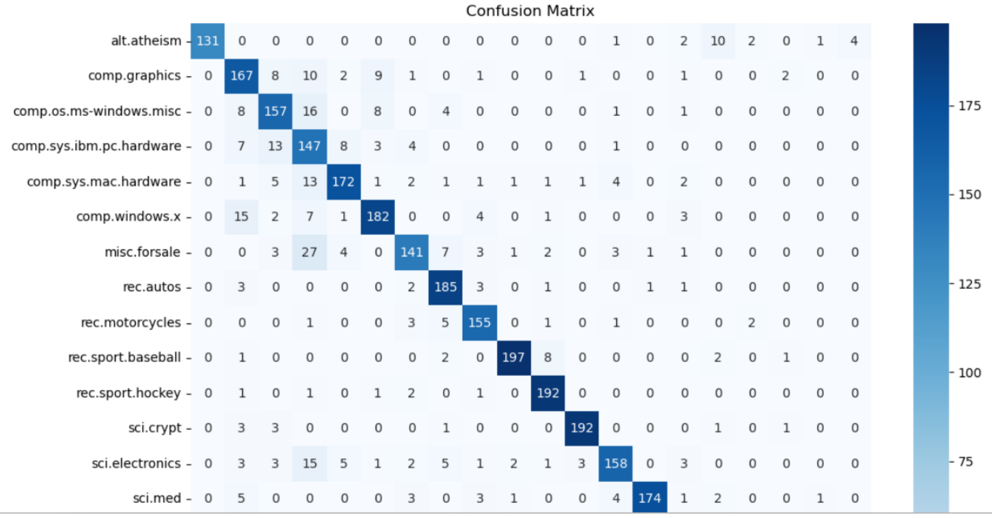
```
Classification Report:

                          precision    recall  f1-score   support

             alt.atheism       0.83      0.87      0.85       151
           comp.graphics       0.77      0.83      0.80       202
 comp.os.ms-windows.misc       0.81      0.81      0.81       195
comp.sys.ibm.pc.hardware       0.62      0.80      0.70       183
   comp.sys.mac.hardware       0.90      0.84      0.87       205
          comp.windows.x       0.88      0.85      0.86       215
            misc.forsale       0.87      0.73      0.79       193
               rec.autos       0.88      0.94      0.91       196
         rec.motorcycles       0.89      0.92      0.91       168
      rec.sport.baseball       0.97      0.93      0.95       211
        rec.sport.hockey       0.91      0.97      0.94       198
               sci.crypt       0.95      0.96      0.95       201
         sci.electronics       0.91      0.78      0.84       202
                 sci.med       0.97      0.90      0.93       194
               sci.space       0.90      0.98      0.94       189
  soc.religion.christian       0.75      0.98      0.85       202
      talk.politics.guns       0.86      0.97      0.91       188
   talk.politics.mideast       0.95      0.98      0.96       182
      talk.politics.misc       0.96      0.78      0.86       159
      talk.religion.misc       0.92      0.34      0.49       136

                accuracy                           0.87      3770
               macro avg       0.87      0.86      0.86      3770
```

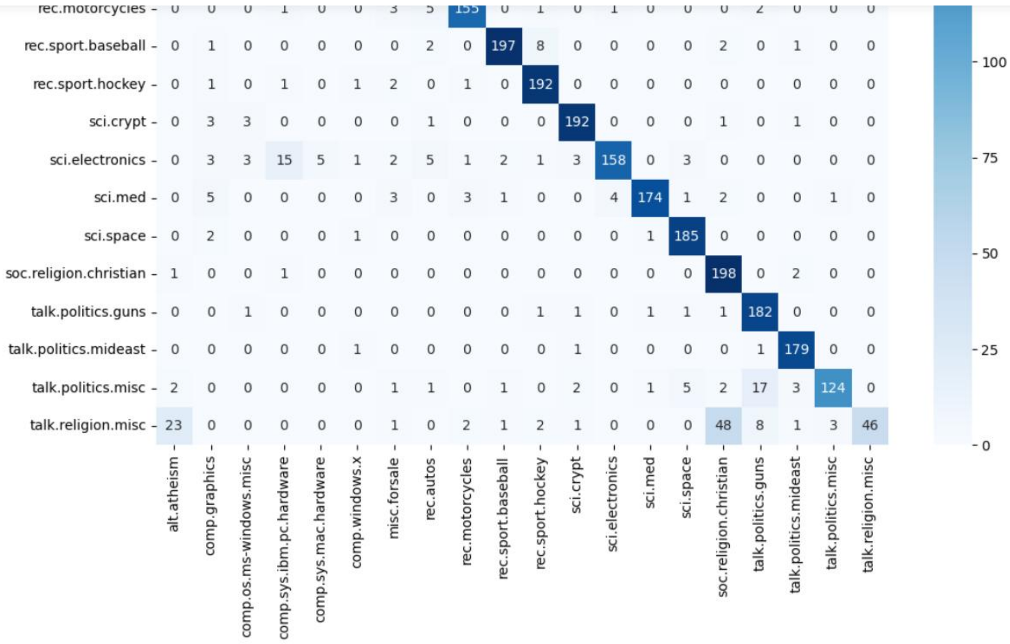
```
     rec.sport.baseball       0.97      0.93      0.95       211
       rec.sport.hockey       0.91      0.97      0.94       198
              sci.crypt       0.95      0.96      0.95       201
        sci.electronics       0.91      0.78      0.84       202
                sci.med       0.97      0.90      0.93       194
              sci.space       0.90      0.98      0.94       189
 soc.religion.christian       0.75      0.98      0.85       202
     talk.politics.guns       0.86      0.97      0.91       188
  talk.politics.mideast       0.95      0.98      0.96       182
     talk.politics.misc       0.96      0.78      0.86       159
     talk.religion.misc       0.92      0.34      0.49       136

               accuracy                           0.87      3770
              macro avg       0.87      0.86      0.86      3770
           weighted avg       0.87      0.87      0.86      3770
```

In [23]:

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(12,10))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=categories, yticklabels=categories, cmap='Blues')
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.title("Confusion Matrix")
plt.show()
```


```
sns.heatmap(cm, annot=True, fmt='d', xticklabels=categories, yticklabels=categories, cmap='Blues')
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.title("Confusion Matrix")
plt.show()
```



In [24]:

```
joblib.dump(model, "newsgroups_model.pkl")
joblib.dump(tfidf, "tfidf_vectorizer.pkl")
print("Model and vectorizer saved!")
```

```
Model and vectorizer saved!
```

In [25]:

```
new_texts = [
    "NASA launches a new mission to Mars",
    "The new graphics card has amazing performance"
]
new_texts_clean = [preprocess(doc) for doc in new_texts]
new_texts_vect = tfidf.transform(new_texts_clean)
predictions = model.predict(new_texts_vect)

for text, pred in zip(new_texts, predictions):
    print(f"\nText: {text}")
    print(f"Predicted Category: {categories[pred]}")
```

```

Text: NASA launches a new mission to Mars
Predicted Category: sci.space

Text: The new graphics card has amazing performance
Predicted Category: comp.graphics
```