جامعة أم القرى
UMM AL-QURA UNIVERSITY

REPORT
DATA ANALYSIS (2)
DS3114

Dr. Omaima Fallatah

Team Members:

- Faten Matouq Almowallad 444000266
- Ayah Bakur Alhawsawi 444006678

https://colab.research.google.com/drive/1u2youtnLZ-UMJued2OKzQtXsc8guBWd3?usp=sharing#scrollTo=c1sAP7hJZ_b4

Tasks: 3

Text Analysis (Extra Model)

(Fake News)

INTRODUCTION

The objective of this project is to develop a text classification model that categorizes news articles as real or fake. Using a dataset of labeled news articles, I implemented various machine learning algorithms to classify text, optimizing performance through data preprocessing and model tuning. This analysis was conducted in Google Colab.

ABOUT DATASET

The dataset comprises two primary files: train.csv with 25,000 rows of labeled news articles and test.csv, which contains the news titles we aim to classify. The labels in the training data represent categories for news such as "fake" or "real," and each news title serves as the feature for classification.

ANALYSIS & RESULTS

In this project, I implemented several machine learning models for text classification using TF-IDF features extracted from the news article titles. After experimenting with models such as **Logistic Regression**, **Multinomial Naive Bayes**, and **Random Forest**, I used **GridSearchCV** to fine-tune the parameters of the Logistic Regression model.

Key Findings:

- **Multinomial Naive Bayes** performed well but slightly underperformed compared to Logistic Regression.
- **Logistic Regression** showed strong performance after optimization using GridSearchCV.
- **Random Forest** provided a solid alternative with competitive accuracy.

Libraries Used:

1. **Pandas**: For data manipulation and analysis.

   ```
   import pandas as pd
   ```

2. **Matplotlib**: For creating static visualizations.

   ```
   import matplotlib.pyplot as plt
   ```

3. **Seaborn**: For more advanced visualizations.

   ```
   import seaborn as sns
   ```

4. **Scikit-learn**: Used for preprocessing, model building, and performance evaluation.
   - `TfidfVectorizer`: For converting text into numerical features.

     ```
     from sklearn.feature_extraction.text import TfidfVectorizer
     ```

   - `LogisticRegression`: Implementing logistic regression classifier.

     ```
     from sklearn.linear_model import LogisticRegression
     ```

   - `MultinomialNB`: For implementing the Naive Bayes classifier.

     ```
     from sklearn.naive_bayes import MultinomialNB
     ```

   - `RandomForestClassifier`: To implement the Random Forest classifier.

     ```
     from sklearn.ensemble import RandomForestClassifier
     ```

   - `GridSearchCV`: To optimize model hyperparameters.

     ```
     from sklearn.model_selection import GridSearchCV
     ```

Data Preprocessing

In this text classification project, data preprocessing was essential to transform the raw text into a format suitable for machine learning models. Here's the step-by-step process:

```
# Quick look at the data
print(train_data.head())
print(test_data.head())
print(submit_data.head())

# Merging data or working with the relevant columns
# Assuming there is a "news" column for the news and a "label" column for the classification
X_train = train_data['title']

y_train = train_data['label']

X_test = test_data['title']
```

```
   id                                              title              author  \
```

1. **Handling Missing Values**: Missing titles in both training and test datasets were replaced with empty strings to prevent errors during text vectorization.

   ```
   X_train = X_train.fillna('')
   X_test = X_test.fillna('')
   ```

2. **TF-IDF Vectorization**: Text data (article titles) were converted into numerical features using **TfidfVectorizer**. This method calculates the importance of a word relative to the document and across the corpus, transforming the text into vectors for model training and predictions.

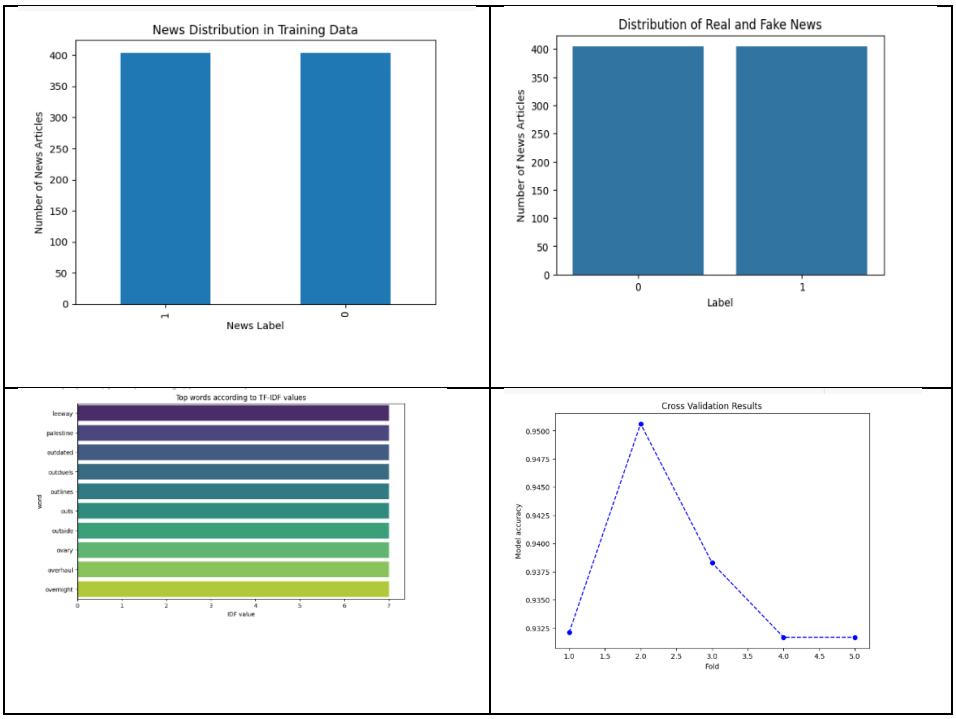
   ```
   vectorizer = TfidfVectorizer(max_features=5000)
   X_train_tfidf = vectorizer.fit_transform(X_train)
   X_test_tfidf = vectorizer.transform(X_test)
   ```

3. **Label Encoding**: The target labels were converted into numeric format for the model using **pd.to_numeric**.

   ```
   y_train = pd.to_numeric(y_train, errors='coerce').fillna(-1).astype(int)
   ```

Visual Analysis

- **Label Distribution**: The count of real vs. fake news was visualized using bar plots to ensure a balanced or imbalanced dataset.
- **Top TF-IDF Words**: I visualized the top 10 most important words by their TF-IDF values to understand which words contributed most to the classification.
- **Cross-validation**: Cross-validation accuracy for each fold was plotted to assess model consistency.



CONCLUSION

The text classification project successfully demonstrated the application of machine learning techniques to categorize news articles. Logistic Regression and Naive Bayes models performed well, and the Random Forest model provided a robust alternative. This analysis highlighted the importance of preprocessing and feature extraction in text classification tasks.