REPORT Classification problem: texts as ‘ham’ or ‘spam’:

Understanding classifier

- The classification is binary class: the target classes being ‘spam’ and ‘ham’.
- Unilabel classifier: one mail can be either spam or ham , not both
- Steps in classifier:
    1. Selection
    2. Training
    3. Prediction
    4. Evaluation

Statistics of the dataset

The dataset contains 5574 instances of SMS labelled messages tagged as SPAM or HAM.The dataset was collected from 4 different sources: from Grumbletext Web site : 425 SPAM , from NUS SMS Corpus : 3375 HAM messages, from PhD thesis: 450 HAM messages and from SMS Spam Corpus : 1002 HAM messages and 322 SPAM messages.

The dataset is such that first word is classification ‘spam’/’ham’ followed by space and the message.

Processing Raw Data

Preprocessing is done as 4 steps:

1. Writing into CSV:
    - For each line in the dataset
    - writing into “preprocessed.csv”
    - while writing, the text is appended to 1 column and it’s corresponding classification is written to another column

2.read_file:

Read each line of the dataset and pass it on to the function defined in 1. Writing into CSV. It also creates a file “preprocessed.csv” before any writes

3. Read from CSV:

All rows are parsed and selected from the given CSV, and the rows are appended to the array all rows. Only after checking conditions like when the length of the message is greater than 1.

4. Preprocess:

Taking the input as text_corpus,

Parses each sentence in a corpus and does the following:

1. Convert all letters to lower case
2. Remove digits and numbers
3. Remove all punctuations
4. Remove all white spaces

An array of preprocessed sentences are returned.

Understanding data and finding a fit

The given problem of sms text classification as spam and ham falls into the category of supervised learning. In supervised learning, Since here the training data, in fact the whole data set has desired outputs. Hence while splitting training and test dataset, we get datasets that have the data as well as their corresponding desired outputs, which here is the classification as spam and ham.
Here the problem can be defined in such a way that:

Given: training examples <x, f(x)> here x is the text message and f(x) is the classification of spam or ham.

Find: approximation for f(x): here the function that maps given text to spam or ham

Choice of the model

I chose SVC(support vector classifier) because of the property of the dataset which is not balanced. The model SVC works particularly well with such unbalanced data. The dataset given is unbalanced in terms that 'HAM' messages in the dataset is much larger compared to the 'SPAM' messages present.

Also SVC is a cost-sensitive classifier that classifies with low memory cost even when positive and negative samples are far apart in size.

Further for binary data: here classification is either spam or ham. Hence in such binary classifier, the classification efficiency is larger compared to any other classifier.

Hyper parameter tuning: SVC provides inbuilt function SVC grid as explained in feature engineering.

Stepwise Explanation of Idea

Preprocessing:

After preprocessing we have in 2 lists: the text messages and their corresponding classification words.

Train Test Split:

The given dataset is split into train and test at 30 percentage ratio as suggested in class. Each preprocessed text is tokenized and passed to train method of Word2Vec

Train model Word2Vec:

Using the hyper parameters, n_epochs and features, the word to vec model is trained based on the data passed, which are tokenized sentences. The number of features represent the number of dimensions in a hyperplane.

Generate training data for SVC model:

The training data is generated from the Word2Vec model designed earlier. Each sentence in a given dataset is tokenised into words. These words are represented as 1*30 array. If the word is not empty, then the vector as predicted by Word2Vec model is taken. The

predicted vectors for each word in a sentence is added through vector addition. And the average is taken. This represents the vector that represents a sentence.

Train_SVC model

The hyperparam tuning is done in an inbuilt manner using GridSearchCV.Which takes for a given train sample and output, The sample here is the vector and the output is 'spam'/'ham', with the hyper-params as explained in hyper_parameter choice.

Preprocess Test Data:

Preprocessing and generation of test data is done as for the train data.

Predict:

The predict function on the SVC model is called, which returns the predicted output values.

Measure metrics:

The output of predict function and the actual output of test data is is compared though the function: classification_report of sklearn.metrics

Feature engineering

Here the features are all unique words in the given dataset! These unique words are plotted as vectors in a n-dimensional space ,given by hyper-parameter as described below

Hyper_parameter choice:

For the model: Word2Vec:

The hyper_params chosen are : features that represent the dimension in which the vector is projected, It is chosen as 30 since the avg length of sms text contains 30 words. The number of epochs is chosen as 50 since it's not too long nor too short. More number of epochs take longer time and less number of epochs result in bad training model.

For SVC:

The hyperparam tuning is done in an inbuilt manner using GridSearchCV. The params that are trained are C: error reward is represented as c., gamma:parameter of fit for nonlinear hyper-planes and kernel: that specifies the type of hyperplane.

Evaluation method used

In built classification_report of sklearn.metrics package, when passed with actual value of test dataset and predicted value, calculates confusion matrix and from the same derives the following measures for each of the classes('spam' and 'ham'):

- Precision:
  - Label an instance positive that is actually positive .
  - For each class - true positives : sum of true and false positives
- Recall:
  - Find all positive instances
  - true positives : sum of true positives and false negatives
-
-

- F1:
    - weighted harmonic mean of precision and recall

- support scores:
    - number of actual occurrences of the class

References:

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html
https://www.geeksforgeeks.org/svm-hyperparameter-tuning-using-gridsearchcv-ml/
https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92