香港城市大學
City University of Hong Kong

CS 4486 Artificial Intelligence HW3 Report

Name: Chan Tsz Fung

SID: 56228799

Background:

Problem Definition

The goal of this project is to train classifiers for predicting whether certain comments are spam or not. To achieve this goal, training dataset(traing.csv) is used for serving as input. The original csv file contains attributes like Video ID, Author, Date and Text. These are the input attributes for training. It also contains attributes CLASS, indicating whether a record is spam or not.

Assumptions

For solving this classification problem, we need to first determine whether date, author and video ID have any influence on the predicted outcome. Some simple counting have been performed and graphs have been plotted for illustrating the relationship between outcome and these attributes:

DATE

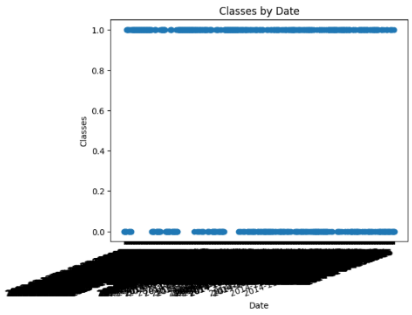
For DATE, we investigate the relationship between input and output attributes by plotting graph

```
import matplotlib.pyplot as plt

print(yt_train.DATE)

plt.plot_date(yt_train.DATE, yt_train.CLASS)
plt.title('Classes by Date')
plt.xlabel('Date')
plt.xticks(rotation=20, ha='right')
plt.ylabel('Classes')
```

```
0      2013-07-12T22:33:27.916000
1      2013-07-18T11:17:52.388000
2      2013-07-15T12:09:51.188000
3      2013-07-13T13:14:30.621000
4      2013-07-13T13:27:39.441000
                  ...
793          2014-11-07T19:27:45
794          2014-11-07T19:33:03
795          2014-11-07T19:33:46
796          2014-11-07T19:33:46
797          2014-11-07T19:45:26
Name: DATE, Length: 798, dtype: object
Text(0, 0.5, 'Classes')
```



It seems that no obvious trend can be generalized. Therefore, We would assumed that all timeslots have likely prob of producing spam.

AUTHOR

Check the author that are common in both training and testing

```
li1=yt_train["AUTHOR"].values
li2=yt_test["AUTHOR"].values
len1=len(li1)
len2=len(li2)
for i in range(len1):
    for j in range(len2):
        if li2[j]==li1[i]:
            print(li2[j])
```

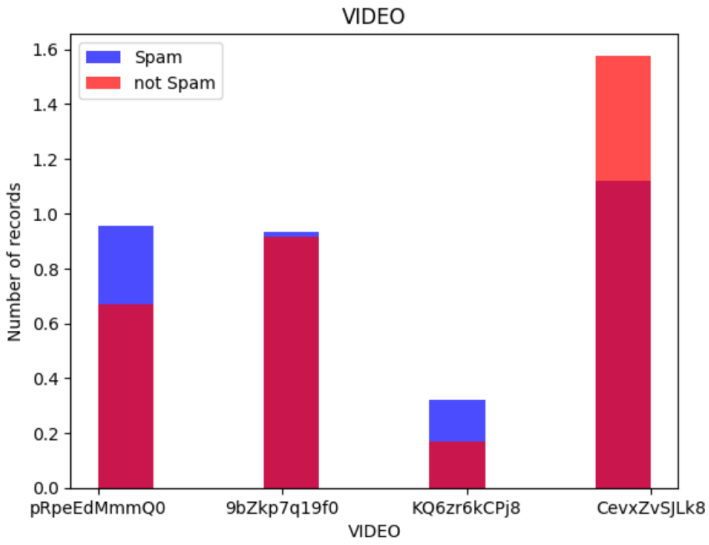
```
Nathan Waterhouse
Mason Sieverding
```

Only 2 authors are common in train and test, therefore we would not consider author as important in this problem (just drop it)

VIDEO

```
import matplotlib.pyplot as plt

plt.hist(yt_train[yt_train['CLASS']==1]["VIDEO"],color='blue',label='Spam',alpha=0.7,density=True)
plt.hist(yt_train[yt_train['CLASS']==0]["VIDEO"],color='red',label='not Spam',alpha=0.7,density=True)
plt.title("VIDEO")
plt.ylabel("Number of records")
plt.xlabel("VIDEO")
plt.legend()
plt.show()
```



From the graph, we can see that for every video, there are good amount of spam exist. Therefore Video ID would not be important attribute to be used for classifying a comment is spam or not as well.

Based on the evidence constructed from these attributes, we assume that Video ID, Author and Date have no influence on the prediction outcome. In other words, a spam can be found on any video, written by anyone and at any time. Comment text is the major predictor for predicting spam or not.

Other checks

We have also checked for the distribution of the spam and non-spam records. It is found that both count are similar to each other, which is good prerequisite for model building.

Furthermore, we have checked for missing values before model building. It is found that there are no missing values in the training set, which is desirable.

METHOD

To solve this classification problem, we have implemented a total of six different algorithms, as well as using different kinds of feature extraction methods.

The implementations and classification accuracies are listed below:

Implementation	Algorithms	TEXT representation
1	Logistic Regression Model	CountVectorizer as vector
		TfidVectorizer as vector
2	Recurrent Neural Network (LSTM)	tf.keras.layers.TextVectorization as vector
3	Feed-Forward Neural Network	pre-trained text embedding model nnlm-en-dim50
4	K-nearest neighbor	CountVectorizer as vector
5	Decision Tree	CountVectorizer as vector
6	Support Vector Machine (SVM)	CountVectorizer as vector

These are all the common approaches in solving classification problems.

We will go over every implementation one by one:

Logistic Regression Model

The reason for using logistic regression model is that it is quick to train and easy to implement. It also gets good performance for typical binary classification problems. In this problem, the accuracy is very high. It achieves 96 % accuracy for test data, which is the highest among all the implemented algorithms.

Recurrent Neural Network (LSTM)

RNN is common algorithm used for classifying text. The reason for us to implement this algorithm is that this kind of network possesses stochastic process with long- and short-range dependence. It uses layers that give the model a short term memory. It is particular good for time series data like text. Our input data is sequence of text, those wordings might be inter-related with each other. For example, the occurrence of a particular text might be affecting the occurrence of another text. By implementing RNN, we can better predict the next wordings and corresponding labels more accurately.

Feed-Forward Neural Network

Before actual implementation, this model is believed to have a lower accuracy compared to other algorithms like recurrent neural network. However, this algorithm is still worthy for implementation as the implementation is similar to the RNN. It is actually easier to implement compared to RNN. The main reason for us to include this algorithm is just to demonstrate the use of pre-trained text embedding model nnlm-en-dim50 introduced by google in a neural network. It helps embedding the text when the text data are feed to the network.

K-nearest neighbor

The reason for implementing this classifier is that there are many similar wordings found in the training data set. For example, those spam records usually contain words like “com”, “www”, “subscribe”. If unseen records possess these wordings and show similarities compared to these spam records, they are very likely to be spam. Moreover, this algorithm is very easy to implement. It also achieves high accuracy for comparing simple and small set of records like this.

Decision Tree

The reason for implementing this classifier is due to the simplicity of implementation. By using decision tree, it is believed that we can list out all the wordings that are commonly found in the text comment. We can use those to produce a list of decision for the classification. The performance is believed to be good as spam wordings are pretty common among spam messages. We are supposed to construct effective rules with high chances.

Support Vector Machine (SVM)

The reason for trying out SVM is that it provides good generalization ability and being robust at high dimension data. For representing our text data, which is split into

2000 columns, we need to make sure our algorithm is robust. SVM automatically regularized high-dimensional data such that data wouldn't be overfit. Another reason is that out dataset is very small. SVM can achieve good performance in handling small dataset.

Experiments & Results

Among the 6 implementations, the classification performances are ranked as follows:

Implementation	Algorithms	Accuracy on test data
1	Logistic Regression Model	96.03
2	Support Vector Machine (SVM)	94.55
3	Decision Tree	92.08
4	K-nearest neighbor	90.10
5	Recurrent Neural Network (LSTM)	86.14
6	Feed-Forward Neural Network	85.64

We can see that for this simple problem, simpler methods like logistic regression, SVM, decision trees provide better generalization performance compared to more complicated models like Neural network.

It does not mean that the network we trained are not good. In fact, classification performance over 85% can be considered as great. This is also the finalized value and the best value we can obtain after finishing multiple times of fine-tuning.

In the beginning of the training process for LSTM, we take only one layer of LSTM with 16 nodes, following by four layers of Dense layer with 16 nodes, it can only achieve about 70 % of classification performance when epochs for training is 10.

When we plot the graph of training accuracy for every epoch, it is shown that the accuracy of training set keeps increasing, finally approaching nearly 1. Meanwhile,

the accuracy of validation set increased for 6-7 epochs. It drops at later epochs. We conclude that the model is having the problem of overfitting.

To solve this problem, we add the dropout layer between every dense layer, such that the accuracy would not increase so fast during the training phase. We also lower the number of epochs, reducing it to 5 from 10, such that the model is less likely to be overtrained. We also fine-tuned the dropout parameters. The generalization performance for the test data is improved afterwards. The accuracy of training does not reach 1 when finished, the accuracy of the validation data is still increasing when we finish training. This is a good sign for having a more generalized model.

For training the support vector machine, we have also fine-tuned different set of parameters for optimization. Originally, we set C to a very small value (0.0001), which means that we prefer larger margin and allow higher misclassification in the training data. It is shown that the model tends to be underfitting the data and the classification performance is roughly about 90%. The performance improves after we gave the model an option to have a higher C value (C=10).

For training the decision tree, the max depth and number of leaf node is not set when we first constructed it. The result tree is more complicated, and the classification accuracy is only about 90%. To make the tree generalized better, we put constraint on max depth and number of leaf nodes. The accuracy is improved from 90% to 92 %.