Individual Final Report
Cristina Giraldo

Introduction

The project aims to predict the hotel rating based on the reviews. In order to achieve this, we used text mining techniques to extract information from the reviews and accurately predict the rating. We used a data set of approximately 10,000 observations of US hotels reviews between 2016 and 2018.

To model and predict results we executed the following different steps to get the information and graphics for this project. First we depurated the data by eliminating unnecessary columns or features, and also cleaning the data by removing null values, noise reduction, lower casing. Furthermore, In the preprocessing stage, we also performed some techniques specifically for text mining that includes remove stop words, count rare words, tokenization, lemmatization and stemming.

The models used to predict the ratings were Naïve Bayes, Support Vector Machine, and Linear Vector Machine with Lasso penalty. In all 3 cases, the models were able to predict about 50% of accuracy. With this accuracy we conclude that more observation would be needed if we can to improve the accuracy of the models.

Description of individual work

The model chosen for me to predict the categories according to the review was Naïve Bayes. This algorithm is based in probability. It is considered reliable and useful to work with big datasets; specially with categorical variables. For this reason, it has been used for a long time in classification problems such as text classification, spam filtering and sentiment analysis. Given that I decided to try this algorithm to verify how precise is the prediction according to the rate review.

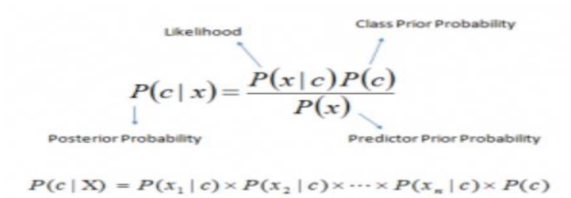

Figure 1. Bayes theorem

$$P(\text{rating}|\text{review}) = \frac{P(\text{rating}|\text{review}) \text{ x } P(\text{rating})}{P(\text{review})}$$

Figure 2. Bayes theorem - Our case

Since, we want to know what are the probabilities of getting a review with one, two, three, four or five categories. We can convert the formula in figure 2 in the following equation:

P(rating|review) x P(rating 1)

P(rating|review) x P(rating 2)

P(rating|review) x P(rating 3)

P(rating|review) x P(rating 4)

P(rating|review) x P(rating 5)

Figure 3. Bayes theorem modified to our case

This algorithm basically tries to find the mean of a word according to the class. In our case the classes that were used for this model were define as 1,2,3,4 and 5; being 5 the best rating for the review. Since we have 5 classifiers a multinomial naive Bayes was implemented in our project and we did use of this variables as a labels to do our prediction.

Description of work / code

In this project, I had an active participation in the creation of classes, methods and UI. To be more precise the following images in a red square shows my participation in the team during the construction of the code implemented for the analysis hotel reviews in the U.S.

File CleaningProcess.py

```python
class CleaningDF:

    def __init__(self, p_df):
        self.p_df = p_df

    def get_info(self):
        return self.p_df.info()

    # cristina. deletes unnecessary features
    def drop_columns(self):
        dropcols = ['id', 'dateadded', 'dateupdated', 'address', 'categories', 'primarycategories', 'keys', 'latitude',
                    'longitude', 'postalcode', 'reviews_date', 'reviews_dateseen', 'reviews_sourceurls',
                    'reviews_usercity', 'reviews_userprovince', 'reviews_username', 'sourceurls', 'websites', 'location']

        self.p_df = self.p_df.drop(dropcols, axis=1)
        return self.p_df

    def missing_val(self):
        self.p_df.isnull().values.any()                                    # cristina
        self.p_df["reviews_text"].isna().sum()                             # Kevin
        self.p_df["reviews_title"].notnull().isna().sum()                  # Kevin
        self.p_df['reviews_text'] = self.p_df['reviews_text'].str.replace('\d+', '') #cristina. delete numbers
        self.p_df['reviews_text'] = self.p_df['reviews_text'].dropna().reset_index(
            drop=True)  # delete NaN and reindex   #cristina
        self.p_df['reviews_text'] = self.p_df['reviews_text'].astype(str)  # Cristina. To assure all are strings.
        return self.p_df
```

```python
class PreprocessReview:

    def __init__(self, pr_df):
        self.pr_df = pr_df


    # cristina. Remove Most frequent words
    def common_words(self, wfilter, n_words):

        self.filter = wfilter
        self.n_words = n_words
        all_words = ' '.join([text for text in wfilter])
        all_words = all_words.split()

        #get word frequency
        fdist = FreqDist(all_words)
        words_df = pd.DataFrame({'word': list(fdist.keys()), 'count': list(fdist.values())}) #converts to df

        # selecting top #terms most frequent words and plot
        d = words_df.nlargest(columns="count", n=self.n_words)
        plt.figure(figsize=(20, 5))
        ax = sns.barplot(data=d, x="word", y="count")
        ax.set(ylabel='Count')
        # plt.show()
        return d
```

```python
    # cristina. tokenization - separates words
    def tokenization(self):
        self.pr_df['reviews_text_token'] = self.pr_df.apply(lambda row: nltk.word_tokenize(row['reviews_text']), axis=1)
        return self.pr_df
```

```python
class Predictors:
    def __init__(self, f_df):
        self.f_df = f_df

    #Cristina. Model and evaluation
    def naivesb(self):
        X_train, X_test, y_train, y_test = train_test_split(self.f_df['reviews_text'],
                                                            self.f_df['reviews_rating'].astype('int'),
                                                            test_size=0.25, random_state=85)
        count_vectorizer = CountVectorizer()  # converts to bags of words and it would remove stop words

        count_train = count_vectorizer.fit_transform(X_train.values)
        count_test = count_vectorizer.transform(X_test.values)

        nb_classifier = MultinomialNB()
        nb_classifier.fit(count_train, y_train)
        pred = nb_classifier.predict(count_test)
        metrics.confusion_matrix(y_test, pred, labels=[1, 2, 3, 4, 5])
        counter = 0
        for review, category in zip(X_test, pred):
            print('%r => %s' % (category, review))
            if (counter == 40):
                break
            counter += 1
        print("Accuracy score Naives Bayes: " + str(metrics.accuracy_score(y_test, pred)))
        return metrics.accuracy_score(y_test, pred)
```

Results

The accuracy of Naives Bayes algorithm is about 48% which is an indicator of the low fit for our data and the results shows clearly that there are some misclassifications

- 5 => pleasant surprise cottages neat clean perfect family toasty warm even temps dipped windchill night full sized oven stove fridge plenty space fit whatever food snacks need bring shower
- 4 => returned week long stay Americana want fancy don’t stay complaints breakfast morning fine us bagel juice roll muffin coffee piece fruit good rooms clean yes older motel
- 4 => bad beds creaky thin walls hear everything room next door hallway housekeeping isn’t consistent laundry delivery wrong room good location
- 5 => bedroom suite right husband daughter enough space spread stay candlewood whenever visiting family nj like kitchen full size refrigerator microwave dishwasher two burner stove like free laundry place go
- 4 => bad bed small two people blanket thin staff ok friendly though good location new England style building room cute
- 4 => smooth early check given upgraded room room gorgeous enjoyed jacuzzi casino little smokey buffet nice although find piece plastic greens manager apologize would rated higher awakened
- 5 => want thank wonderful service received best western plus encino visiting mother cottage hospital front desk clerks housekeeping cordial courteous best western

Summary and conclusions

Our project used text mining techniques to draw meaning out of the written online reviews. Unlike normal data mining, most of the text mining data is unstructured with a content that can be valuable. However, it requires to implement several steps of preprocessing to extract the meaningful information.

We all participated in different steps of the project however the particular codes made for me are listed in the screens in the section " Description of work/code" and my biggest participation was the UI with about 85% of the work.

Percentage of code

35/240 = 14%

References

García, S., Luengo, J., & Herrera, F. (2015). Data preprocessing in data mining. Cham: Springer. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

A. N. (n.d.). *Support Vector Machines*. Lecture http://cs229.stanford.edu/notes/cs229-notes3.pdf

Jain, S. (2019, March 11). Ultimate guide to deal with Text Data (using Python) - for Data Scientists and Engineers. Retrieved from https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/

R. (n.d.). *An Idiot's guide to Support vector machines (SVMs)*. Lecture.

Text Analytics for Beginners using NLTK. (n.d.). Retrieved from https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk

Tibshirani, R. (1996) Regression shrinkage and selection via the lasso. J.R.S.S.B. 58, 267-288

A. (2016, February 24). *Topics in sparse Support Vector Machines*. University of Arizona. https://www.math.arizona.edu/~wammonj/talks/topics-sparse-svms.pdf

J., S., T., & R. (n.d.). *1-norm Support Vector Machines*. Stanford University https://papers.nips.cc/paper/2450-1-norm-support-vector-machines.pdf