NLP Classification Task Report

I. Introduction

The main objective of this report is to provide a comprehensive documentation on the process of finding, preprocessing, and analyzing dataset for binary classification task The dataset we deal with contains various documents and each of these documents has a class label, . title , date, and body. The main goal of this project is to develop a robust and reliable classifier. This classifier can be trained to predict the class score for a given set of tests. Predictions are based on carefully extracted features from the training set. The process will include starting with an initial data analysis, followed by preliminary work to clean and prepare the data for analysis, and finally applying machine learning algorithms to develop predictive models.

II. Exploratory Data Analysis (EDA)

A. Dataset Overview

First Step loaded dataset by using Panda Library

```
ath = 'trainset.txt'
column_names = ['CLASS', 'TITLE', 'DATE', 'BODY']
news = pd.read_csv(ath, sep='\t', names=column_names)
news.head()
```

So I loaded the dataset successfully and the next step is to check for any missing values.

CLASS	TITLE	DATE	BODY
1	JAPAN FIRM PLANS TO SELL U.S. FARMLAND TO JAP...	MORIOKA, Japan, March 12 -	A Japanese real estate company said it will ...
-1	NORTH BH SETS ONE-FOR-FIVE OFFER FOR NORGOLD ...	MELBOURNE, March 12 -	North Broken Hill Holdings Ltd &lt;NBHA.ME ...
-1	OUTOKUMPU IN COPPER DEAL WITH IBERICA DEL COBRE	HELSINKI, June 26 -	Finland's state-owned mining company Outokum...
1	ROTTERDAM GRAIN HANDLER SAYS PORT BALANCE ROSE	ROTTERDAM, April 13 -	Graan Elevator Mij, GEM, said its balance in...
1	U.S. SENATE PANEL VOTES TO LIMIT COUNTY LOAN ...	(NO DATE)	(NO TEXT)

B. Handling Missing Values

When I Look into dataset I saw some missing values in ‘BODY’ and ‘DATE’ so i need to check how many missing values in data set

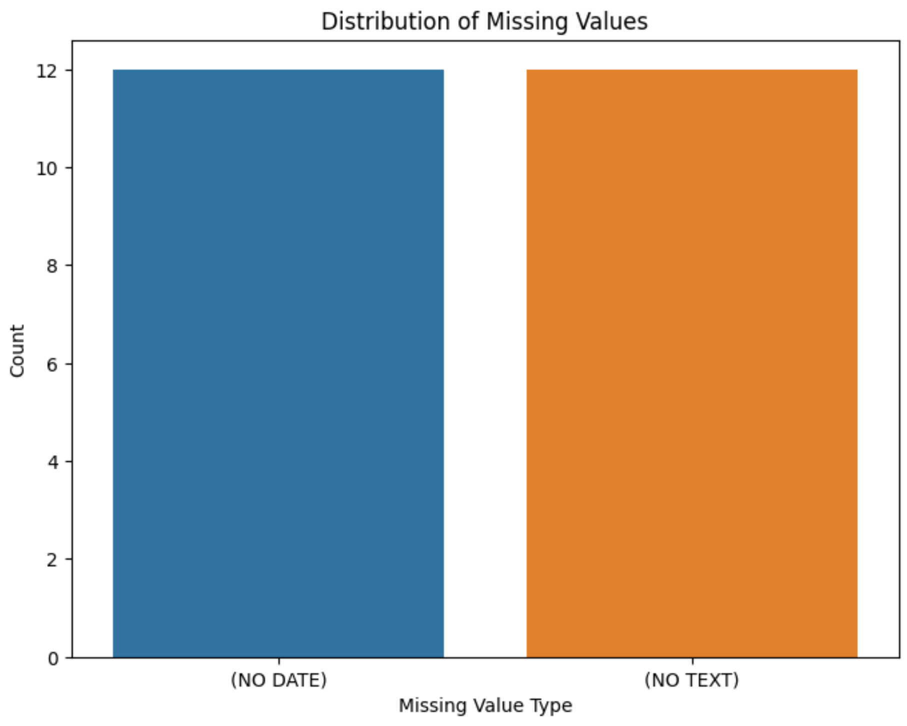
```
no_date_count = (news['DATE'] == '(NO DATE)').sum()
no_text_count = (news['BODY'] == '(NO TEXT)').sum()

print(f'Number of occurrences of "(NO DATE)" in the DATE column: {no_date_count}')
print(f'Number of occurrences of "(NO TEXT)" in the BODY column: {no_text_count}')
```

I viewed that there were 12 instances where the 'DATE' column showed "(NO DATE)" and another 12 where the 'BODY' column had "(NO TEXT)". This totaled to 24 missing values. My next step was to visualize the distribution of these missing values.

C. Visualizations

I extracted the missing values and viewed them in a table view and graph view because I needed to clarify if these values were in the same row.

```
plt.figure(figsize=(8, 6))
sns.barplot(x=['(NO DATE)', '(NO TEXT)'], y=[no_date_count, no_text_count])
plt.title('Distribution of Missing Values')
plt.xlabel('Missing Value Type')
plt.ylabel('Count')
plt.show()

filtered_rows = news[(news['DATE'] == '(NO DATE)') | (news['BODY'] == '(NO TEXT)')]
filtered_table = filtered_rows[['CLASS', 'TITLE', 'DATE', 'BODY']]
filtered_table
```



My guess here was correct because those values are in the same row. However, I had doubts because some class values were not the same. I thought that the prediction might be based on the title. So, I decided to predict with the title as well to clarify my doubts.

III. Data Preprocessing

A. Removal of Irrelevant Rows

I decided to remove rows with missing DATE and BODY.

```
news_clean = news[(news['DATE'] != '(NO DATE)') & (news['BODY'] != '(NO TEXT)')]
news_clean.head()
```

B. Text Preprocessing

In the preprocessing step, I used NLTK libraries to tokenize, extract English vocabulary, and lemmatize with text, resulting in an enhanced representation of document content use in general bae Function preprocess(text) optimizes these operations to improve the quality of text data for subsequent analysis .

```
def preprocess(text):
  tokens = word_tokenize(text)
  stop_words = set(stopwords.words('english'))
```

```
    tokens= [token for token in tokens if token.lower() not in stop_words]
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(token.lower()) for token in tokens]
    return' '.join(tokens)
```

Tokenize : Separate Words for example “ I am NIBM student” to ['I' , ‘am’ ,’NIBM’ ,’Student’]

Stopwords : Remove common words that not effect for training “The car” to “car”

lower() : used lower for avoid duplicates because Uppercase text like Car , car

lemmatizer : Change word in to root form ”Better” to “good”

After that I Apply Preprocess to BODY and ADD it as BODY2 in Table

C. Vectorization and Feature Selection

In this step I used the CountVectorizer from scikit-learn to numerically transform the preprocessed character data (‘BODY2’). The resulting coefficient, X, represents the frequency of each word in the document system, which provides the basis for extracting segments in the subsequent classification task

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()

X=vectorizer.fit_transform(news_clean['BODY2'])
print(X.toarray())
X.shape
```

IV. Model Training and Evaluation

A. Model Selection

1. **Random Forest**
    - random forest is chosen because of its robustness and ability to handle complex data sets. Using a group of decision trees reduces overfitting and provides higher prediction accuracy.
2. **Decision Tree**
    - Using a decision tree for its definition and frequency. It is useful to understand the importance of features, which makes it a valuable first detection algorithm.
3. **Support Vector Machine**
    - SVM was chosen because of its effective ability in dealing with nonlinear relationships and high-dimensional data. Its ability to find the optimal hyperplane makes it suitable for complex classification tasks.
4. **K-Nearest Neighbor**
    - KNN is used for simple and intuitive classification method. It is effective in capturing local patterns and is well suited for situations where similar patterns with similar features are available.

So I need to explore how these models are react on this dataset and find which Classifier Model will give higher Accuracy

B. Model Training

Here I Provide my Libraries and Training Codes

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
X_top = X[:, top_feature_indices]

X_train, X_test, y_train, y_test = train_test_split(X_top, news_clean['CLASS'], test_size=0.2

# Initialize classifiers
decision_tree = DecisionTreeClassifier(random_state=42)
svm_classifier = SVC(random_state=42)
knn_classifier = KNeighborsClassifier()
model = RandomForestClassifier(random_state=42)

#Random Forest
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Random Forest Accuracy : {accuracy}')
print(f'Random Forest Classification Report:\n{classification_report(y_test, y_pred)}')

# Train and evaluate Decision Tree
decision_tree.fit(X_train, y_train)
y_pred_dt = decision_tree.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print(f'Decision Tree Accuracy: {accuracy_dt}')
print(f'Decision Tree Classification Report:\n{classification_report(y_test, y_pred_dt)}')

# Train and evaluate SVM
svm_classifier.fit(X_train, y_train)
y_pred_svm = svm_classifier.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print(f'SVM Accuracy: {accuracy_svm}')
print(f'SVM Classification Report:\n{classification_report(y_test, y_pred_svm)}')

# Train and evaluate KNN
knn_classifier.fit(X_train, y_train)
y_pred_knn = knn_classifier.predict(X_test)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
print(f'KNN Accuracy: {accuracy_knn}')
print(f'KNN Classification Report:\n{classification_report(y_test, y_pred_knn)}')
```

During the model training and evaluation phase, I applied various classifiers including random forest, decision tree, SVM, and KNN using the top selected features and the regular random forest model proved to be very high accuracy, and I got 100% Accuracy (I had doubt here, is this model over fit but random forest model is good foe avoid over fitting) . The detailed distribution report for each model further illustrates their performance metrics I Provide it below

Model	Accuracy	Precision (Class **(-1)**)	Precision (Class **1**)	Recall (Class **(-1)**)	Recall (Class **1**)	F1-Score (Class **(-1)**)	F1-Score **1**)
Random Forest	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Decision Tree	0.82	0.89	0.75	0.76	0.88	0.82	0.81
SVM	0.97	1.00	0.94	0.95	1.00	0.98	0.97
KNN	0.92	1.00	0.85	0.86	1.00	0.92	0.92

C. Feature Importance Analysis

1. **Feature selection**
   - For information gain I Used `mutual_info_classif` from scikit-learn to calculate the mutual information derived from each object in the data set and top 10 factors were identified Here code below.

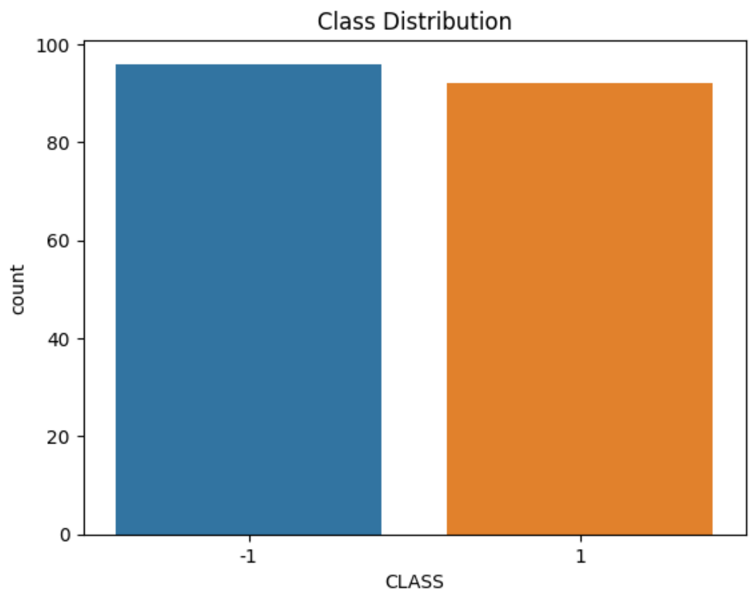
```
from sklearn.feature_selection import mutual_info_classif

info_gain = mutual_info_classif(X,news_clean['CLASS'])
import numpy as np
sor = np.argsort(info_gain)[::-1]
for i in sor[:10]:
  print(f'Features:{[i]}:{info_gain[i]}')
```

```
Features:[2740]:0.15552721522607502
Features:[2040]:0.155527215226075
Features:[1211]:0.15479727101242907
Features:[4336]:0.1400514708825369
Features:[637]:0.12504393368706637
Features:[2260]:0.11719308498561734
Features:[1331]:0.11261902205149575
Features:[2747]:0.1126190220514957
Features:[2064]:0.1057428866553785
Features:[2593]:0.10480470016168186
```

2. **Class distribution diagram**

    - I visualized class distribution This gives us insight into how balanced or unbalanced the class label is, which is important for understanding the quality of the dataset.

```
sns.countplot(x='CLASS', data=news_clean)
plt.title('Class Distribution')
plt.show()
```

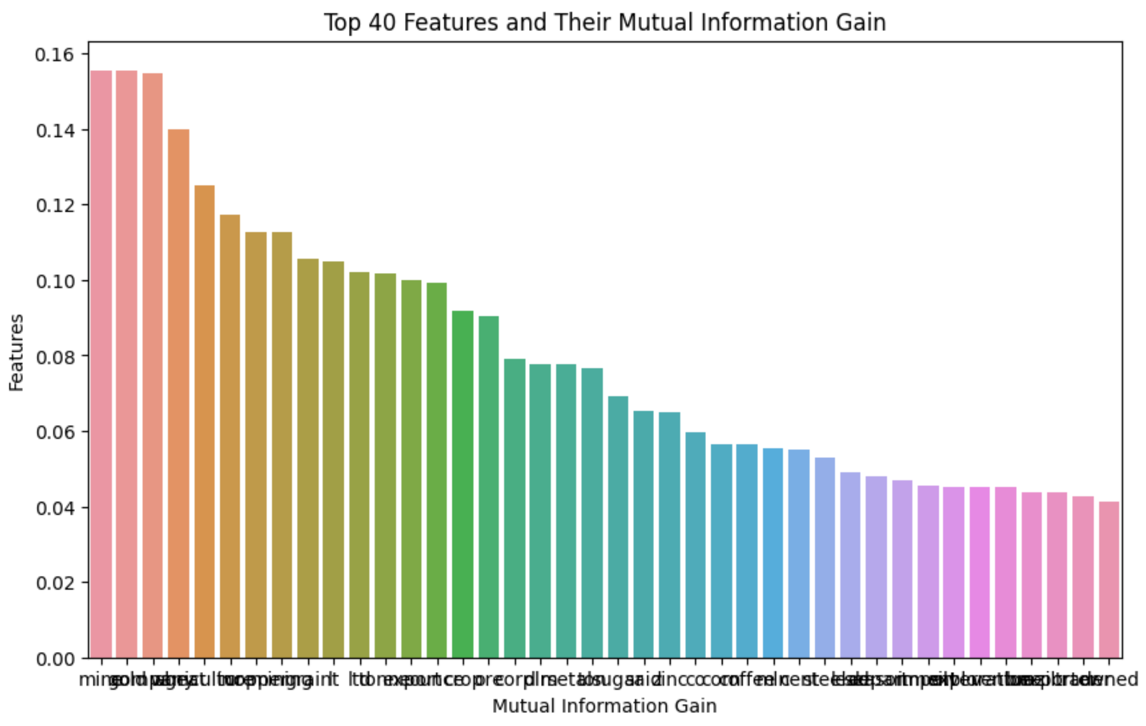


3. **Top Features with Bar Plot**

    - Here I set 40 Features to identified and displayed based on mutual information. In order to show the importance of these factors in the classification task, a bar plot was constructed which clearly shows the importance.

```
top_feature_indices = sor[:40]
top_feature_values = [info_gain[i] for i in top_feature_indices]
top_feature_names = [vectorizer.get_feature_names_out()[i] for i in top_feature_indices]

plt.figure(figsize=(10, 6))
```

```
sns.barplot(y=top_feature_values, x=top_feature_names)
plt.title('Top 40 Features and Their Mutual Information Gain')
plt.xlabel('Mutual Information Gain')
plt.ylabel('Features')
plt.show()
```



V. Testing Set Predictions

A. Preprocessing

In this step i uploaded test and preprocess it like previous same method

```
path1 = 'testing set.txt'
column_names = [ 'TITLE', 'DATE', 'BODY']
newstest = pd.read_csv(path1, sep='\t', names=column_names)
newstest.head()

def preprocess(text):
  tokens = word_tokenize(text)
  stop_words = set(stopwords.words('english'))
  tokens= [token for token in tokens if token.lower() not in stop_words]
  lemmatizer = WordNetLemmatizer()
  tokens = [lemmatizer.lemmatize(token.lower()) for token in tokens]
  return' '.join(tokens)

newstest['BODY2'] = newstest['BODY'].apply(preprocess)
newstest.head()

xtest=vectorizer.transform(newstest['BODY2'])#here don,t use fit transform
print(X.shape)
print(xtest.shape)
```

I Used same vectorizer before used in training set else it will create another metrics

B. Model Prediction (Using Body)

Make predictions on the testing set using the trained Random Forest model with the body.
I Used Random Forest model because it gave higher accuracy and RF model good for Overfitting issue

```
X_top_test = xtest[:, top_feature_indices]

newstest['PREDICTED CLASS'] = model.predict(X_top_test)
newstest.head(2)
```

TITLE	DATE	BODY	BODY2	PREDICTED CLASS
USSR WHEAT BONUS OFFER SAID STILL UNDER DEBATE	WASHINGTON, March 6	The Reagan administration continues to debat...	Reagan administration continues debate whether...	1
HAITIAN CANE PLANTERS PROTEST SUGAR MILL CLOSURE	PORT-AU-PRINCE, April 13	About 2,000 sugar cane planters marched to P...	2,000 sugar cane planter marched port-au-princ...	1

C. Model Prediction (Using Title)

In a separate notebook, I recreated the entire class prediction process based on the 'BODY' field. following the same steps, I used the ‘TITLE’ field to create a class prediction, for consistency of methodology else get error in predictions

```
X_top_test = xtest[:, top_feature_indices]

newstest['PREDICTED CLASS'] = model.predict(X_top_test)
newstest.head(2)
```

TITLE	DATE	BODY	TITLE2	PREDICTED CLASS
USSR WHEAT BONUS OFFER SAID STILL UNDER DEBATE	WASHINGTON, March 6 -	The Reagan administration continues to debate...	ussr wheat bonus offer said still debate	1
HAITIAN CANE PLANTERS PROTEST SUGAR MILL CLOSURE	PORT-AU-PRINCE, April 13 -	About 2,000 sugar cane planters marched to P...	haitian cane planter protest sugar mill closure	1

VI. Comparing Predictions with Actual Labels

Common Dates Analysis

I guess test dataset also in trained dataset for that i get analyze it using DATE as Common For check Test Accuracy

```
common_dates = []

for date_train in news['DATE']:
    if date_train in newstest['DATE'].values:
        common_dates.append(date_train)

print("Common Dates:")
common_dates
```

VII. Accuracy Calculation

A. Result Table

In the result_table DataFrame I cross-checked the predictions on the test set with the corresponding class labels from the training set based on match dates. Each entry in the result_table contains a date, an actual class label ('CLASS'), and a predicted class label ('PREDICTED CLASS') for comparison.

```
result_table = pd.DataFrame(columns=['DATE', 'CLASS', 'PREDICTED CLASS'])


for index, row_test in newstest.iterrows():
    matching_rows = news[news['DATE'] == row_test['DATE']]

    for _, row_train in matching_rows.iterrows():
        result_table = result_table.append({
            'DATE': row_test['DATE'],
            'CLASS': row_train['CLASS'],
            'PREDICTED CLASS': row_test['PREDICTED CLASS']
        }, ignore_index=True)


result_table.head(2)
```

DATE	CLASS	PREDICTED CLASS
WASHINGTON, March 6 -	1	1
WASHINGTON, March 6 -	1	1

B. Accuracy Calculation

Calculate accuracy using the result table.

```
from sklearn.metrics import accuracy_score

result_table['CLASS'] = pd.to_numeric(result_table['CLASS'])
result_table['PREDICTED CLASS'] = pd.to_numeric(result_table['PREDICTED CLASS'])

accuracy_test = accuracy_score(result_table['CLASS'], result_table['PREDICTED CLASS'])

print(f'Accuracy: {accuracy_test}%')on
```

Accuracy BODY: 0.8514851485148515%
Accuracy TITLE: 0.7722772277227723%
So i got 85% Accuracy in use BODY to predict CLASS and 77% Accuracy in used TITLE to predict CLASS   for testing

VIII. Conclusion

A. Summary

In the Summary, the models I created did pretty well in predicting the document classes. When I used the main text (BODY), I got a good 85.15% accuracy. But when I used just the title (TITLE), the accuracy dropped a bit to 77.23%. This tells us that what's written in the main part of the document is more helpful for the computer to figure out the class. so here the body gives a very accurate response to predict its class

B. Prediction Comparison

First I thought when i saw there NO TEXT , NO DATE values Sometime tittle will give highest accuracy but when I work on title i realized that BODY predict class better than TITLE so when I comparing how well the models did, it is clear that using the title alone didn't work as well as using the whole document's text. The main part of the document has more important information that the computer can learn from i mean computer understand BODY feature well .So It's like the body of the document gives the computer more clues to do a better job in figuring out the class.

C. Future Improvements

Looking ahead, there are ways to make the models even better. We could try combining information from both the title and the body. Also, using more advanced language(for example Neural Network) tricks could help the computer understand things even better. If we take a closer look at the documents that the computer got wrong, we might learn more and make the models even smarter. So, there will be ways to keep making things better, like getting a higher accuracy on a test set.

IX. References

https://www.researchgate.net/publication/349576515_Feature_selection_methods_for_text_classification_a_systematic_literatur

https://medium.com/@abdallahashraf90x/feature-engineering-in-nlp-a784d683bfce

https://spotintelligence.com/2023/10/09/binary-classification/#:~:text=Binary classification is frequently used,public opinion and customer feedback.

https://inside-machinelearning.com/en/a-simple-and-efficient-model-for-binary-classification-in-nlp/