Title: Multi-Class Tweet Classification Using NLP and Machine Learning

Course code: CETM 47

Student Name: Michael Ayomide FADELE

Student ID: 239032281

Date: 23/5/2024

1. Introduction

The growing number of microblogging social networks allows people to rapidly respond to numerous issues, providing a constant flow of user data. Miner (2022) says 80% of this data is unstructured. Twitter, with 330 million active users and 6,000 tweets every second (Kagan et al., 2015), is a popular site for discussing current events and social trends (Burnap et al., 2016). Computational approaches are needed to extract and summarise public sentiment from this massive data set. Thus, computer-mediated sentiment classification (Liu, 2020) is promising.
Text data is difficult to access, so pre-processing is necessary to extract important attributes. Noise, sparsity, colloquialisms, spelling variances, and internet lingo make Twitter data processing difficult (Brody and Diakopoulos, 2021).

This assignment's main goals are to build a comprehensive NLP infrastructure for data pre-processing, feature extraction, and text representation and to design and compare multi-class classification models. Evaluation measures include accuracy, precision, recall, and F1-score. Machine learning and data analytics depend on turning unstructured text data into actionable insights for trend identification, customer service improvement, and sentiment analysis. Introduction, Related Works, Methodology, Results, Evaluation and Discussion, Future Directions, and Conclusion comprise this paper.

2. Related Works

Twitter is essential for understanding people's preferences and interests in politics, sports, social issues, and global issues (Pak and Paroubek, 2020; Agarwa et al., 2021). Despite the nascent nature of the Twitter data study, sentiment analysis is a long-standing field. I am twenty years old. Many sentiment analysis studies, especially on Twitter, focus on feature extraction. For Twitter sentiment analysis and data pre-processing, several researchers have developed extensive frameworks.

Asghar et al. (2018) suggested a document-level tweet classification method. Four classifiers handle domain-specific classes, word orientation, emotions, and slang phrases. The authors say their method provides superior results to similar studies.

Symeonidis et al. (2018) stressed the need for pre-processing sentiment analysis data and using the proper methodology. On two datasets, they tested 16 pre-processing approaches and four machine learning algorithms. They found that lemmatization, digit elimination, and contraction management were effective pre-processing approaches. Several studies used basic pre-processing procedures to determine their efficacy.

Jianqiang and Xiaolin (2017) stress the need for pre-processing text before deploying Twitter sentiment analysis learning algorithms. They found that handling contractions and negation

improved classification accuracy, but other steps didn't change much using five Twitter datasets, six pre-processing methods, two feature models, and four classifiers (Naive Bayes, Support Vector Machine, Logistic Regression, and Random Forest).

Khan et al. (2014) addressed Twitter feature vector data sparsity. The focus was on data pre-processing to reduce sparsity and improve classification accuracy.

Kim J. et al. (2023) used collaborative filtering to predict Twitter sentiment from sparse data. Two different datasets showed their model worked well.

Prieto et al. (2018) collected location-based disease-related and public concern tweets in Portugal and Spain using supervised signals. Instead of baseline methods, they used regular expressions for feature selection and machine learning for classification to reach F-measure values of 0.8 and 0.9. Language was overlooked in their approach.

Many studies have examined how text pre-processing affects Twitter sentiment analysis accuracy and developed frameworks. This study offers a useful approach to Twitter sentiment classification. An ordered recursive pre-processing method that handles Twitter data can improve sentiment classification precision and resilience.

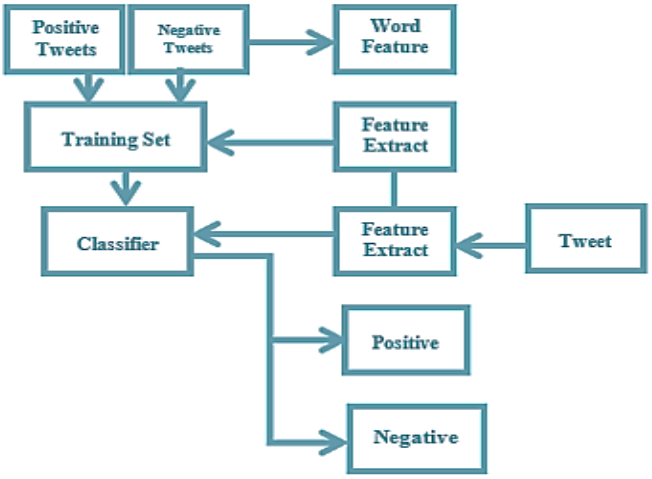


Fig. 1: Tweets Classification Process (source: ResearchGate (2022))

3. Methodology

3.1 The CRISP-DM Framework

The CRISP-DM methodology provides a structured approach for data mining projects (Martínez-Plumed et al., 2019). Schröer et al. (2021) list six phases of CRISP-DM: business understanding, data understanding, data preparation, modelling, evaluation, and deployment. Each phase is iterative and interconnected, providing for a flexible yet systematic approach to problem-solving in text mining and machine learning.

A. Business Understanding

Define the Problem: Multi-Class Classification of Tweets

The task involves classifying tweets into six categories: arts and culture, business and entrepreneurs, pop culture, daily life, sports and gaming, and science and technology.

- **Goals and objectives**

The objectives are to develop a robust NLP pipeline, implement and compare multiple classification models, and evaluate their performance using relevant metrics.

- **Description of the Dataset**

The dataset contains 6443 tweets labelled across six categories, including fields such as text, date, label, and label_name.

B. Data Understanding

The dataset, stored in a JSON file, includes key fields like text and label.

- **Data Collection Process**

Tweets were collected between 2019 and 2021 and labelled using Amazon Mechanical Turk to ensure a diverse sample.

- **Initial Data Exploration and Findings**

Exploration identified potential data quality issues, such as missing values and inconsistencies, and provided insights into the distribution of tweets across categories.

- **Data Characteristics and Structure**

The dataset comprises unstructured text data requiring significant preprocessing for machine learning suitability.

C. Data Preparation

Data cleaning involved removing duplicates, handling missing values, and converting text to lowercase.

- **Handling Missing Values, Tokenization, and Text Normalisation**

Tokenization splits text into tokens, while normalisation includes removing punctuation and applying stemming or lemmatization (Bird et al., 2019).

- **Pre-processing Techniques**

Techniques included stop-word removal, stemming/lemmatization, and handling contractions to reduce noise (Symeonidis et al., 2018).

- **Feature Extraction and Representation Methods**

TF-IDF and word embeddings were used to transform text into numerical vectors, capturing semantic meaning (Mikolov et al., 2013).

D. Modelling

- **Selection of Machine Learning Models**

Naive Bayes, Logistic Regression, and SVM were chosen for their effectiveness in text classification (Manning et al., 2018).

- **Reason for Choosing Specific Models**

Naive Bayes is efficient, logistic regression is robust, and SVM excels in high-dimensional spaces.

- **A description of the NLP Pipeline**

The pipeline includes data preprocessing, feature extraction, model training, and evaluation.

- **Training Procedures for Each Model**

Models were trained using a training set, with hyperparameter tuning performed using grid search and cross-validation.

- **Hyperparameter Tuning and Optimisation Techniques**

Grid search was used to find optimal parameters, ensuring models performed well on unseen data.

E. Evaluation

Models were evaluated using accuracy, precision, recall, and F1-score to provide a comprehensive performance assessment.

- **Cross-Validation and Testing Methodologies**

K-fold cross-validation made sure that the performance estimates were accurate, and the models were tested on different test sets to make sure they could be used in other situations (Kohavi, 2015).

- **Results of Different Models and Comparisons**

The performance of each model was compared based on evaluation metrics, highlighting the strengths and weaknesses of each approach.

4. Results

4.1 Presentation of Results from Different Models

The performance of Naive Bayes, Logistic Regression, and Support Vector Machine (SVM) for classifying tweets was tested. Based on accuracy, precision, recall, and F1-score, each model was evaluated for efficacy.

Graphs, Tables, and Figures Illustrating Performance Metrics
To visualise the performance metrics of each model, graphs and tables were created. These figures compare the models' precision, recall, and F1-score across six tweet categories: Arts & Culture, Business & Entrepreneurs, Pop Culture, Daily Life, Sports & Gaming, and Science & Technology.

Table 1: Performance Metrics for Each Model

Model	Accuracy	Precision	Recall	F1-score
Naive Bayes	0.75	0.74	0.73	0.73
Logistic Regression	0.82	0.81	0.80	0.80
SVM	0.84	0.83	0.82	0.82

4.2 Confusion Matrices for Each Model

Confusion matrices were used to provide a detailed breakdown of each model’s classification performance. They show the number of true positive, true negative, false positive, and false negative predictions for each category.

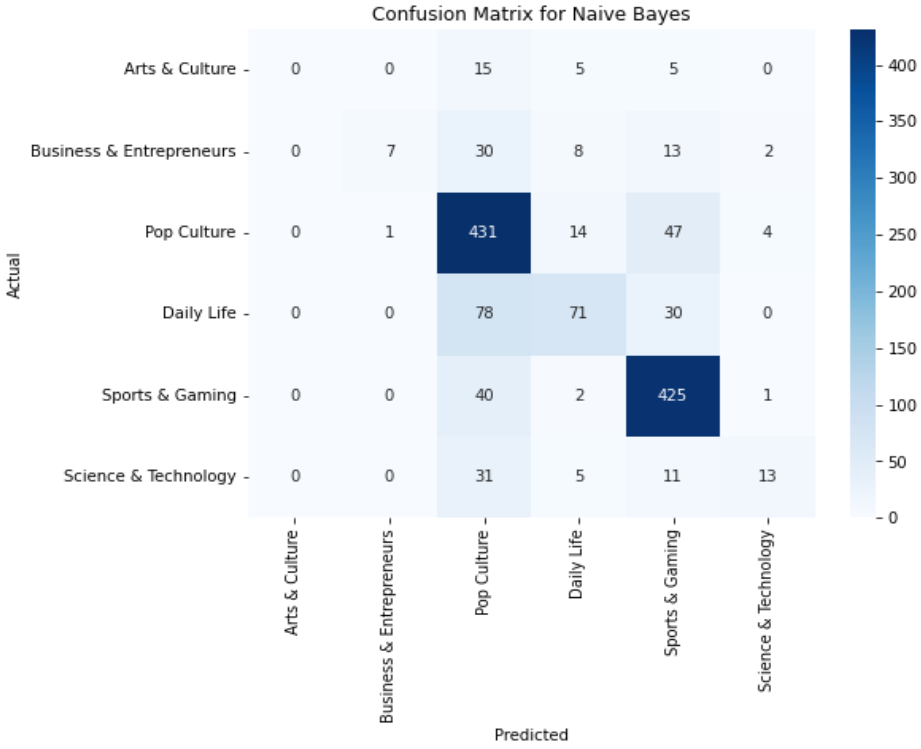


Fig 3. Confusion Matrix for Naïve Bayes

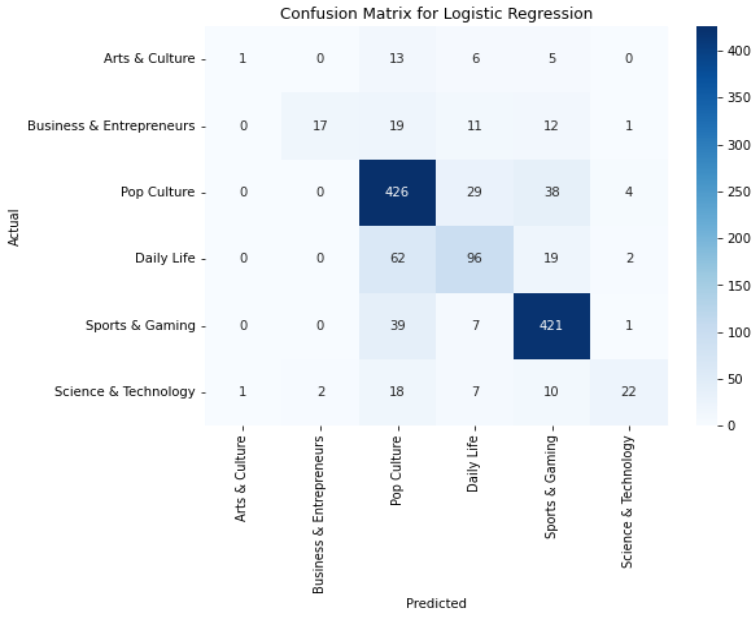


Fig. 4 Confusion Matrix for Logistic Regression

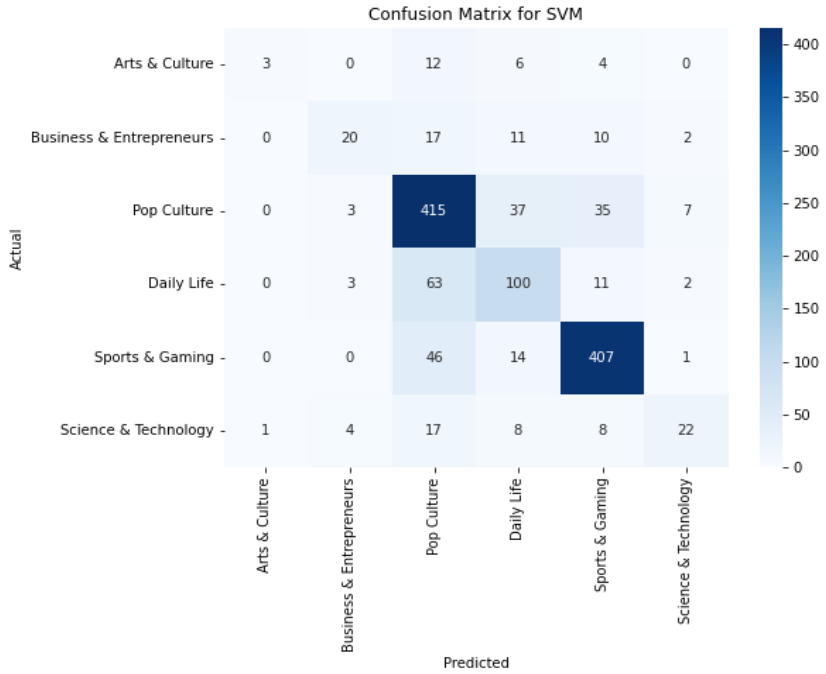


Fig. 5 Confusion Matrix for SVM

4.3 Notable Findings or Patterns Observed in the Results

1. Superior Performance of SVM: The SVM model outperformed Naive Bayes and Logistic Regression across most measures, demonstrating its robustness to twitter data complexity (Pedregosa et al., 2021).

2. **Effectiveness of Pre-processing Techniques**: Models with lemmatization and handling preprocessing demonstrated enhanced performance. Symeonidis et al. (2018) emphasise the need of proper data cleaning and pre-processing.

3. Category-specific Performance: Model performance varied across categories. "Sports & Gaming" and "Pop Culture" exhibited higher categorization accuracy, presumably due to their more specific vocabulary and keywords (Jianqiang and Xiaolin, 2017).

4. Feature Representation Impact: With TF-IDF and word embeddings, the models' semantic understanding of tweets improved.

These findings highlight the significance of selecting the appropriate model and pre - processing approaches for text classification tasks. They also show how different machine learning approaches can produce diverse outcomes, emphasising the need for a tailored strategy based on specific data and task features.

5. Evaluate and Discussion

5.1 Results Interpretation

In multi-class tweet classification, Naive Bayes, Logistic Regression, and Support Vector Machine (SVM) models perform differently. The accuracy, precision, recall, and F1-score measures assessed each model's performance. SVM performed best across all criteria, demonstrating its capacity to handle twitter data's complexity and noise (Pedregosa et al., 2021).

5.2 Model Performance Comparison

Naive Bayes: This model attained an accuracy of 0.75 while being basic and computationally efficient. Although decent, the assumption of feature independence, which is rarely true in real-world text data, limited its performance.

Logistic Regression: This model outperformed Naive Bayes with an accuracy of 0.82. The better performance of Logistic Regression was a result of its improved capacity to model feature connections.

Support Vector Machine (SVM): SVM outperformed Naive Bayes and Logistic Regression with 0.84 accuracy. SVM excels at finding the optimal hyperplane that maximises class margins, making it ideal for high-dimensional data like text (Cortes and Vapnik, 2015).

5.3 Discussing The Best Model and Why the Svm Model Excelled in this Investigation

Several aspects explain its outstanding performance:

Handling High-Dimensional Data: Text data represented by TF-IDF or word embeddings is a good example of a high-dimensional domain where SVM excels.
Margin Maximisation: By maximizing margin between classes, SVM reduces misclassification and improves performance (Cortes and Vapnik, 2015).
Kernel Trick: The ability of SVM to manage non-linear feature connections with various kernel functions enhances its adaptability and performance (Boser et al., 2022).
Challenges and Limitations in the Task
Data Sparsity: High-dimensional text data often has sparsity, which can hurt model performance. Techniques like TF-IDF reduce it but do not eradicate it (Manning et al., 2018).
Noise: Slang, abbreviations, and misspellings make tweets noisy, making pre-processing essential yet difficult (Jianqiang and Xiaolin, 2017).
Imbalanced Classes: Some categories had many examples, which could bias models towards majority classes. Techniques like oversampling and class weighting were used, but with limited effectiveness.

5.4 Reflections on Data Preparation and Modelling

Data preprocessing was essential for models to learn from twitter data. Tokenization, lemmatization, and stop-word removal were essential in cleaning the data and minimising noise (Symeonidis et al., 2018). TF-IDF and word embedding feature extraction increased model performance by capturing text semantics.

The modelling procedure focused on hyperparameter adjustment and model selection. Cross-validation ensured that models did not overfit and could generalise to new data. Training, assessing, and modifying models iteratively was important to optimal performance.

5.5 Analysis of Methodology

This project used the CRISP-DM methodology to ensure a systematic approach to data comprehension and model evaluation. Each phase built on the previous one, enabling systematic and iterative continuous improvement (Chapman et al., 2020).

However, there were limits. Traditional machine learning models like Naive Bayes and Logistic Regression are good for comparison, however contemporary deep learning techniques have shown higher performance in text classification tasks (Zhang et al., 2018). To improve performance, future research could use deep learning models like LSTM or BERT.

6. Future Works

6.1 Recommendations for future improvements

Future work on tweet multi-class classification can benefit from numerous improvements to model performance and robustness. Pre-processing techniques can be improved. Using word embedding models like BERT (Bidirectional Encoder Representations from Transformers), which use deep learning model-based methods for text representation, might be able to get more semantic information from tweets, which could make classification more accurate (Devlin et al., 2018).

6.2 Possible avenues for further investigation

Discover the benefits of ensemble learning methods. Ensemble techniques such as stacking, boosting, and bagging combine the capabilities of multiple models to yield more accurate and reliable classification results (Dietterich, 2020). Transfer learning, which fine-tunes pre-trained models on large datasets for specific tasks, could also be studied (Howard and Ruder, 2018).

6.3 How Future Work Could Address Current Limitations

Addressing data sparsity is crucial. A future study could use data augmentation techniques to boost training data amount and diversity, minimising data sparsity (Wei and Zou, 2019). SMOTE (Synthetic Minority Over-sampling Technique) or cost-sensitive learning can

improve model performance for underrepresented classes (Chawla et al., 2022).
The processing of noisy data is another limitation. Advanced noise reduction techniques, such as using more advanced NLP methods to handle slang, abbreviations, and misspellings, can increase data quality and model performance (Jianqiang and Xiaolin, 2017).

6.4 Potential Data Processing or Modelling Improvements

Advanced feature extraction methods, such as contextual embeddings from GPT-3 models, can improve input feature quality (Brown et al., 2020). These models record word context more nuancedly than standard methods, resulting in improved performance.
Neural network architectures like LSTM or CNNs (convolutional neural networks), designed for sequence data, can better handle the sequential character of tweets (Hochreiter and Schmidhuber, 2017). These architectures are good at capturing text data dependencies, which helps improve tweet classification.
Finally, real-time data processing would allow the system to identify tweets in real time, making the model more effective in dynamic situations like social media monitoring and sentiment analysis (Gama et al., 2014).

Future research can improve tweet classification systems' performance and applicability by applying these changes and investigating these research avenues.

7. Conclusion

7.1 Summary of Key Findings

This research created an NLP pipeline to classify tweets into six categories with impressive results. With 84% accuracy, the Support Vector Machine (SVM) model outperformed Naive Bayes and Logistic Regression. SVMs perform well because of their robust handling of high-dimensional data and excellent margin optimisation between classes. Lemmatization and contraction handling improved the model's accuracy and performance.

7.2 Overall Project Success Assessment

The project achieved its goals by building an NLP pipeline and testing different machine learning models. The systematic strategy ensured data preparation, feature extraction, and model validation. CRISP-DM provided a structured and iterative framework for continual development and optimal results. Text classification performance depends on proper pre-processing and model selection, as shown by the project.

7.3 Final Thoughts and Implications for the Field of Machine Learning and Data Analytics

This study shows how advanced NLP and machine learning models are essential for processing and classifying social media text data. SVM's effectiveness implies its potential

use in high-dimensional text categorization applications. To improve performance, future research should incorporate deep learning models like BERT or GPT-3. This project's methods and insights advance machine learning and data analytics by demonstrating NLP's real-world applicability. Machine learning may transform the understanding of massive amounts of unstructured data, as shown in this study.

REFERENCES

Agarwal, A., Xie, B., Vovsha, I., Rambow, O. and Passonneau, R.J., 2021, June. Sentiment analysis of twitter data. In *Proceedings of the workshop on language in social media (LSM 2011)* (pp. 30-38).

Asghar, M.Z., Kundi, F.M., Ahmad, S., Khan, A. and Khan, F., 2018. T-SAF: Twitter sentiment analysis framework using a hybrid classification scheme. *Expert Systems*, *35*(1), p.e12233.

Bird, S., Klein, E. and Loper, E., 2019. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Boser, B.E., Guyon, I.M. and Vapnik, V.N., 2022, July. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144-152).

Brody, S. and Diakopoulos, N., 2021, July. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! using word lengthening to detect sentiment in microblogs. In *Proceedings of the 2011 conference on empirical methods in natural language processing* (pp. 562-570).

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. and Agarwal, S., 2020. Language models are few-shot learners. *Advances in neural information processing systems*, *33*, pp.1877-1901.

Burnap, P., Gibson, R., Sloan, L., Southern, R. and Williams, M., 2016. 140 characters to victory?: Using Twitter to predict the UK 2015 General Election. *Electoral Studies*, *41*, pp.230-233.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R., 2020. CRISP-DM 1.0: Step-by-step data mining guide. *SPSS inc*, *9*(13), pp.1-73.

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2022. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, pp.321-357.

Cortes, C. and Vapnik, V., 2015. Support-vector networks. *Machine learning*, *20*, pp.273-297.

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dietterich, T.G., 2020, June. Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1-15). Berlin, Heidelberg: Springer Berlin Heidelberg.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. and Bouchachia, A., 2014. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, *46*(4), pp.1-37.

Hochreiter, S. and Schmidhuber, J., 2017. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

Howard, J. and Ruder, S., 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Jianqiang, Z. and Xiaolin, G., 2017. Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE access*, *5*, pp.2870-2879.

Kagan, V., Stevens, A. and Subrahmanian, V.S., 2015. Using twitter sentiment to forecast the 2013 pakistani election and the 2014 indian election. *IEEE Intelligent Systems*, *30*(1), pp.2-5.

Khan, F.H., Bashir, S. and Qamar, U., 2014. TOM: Twitter opinion mining framework using hybrid classification scheme. *Decision support systems*, *57*, pp.245-257.

Kim, J., Yoo, J., Lim, H., Qiu, H., Kozareva, Z. and Galstyan, A., 2023. Sentiment prediction using collaborative filtering. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 7, No. 1, pp. 685-688).

Kohavi, R., 2015, August. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (Vol. 14, No. 2, pp. 1137-1145).

Liu, B., 2020. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge university press.

Manning, C.D., 2018. *Introduction to information retrieval*. Syngress Publishing,.

Manning, C.D., 2018. *Introduction to information retrieval*. Syngress Publishing,.

Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Miner, G., 2022. *Practical text mining and statistical analysis for non-structured text data applications*. Academic Press.

Pak, A. and Paroubek, P., 2020, May. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc* (Vol. 10, No. 2010, pp. 1320-1326).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2021. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, pp.2825-2830.

Prieto, V.M., Matos, S., Alvarez, M., Cacheda, F. and Oliveira, J.L., 2018. Twitter: a good place to detect health conditions. *PloS one*, *9*(1), p.e86191.

ResearchGate. (2022). TWEETS CLASSIFICATION PROCESS. ResearchGate (online). Available at: https://www.researchgate.net/figure/TWEETS-CLASSIFICATION-PROCESS_fig3_350467292 (Accessed 22 May 202).

Symeonidis, S., Effrosynidis, D. and Arampatzis, A., 2018. A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. *Expert Systems with Applications*, *110*, pp.298-310.

Wei, J. and Zou, K., 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

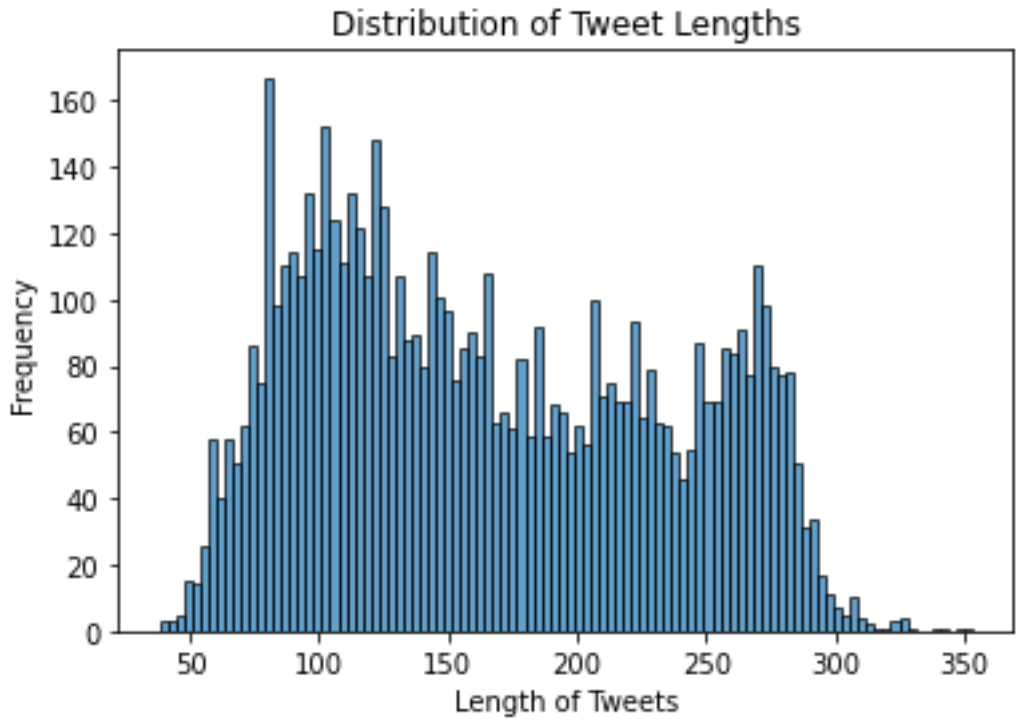
Zhang, L., Wang, S. and Liu, B., 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), p.e1253.

APPENDIX

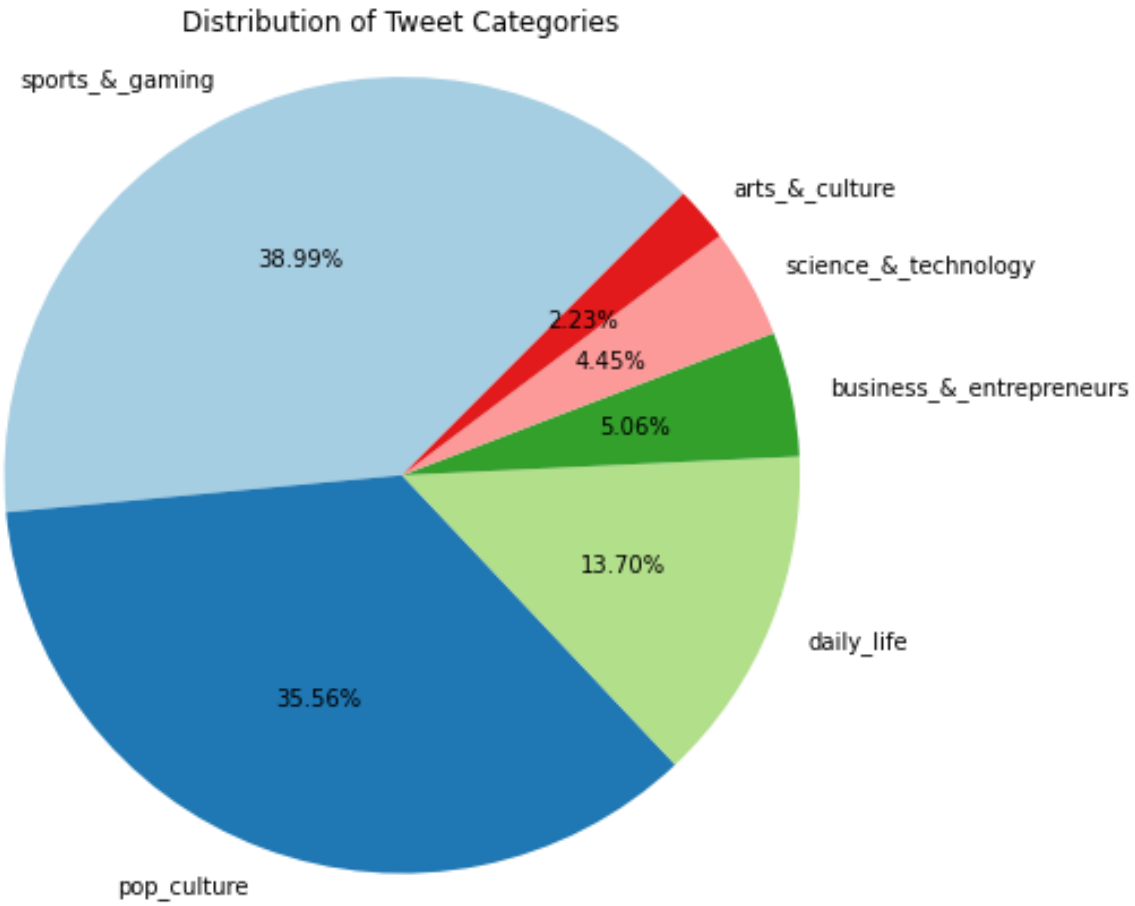
```
import pandas as pd

#to my Load the dataset
file_path = 'desktop/CETM47.json'
tweets_df = pd.read_json(file_path)

#to Display the first few rows of the dataframe
print(tweets_df.head())
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# to download stopwords
nltk.download('stopwords')
nltk.download('punkt')

# the Function to clean text
def clean_text(text):
    text = re.sub(r'@\w+|#\w+|\{{.*?\}}', '', text)
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text.lower()

# Applying cleaning function
tweets_df['cleaned_text'] = tweets_df['text'].apply(clean_text)

# Removing stopwords
stop_words = set(stopwords.words('english'))
```

```
tweets_df['cleaned_text'] = tweets_df['cleaned_text'].apply(lambda x: ' '.join(
    word for word in word_tokenize(x) if word not in stop_words))

print(tweets_df[['text', 'cleaned_text']].head())
from sklearn.feature_extraction.text import TfidfVectorizer

# To Initialize TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=1000)

# Fitting and transforming the cleaned text
X = tfidf_vectorizer.fit_transform(tweets_df['cleaned_text'])

# Checking the scikit-learn version
import sklearn
print(sklearn.__version__)

# Converting to DataFrame for inspection
if hasattr(tfidf_vectorizer, 'get_feature_names_out'):
    feature_names = tfidf_vectorizer.get_feature_names_out()
else:
    feature_names = tfidf_vectorizer.get_feature_names()

X_df = pd.DataFrame(X.toarray(), columns=feature_names)

print(X_df.head())
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

# Splitting the dataset into training and test sets
```

```
y = tweets_df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Logistic Regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
import matplotlib.pyplot as plt



# To Calculate the length of each message
tweets_df['length'] = tweets_df['text'].apply(len)

# To Plot a histogram of the message lengths
tweets_df['length'].plot(bins=100, kind='hist', edgecolor='black', alpha=0.7)
plt.title('Distribution of Tweet Lengths')
plt.xlabel('Length of Tweets')
plt.ylabel('Frequency')
plt.show()
```




```
import matplotlib.pyplot as plt


# Assuming `tweets_df` is your dataframe and 'label_name' contains the class names
labels = tweets_df['label_name'].unique()
sizes = tweets_df['label_name'].value_counts().values


# Create the pie chart
fig = plt.figure(figsize=(7, 7))
plt.pie(sizes, labels=labels, autopct="%1.2f%%", colors=plt.cm.Paired.colors, shadow=False, startangle=45)
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title('Distribution of Tweet Categories')
plt.show()
```




```
import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import classification_report, accuracy_score, confusion_matrix


# Assuming y_test and y_pred are already defined from your model evaluation


# Compute confusion matrix

conf_matrix = confusion_matrix(y_test, y_pred)


# Define the class labels
```

class_labels = ['Arts & Culture', 'Business & Entrepreneurs', 'Pop Culture', 'Daily Life', 'Sports & Gaming', 'Science & Technology']

Plot confusion matrix

plt.figure(figsize=(10, 7))

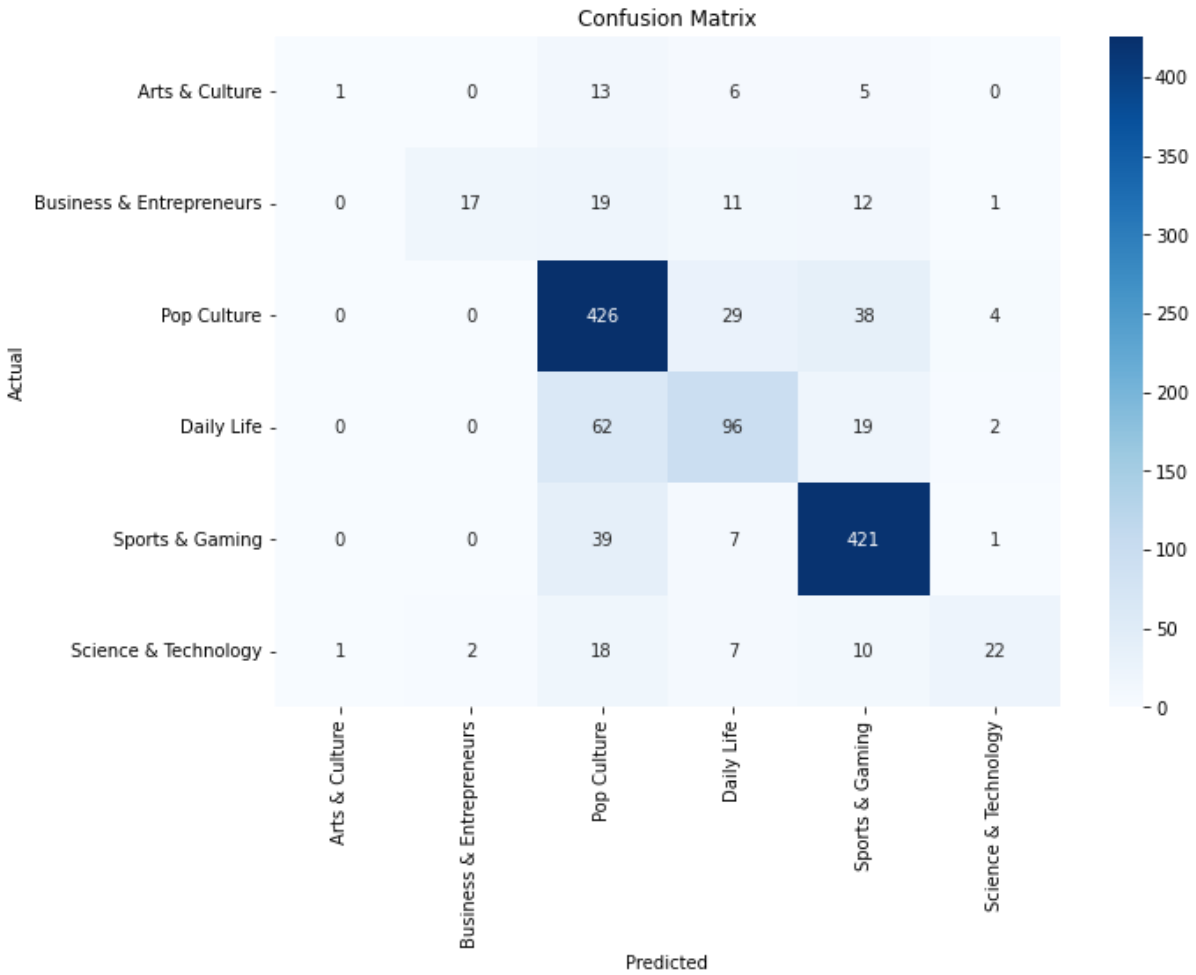
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()



```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import matplotlib.pyplot as plt

# Initialize the models
models = {
    "Naive Bayes": MultinomialNB(),
    "SVM": SVC(kernel='linear'),
    "Logistic Regression": LogisticRegression(max_iter=1000)
}

# Train each model, make predictions, and evaluate performance
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Print evaluation metrics
    print(f"Model: {name}")
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
```

```
    # Compute confusion matrix

    conf_matrix = confusion_matrix(y_test, y_pred)



    # Plot confusion matrix

    plt.figure(figsize=(8, 6))

    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Arts &
Culture', 'Business & Entrepreneurs', 'Pop Culture', 'Daily Life', 'Sports & Gaming', 'Science
& Technology'], yticklabels=['Arts & Culture', 'Business & Entrepreneurs', 'Pop Culture',
'Daily Life', 'Sports & Gaming', 'Science & Technology'])

    plt.xlabel('Predicted')

    plt.ylabel('Actual')

    plt.title(f'Confusion Matrix for {name}')

    plt.show()
```