Medical Text Classification

Approach:

In order to classify the documents into 5 categories, following steps were performed:

- Import and extract Training Data from .dat file
- Import Test Data from .dat file
- Convert the training data into pandas DataFrame, having two columns; one for the type of category and other for the text/abstract.
- Preprocess both the training and the test data
- Specify the input features and response variable for training data.
- Split the training data using train-test split library
- Import various libraries for classifying the documents.
- Perform typical 4 step method:
  Step 1: Import Model
  Step 2: Instantiate the Model
  Step 3: Fit the Model
  Step 4: Predict the values
- Compare predicted values with the known output values and get F1 score.
- Finally, using the entire training set predict the values for Test Data.

Methodology:

- Technology used: Scikit-Learn, Pandas, Numpy, NLTK, re

- Importing data:
  Scanned the input training .dat file line by line and extracted response variable and data into two lists. Then, created a pandas dataframe using these lists. For test data, followed the same procedure for the abstract-text field.

- Data Preprocessing:
  Using default libraries like ‘re’ and using ‘stopwords’ and ‘SnowBallStemmer’ from NLTK (corpus and stem), I convert the text into lower case, also removed stopwords from the text; including escape characters.
- Train_Test_Split:
  Train data is split into two parts; training sub data and training-test sub data. (50%-50% proporation: Parameter used: test_size=0.5) In order keep the results constant; I used the parameter ‘random_state’ and equated it to 4. So that for every computation the samples in training set and test set after splitting the training set must be constant.

- Model Creating:
  In order to classify the documents; I used SGDClassifier that is ‘Stochastic Gradient Descent Classifier’.

- 4 Step Procedure:
  - Step 1: Import Model
    I imported various libraries such as,
    CountVectorizer (In order to convert the text into manipulative values). 'Bag of Words' principle in used.
    TFIDTransformer: To transform a count matrix to normalized tf or tf-idf representation.
    Pipeline from sklearn: This is used to perform transform operations. It accepts parameters such as CountVectorizer, TfidfTransformer, SGDClassifier.
    I mentioned parameters separately in a list.

  - Step 2: Instantiated the Model
    I instantiated the model inside the parameter list of Pipeline.

  - Step 3: Fit the Model
    I fitted the model using splitted training data first in order to computer F1 score and then fitted the model with entire training set in order to predict the types of various documents mentioned in test data.

  - Step 4: Predict the category
    After the machine has learned the model, in order to predict the category; we feed the predictor with test data.

- Capturing the response:
  In order to capture the response; I save the output of the predictor in the .dat file.

- Parameter details:
  - 'vect__max_df': (0.5, 0.75, 1.0):
    max_df is set to automatically detect and filter stop words based on intra corpus document frequency of terms.

  - 'vect__max_features': (None, 5000, 10000, 50000):
    It only considers the top max_features ordered by term frequency across the corpus.

  - 'tfidf__use_idf': (True, False),
    use_idf=True, smooth_idf=True.

Ref.: Scikit Learn Documentation