# News Classification Based on Their Headlines

ITE-G7-E2: **Group 05**

Supervised by: **Chap Chanpiseth**

January 9, 2024

## 1. Objective:

The aim of this project is to classify news based on headlines using various machine learning techniques such as Decision Tree, Multinomial Naive Bayes, and Artificial Neural Network. The objective is to accurately categorize news into predefined classes using only the information present in their headlines.

## 2. Requirement Libraries:

- pandas
- scikit-learn
- numpy
- seaborn
- jupyter
- nltk

```
#import importance libraries
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn import set_config
import nltk
import re
import string
```

## 3. **Read Dataset to dataframe**

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

src_file = 'uci-news-aggregator.csv'

path_to_file = '/Colab Notebooks/'

src_filepath = drive_path + '/My Drive' + path_to_file + src_file

# Replace src_filepath with the actual file path if you use other IDE beside
colab
dataframe = pd.read_csv(src_filepath, encoding="utf8",
usecols=['TITLE', 'CATEGORY']) dataframe.columns
```

OUTPUT:

```
Mounted at /content/drive
Index(['TITLE', 'CATEGORY'], dtype='object')
```

```
# Display the dataset
Dataframe
```

OUTPUT:

```
                                                 TITLE CATEGORY
0         Fed official says weak data caused by weather,...          b 1
Fed's Charles Plosser sees high bar for change...        b 2       US open:
Stocks fall after Fed official hints ...        b 3       Fed risks falling
'behind the curve', Charles ...          b 4        Fed's Plosser: Nasty
Weather Has Curbed Job Gr...        b

...                                                    ...       ...
137660  An 'Orphan Black' primer: Everything that happ...        e 137661
One Actress, Many Amazing Performances: Orphan...        e 137662  Orphan
Black's Tatiana Maslany On How The Show...        e 137663  BBC's 'Orphan
Black' returns, engineered to ne...        e 137664  Orphan Black' season
2 review: Tatiana Maslany...        e

[137665 rows x 2 columns]
```

## 4. **Data PreProcessing**

```
# Preprocessing #check for missing data
if(any(dataframe.isnull().any())):
 print('Missing Data\n')
 print(dataframe.isnull().sum())
else:
```

```
 print('NO missing data')
```

OUTPUT:

```
NO missing data
```

```
# check for duplicate
if(any(dataframe.duplicated())==True):
print('Duplicate rows found')
print('Number of duplicate rows= ',
dataframe[dataframe.duplicated()].shape[0])
    dataframe.drop_duplicates(inplace=True,keep='first')
dataframe.reset_index(inplace=True,drop=True)
    print('Dropping duplicates\n')
print(dataframe.shape) else:
print('NO duplicate data')
```

OUTPUT:

```
Duplicate rows found
Number of duplicate rows=  4866
Dropping duplicates
(132799, 2)
```

```
# download the library to for the nltk functions to use in the
cleaning process nltk.download('stopwords')
nltk.download('punkt') nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
set_config(transform_output="pandas")

wnl = WordNetLemmatizer()


# Function for cleaning and tokenize the headline def tokenize(doc):
document = doc.lower()  # convert the content of the headline to
lowercase
    document = re.sub(r'\d+', '',
                      document)  # remove all of the digits inside of
the content (using regular expressions)
    document = document.translate(str.maketrans('', '',
string.punctuation))  # remove the puntuations (, . ! # ...)
```

```
document = document.strip()  # remove the spaces at the start and end
of the headline
    return [wnl.lemmatize(token) for token in word_tokenize(document)
if token not in stopwords.words('english')]
    # tokenize the headlines
```

```
    # and then filter only the words that are not in the english
stopwords (words that are commonly used and give no benifits to the
classifier)
    # and finally lemmatize all of the tokens


# The preprocess pipeline preprocessor
= Pipeline([
    ('vect', CountVectorizer(tokenizer=tokenize)),  # passing custom
tokenizer method for the CountVectorizer to use
    ('tfidf', TfidfTransformer()),
])

tfidf_dataset = preprocessor.fit_transform(dataframe["TITLE"].values) #
process the training dataset
```

OUTPUT:

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/
text.py:528: UserWarning: The parameter 'token_pattern' will not be
used since 'tokenizer' is not None'
  warnings.warn(
```

# 5. Model Implementation and Evaluation

## 5.1 Label Encoder

```
le = LabelEncoder()
class_label = le.fit_transform(dataframe["CATEGORY"])
# list(le.classes_)
class_label
array([0, 0, 0, ..., 1, 1, 1])
X_train, X_test, y_train, y_test = train_test_split(
tfidf_dataset.toarray(),     class_label,
test_size = 0.3,     random_state=42
)
```

## 5.2 Decision Tree Classifier

```
dt_classifier = DecisionTreeClassifier(criterion="gini",
splitter="best")
dt_classifier.fit(X_train, y_train) dt_predictions =
dt_classifier.predict(X_test)

# Evaluation
print("accuracy score of Decision Tree:")
print(accuracy_score(y_test, dt_predictions))
```

OUTPUT:

```
accuracy score of Decision Tree:
0.9151606425702812
```

## 5.3 Multinomial Naive Bayes Classifier

```
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)
nb_predictions = nb_classifier.predict(X_test)

# Evaluation
print("accuracy score of Multinomial Naive Bayes:")
print(accuracy_score(y_test, nb_predictions))
```

OUTPUT:

```
accuracy score of Multinomial Naive Bayes:
0.9354166666666667
```

### 5.4 **Artificial Neural Network**

```
nn_classifier = MLPClassifier()
nn_classifier.fit(X_train, y_train)
nn_predictions = nn_classifier.predict(X_test)

# Evaluation
print("accuracy score of Artificial Neural Network:")
print(accuracy_score(y_test, nn_predictions))
```

OUTPUT:

```
accuracy score of Artificial Neural Network:
0.9492720883534137
```

## 7. **Conclusion**

In this project, we explored three machine learning models - Decision Tree, Multinomial Naive Bayes, and Artificial Neural Network - for news headline classification. Each model brought unique strengths to the task:

- The Decision Tree likely offered a clear, interpretable model but might have faced limitations in handling complex text data.
- The Multinomial Naive Bayes, often effective for text classification, probably performed well due to its simplicity and efficiency.
- The Artificial Neural Network, with its advanced pattern recognition capabilities, was expected to provide the highest accuracy.

While each model has its advantages, the choice often depends on the specific requirements and nature of the dataset. Future work could focus on further tuning these models, exploring advanced techniques like deep learning, and experimenting with different feature extraction methods for improved accuracy.