Practical 10

Name – Riya konde

Batch – T3

Roll No – 0046

```
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('wordnet')
nltk.download('stopwords')
```

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer, PorterStemmer
```

```
from google.colab import files

uploaded = files.upload()
```

```
Choose Files spam.csv
• spam.csv(text/csv) - 480130 bytes, last modified: 4/9/2025 - 100% done
Saving spam.csv to spam (1).csv
```

```
[29] df = pd.read_csv(r"spam.csv", encoding="latin1")
     df
```

```
df["POS_Tags"] = df["Tokenized"].apply(nltk.pos_tag)
df
```

```
stopw = set(stopwords.words('english'))
df['Filter'] = df['Tokenized'].apply(lambda tokens: [word for word in tokens if word.lower() not in stopw])
df
```

```
[36] from sklearn.feature_extraction.text import TfidfVectorizer
```

```
df["Processed"] = df['Lemmatized'].apply(lambda tokens: ' '.join(tokens))
df
```

```
df.columns = ["Type", "Messege"]
df
```

```
df["Tokenized"] = df["Messege"].apply(word_tokenize)
df
```

```
stemmer = PorterStemmer()
df['Stemmed'] = df['Filter'].apply(lambda tokens: [stemmer.stem(word) for word in tokens])
df
```

```
[38] tf_vector = TfidfVectorizer()
tf_matrix = tf_vector.fit_transform(df['Processed'])
```

```
[39] tfidf = pd.DataFrame(tf_matrix.toarray(), columns=tf_vector.get_feature_names_out())
```

```
tfidf['Type'] = df['Type']
tfidf
```

```
lemma = WordNetLemmatizer()
df['Lemmatized'] = df['Filter'].apply(lambda tokens: [lemma.lemmatize(word) for word in tokens])
df
```

```
print(tfidf)
```

```
       00  000  000pes  008704050406  0089  0121  01223585236  01223585334  \
0     0.0  0.0     0.0           0.0   0.0   0.0          0.0          0.0
1     0.0  0.0     0.0           0.0   0.0   0.0          0.0          0.0
2     0.0  0.0     0.0           0.0   0.0   0.0          0.0          0.0
3     0.0  0.0     0.0           0.0   0.0   0.0          0.0          0.0
4     0.0  0.0     0.0           0.0   0.0   0.0          0.0          0.0
...   ...  ...     ...           ...   ...   ...          ...          ...
5567  0.0  0.0     0.0           0.0   0.0   0.0          0.0          0.0
5568  0.0  0.0     0.0           0.0   0.0   0.0          0.0          0.0
5569  0.0  0.0     0.0           0.0   0.0   0.0          0.0          0.0
5570  0.0  0.0     0.0           0.0   0.0   0.0          0.0          0.0
5571  0.0  0.0     0.0           0.0   0.0   0.0          0.0          0.0
```

Conclusion

Document preprocessing is essential for cleaning and structuring text data. Techniques like tokenization, stop word removal, stemming, and lemmatization help simplify and standardize text. TF-IDF then converts this text into numerical form, highlighting important terms for analysis. Together, they enable more accurate and efficient NLP and machine learning outcomes.