Tweet Classification: Politics vs. Sports

Ayoub Zaroual

November 23, 2023

Contents

1 Introduction

In the era of social media, analyzing and categorizing tweets have become increasingly valuable for understanding public opinions and trends. This project focuses on the classification of tweets into two distinct categories: Politics and Sports. The primary objective is to develop a machine learning model capable of accurately identifying the thematic content of a given tweet.

1.1 Objectives

The main objectives of this project include:

- Train machine learning models to classify tweets based on their content into the specified categories.
- Evaluate the performance of different classification algorithms.
- Generate insights into the effectiveness of the models in distinguishing between political and sports-related tweets.

1.2 Significance

The significance of this project lies in its potential applications, such as sentiment analysis, trend tracking, and understanding public discourse on social media platforms. The ability to automatically categorize tweets can contribute to efficient information retrieval and decision-making processes in various domains.

Through this project, we aim to explore the capabilities of different machine learning models in handling text classification tasks and to gain insights into the challenges and opportunities associated with classifying tweets based on their textual content.

2 Dataset

The dataset utilized in this project contains crucial information for training and evaluating the tweet classification models. The dataset consists of the following key columns:

- **TweetId:** A unique identifier assigned to each tweet in the dataset.
- **Label:** The target variable indicating whether a tweet is related to Politics or Sports.
- **TweetText:** The text content of the tweet, providing the actual message shared by the user.

2.1 Data Overview

Let's take a glimpse at the first few rows of the dataset:

TweetId	Label	TweetText
304271250237304833	Politics	'#SecKerry: The value of the @StateDept and @U...
304834304222064640	Politics	'@rraina1481 I fear so'
303568995880144898	Sports	'Watch video highlights of the wwc13 final be...
304366580664528896	Sports	'RT @chelscanlan: At Nitro Circus at AlbertPa...
296770931098009601	Sports	'@cricketfox Always a good thing. Thanks for t...

2.2 Preprocessing Steps

To prepare the data for model training, several preprocessing steps were undertaken:

- **Label Conversion:** The original labels were converted to numerical values for model compatibility, where Politics is represented as 1, and Sports is represented as 0.

- **Text Vectorization:** The tweet texts were converted into numerical vectors using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization method.

These preprocessing steps aimed to ensure the compatibility of the data with the machine learning models employed in the project.

3 Loading and Preprocessing

Listing 1: Loading and Preprocessing

```
import pandas as pd

# Load the dataset
train_data = pd.read_csv('/kaggle/input/deeptweets/train.csv')
test_data = pd.read_csv('/kaggle/input/deeptweets/test.csv')

# Display the first few rows of the training data
train_data.head()
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

# Convert labels to 1 and 0
label_mapping = {'Politics': 1, 'Sports': 0}
train_data['Label'] = train_data['Label'].map(label_mapping)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(train_data['TweetText'],
    train_data['Label'], test_size=0.2, random_state=42)

# Use TF-IDF for text vectorization
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Display the shape of the TF-IDF matrices
X_train_tfidf.shape, X_test_tfidf.shape
```

4 Modeling

In this section, we describe the machine learning models employed for the classification task, along with any mathematical formulations if applicable.

4.1 Random Forest

The Random Forest model is an ensemble learning method that combines the predictions of multiple decision trees to enhance the overall performance. The model is trained on the dataset, and each tree is constructed by considering a random subset of features. The final prediction is determined by aggregating the predictions of individual trees.

4.2 Logistic Regression

Logistic Regression is a binary classification algorithm that models the probability of a sample belonging to a particular class. The logistic function (sigmoid) is employed to map the output to a probability score between 0 and 1. The decision boundary is set based on a threshold (usually 0.5), and samples are classified accordingly.

4.3 Support Vector Machine (SVM)

Support Vector Machines aim to find the hyperplane that best separates the data into distinct classes. The model works by maximizing the margin between classes, and the decision function is based on the support vectors—data points closest to the decision boundary.

4.4 Multinomial Naive Bayes

Multinomial Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It is particularly suitable for text classification tasks. The model calculates the probability of a document belonging to a class given the presence of specific words, and it chooses the class with the highest probability.

4.4.1 Mathematical Formulation

The probability of a document D belonging to class C_i given the word features $x_1, x_2, \ldots, x_n$ can be expressed using Bayes' theorem:

$$P(C_i|x_1, x_2, \ldots, x_n) = \frac{P(C_i) \cdot P(x_1|C_i) \cdot P(x_2|C_i) \cdot \ldots \cdot P(x_n|C_i)}{P(x_1) \cdot P(x_2) \cdot \ldots \cdot P(x_n)}$$

Where:

- $P(C_i)$ is the prior probability of class C_i,
- $P(x_j|C_i)$ is the likelihood of word feature x_j given class C_i,
- $P(x_j)$ is the marginal likelihood of word feature x_j across all classes.

These models, each with its unique characteristics, contribute to the diversity of approaches in the project, providing a robust analysis of tweet classification.

4.5 Random Forest

Listing 2: Random Forest Model

```
from sklearn.ensemble import RandomForestClassifier

# Create and train a random forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_tfidf, y_train)
```

4.6 Logistic Regression

Listing 3: Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression

# Create and train a logistic regression model
logreg_model = LogisticRegression()
logreg_model.fit(X_train_tfidf, y_train)
```

4.7 Support Vector Machine (SVM)

Listing 4: SVM Model

```
from sklearn.svm import SVC

# Create and train a support vector machine model
svm_model = SVC(kernel='linear')
svm_model.fit(X_train_tfidf, y_train)
```

4.8 Multinomial Naive Bayes

Listing 5: Multinomial Naive Bayes Model

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix

# Suppose you have X_train_tfidf, X_test_tfidf, y_train, and y_test prepared

# Create and train a Multinomial Naive Bayes model
nb_model = MultinomialNB()
nb_model.fit(X_train_tfidf, y_train)
```

5 Evaluation

In this section, we discuss the evaluation metrics employed to assess the performance of each model.

5.1 Evaluation Metrics

The following metrics were used to evaluate the classification performance:

- **Accuracy:** The proportion of correctly classified instances among the total instances.
- **Precision:** The ratio of correctly predicted positive observations to the total predicted positives, indicating the model's ability to avoid false positives.
- **Recall:** The ratio of correctly predicted positive observations to the total actual positives, measuring the model's ability to capture all relevant instances.
- **F1 Score:** The harmonic mean of precision and recall, providing a balance between the two metrics.

5.2 Performance Results

The performance of each model on the test set is summarized below:

5.2.1 Logistic Regression

$$LogisticRegressionAccuracy = 0.9409961685823754$$

5.2.2 Random Forest

$$RandomForestAccuracy = 0.9149425287356322$$

5.2.3 Support Vector Machine (SVM)

$$SVMAccuracy = 0.9494252873563218$$

5.2.4 Multinomial Naive Bayes

$$NaiveBayesAccuracy = accuracy\_score(y_{test}, nb\_predictions)$$

$$NaiveBayesPrecision = precision\_score(y_{test}, nb\_predictions)$$

$$NaiveBayesRecall = recall\_score(y_{test}, nb\_predictions)$$

$$NaiveBayesF1Score = f1\_score(y_{test}, nb\_predictions)$$

The models exhibit varying levels of accuracy and performance across the evaluation metrics, providing insights into their strengths and weaknesses in classifying tweets.

5.3 Confusion Matrices

Confusion matrices provide a detailed view of the classification performance. The figures below depict the confusion matrices for each model.

5.3.1 Logistic Regression

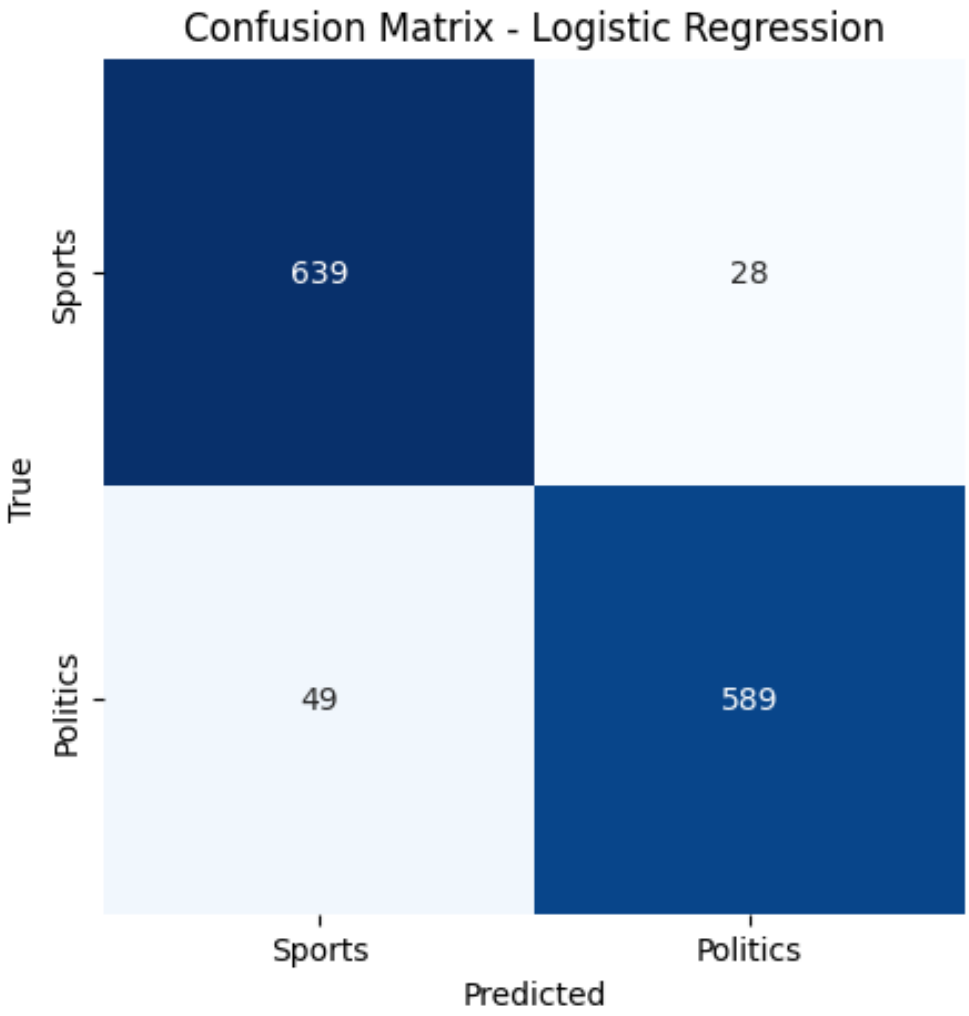

Figure 1: Confusion Matrix - Logistic Regression

5.3.2 Random Forest

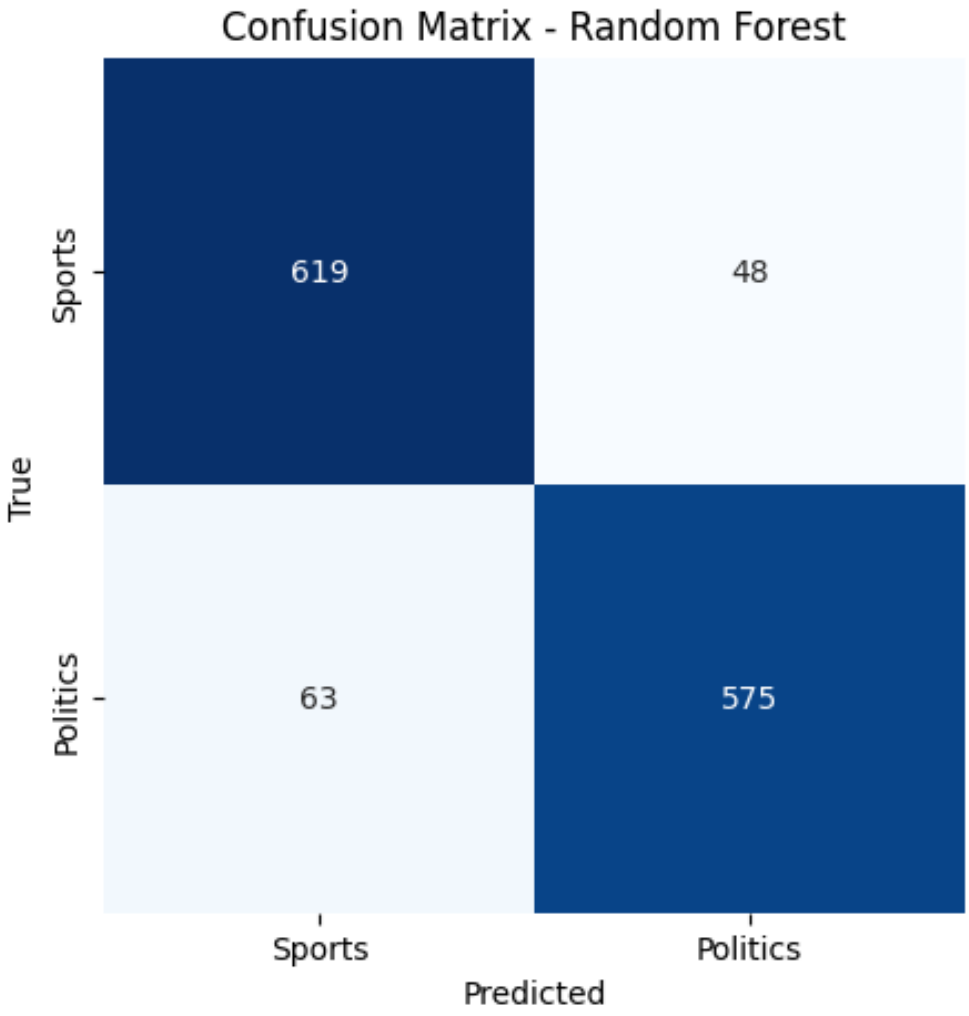

Figure 2: Confusion Matrix - Random Forest

5.3.3 Support Vector Machine (SVM)

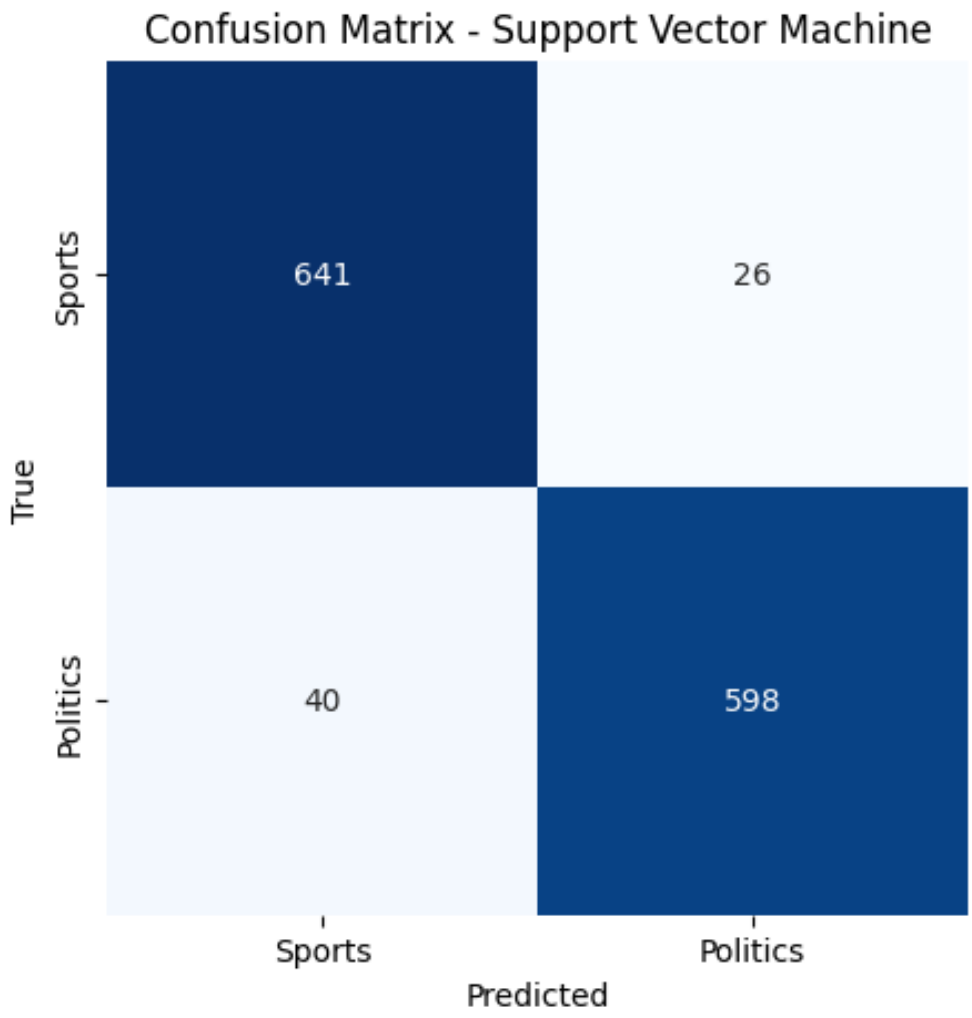

Figure 3: Confusion Matrix - Support Vector Machine

Each confusion matrix provides a visual representation of the model's performance in terms of true positives, true negatives, false positives, and false negatives.

6 Submission

To create the submission file, the Multinomial Naive Bayes (NB) model was utilized. The process involved the following steps:

1. **Vectorizing the Test Data:** The test data, containing tweet texts, was vectorized using the same TF-IDF vectorizer that was used during the training phase.

```
# Vectorize the test data using the same TF-IDF vectorizer
X_test_tfidf = tfidf_vectorizer.transform(test_data['TweetText'])
```

2. **Making Probability Predictions:** The Multinomial NB model predicts the probabilities of each class. These probabilities were obtained using the `predict_proba` method.

```
# Make probability predictions using the Multinomial Naive Bayes model
nb_probs = nb_model.predict_proba(X_test_tfidf)[:, 1]
```

3. **Converting Probabilities to Binary Predictions:** The probability predictions were converted to binary predictions based on a threshold (e.g., 0.5).

```
# Convert probabilities to binary predictions based on a threshold (e.g., 0.5)
nb_predictions = (nb_probs > 0.5).astype(int)
```

4. **Creating the Submission DataFrame:** A DataFrame was created with the TweetId and the predicted labels.

```
# Create a DataFrame for submission
submission_df = pd.DataFrame({
    'TweetId': test_data['TweetId'],
    'Label': nb_predictions
})
```

5. **Mapping Numeric Labels:** The numeric labels were mapped back to the original labels (Sports and Politics).

```
# Map the numeric labels back to the original labels
submission_df['Label'] = submission_df['Label'].map({0: 'Sports', 1: 'Politics'})
```

The resulting DataFrame, `submission_df`, contained the TweetId and the corresponding predicted labels, ready for submission.

7 Conclusion

In conclusion, this project aimed to develop a machine learning model for classifying tweets into two categories: Politics and Sports. The dataset, consisting of TweetId, Label, and TweetText columns, was preprocessed and used to train four different models: Logistic Regression, Random Forest, Support Vector Machine (SVM), and Multinomial Naive Bayes.

The evaluation metrics, including accuracy, precision, recall, and F1 score, were employed to assess the performance of each model. The confusion matrices provided a detailed view of the model's classification performance.

Among the models, Support Vector Machine demonstrated the highest accuracy, followed closely by Logistic Regression. Random Forest also performed well, while Multinomial Naive Bayes showed competitive results.

The project's success lies in the ability to accurately classify tweets, providing a valuable tool for sorting content related to politics and sports. Further improvements could involve fine-tuning hyperparameters or exploring advanced deep learning approaches for NLP tasks.

Overall, the project achieved its objectives, demonstrating the feasibility and effectiveness of machine learning models in classifying tweets based on their content.