19Z610 – Machine Learning Laboratory

Real and Fake News Classification

Team,

21Z213 – C O Avina

21Z241 – Rakesh Kumar S

21Z250 – Santhosh A

21Z258 – Soorya Subramani

21Z260 – Subhasri Shreya S L

21Z264 – V Harsheni

Problem Statement

The project's objective is to distinguish between genuine and false information in dynamic news sourced from web and social media platforms. The model is trained on a pre-labeled dataset, where each news item is classified as either real or fake. Natural Language processing is applied to preprocess the dataset, focusing on extracting relevant keywords. To enhance adaptability, the model integrates Online Learning, allowing it to dynamically retrieve news for real-time classification. The project outputs include the binary classification of dynamic news and the corresponding prediction accuracy.

Dataset Description

The dataset[1] consists of 38,654 rows and 2 columns. The first column contains news articles, while the second column indicates the label associated with each article, specifying whether it is categorized as real or fake news. This dataset is curated and structured for the specific purpose of training classification models, particularly in the domain of fake news detection. With a substantial number of labeled instances, it offers ample data for supervised learning approaches, enabling machine learning algorithms to learn patterns and features that distinguish between genuine and fabricated news articles. Researchers and practitioners can leverage this dataset to develop, test, and refine classification algorithms aimed at identifying and mitigating the spread of misinformation in various media contexts.

Models used

Logistic Regression

Logistic regression[2][3] is a statistical method used for binary classification tasks, where the goal is to predict a categorical outcome with two possible classes. It's a supervised learning algorithm that models the relationship between one or more independent variables (features) and a binary dependent variable (target) by estimating the probability that a given input belongs to a particular class. Despite its name, logistic regression is a classification algorithm rather than a regression algorithm.

The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Where:

- σ(z) represents the output (probability) of the sigmoid function.

- z is the input to the function, which is the linear combination of the features and their corresponding weights in logistic regression.

- e is the base of the natural logarithm (Euler's number, approximately equal to 2.71828).

The sigmoid function has an S-shaped curve, with values close to 0 for very negative inputs and values close to 1 for very positive inputs. This property makes it suitable for converting the output of linear regression models into probabilities.

Passive Aggressive Classifier

The Passive Aggressive Classifier[4] (PAC) is a type of online learning algorithm used for binary classification tasks. It belongs to the family of algorithms known as "lazy learning" or "memory-based learning," meaning it does not explicitly build a model during training but rather updates its parameters incrementally, making it particularly useful for processing large streams of data or situations where memory resources are limited.

The Passive Aggressive Classifier operates by updating its model parameters in a "passive" manner when the predictions are correct, meaning it doesn't make significant changes to its weights. However, when a misclassification occurs, it updates its parameters in an "aggressive" manner to correct the mistake, making larger adjustments to the model.

The core idea behind the Passive Aggressive Classifier is to minimize the loss function while adapting to the data stream in an efficient and adaptive manner. It iteratively updates its weight vector based on a loss function, typically using a hinge loss or squared hinge loss, to maximize margin and correctly classify instances.

One of the key advantages of the Passive Aggressive Classifier is its ability to handle non-stationary data and adapt to changing environments, making it suitable for online learning scenarios where data distribution may shift over time.

Tools and Libraries used

nltk

NLTK, or Natural Language Toolkit, is a comprehensive library for natural language processing (NLP) tasks in Python. It provides a wide range of tools and resources for processing and analyzing human language data. NLTK includes functionalities for tasks such as tokenization, stemming[8], stopword removal[9], lemmatization, part-of-speech tagging, parsing, and named entity recognition, among others.

One of the key strengths of NLTK is its extensive collection of corpora, lexical resources, and pre-trained models, which can be used for various NLP tasks and research purposes. These resources cover a wide range of languages and domains, enabling researchers and practitioners to work with diverse linguistic data.

sklearn

Scikit-learn, often abbreviated as sklearn, is a popular machine learning library in Python that provides a wide range of tools for building and deploying machine learning models. It's built on top of other scientific computing libraries in Python such as NumPy, SciPy, and matplotlib, making it seamlessly integrated into the Python data science ecosystem.

Scikit-learn offers support for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. It provides a consistent and user-friendly API, making it easy to use and learn for both beginners and experienced machine learning practitioners.

One of the key strengths of scikit-learn is its extensive collection of well-implemented machine learning algorithms, ranging from traditional methods like linear regression and support vector machines to state-of-the-art techniques such as random forests, gradient boosting, and deep learning through integration with other libraries like TensorFlow and PyTorch.

Challenges faced

- **Invalid ascii characters in dataset**

Handling invalid ASCII characters involves in the dataset posed a challenge. The preprocessing involves identifying and replacing them using techniques like Unicode encoding.

- **Time taken for Stemming and Stopword removal**

Going through each word in the dataset and stemming each of them was a time consuming job.

- **Choosing the correct encoding technique for the natural language**

Selecting the appropriate encoding technique for natural language data is challenging due to the diverse character sets and languages present. Balancing efficiency, Unicode compatibility, and preservation of linguistic nuances further complicates the decision-making process, necessitating careful evaluation and testing to ensure accurate representation and processing of text data.

Contribution of Team Members

Roll No	Name	Contribution
21Z213	C O Avina	Study and implementation of vectorizing Natural Languages.
21Z241	Rakesh Kumar S	Exploration and implementation of data preprocessing for Natural Language.
21Z250	Santhosh A	Exploration and implementation of training models.
21Z258	Soorya Subramani	Study of feasibility of the idea and training models.
21Z260	Subhasri Shreya S L	Exploration and implementation of training models.
21Z264	V Harsheni	Study and implementation of vectorizing[5] Natural Languages.

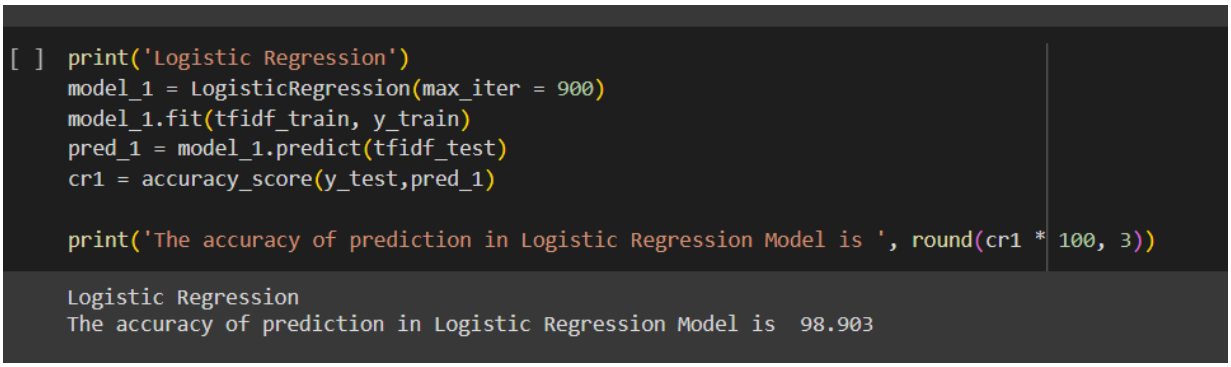
Annexure 1:

```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer
import pandas as pd
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.linear_model import PassiveAggressiveClassifier
```

```
print('Importing the data set')
data = pd.read_csv("Training.csv")
data.head()
data['label'] = data['label'].map({'Real': 1, 'Fake': 0})
data.head()
print('Tokenizing')
data['Text']=data['Text'].apply(word_tokenize)
print()
print("Tokenized successfully.")
print('Stemming')
porter=SnowballStemmer("english",ignore_stopwords=False)
def stem_it(text):
    return [porter.stem(word) for word in text]
data['Text']=data['Text'].apply(stem_it)
print()
print("Stemmed successfully.")
print('Stopword Removal')
def stop_it(t):
    dt=[word for word in t if len(word)>2]
    return dt
data['Text']=data['Text'].apply(stop_it)
data['Text']=data['Text'].apply(' '.join)
print()
print("Stop words removed successfully.")
print('Splitting dataset')
X_train,X_test,y_train,y_test = train_test_split(data['Text'],data['label'],test_size=0.25)
print('Vectorization (TF-IDF)')
my_tfidf = TfidfVectorizer(max_df=0.7)
tfidf_train = my_tfidf.fit_transform(X_train)
tfidf_test = my_tfidf.transform(X_test)
tfidf_train.head()
print('Logistic Regression')
```

```
model_1 = LogisticRegression(max_iter = 900)
model_1.fit(tfidf_train, y_train)
pred_1 = model_1.predict(tfidf_test)
cr1 = accuracy_score(y_test,pred_1)
print('The accuracy of prediction in Logistic Regression Model is ', round(cr1 * 100, 3))
model = PassiveAggressiveClassifier(max_iter=50)
model.fit(tfidf_train,y_train)
y_pred = model.predict(tfidf_test)
cr2 = accuracy_score(y_test,y_pred)
print('The accuracy of prediction in Passive Aggressive Model is ', round(cr2 * 100, 3))
```

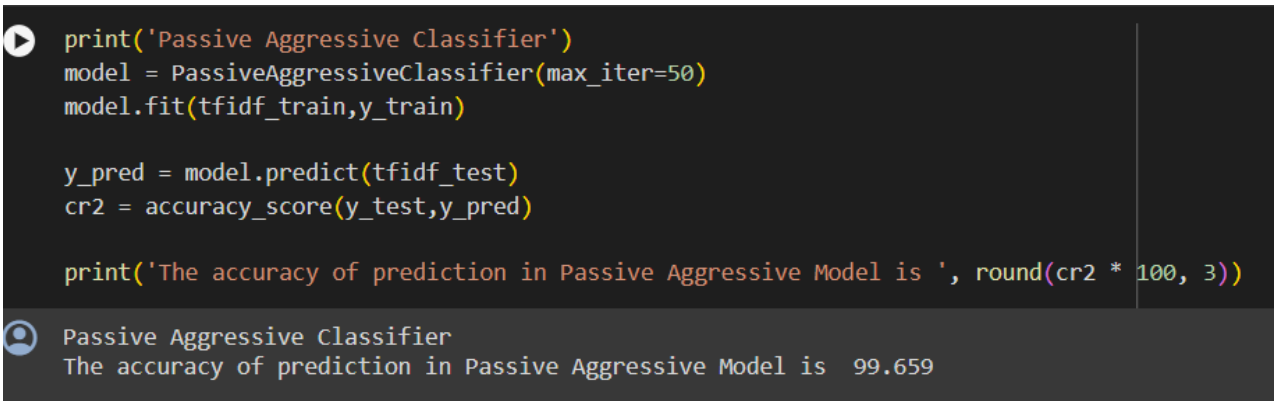
Annexure 2:



```
[ ]  print('Logistic Regression')
     model_1 = LogisticRegression(max_iter = 900)
     model_1.fit(tfidf_train, y_train)
     pred_1 = model_1.predict(tfidf_test)
     cr1 = accuracy_score(y_test,pred_1)

     print('The accuracy of prediction in Logistic Regression Model is ', round(cr1 * 100, 3))

     Logistic Regression
     The accuracy of prediction in Logistic Regression Model is  98.903
```



```
     print('Passive Aggressive Classifier')
     model = PassiveAggressiveClassifier(max_iter=50)
     model.fit(tfidf_train,y_train)

     y_pred = model.predict(tfidf_test)
     cr2 = accuracy_score(y_test,y_pred)

     print('The accuracy of prediction in Passive Aggressive Model is ', round(cr2 * 100, 3))

     Passive Aggressive Classifier
     The accuracy of prediction in Passive Aggressive Model is  99.659
```

References:

[1] https://www.kaggle.com/datasets/sadmansakibmahi/fake-news-detection-dataset-with-pre-trained-model

[2] https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/#:~:text=Logistic%20Regression%20is%20another%20statistical,pass%20this%20exam%20or%20not.

[3] https://www.geeksforgeeks.org/understanding-logistic-regression/

[4] https://www.geeksforgeeks.org/passive-aggressive-classifiers/

[5] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[6] https://scholarspace.manoa.hawaii.edu/bitstreams/2eb9a917-e4dc-4371-bb5a-dd894001bf57/download#:~:text=They%20categorized%20fake%20news%20into,increased%20use%20of%20fake%20information.

[7] https://scholarspace.manoa.hawaii.edu/bitstreams/2eb9a917-e4dc-4371-bb5a-dd894001bf57/download#:~:text=They%20categorized%20fake%20news%20into,increased%20use%20of%20fake%20information.

[8] https://www.geeksforgeeks.org/introduction-to-stemming/

[9] https://www.analyticsvidhya.com/blog/2019/08/how-to-remove-stopwords-text-normalization-nltk-spacy-gensim-python/