RATING PREDICTION

Best Laptop

"Super build quality with good hardware configuration!"

Value for Money

"Bought it today. Find it cool and smooth. Best features in this budget"

Great Watch

"This is a great product and offers alot of insight into ones health"

@ by Vijayaraghavan S

Introduction

Business Problem Framing:

Websites and online stores increasingly rely on rating systems and interactive elements for visitors and customers. Users leave ratings on websites or give their opinions on products and companies using the comment boxes embedded on the page. The added value for users is clear. Customers and website visitors often gain important information through ratings and can read other users' experiences before investing in a product, service, or company. Since it's not possible to take a closer look at products online, these ratings and reviews fill information gaps. Online shopping is quite convenient, practical, time-saving, and fast. But nonetheless, there's a distance between the provider and the customer. However, if a website contains ratings or a comment box, this can help to close the gap between the provider and the consumer. Customers can then use the feedback to help each other decide whether to go ahead with the purchase by providing information on the function, range, and value of a product.

The rise in E-commerce has brought a significant rise in the importance of customer reviews. There are hundreds of review sites online and massive amounts of reviews for every product. Customers have changed their way of shopping and according to a recent survey, 70 percent of customers say that they use rating filters to filter out low-rated items in their searches. The ability to successfully decide whether a review will be helpful to other customers and thus give the product more exposure is vital to companies that support these reviews like Google, Amazon, Flipkart, Myntra, Reliance, etc. There are two main methods to approach this problem. The first one is based on review text content analysis and uses the principles of natural language processing (the NLP method). This method lacks the insights that can be drawn from the relationship between customers and items. The second one is based on recommender systems specifically on collaborative filtering and focuses on the reviewer's point of view.

Introduction

Conceptual Background of the Domain Problem :

Rating prediction is a well-known recommendation task aiming to predict a user's rating for those items which were not been rated yet by customers. Predictions are computed from users' explicit feedback i.e., their ratings provided on some items in the past. Another type of feedback is user reviews provided on items which implicitly express users' opinions on items. Recent studies indicate that opinions inferred from users' reviews on items are strong predictors of users' implicit feedback or even ratings and thus, should be utilized in computation. As far as we know, all the recent works on recommendation techniques utilizing opinions inferred from users' reviews are either focused on the item recommendation task or use only the opinion information, completely leaving users' ratings out of consideration.

The approach proposed in this project is filling this gap, providing a simple, personalized, and scalable rating prediction framework utilizing both ratings provided by users and opinions inferred from their reviews. Experimental results provided on datasets containing user ratings and reviews from the real-world Amazon and Flipkart Product Review Data show the effectiveness of the proposed framework.

Review of Literature:

Rating is a classification or ranking of someone or something based on a comparative assessment of their quality, standard or overall performance.

This project is more about exploration, feature engineering, and classification that can be done on this data. Since we scrape a huge amount of data that includes five stars rating, we can do better data exploration and derive some interesting features using the available columns.

We can categorize the ratings as 1, 2, 3, 4, and 5 stars respectively. 1 stand for Bad and 5 stand for Excellent.

Introduction

Motivation for the Problem Undertaken:

Every day we come across various products in our lives, on the digital medium we swipe across hundreds of product choices under one category. It will be tedious for the customer to make a selection. Here come 'reviews' where customers who have already got that product leave a rating after using them and brief their experience by giving reviews. As we know ratings can be easily sorted and judged whether a product is good or bad. But when it comes to sentence reviews, we need to read through every line to make sure the review conveys a positive or negative sense. In the era of artificial intelligence, things like that have got easy with the Natural Language Processing (NLP) technology. Therefore, it is important to minimize the number of false positives our model produces, to encourage all constructive conversation. Our model also provides beneficence for the platform hosts as it replaces the need to manually moderate discussions, saving time and resources. Employing a machine learning model to predict ratings promotes an easier way to distinguish between product qualities, costs, and many other features.

Many product reviews are not accompanied by a scale rating system, consisting only of a textual evaluation. In this case, it becomes daunting and time-consuming to compare different products in order to eventually make a choice between them. Therefore, models able to predict the user rating from the text review are critically important. Getting an overall sense of a textual review could in turn improve the consumer experience.

Analytical Problem Framing

Mathematical / Analytical Modeling of the Problem:

As per the client's requirement for this rating prediction project I have scraped reviews and ratings from well-known e-commerce sites. This is then saved into a CSV format file. Also, I have shared the script for web scraping into the GitHub repository.

Then loaded this data into a data frame and did some of the important natural language processing steps and gone through several EDA steps to analyse the data. After all the necessary steps I have built an NLP ML model to predict the ratings.

In our scrapped dataset our target variable column "Ratings" is a categorical variable i.e., it can be classified as 1, 2, 3, 4, and 5 stars. Therefore, we will be handling this modeling problem as a multi-class classification project.

Data Sources and their format:

The project is done in two parts 1. Data Collections using Web-scrapping and 2. Model Building Phase.

Total Reviews collected 74000 which includes some blank rows which is been treated already while building a model.

Have concentrated on three columns mainly only 1. Review Title, 2. Detailed Review and 3. Ratings.

Model Building Phase

After collecting the data, you need to build a machine learning model. Before the model, the building does all data pre-processing steps involving NLP. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the steps mentioned below:

1. *Data Cleaning*
2. *Explanatory Data Analysis*
3. *Data Pre-Processing*
4. *Model Building*
5. *Model Evaluation*
6. *Select the best classification model.*

Data Preprocessing

Checked the rating column and it had 10 values instead of 5 so had to clean it through and ensure that our target label was updated as a numeric datatype instead of the object datatype value. Made sure that the string entries were replaced properly.

```
df['Ratings'] = df['Ratings'].replace('2.0 out of 5 stars',2)
df['Ratings'] = df['Ratings'].replace('3.0 out of 5 stars',3)
df['Ratings'] = df['Ratings'].replace('4.0 out of 5 stars',4)
df['Ratings'] = df['Ratings'].replace('5.0 out of 5 stars',5)
df['Ratings'] = df['Ratings'].astype('int')
df['Ratings'].unique()
```

Combined the Review Title & Review Text into a single column as Review.

```
# Now combining the "Review_title" and "Review_text" columns into one single column called "Review"
df['Review'] = df['Review_title'].map(str)+' '+df['Review_text']
df
```

Model Building Phase

Removing the unwanted text, URLs and replacing some of the contracted words.

Removing all Stop Words as well.

Text Processing to remove unwanted punctuations and special characters ¶

```
'''defining a function to replace some of the contracted words to their full form and removing urls and some
unwanted text'''

def decontracted(text):
    text = re.sub(r"won't", "will not", text)
    text = re.sub(r"don’t", "do not", text)
    text = re.sub(r"can't", "can not", text)
    text = re.sub(r"im ", "i am", text)
    text = re.sub(r"yo ", "you ",text)
    text = re.sub(r"doesn’t", "does not",text)
    text = re.sub(r"n\'t", " not", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'t", " not", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'m", " am", text)
    text = re.sub(r"<br>", " ", text)
    text = re.sub(r'http\S+', '', text) #removing urls
    return text

# Lowercasing the alphabets
df['Review'] = df['Review'].apply(lambda x : x.lower())
df['Review'] = df['Review'].apply(lambda x : decontracted(x))

# Removing punctuations from the review
df['Review'] = df['Review'].str.replace('[^\w\s]','')
df['Review'] = df['Review'].str.replace('\n',' ')
```

```
# Removing all the stopwords
stop = stopwords.words('english')
df['Review'] = df['Review'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

Lemmatizing is the process of grouping together the inflected forms of a word so they can be analyzed as a single item. This is quite similar to stemming in its working but differs since it depends on correctly identifying the intended part of speech and the meaning of a word in a sentence. As well as within the larger context surrounding that sentence such as neighboring sentences or even an entire document. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

Model Building Phase

Lemmatization

```
# Defining functiom to convert nltk tag to wordnet tags
def nltk_tag_to_wordnet_tag(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None

# Defining function to lemmatize our text
def lemmatize_sentence(sentence):
    # tokenize the sentence and find the pos_tag
    nltk_tagged = nltk.pos_tag(nltk.word_tokenize(sentence))
    # tuple of (token, wordnet_tag)
    wordnet_tagged = map(lambda x : (x[0], nltk_tag_to_wordnet_tag(x[1])), nltk_tagged)
    lemmatize_sentence = []
    for word, tag in wordnet_tagged:
        if tag is None:
            lemmatize_sentence.append(word)
        else:
            lemmatize_sentence.append(lemmatizer.lemmatize(word,tag))
    return " ".join(lemmatize_sentence)

df['Review'] = df['Review'].apply(lambda x : lemmatize_sentence(x))
```

Dealt with very lengthy comments that may have been detected as an outlier for our classification model.

Removing Outliers

```
# Applying zscore to remove outliers
z_score = zscore(df[['Review_WC']])
abs_z_score = np.abs(z_score)
filtering_entry = (abs_z_score < 3).all(axis = 1)
df = df[filtering_entry]
print("We have {} Rows and {} Columns in our dataframe after removing outliers".format(df.shape[0], df.shape[1]))
```

```
We have 59834 Rows and 6 Columns in our dataframe after removing outliers
```

Model Building Phase

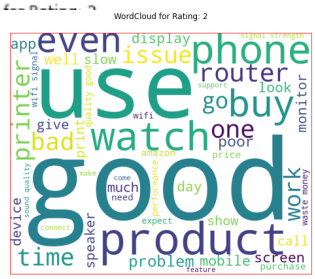
Have analysed the input-output logic with a word cloud and I have word clouded the reviews as per their rating classification. A tag/word cloud is a novelty visual representation of text data, typically used to depict keyword metadata on websites or to visualize free-form text. It's an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance.

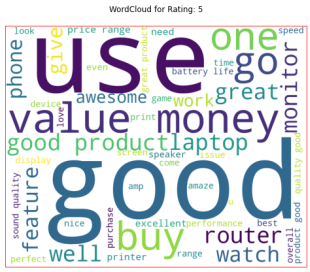
Displaying loud words with Word Cloud information

```
# Getting insight of loud words in each rating
cols = 2
ratings = np.sort(df.Ratings.unique())
rows = len(ratings)//2
if len(ratings) % cols != 0:
    rows += 1
fig = plt.figure(figsize=(15,20))
plt.subplots_adjust(hspace=0.3)
p = 1
for i in ratings:
    word_cloud = WordCloud(height=800, width=1000, background_color="white", max_words=50).generate(' '.join(df.Review[df.Ratings
    axis = fig.add_subplot(rows,cols,p)
    axis.set_title(f"WordCloud for Rating: {i}\n")
    axis.imshow(word_cloud)
    for spine in axis.spines.values():
        spine.set_edgecolor('r')
    axis.set_xticks([])
    axis.set_yticks([])

    plt.tight_layout(pad=5)
    p += 1
plt.show()
```

These are the comments that belong to different rating types so with the help of these word clouds we can see all the frequently used words in each and every rating class. It is observed that 5-star rating comments have mostly positive words while the 1-star rating comments are loaded with negative descriptions.





Model Development and Evaluation

Identification of possible problem-solving approaches (methods)

Checked for missing values in our originally imported dataset and were able to notice NaN values in them.

```
df.isna().sum() # checking for missing values
```

```
Review_title    13342
Review_text     11559
Ratings         13340
dtype: int64
```

We dropped all the NaN values from our data frame since we could afford to lose that much data.

```
print("We have {} Rows and {} Columns in our dataframe before removing NaN".format(df.shape[0], df.shape[1]))
df.dropna(inplace=True)
print("We have {} Rows and {} Columns in our dataframe after removing NaN".format(df.shape[0], df.shape[1]))
```

```
We have 74523 Rows and 3 Columns in our dataframe before removing NaN
We have 60778 Rows and 3 Columns in our dataframe after removing NaN
```

We then checked for the data-type details present in our data frame. Using the info method, we are able to confirm the non-null count details as well as the datatype information. We noticed all the 3 columns showing as object datatype along with our target label.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 60778 entries, 0 to 74522
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Review_title  60778 non-null  object
 1   Review_text   60778 non-null  object
 2   Ratings       60778 non-null  object
dtypes: object(3)
memory usage: 1.9+ MB
```

Model Development and Evaluation

Testing of Identified Approaches (Algorithms):

The complete list of algorithms that were used in training and testing the classification model are listed below:

1. Logistic Regression, 2. Linear Support Vector Classifier, 3. Random Forest Classifier
4. Bernoulli Naïve Bayes, 5. Multinomial Naïve Bayes, 6. Stochastic Gradient Descent Classifier, 7. LGBM Classifier

From all of these above models Random Forest Classifier gave me good performance and I performed a parameter tuning process to further improve the model confidence.

Run and Evaluate the selected model:

Created a classification function that included the evaluation metrics details for the generation of our Classification Machine Learning models. We defined the various classification models mentioned above and assigned them to user-created variables. Then we defined the function that would train and predict the multiclass labels for us along with the evaluation metrics. I did not include the cross-validation part in the function and rather created a separate function for that metrics to evaluate only the best-scored classification models amongst the original list. After calling the classification function we were able to obtain the accuracy score, classification report, and confusion matrix details.

```
# Defining the Classification Machine Learning Algorithms
rf = RandomForestClassifier()
lr = LogisticRegression(solver='lbfgs')
svc = LinearSVC()
bnb = BernoulliNB()
mnb = MultinomialNB()
sgd = SGDClassifier()
lgb = LGBMClassifier()

# Creating a function to train and test the model with evaluation metrics
def BuiltModel(model):
    print('*'*30+model.__class__.__name__+'*'*30)
    model.fit(x_train, y_train)
    y_pred = model.predict(x_train)
    pred = model.predict(x_test)
    accuracy = accuracy_score(y_test, pred)*100
    print(f"ACCURACY SCORE PERCENTAGE:", accuracy)
    # Confusion matrix and Classification report
    print(f"CLASSIFICATION REPORT: \n {classification_report(y_test, pred)}")
    print(f"CONFUSION MATRIX: \n {confusion_matrix(y_test, pred)}\n")
    print("-"*120)
    print("\n")
```

Model Development and Evaluation

```
*****************************RandomForestClassifier*******************************
ACCURACY SCORE PERCENTAGE: 65.95976073953236
CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           1       0.68      0.79      0.73      2197
           2       0.68      0.62      0.65      2222
           3       0.66      0.56      0.61      2203
           4       0.62      0.59      0.61      2269
           5       0.66      0.74      0.70      2143

    accuracy                           0.66     11034
   macro avg       0.66      0.66      0.66     11034
weighted avg       0.66      0.66      0.66     11034

CONFUSION MATRIX:
 [[1746  249  117   36   49]
 [ 423 1374  238  123   64]
 [ 236  271 1235  291  170]
 [  95   96  209 1342  527]
 [  71   41   80  370 1581]]
```

Key Metrics for success in solving the problem under consideration:

The key metrics used here were accuracy_score, cross_val_score, classification report, and confusion matrix. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

Cross-Validation:

Cross-validation helps to find out the overfitting and underfitting of the model. In the cross-validation, the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part is 20% of the full dataset. While running the Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else be used for training data. This way we will get the first estimate of the model quality of the dataset.

In a similar way, further iterations are made for the second 20% of the dataset is held as a holdout set and the remaining 4 parts are used for training data during the process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

Model Development and Evaluation

Cross validation score for best score models

```
# Checking cross-validation score only for those algorithms which are giving us better accuracy.

def cross_val(model):
    print('*'*30+model.__class__.__name__+'*'*30)
    scores = cross_val_score(model,train_features,y, cv = 3).mean()*100
    print("Cross validation score:", scores)
    print("\n")

for model in [lr,svc,sgd,rf,lgb]:
    cross_val(model)
```

```
******************************LogisticRegression******************************
Cross validation score: 63.89885807504078


******************************LinearSVC******************************
Cross validation score: 64.26318651441001


******************************SGDClassifier******************************
Cross validation score: 63.5562805872757


******************************RandomForestClassifier******************************
Cross validation score: 65.06797172376292


******************************LGBMClassifier******************************
Cross validation score: 63.088635127786844
```

Confusion Matrix:

A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e., commonly mislabelling one as another). It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

Model Development and Evaluation

Classification Report::

The classification report visualizer displays the precision, recall, F1, and support scores for the model. There are four ways to check if the predictions are right or wrong:

a. TN / True Negative: the case was negative and predicted negative
b. TP / True Positive: the case was positive and predicted positive
c. FN / False Negative: the case was positive but predicted negative
d. FP / False Positive: the case was negative but predicted positive

Precision: Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a truly positive and false positive. It is the accuracy of positive predictions. The formula of precision is given below: Precision = TP/ (TP + FP) .

Recall: Recall is the ability of a classifier to find all positive instances. For each class, it is defined as the ratio of true positives to the sum of true positives and false negatives. It is also the fraction of positives that were correctly identified. The formula of recall is given below: Recall = TP/(TP+FN)

F1 score: The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy. The formula is: F1 Score = 2(Recall * Precision) / (Recall + Precision) .*

Support: Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

Model Development and Evaluation

Hyperparameter Tuning:

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model. We are not aware of optimal values for hyperparameters that would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as Hyperparameter Tuning. We can do tuning by using GridSearchCV.

GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have the Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

Interpretation of the Results

Starting with univariate analysis, with the help of count plot it was found that the data consists of an equal amount for each rating (i.e., from 1 to 5). Moving further with the removal and replacement of certain terms (like punctuations, extra spaces, numbers, money symbols, smiley etc) as well as removal of stop words. It was evident that the length of review text decreases by a large amount.

This was also depicted by using distribution plots and histograms. With the help of word cloud, it was found that rating 1 consists of words like waste, money, slow, worst, issue, horrible, etc, rating 2 consists of words like problem, issue, bad, poor, slow, etc, and rating 3 consists of a word like a problem, bad, issue, slow, life, average, nice, etc, rating 4 consists of a word like good, value, money, nice, performance, great, better, wonderful, etc and rating 5 consists of words like excellent, must buy, great, perfect, super, awesome, mind-blowing, etc.

Conclusions

Key Findings and Conclusions of the Study:

This research evaluated the rating of a product classification using machine learning and deep learning techniques. Using real data, we compared the various machine learning algorithms' accuracy by performing detailed experimental analysis while classifying the text into 5 categories.

Generally, Random Forest Classification machine learning algorithms have shown a better performance with our real-life data than others, and the most performing models are all ensemble classifiers.

Learning Outcomes of the Study in respect of Data Science:

In this project we were able to learn various Natural Language Processing techniques like lemmatization, stemming, removal of Stop Words, etc. This project has demonstrated the importance of sampling effectively, modeling, and predicting data. Through different powerful tools of visualization, we were able to analyze and interpret different hidden insights about the data. The few challenges while working on this project are:

1. *Imbalanced Dataset.*
2. *Lots of Text data.*

The dataset was highly imbalanced so we balanced the dataset using smote technique. We converted text data into vectors with the help of TfidfVectorizer.

***Feedback**:*

Customers like to see lots of reviews. A single review with a few positive words makes up an opinion, but a few dozen that say the same thing make a consensus. The more reviews, the better, and one study found that consumers want to see at least 40 reviews to justify trusting an average star rating. However, a few reviews are still better than no reviews.

Acknowledgment

I have utilized a few external resources that helped me to complete this project. I ensured that I learn from the samples and modify things according to my project requirement.