Jyotsna Nakte jnn2078
Jairaj Tikam jpt1342

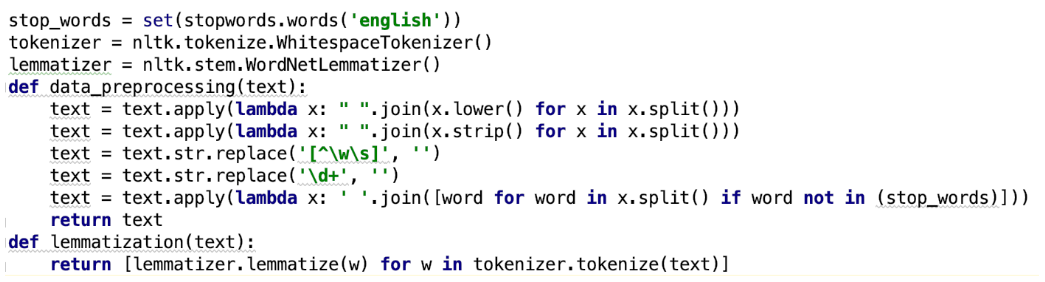
Milestone 03: Testing of Candidate Algorithms

Overview:

We test four machine learning algorithms keeping the goal of the project to predict the product would be recommended or not based on review text. The algorithms tested were Gaussian Naive Bayes, Random Forest Classifier, Logistic Regression, Linear SVC. The sections below explain the step-by-step process to achieve the aim of this phase.

I.] Data-Preprocessing:

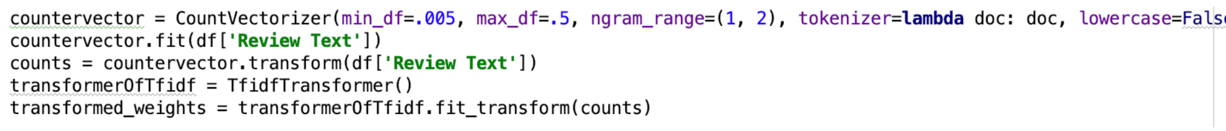
The important columns were filtered for the task from the dataset using Clothing ID, Review Text, Recommended IND of the dataset. Later, we checked for empty or null values in the text. To perform natural language processing, we cleaned the text by using techniques of removing stop words of text, converting all the text to lowercase, removing extra spaces and special characters, removing punctuations and numbers. Further, we applied lexicon normalization by Lemmatization that returned root form of each word. We carried out the process by making use of nltk (natural language toolkit) python package.



```
stop_words = set(stopwords.words('english'))
tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()
def data_preprocessing(text):
    text = text.apply(lambda x: " ".join(x.lower() for x in x.split()))
    text = text.apply(lambda x: " ".join(x.strip() for x in x.split()))
    text = text.str.replace('[^\w\s]', '')
    text = text.str.replace('\d+', '')
    text = text.apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))
    return text
def lemmatization(text):
    return [lemmatizer.lemmatize(w) for w in tokenizer.tokenize(text)]
```

II.]TF-IDF Matrix:

To use 'Review text' the text field column was converted into text features; we made use of TF-IDF (Term Frequency-Inverse Document Frequency) concept. For information retrieval from the text, tf-idf was used to discover how important a word is to a document in a collection or corpus. With the help of python package sci-kit-learn, we made use of CountVectorizer class to count how many times each term shows in every document and Tfidf Transformer class producing weight matrix.



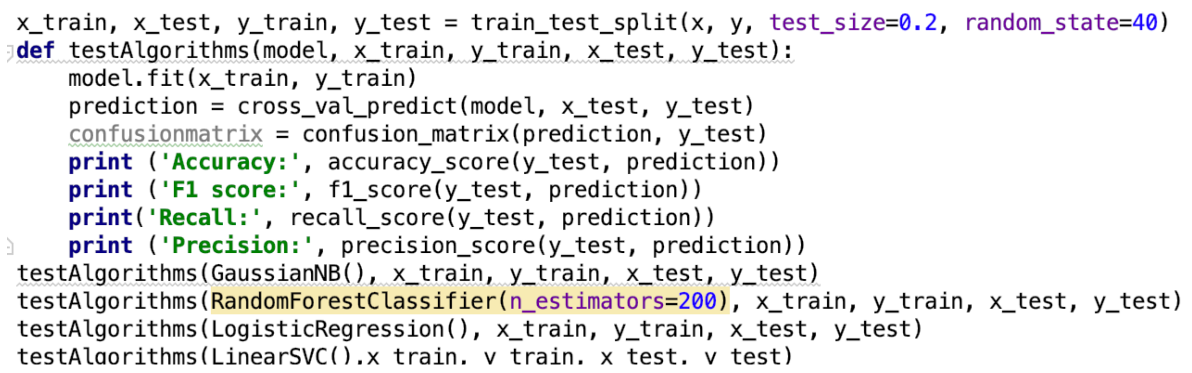
```
countervector = CountVectorizer(min_df=.005, max_df=.5, ngram_range=(1, 2), tokenizer=lambda doc: doc, lowercase=False)
countervector.fit(df['Review Text'])
counts = countervector.transform(df['Review Text'])
transformerOfTfidf = TfidfTransformer()
transformed_weights = transformerOfTfidf.fit_transform(counts)
```

The vocabulary for the count of words found in the document found was 880. Therefore 880 features were added to predict the recommendation.

III.] Algorithms Tested:

To test the algorithms we first split the data into training, testing set using python package function sklearn.model_selection.train_test_split() with random_state= 40 and test_size=0.2.

The training set has 18788 values, and the testing data set has 4698 values from the original data. The training set was trained and used to predict the recommendation value of testing dataset using machine learning classification algorithms i.e., using supervised learning. The models used were Gaussian Naive Bayes, Random Forest Classifier, Logistic Regression, Linear SVC were trained to fit and predict by using python package sklearn. The prediction model gave information of confusion matrix which gives information about true positives (TP), true negatives (TN), false positive (FP), false negatives (FN) which helped to calculate accuracy, recall, precision, f-score. These measures help us to evaluate the model. Accuracy is the performance measure ratio of correctly predicted observations to the total observation (TP+TN/TP+TN+FN+FP). Recall refers to the ratio of correctly predicted positive observations to all observations in actual class – yes (TP/TP+FN). Precision is the ratio of correctly predicted positive observations to the total predicted positive observations (TP/TP+FP). F-score refers to the weighted average of recall and precision(2*TP/2TP+FN+FP).



```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=40)
def testAlgorithms(model, x_train, y_train, x_test, y_test):
    model.fit(x_train, y_train)
    prediction = cross_val_predict(model, x_test, y_test)
    confusionmatrix = confusion_matrix(prediction, y_test)
    print ('Accuracy:', accuracy_score(y_test, prediction))
    print ('F1 score:', f1_score(y_test, prediction))
    print('Recall:', recall_score(y_test, prediction))
    print ('Precision:', precision_score(y_test, prediction))
testAlgorithms(GaussianNB(), x_train, y_train, x_test, y_test)
testAlgorithms(RandomForestClassifier(n_estimators=200), x_train, y_train, x_test, y_test)
testAlgorithms(LogisticRegression(), x_train, y_train, x_test, y_test)
testAlgorithms(LinearSVC(),x_train, y_train, x_test, y_test)
```

IV.] Results:

Algorithm	Accuracy	Confusion Matrix	Recall	Precision	f-score
Gaussian Naive Bayes	61.70%	706 1655 145 2192	0.5697	0.93	0.70
Linear SVC	86.07%	424 227 427 3620	0.9409	0.89	0.92
Random Forest Classifier	85.41%	264 98 587 3749	0.9792	0.86	0.91
Logistic Regression	84.92%	285 77 566 3770	0.9799	0.869	0.91

V.] Analysis of the Algorithms and Results:

Gaussian Naive Bayes: The Naive Bayes classifier is a simple classification based on Bayes Theorem. It assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature i.e., they are mutually independent. It is useful to build classifiers on large datasets. As the dataset we used here has a large number of features of the words and Naive Bayes handles such cases. Here we have 880 words that behave as features. The algorithm offers simplicity with the advantage of fast model training and prediction time. The model gave the lowest accuracy of 61.70% with highest recall value 0.5697.

Linear SVC (Support Vector Classification): SVC is used to find a hyperplane in N-dimensional space (N—the number of features) that distinctly classifies the data points. The aim is to find a plane that has the maximum margin to classify data points in the future with more confidence. Text classification works best with linear SVC providing higher speed and better performance. The dataset here has less than 1000 features makes the algorithm work efficiently for NLP. The model gave the best accuracy of all 86.07% with approximate similar recall, precision values.

Random Forest Classifier: Random forest classifier are random decision makers popularly known as an ensemble learning method for classification that estimates based on the combination of different decision trees. Every tree in the forest is built on a random best subset of features by voting. It has no requirement of feature normalization and reduces over-fitting. It is said to be one of the most accurate learning classifiers. In the algorithms used, this algorithm performs the second best with accuracy 85.41% and precision 0.91 that is percent of result returned which are relevant.

Logistic Regression: The predictive learning model usually used to analyze data with one or more independent variables that determine an outcome. It works as the best fitting model to find the relationship between the dichotomous target variable and set of independent variables. The target/response is dichotomous, that is having only two possible outcomes. The dataset here has recommended target variable having two values 0/1 and a massive set of independent variable features. The algorithm works best for such datasets, and most of the times work better than binary classifications. The model performed third best with the accuracy of 84.92% and recall 0.97 which is the percent of total relevant result returned.

As we can see from the above table, Linear SVC is the algorithm that has the best accuracy rate among all the candidate algorithms. It also has the best F-score metric because of its Recall and Precision rates. Gaussian Naive Bayes is the worst performing algorithm with accuracy well below the rest of the candidate algorithms. This is also complimented with the fact that it has the lowest F-score. Our core algorithm, Random Forest Classifier is a balance between efficiency and performance. It has the second highest accuracy score. The results show the recall being high in the other three algorithms except Naive Bayes because of imbalanced class. As mentioned in the prior phases the recommended class 1 occupied 82% of the data. As the precision increases recall is less and vice-versa. The accuracy can be improved by using balanced classes in the next phase of the project.

Core algorithm:

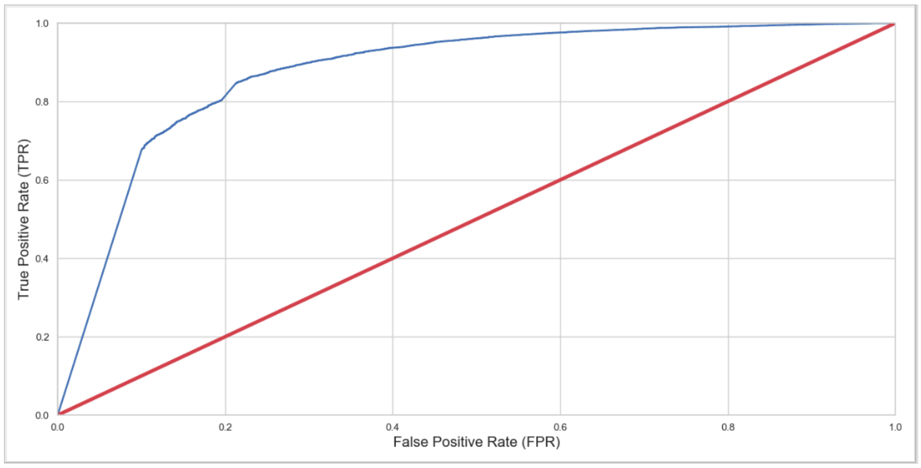
We have selected Random Forest Classifier to be our core algorithm. Looking at the results provided above, we can see that it is not the best performing algorithm in terms of accuracy since the Linear SVC is just a tad bit more accurate. One of the main reasons we have chosen Random Forest as our core algorithm and not Linear SVC is because we wanted our algorithm to be scalable

Jyotsna Nakte jnn2078
Jairaj Tikam jpt1342

for large amounts of data. Linear SVC, though quick enough for the current number of dimensions, will not scale well for higher dimensional data.

ROC (Receiver Operator Curve) applied to the Random Forest Classifier:

The Receiver Operating Characteristics curve gives us the plot of true positive rate against the false positive rate for the various possible breakpoints in a diagnostics test. The ROC curve determines the trade-off between sensitivity and specificity of the results. As we can see in the plot below, there is still a significant gap between the top left corner and the curve. This is because of the imbalance in the class distribution. However, the area under the curve being so high is expected because of the high accuracy obtained by the algorithm.



Future work:

The precision and recall rates are not good enough at this point of time, but we believe that data resampling and class balancing would further help improve the numbers. We plan to do this by decreasing samples in the larger value class and this way the classes would be balanced. The goal is next phase would be to improve the model by balancing class using techniques like cross validation. To make the most of the available data samples, we are planning to run the algorithm using random batches of data from the most probable class and use an average of the results obtained.

References:

[1] https://scikit-learn.org/stable/
[2] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
[3] https://www.nltk.org/book/ch01.html
[4] https://www.toptal.com/machine-learning/nlp-tutorial-text-classification