PROJECT-2
FAKE NEWS CLASSIFICATION

Kishore Jagan Jothi Kumar
and
Sanjay Vishal Velmurugan

Contents

I Introduction

The aim of this project is to classify fake news titles into three different categories: unrelated, agreed, disagreed. This will be achieved by training a model on the given training data set and then verifying its accuracy on the validation data set. Finally, we'll predict the labels for the test data set using our trained model.
There are 3 stages in the project:

1. Data Pre-processing
2. Feature Extraction and Engineering
3. Model Selection and Prediction

II Data Pre-processing

To clean the given data, we completed the following steps:

1. We created a dataframe using pandas and added all the data from the training set given in 'train.csv'. Then, we removed the Chinese news title columns.
2. We came across discrepancies in the label column of the data. There were multiple entries whose label didn't fall into the pre-determined valid labels which were agreed, disagreed or unrelated.
3. So, we removed all those entries with incorrect labels and the resulting data is stored in 'cleanedtrain.csv'. This is done in the code file named as 'refining_step1.ipynb'.
4. Later, we moved on to cleaning the news title data.
5. We first split the sentences into individual words using word_tokenize, then removed all the common stop words. Finally, we used Porter'stemmer algorithm to get the root words.
6. After completing the above mentioned cleaning processes, the data is stored in 'cleanedtrain2.csv'. These steps are executed in the file 'refining_step2.ipynb'.
7. Similar steps are followed for the validation data set and the final result is stored in 'cleanedvalidation2.csv'.

III Feature Engineering

1. Vectorization of the words and sentences is done using text embedding. This is done because once data is converted into a vector, we can predict whether two data points are similar or not by calculating their distance.

2. There are various different techniques to get the vectors. We have used Google's Universal Sentence Encoder.

3. The Universal Sentence Encoder encodes text into high dimensional vectors that can be used for text classification, semantic similarity, clustering, and other natural language tasks.

4. It comes with two variations i.e. one trained with Transformer encoder and other trained with Deep Averaging Network (DAN). The two have a trade-off of accuracy and computational resource requirement. While the one with Transformer encoder has higher accuracy, it is computationally more intensive. The one with DAN encoding is computationally less expensive with little lower accuracy.

5. We have used the transformer encoder by importing the pre-trained Universal Sentence Encoder available in Tensorflow-hub.

6. The relevant code to extract the sentence vectors is in 'sentence_encoder.py'.

7. We take the data from 'cleanedtrain2.csv' file as the input, convert it into vectors and store the vectors of title1_en and title2_en columns are saved in 'test3.npy' and 'test4.npy' files respectively.

8. Similarly, vectorization of the validation data stored in 'cleanedvalidation2.csv' is carried out by making the requisite changes to the code stored in the same file, 'sentence_encoder.py'. And, the vectors of of title1_en and title2_en columns are saved in 'valid1.npy' and 'valid2.npy' files respectively.

IV Model Description

We trained various models on the training data set and saved them. Later, we implemented the saved model on the given validation set to varying degrees of success. The relevant code is present in the 'classification.ipynb' file.

1. Logistic Regression Model
   Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

2. Gaussian Naive Bayes Model
   In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values.

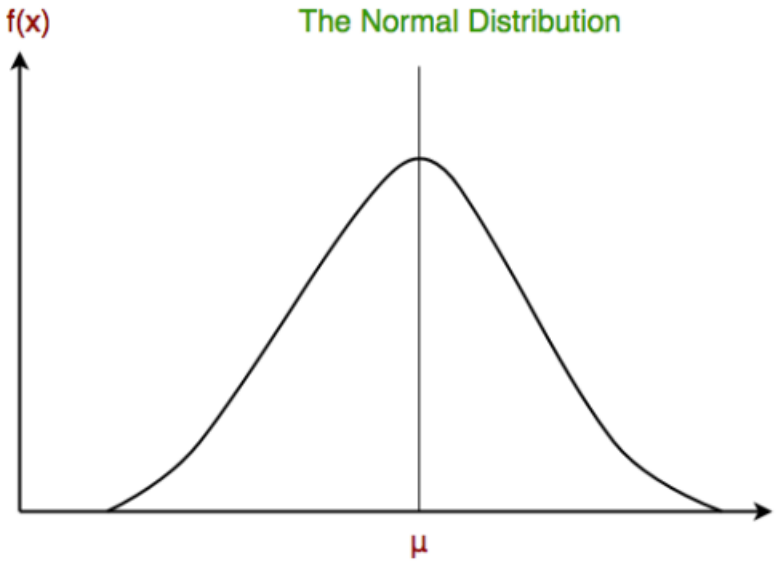

Figure 1: Plot of Gaussian Distribution

3. Decision Tree Model
   A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

4. Random Forest Model
   Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

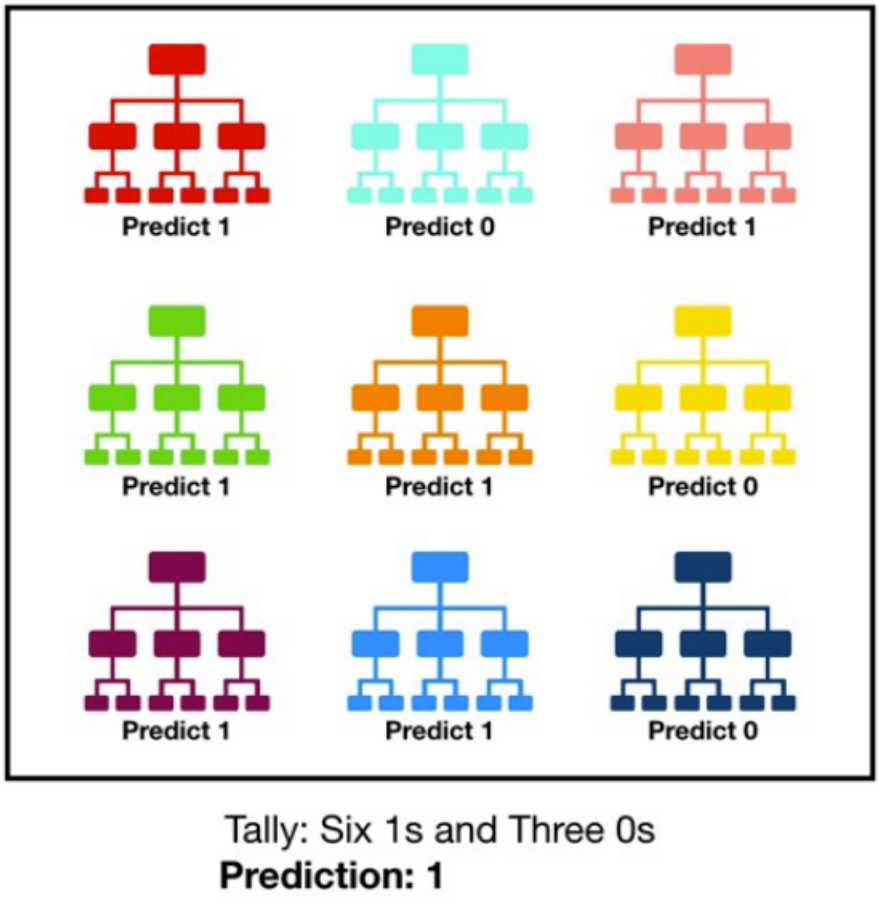

Figure 2: Visualization of a Random Forest Model Making a Prediction

5. AdaBoost Classifier

   Ada-boost or Adaptive Boosting is a type of ensemble boosting classifier. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations.

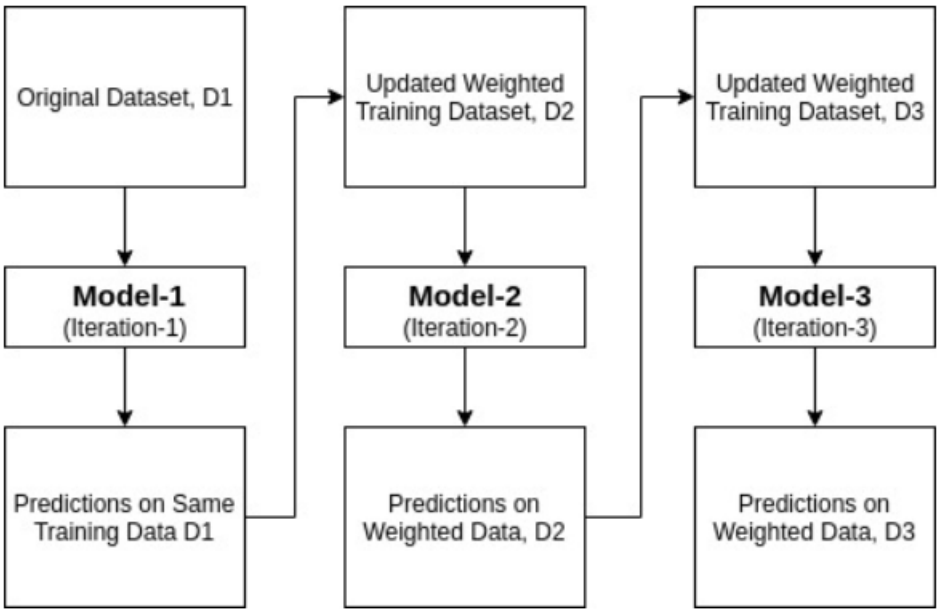

Figure 3: Flow chart of AdaBoost implementation

V Results

Classification Model	Accuracy	F1 Score
Logistic Regression	70.27%	0.66
Gaussian Naive Bayes	41.47%	0.43
Decision Tree	68.39%	0.55
Random Forest	81.66%	0.79
AdaBoost Classifier	63.04%	0.61

Figure 4: Accuracy and F1 Score of various classification models

After experimenting with different models, we selected the Random Forest Classifier model to predict the labels of the testing data set since, it had the highest accuracy and F1 score of 81.66 % and 0.79 respectively.

VI References

1. Tokenize text using NLTK in python,
   Available: https://www.geeksforgeeks.org/tokenize-text-using-nltk-python/

2. How to Clean Text for Machine Learning with Python,
   Available: https://machinelearningmastery.com/clean-text-machine-learning-python/

3. Use-cases of GoogleâĂŹs Universal Sentence Encoder in Production,
   Available: https://towardsdatascience.com/use-cases-of-googles-universal-sentence-encoder-in-production-dd5aaab4fc15

4. Classifier comparison,
   Available: https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparis

5. A guide to Text Classification(NLP) using SVM and Naive Bayes with Python,
   Available: https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34

6. Implementing SVM and Kernel SVM with Python's Scikit-Learn,
   Available: https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/