Project Presentation

SLIDE-1

What did you do?

I will present my capstone project which is related to the sentiment analysis on Amazon Beauty Products Review.

Sentiment analysis ,which is a subtopics of Natural Language Processing (NLP), has been gradually becoming more and more popular. It is a contextual mining of text which identifies and extracts subjective information in source material and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations.

Companies, especially in ecommerce, also do sentiment analysis to collect and analyze customer feedback about their products in order to maximize their sales and services. Besides that, potential customers prefer to review the opinions of existing customers before they purchase a product or use a service of a company in order to have best product or services over alternatives.

SLIDE-2

Problem Definition

In this project, I assumed that Amazon is our client. The company wants to develop a software tool that will identify the positive and negative words which customers use when they write reviews for the beauty products as their purchase inclination.

For that, they gave their 9 years beauty products' reviews between 2005-2014 and asked us to develop a model which will identify positive and negative words used in the reviews as a component of customer's sentiment towards to the company's beauty products.

According to the customer request, we will build a sentiment analysis model as part of natural language processing, based on their reviews on the beauty product online purchases. Our dataset consists mainly of customers' reviews and ratings.

SLIDE-3

Problem Definition

I find this problem important and interesting. Because the selling rates mostly depend on the reviews and ratings left by the customers which shows how they are satisfied with the product. That is the reason why it becomes crucial to predict whether customers will leave a good or bad rating based on their reviews.

SLIDE-4:

Data

Woman's Beauty Products e-commerce dataset revolving around the reviews written by the customers. It is real commercial data, it has been anonymized, and references to the company in the review text.

In this problem, our predictor variables will be the review text in order to classify the bad and good sentiments about the beauty products.

Dependent variable will be created based on the rates for each review.

As it might be clearly seen, this will be supervised machine learning problem and classification model which uses Natural Language Processing Sentiment Analysis technique will be applied for providing requested application.

SLIDE-5:

Data Retrieval

The data was in Standford Analysis Project webpage. The original data was in a JSON format there. In order to analyze the data, I should change the data format. For that, I import JSON and decode JSON file with using query in order to convert JSON file to csv file format .

SLIDE-6:

Data Information

Amazon beauty products data includes 28798 rows(observations) and 9 columns(feature variables) and its memory usage is 3.4+ MB. In the dataset, we have 7 object, 1 float64 and 1 int64 data types.

Each row corresponds to a customer review, and each column include the variables:

reviewerID : ID of the reviewer, - type: object
asin : ID of the product ,– type: object
reviewerName : name of the reviewer – type: object
helpful : helpfulness of the review,– type: object
reviewText : text of the review – type: object
overall : Rating (1,2,3,4,5)– type: float64
summary : summary of the review – type: object
unixReviewTime : time of the review (unix time) – type: int64
reviewTime : time of the review (raw) – type: object

SLIDE-7:

Data Wrangling – Missing & Duplicate Values

I checked for duplicate values in the dataset. We don't have any duplicate values in the data.

Heatmap for missing values show that we 222 reviewerName missing in the dataset. Since customers don't give their identity, it may not be reliable to make an analysis on their reviews and ratings. I would prefer to drop the missing values from dataset since we have enough observations to conclude a prediction for sentiment analysis.

SLIDE-8:

Data Wrangling

- We concatenated 'reviewText' and 'summary' since both gave the approximately same type of information about product in text format, and later dropped both 'reviewText' and 'summary' columns.
- 'helpful' feature was dropped since we didn't need that column for our model.
- We classified the 'overall' (ratings) as good and bad in order to make sentiment analysis. For that, we dropped observations whose 'overall' columns' values are equal to 3 since that rating group doesn't give an exact opinion about product whether it is good or bad. We created a new column named as 'rate_class' from 'overall' column and converted its' values as 'good' and 'bad'. Later, we dropped 'overall' column.
- In the dataset, 'reviewerID' and 'reviwerName' were used both for identification of customers. We dropped one of them from the dataset. Preferably, I dropped 'reviewerName' since customer names were not standardized and there were lots of different style to represent them in it.
- 'unixReviewTime' was dropped since it has already been represented in 'reviewTime' feature in a more understandable format. Also, 'reviewTime' was converted to datetime data type and a new 'year' column was created to make analysis between other variables in the future work. After that, 'reviewTime' column was also dropped.

SLIDE-9:

Data Wrangling- Rename Columns

We renamed the columns in order to improve practicality/readability of coding:
reviewerID : "customer"
asin : "product"
reviewText: This will be concatenated with "summary" and renamed as "review_text"
overall: "rating_class"
reviewTime: "year"

SLIDE-10:

Descriptive Statistics

In our dataset, we have 2084 reviews which have bad ratings whereas 22425 reviews which have good ratings. That is an imbalanced dataset.

SLIDE-11:

Descriptive Statistics

We have 1340 unique customers and 733 products in this dataset. Each customer averagely gives 18 reviews for products and on the other hand, there is averagely 34 reviews for each product in the website.

SLIDE-12:

Preprocessing Text

Since, text is the most unstructured form of all the available data, various types of noise are present in it and the data is not readily analyzable without any pre-processing. The entire process of cleaning and standardization of text, making it noise-free and ready for analysis is known as text preprocessing.

SLIDE-13:

Preprocessing Text

In this section, I applied the text preprocessing.
Removing tags: Our text contains unnecessary content like HTML tags, which do not add much value when analyzing text. I removed the HTML tags.
Removing accented characters: In text corpus, I have to deal with accented characters\letters. Hence I converted and standardized accented characters/letters into ASCII characters. A simple example would be converting é to e.
Expanding contractions: I converted each contraction to its expanded, original form in order to help with text standardization. Contractions are basically shortened versions of words or syllables. These shortened versions of existing words or phrases are created by removing specific letters and sounds. Examples would be, do not to don't and I would to I'd.

Removing special characters: We used simple regular expressions(regexes) to remove special characters and symbols which are usually non-alphanumeric characters or even occasional numeric characters.

Stemming: Stemming is the process of reducing the words(generally modified or derived) to their word stem or root form. The objective of stemming is to reduce related words to the same stem even if the stem

is not a dictionary word. Stemming is faster than lemmatisation.

Lemmatisation: Lemmatisation is the process of reducing a group of words into their lemma or dictionary form.
Stemming versus Lemmatization:
Meaning of the word – Lemmatization
Fast, if I have large data – Stemming
I have to choose one of them, I can't use both of them together.
Removing stopwords: Words which have little or no significance especially when constructing meaningful features from text are known as stopwords. These are usually words that end up having the maximum frequency if you do a simple term or word frequency in a corpus. Words like a, an, the, and so on are considered to be stopwords. There is no universal stopword list but we use a standard English language stopwords list from NLTK. You can also add your own domain specific stopwords as needed.
Besides: I also did other standard operations like tokenization, removing extra whitespaces, digits, and more advanced operations like spelling corrections, grammatical error corrections, removing repeated characters and text lower casing the text corpus.

Based on the functions which I underlined here and with additional text correction techniques, I built a text normalizer in order to help me to preprocess the text.

SLIDE-14:

Preprocessing Text

After applying text normalizer to 'text' document, I applied tokenizer to create tokens for the clean text.
Then, I saved the dataframe to csv file as a cleaned_dataset.
A clean dataset will allow the machine learning models to learn meaningful features and not overfit on irrelevant noise. After following these steps, I can start using the clean, labelled data to train models in modeling section. Before starting modeling section, I did exploratory data analysis to get insight from the data.

SLIDE-15:

Exploratory Data Analysis – Target Variable

Considering the reviews, 91.5% of them (22425) are classified as good, whereas 8.5% of them (2084) are bad.

SLIDE-16:

Exploratory Data Analysis – Year Feature

In this section, I univariate analysis for target variable, and bivariate analysis for year, review length and text review with target variable in order to get insight about relationship between each other. I used bar chart and with two axis and wordcloud to visualize it. I checked other customer and product variable but did not get insightful info from them.

After 2012, good ratings' percentage is progressing over 90%. Before 2012, only 2009 also shows a slightly rapid increase in good ratings from 87.5% to 90.7%. Besides those, 2011 has the lowest good ratings with 85% overall. As it might be seen in the graph, the overall good rating is progressing between 85% and 93% in beauty products. This is an imbalanced dataset.

SLIDE-17:

Exploratory Data Analysis – Review Length Feature

As it might be seen the graph, the highest percentage of good rating reviews lies between 0-50 bin with 96.4% whereas lowest percentage of good rating reviews lies between 400450 bin with 88.3%. As the review length extends, the good rating tends to decrease slightly. The difference between good rating review percentage with shortest and longest of review length bin is 8.1%. Insightfully, the customers who have complains about the products are more willingly to write longer reviews than other customers who are satisfied with company's products.

SLIDE-18:

Exploratory Data Analysis – Text Review Feature

Good Rating Words

Fixing the rating count value is above 100, the most common 50 words which belong to good rating class are shown in the wordcloud representation. Each of these words define which products what kind of good impression have on the customers. For example, 'mencare' and 'men' words tell male beauty products are more appreciated in the reviews. On the other hand, 'eczema', and 'scar' tell some beauty products are praised for covering them. 'economical' and 'shipping' words might give the insight that products are accepted as reasonably priced and conveniently shipped to the customers.

SLIDE-19:

Exploratory Data Analysis – Text Review Feature

Bad Rating Words

Same standards as I mentioned earlier, the most common 50 words which belong to bad rating class are shown in the wordcloud. Likewise, in good ratings, each of these words define which products what kind of bad impression have on the customers. For example, 'wax' products have mostly used to complain. 'description' word gives insight that the product's usage is not clearly depicted in the description or beauty product has side effects which the description fails to explain.

Controversial Cases The controversial case such as "I was expecting better - negative meaning" or "it was better than my expectation - positive meaning " will be handled in the modelling section via using deep learning technique (Keras with Word2Vec).

SLIDE-20:

Feature Selection

- I defined a threshold, and ignore the words which are the under this threshold. Words shouldn't occur less than

- Apply the PCA

PCA is a Dimensionality Reduction technique that transforms a high dimensional feature space into smaller components which still retain most of the relevant information. The components are such that the first component has the maximum variance in the data, followed by the second component and so on with the constraint that all the components are orthogonal to each other.

- Apply Singular Value Decomposition

SVD (Singular Value Decomposition) in NLP can be used for text mining. When you have a matrix of documents/words, SVD can reduce the dimension of your matrix. In NLP , it is used to study the relation between documents and words.

SLIDE-21:

Modeling – Data Preprocessing

I separated the response variable 'rating class' and feature 'clean text' as y and X respectively.

We split our data in to a training set used to fit our model and a test set to see how well it generalizes to unseen data. I used test size as 25%. Considering imbalanced dataset, I also used stratified sampling in order to have same class proportion in both training and test set.

Train Set Shape :(18381,)

Test Set Shape :(6128,)

SLIDE-22:

Modeling – Selecting the Right Evaluation Metric

As I emphasized the data imbalance earlier, the evaluation of the classifier performance must be carried out using adequate metrics in order to take into account the class distribution and to pay more attention to the minority class. When the positive class is smaller and the ability to detect correctly positive samples is our main focus (correct detection of negatives examples is less important to the problem) we should use precision and recall. For our particular case, based on this thought I used f1 score which is harmonic average of precision and recall as my evaluation metric.

I used also confusion matrix to understand the types of errors our model makes, and which kind of errors are least desirable.

Recall: *It is known as a true positive rate. The number of positives that your model has claimed compared to the actual defined number of positives available throughout the data. TP / (TP+FN)*

Precision: *It is also known as a positive predicted value. This is more based on the prediction. It is a measure of a number of accurate positives that the model claims when compared to the number of positives it actually claims. TP / (TP + FP)*

SLIDE-23:

Modeling – Applying Vectorizing

Raw data, a sequence of symbols cannot be fed directly to the algorithms themselves as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length.

We call **vectorization** the general process of turning a collection of text documents into numerical feature vectors.

I applied three vectorization techniques to the classification models in order to select the highest giving f1 score vectorizer.

Count Vector: Documents are described by word occurrences while completely ignoring the relative position information of the words in the document. The bag of words model represents each text document as a

numeric vector where each dimension is a specific word from the corpus and the value could be its frequency in the document, occurrence (denoted by 1 or 0) *or even weighted values.*

TF-IDF: Common words like 'is', 'the', 'a' etc. tend to appear quite frequently in comparison to the words which are important to a document.
Ideally, what we would want is to down weight the common words occurring in almost all documents and give more importance to words that appear in a subset of documents.
TF-IDF works by penalizing these common words by assigning them lower weights while giving importance to words which appear in a subset of a particular document.

TF = (Number of times term t appears in a document)/(Number of terms in the document)
It denotes the contribution of the word to the document i.e words relevant to the document should be frequent.

IDF = log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in.
Ideally, if a word has appeared in all the document, then probably that word is not relevant to a particular document. But if it has appeared in a subset of documents then probably the word is of some relevance to the documents it is present in.

Hashing vectorizer: It turns a collection of text documents into a scipy.sparse matrix holding token occurrence counts. Hash Vectorizer is designed to be as memory efficient as possible. Instead of storing the tokens as strings, the vectorizer applies the hashing trick to encode them as numerical indexes. The downside of this method is that once vectorized, the features' names can no longer be retrieved.

SLIDE-24:

Modeling – Models

We use training dataset X and labeled dataset y as input. Labeled data set has binary discrete value. Considering all these, I will apply supervised machine learning classification model on the training dataset.
For NLP text data, I used;
- Logistic Regression
- Random Forest
- Naïve Bayes
- XGBoost
- CatBoost

SLIDE-25:

Dummy classifier f-1 score is 0.84.

SLIDE-26:

Count-Vectorizing and Algorithms

See the chart and graph.

SLIDE-27:

TF-IDF and Algorithms

See the chart and graph.

SLIDE-28:

Hashing Vectorizing and Algorithms

See the chart and graph.

SLIDE-29:

Adding Most and Least Common Words to Stopwords List

See the chart and graph.

SLIDE-30:

SMOTE

See the chart and graph.

SLIDE-31:

PCA + SMOTE

PCA is a Dimensionality Reduction technique that transforms a high dimensional feature space into smaller components which still retain most of the relevant information. The components are such that the first component has the maximum variance in the data, followed by the second component and so on with the constraint that all the components are orthogonal to each other.

SLIDE-32:

Truncated SVD + SMOTE

See the chart and graph.

SLIDE-33:

Word2Vec and Simple Neural Network

We created word vectors using Word2Vec and the model has 25026 unique words where each word has a vector length of 100. Then we used these dense vectors - word embeddings - in a simple neural network to predict. In training and validation accuracy graph, the model starts to overfit after 3th epoch. The accuracy for this simple neural network is 0.9235.

Visualization of the words of interest and their similar words using their embedding vectors after reducing their

dimensions to a 2-D space with t-SNE is presented in the graph. Similar words based on gensim's model can be viewed as well.

SLIDE-34:

Conclusion

* In this project, I tried to predict the good and bad words based on the reviews left by the e-customers.
* We applied;
  - Count Vector, TF-IDF, Hashing Vector, Word2Vec
  - Classification Models and Simple Neural Network
  - Adding most and least common words to CountVect,
  - SMOTE,
  - PCA + SMOTE
  - Truncated SVD + SMOTE

SLIDE-35:

Conclusion

*Boosting algorithms were the winners almost each combination.

*I will go with Naïve Bayes with Count Vectorizing in deploying section (f1 score is 0.924423).

*Adding most and least common words to the stopword list didn't have impact on models' performance.

*SMOTE and Linear Dimensionality Reduction techniques decreased the f-1 scores.

SLIDE-36:

Recommendations

* We recommend the client to use the model as it is and give us some time to develop neural network algorithms to get better scores. By the way, prediction time and the size of the data set are also important and need to be considered. Especially neural network algorithms may not be outperforming with the low size data sets.

SLIDE-37:

Recommendations

*Provide a system that user can write their reviews/feedbacks without mistakes such as not accepting the review if it does not satisfy the word/grammar correctness.

*Encourage users to reflect their experience with products via giving incentives.

*Use the reviews as a feedback mechanism, take actions towards them, and let the customers know about the conclusion.

SLIDE-38:

Future Study

*Implementation of Dask library for parallel processing to decrease run time.

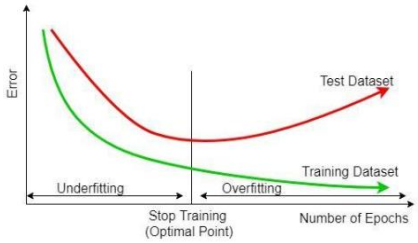
*After decreasing run time, focusing on hyperparameter tuning more.

*Implementation of Deep Learning with different neural network types and different layer combinations to get better results. (Keras+Word2Vec, Fastext)

Additional Notes:

OVERFITTING:

A simple way to check whether the model overfits is to measure the error on the training and test datasets. If the error on training dataset is low and high on test and/or validation dataset, then there likely is overfitting in the model.



Overfitting: Overfitting occurs when the model learns the training set too well. It takes up random fluctuations in the training data as concepts. These impact the model's ability to generalize and don't apply to new data. Hence the model is overfit to the training dataset and will give error when new testing dataset is introduced. High loss and low accuracy is seen in the test dataset.

Below are some of the ways to avoid overfitting:

Keep the model simple: With lesser variables and parameters, the variance can be reduced.

Simplify the model: regularization, controlled by hyperparameter

Use regularization techniques such as LASSO that penalizes certain model parameters if they are likely to cause overfitting.

Gather more training data

Use cross-validation techniques such as k-folds cross-validation.