Fake Social Media Text Classification using NLP

Sanskriti Shevgaonkar

January 2024

1 Introduction

In this report, we endeavor to tackle the pressing issue of classifying posts within the MediaEval 2015 ”verifying multimedia use” dataset, aiming to discern between genuine and deceptive content. The task involves designing three distinct pipelines using Machine Learning Algorithms(KNN, Random Forest, SVM). The data pipeline incorporates essential pre-processing steps, feature selection methods, and dimensionality reduction techniques to optimize the models' efficiency. We train each algorithm and evaluate its performance in distinguishing real from fake posts using various metrics like accuracy and F1-score. This project endeavours to contribute to the field of multimedia content verification by implementing robust Machine Learning solutions tailored to the complexities of the provided dataset. help library,

2 Data Exploration

2.1 Dataset

The dataset is a text file of The MediaEval 2015 ”verifying multimedia use” dataset consisting of social media posts (e.g., Twitter, Facebook and blog posts) for which the social media identifiers are shared along with the post text and some additional characteristics of the post. However, the dataset only consists of text and metadata features. The training dataset has 14277 rows and 7 columns. whereas, the testing dataset has 3755 rows and 7 columns. There are 6742-real, 4921-humor, and 2614-fake tweets labelled in the training dataset. There are no missing values in the training dataset. The task is to train an NLP model to classify the tweet text into real or fake.

2.2 Detecting the Languages

The tweetText column seems to have text in different languages. *langid* library is used for the detection of all the language codes and then mapped with the language names using a dictionary. There are 79 languages in the training dataset, whereas 48 languages are in the test dataset. The language names are stored in a different column and copied to a text file for future use i.e. Removing stopwords.

3 Pre-processing

This process involves cleaning the data and making it ready for feature extraction. For this step, I have used both train and test datasets. I have also made sure to label all the humor labels are fake as the task is for binary classification.

3.1 Data Cleaning

The tweetText column has a lot of redundant data such as links, emojis, hashtags, and special characters. *isalnum()* method is used to check for the alphanumeric characters and numbers in the dataset, also all the links starting with HTTP have been removed.

3.2 Tokenization

Tokenization is the process of breaking down a text into smaller units called tokens. The data is now tokenized into a list of words also called tokens.

3.3 Removing Stop words

Removing stop words is a common preprocessing step in natural language processing (NLP) to eliminate common words that often do not contribute much to the meaning of a sentence. Stop words are words like ”the,” ”and,” ”is,” ”of,” etc. that occur frequently in a language but typically don’t carry significant meaning on their own. These stopwords are removed using *nltk.corpus* library. The function *get_stopwords()* takes the language text file and downloads the corpus for all the supported languages which helps to recognize the stopwords from a particular language. These stop words are then removed using *stop_words()*

3.4 Stemming the Data

Stemming is a text normalization technique used in natural language processing (NLP) and information retrieval to reduce words to their base or root form. The goal of stemming is to map words with similar meanings to a common root. It helps in capturing the essential meaning of words and reduces the dimensionality of the feature space. Here, *SnowballStemmer* is used which is provided by *nltk* library.

3.5 Train-Test Split

Now, we have arrived at the last step of data pre-processing which is splitting data in training and testing datasets. Here we define x and y train and test set.

4 Hyperparameter Tuning

Hyperparameter tuning is the process of finding the optimal set of hyperparameters for a machine learning model to achieve better performance. Hyperparameters are external configuration settings that are not learned from the data but are set before the training process. They influence the behaviour of the learning algorithm and can significantly

impact the model's performance. Here, we have defined a machine learning pipeline for text classification using *scikit-learn*. The pipeline includes two major components TF-IDF vectorizer and a classifier for the algorithms (KNN, Random Forest, SVM) we are using.

I have also created a dictionary 'parameters' that defines the hyperparameter grid for performing a grid search. In this case, it specifies different values for the max-features parameter in the TFIDF vectorizer. I have also defined a scorer for model evaluation. It uses the F1 score with micro averaging and sets the positive label to "fake." This scorer is likely to be used in the grid search for hyperparameter tuning. Now, to find the best parameters for both the feature extraction and the classifier I have used Grid Search. For e.g. In SVM, the grid search process explored two combinations of hyperparameters and one with max-features=10000 resulted in the best performance according to the specified evaluation metric. The accuracy of the micro-averaged F1 score for the predictions made by the model on the test set is 0.79. Hence, the grid search is a systematic way of finding the optimal hyperparameters for our machine learning model.

5 Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is a technique that prepares the text data for a machine learning model by converting it into a numerical representation. The resulting TF-IDF matrices will be used as input features for our machine learning model for text classification.

6 Machine Learning Algorithms

The basic idea of the task is to classify the labels as real and fake. While selecting these algorithms I have taken into consideration the algorithms which are easy to implement, training time complexity is less, can provide satisfactory results, and are best for classification tasks on large datasets.

6.1 Support Vector Machine (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems[1]. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Why SVM?

- SVMs are a good choice for certain tasks, especially binary classification.

- SVMs perform well in high-dimensional spaces, making them suitable for text classification problems where the feature space can be large (e.g., TF-IDF vectors for text).
- SVMs are less prone to overfitting for high-dimensional spaces. It consists of a regularization parameter (C) which helps u manage the decision boundary.
- The hyperplane tries to maximize the margin between the classes which leads to better generalization performance on unseen data

6.2 Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique[3]. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model.

As the name suggests," Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Why Random Forest?

- Random Forst is a Bagging (Bootstrap Aggregation) Algorithm, which leads to robust and accurate models in terms of classification tasks.
- It helps to capture the non-linear data trends, which are beneficial when the decision boundaries are complex.
- Random Forest works well on problems regarding imbalanced classes providing more balanced predictions
- It uses parallelism, to process large data spaces.

6.3 K-Nearest Neighbour

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique[2]. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner

algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Why KNN?

- KNN doesn't build a model during the training phase, rather it memorizes the training instances and makes predictions during the testing phase. This is useful when the data patterns are complex and not easily captured by a predefined model.
- KNN can handle non-linear decision boundaries and remain unaffected by the shape.
- It is simple and doesn't take much time for training. Hence, can be used on large datasets requiring quick solutions.

7 Evaluation

Here is a brief evaluation of all the three models with their strengths and weakness and other evaluation metrices recorded during implementation.

Criteria	KNN	Random Forest	SVM
Strengths	• Simple and intuitive • Lazy Learning, Minimal learning time • Can capture non-linear decision boundaries	• Handles complex relationships well • Uses parallel processing • Ensemble of decision trees	• Perform well in high-dimensional spaces • Better generalization performance on unseen data • Less prone to overfitting
Weakness	• High computational cost during prediction • Sensitive to noise and irrelevant features • Sensitive to the choice of k	• Computational cost during training • consume a significant amount of memory • Sensitive to hyperparameter tuning	• More time for training • Sensitive to outliers and noise • Sensitive to kernel and C parameter
Evaluation Metrics	F1 score: 0.56 Accuracy(%): 66.07 Time(sec): training-0.017, pred-1.864	F1 score: 0.73 Accuracy(%): 70.9 Time(sec): training-81.29, pred-1.44	F1 score: 0.24 Accuracy(%): 81.75 Time(sec): training-7.25, pred-1.24

Ranking (Lower is better)			
Accuracy	Moderate	High	High
Training Time	Low	Moderate	Moderate
Prediction Time	High	Moderate	Moderate
Interpretability	High	Moderate	Low (if data non-linear)
Robustness to Noise	Moderate	High	Moderate

Conclusion

From the evaluation metrices we can conclude that all the three algorithms (KNN, Random Forest and SVM) have their own advantages and disadvantages. Of all the predictions, we can observe that the SVM model has performed the best in terms of accuracy and robust nature. But it had faced a major setback when it comes to interpretability. Random forest is heavy on resources as it uses parallel processing for generation of nodes. KNN is very Simple to implement, but has high sensitivity to the choice of k.

Thus, the three algorithms have been justified, implemented, critically reviewed and compared for the given MediaEval 2015 "verifying multimedia use" dataset for classification of text into real or fake.

References

[1] J. Shaikh and R. Patil, "Fake News Detection using Machine Learning," 2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC), Gunupur Odisha, India, 2020, pp. 1-5, doi: 10.1109/iSSSC50941.2020.9358890.

[2] A. Kesarwani, S. S. Chauhan and A. R. Nair, "Fake News Detection on Social Media using K-Nearest Neighbor Classifier," 2020 International Conference on Advances in Computing and Communication Engineering (ICACCE), Las Vegas, NV, USA, 2020, pp. 1-4, doi: 10.1109/ICACCE49060.2020.9154997.

[3] Z Khanam et al 2021, “Fake News Detection Using Machine Learning Approaches” IOP Conf. Ser.: Mater. Sci. Eng. 1099 012040, doi: 10.1088/1757-899X/1099/1/012040