

Exploratory Data Analysis and Sentiment Analysis on Brazilian E-Commerce Website

A Project

Presented to

Department of Computer and Information Sciences

State University of New York Polytechnic Institute

At Utica/Rome

Utica, New York

In partial fulfillment
of the requirements of the
Master of Science Degree

By

Mihir Patel

May 2020

DECLARATION

I declare that this project is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Mihir Patel

SUNY POLYTECHNIC INSTITUTE

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

Approved and recommended for acceptance as a project in partial fulfillment of the requirements
for the degree of Masters of Science in Computer and information science

DATE

Dr. Bruno R. Andriamanalimanana
Advisor

ABSTRACT

In the past few years, the growth of e-commerce and digital marketing has generated a huge volume of opinionated data. Analyzing those data would provide enterprises with insight for better business decisions. E-commerce web applications are almost ubiquitous in our day to day life, however as useful as they are, most of them have little to no adaptation to user needs, which in turn can cause both lower conversion rates as well as unsatisfied customers. We propose a machine learning system which learns the user behavior from multiple previous sessions and predicts useful metrics for the current session. In turn, these metrics can be used by the applications to customize and better target the customer, which can mean anything from offering better offers of specific products, targeted notifications or placing smart ads.

With recent advances in every field, the need for developing efficient techniques for analytics as well as predictions have increased to larger extend. As the data gets large it becomes difficult for companies to handle such large volume of data, therefore new approaches are developed. Here we work with the dataset from olist e-commerce website taken from year 2016 to 2018. In this work, we study sentiment analysis of product reviews in Portuguese since this dataset contains data from Brazilian supermarkets. Understanding customer sentiments is of paramount importance in marketing strategies today. Not only will it give companies an insight as to how customers perceive their products and/or services, but it will also give them an idea on how to improve their offers. This project attempts to understand the correlation of different variables in customer reviews e-commerce products, and to classify each review whether it recommends the reviewed product or not and whether it consists of positive, negative, or neutral sentiment.

ACKNOWLEDGEMENTS

I take this opportunity with much pleasure to thank all the people who have helped me through the course of our journey towards this report.

I sincerely thank my project advisor Dr. Bruno Andriamanalimanana for his guidance, help and motivation. Apart from the subject of my project, I learned a lot from him, I am sure it will be useful to me in different stages of my life. I would like to express my gratitude to him for providing necessary information regarding the project and also for his constant guidance, supervision, kind co-operation and invaluable support in all aspects.

I would like to express my sincere thanks to Dr. Jorge E. Novillo and Dr. Chengfu for being on the project review committee. I have always enjoyed your classes.

Finally, I would like to express my heartfelt thanks to the God, my parents and colleagues for their blessings and constant support.

Mihir Patel

Table Of Contents

1 Introduction	1
1.1 What is Natural Language Processing?	1
1.2 Machine Learning	2
1.3 What is Sentiment Analysis?	3
1.4 Aim of the project	4
1.5 Scope	4
1.6 Motivation	5
2 Background	6
2.1 Natural Language Processing Overview	6
2.2 Sentiment Analysis Overview	7
2.3 Literature Overview	8
3 Design	9
3.1 Dataset Files	9
3.2 Tools and Technology Used	10
4 Implementation	12
4.1 Databass	12
4.1.1 Data Preprocessing	13
4.2 Data Visualization	16
4.2.1 Time Series Analysis	16
4.2.2 Geospatial Data Analysis	20
4.2.3 Review Score Distribution	22
4.2.4 Customer Satisfaction Analysis	23
4.2.5 Top Products and Sellers	26
4.2.6 Customer Review Analysis	28
4.3 Sentiment Analysis	32
4.3.1 NLP Tasks	32
4.3.2 Feature Extraction Phase	33
4.3.3 Sentiment Classifier	34

	4.3.4 Model Building	35
5 Conclusion		38
	5.1 Accomplishments and lessons learned	38
	5.2 Future Work	39
Appendix		40
	Appendix A	40
	Appendix B	60
References		61

Introduction

1.1 What is Natural Language Processing?

Natural language processing is a subfield of artificial intelligence, which allows computers to learn and understand human languages and natural way of communication. Speech recognition, text generation and language modeling are major challenges and modern research areas in natural language processing. As human beings, we are capable of generating and understanding languages, while computers were invented with capability of understanding a strict set of rules provided by the programmers. The codes and routines that drive a machines lifecycle are in structured form, however the language through which humans communicate is highly unstructured and requires improvisation. Natural language processing thus enables machines to analyze the unstructured data by providing a vast market of modern researches and advancements in machine learning.

Natural language processing is used mostly in every other IT company for even a smallest feature that proves to be very important. Some customer service companies use customer reviews from Twitter and work on the requests. A totally automated task is created with the help of simple sentiment analysis of natural language processing. Tasks of natural language processing range from simple sentiment analysis to natural language generation. There are quite a few categories of natural language processing tasks that are sub divided into tasks like the following:

- Sentiment Analysis - Classification of a text given the input

- Part of Speech Tagging (POS Tag) - Identifying words in a sentence and classifying them into the language grammar like Noun, Verb etc.
- Named Entity Recognition - Identifying a word to be a person's or place's name
- Machine Translation - Translating a given sequence into different language
- Text Summarization - Generating summary text from an entire essay
- Various tasks such as speech recognition, segmentation of speech, language modeling etc. are major areas of modern research [2].

1.2 Machine Learning

Another subfield of artificial intelligence is machine learning. Machine learning involves study of algorithms and patterns using large datasets and apply them on specific data to recognize, classify or predict certain results. Statistical computation is a major base for machine learning. In the modern world, data analysis is severely important for technology giants and even small-scale information technology companies. Machine learning allows these companies to categorize data and make it useful for future use. There are several machine learning models like SVM (support vector machines), Naïve Bayes, random forests etc. that are used for certain tasks like sentiment analysis i.e. our case here. These models can be further improved by training on not only individual

tokens, but also bigrams or tri-grams. This allows the classifier to pick up on negations and short phrases, which might carry sentiment information that individual tokens do not. Of course, the

process of creating and training on n-grams increases the complexity of the model, so care must be taken to ensure that training time does not become prohibitive.[1]

1.3 What is Sentiment Analysis?

Sentiment analysis is the interpretation and classification of emotions (positive, negative and neutral) within text data using text analysis techniques. Sentiment analysis allows businesses to identify customer sentiment toward products, brands or services in online conversations and feedback. Understanding people's emotions is essential for businesses since customers are able to express their thoughts and feelings more openly than ever before. By automatically analyzing customer feedback, from survey responses to social media conversations, brands are able to listen attentively to their customers, and tailor products and services to meet their needs. When reviewing the customer reviews text, let's say product reviews (here in our case), we would want to know which particular aspects or features people are mentioning in a positive, neutral, or negative way. That's where aspect-based sentiment analysis can help, for example in this text: "*The battery life of this camera is too short*", an aspect-based classifier would be able to determine that the sentence expresses a negative opinion about the feature battery life.[3]

1.4 Aim of the project

This project has in primary focus the exploratory data analysis of the E Commerce business keeping in mind the user satisfaction. It utilizes a large dataset used for decision making which is beyond human capabilities. EDA(Exploratory data analysis) is performed on customer attributes dataset and the sentiment analysis is performed on the customer reviews dataset. It aims to increase the sales productivity with understanding the customer sentiments about the products.

1.5 Scope of the project

This project considers the E Commerce market and its operations which affect the customers in buying various products. It explores various aspects of this business like the freight value of the goods, order payment details, Geolocation of the business, the seller information and mainly the customer reviews. All this information considered collectively would contribute in the prosperous business and customer satisfaction. Visualization makes this task very manageable when it comes to dealing with complex dataset. The type of algorithm used here is classification algorithm. By understanding the user sentiments various changes can be implemented to improve the output of the business.

1.6 Motivation

Data Exploration has opened new ways to explore and get the insights of the business. Many organizations have started to implement this method. The organizations are trying to focus particular group of

users to increase the efficiency of the product. When an individual wants to make a decision about buying a product or using a service, they have access to a huge number of user reviews, but reading and analyzing all

of them is a tedious task. Also when an organization wants to benefit by obtaining the public opinion or to market its products, even to identify new opportunities, predict sales trends, or manage its reputation, it needs to deal with an overwhelming number of available customer comments. With sentiment analysis techniques, it is possible to analyze a large amount of available data, and extract opinions from them that may help both customers and organization to achieve their goals. [4]

Background

2.1 Natural Language Processing Overview

The history of natural language processing (NLP) generally started in the 1950s, although work can be found from earlier periods. In 1950, Alan Turing published an article titled "Computing Machinery and Intelligence" which proposed what is now called the Turing test as a criterion of intelligence.

The Georgetown experiment in 1954 involved fully automatic translation of more than sixty Russian sentences into English. The authors claimed that within three or five years, machine translation would be a solved problem. However, real progress was much slower, and after the ALPAC report in 1966, which found that ten-year-long research had failed to fulfill the expectations, funding for machine translation was dramatically reduced. Little further research in machine translation was conducted until the late 1980s when the first statistical machine translation systems were developed.^[22]

2.2 Sentiment Analysis Overview

The basic application of sentiment analysis lies in gathering the opinion of people. Such opinions are precursors to many business

decisions. Similar to the stop-words, sentiment analysis domain depends on list of words that describe the affect of the writer. (Nielson, 2011) explains how ANEW list is used to classify the opinions of users as negative, neutral or positive. (Wang et. al., 2011) applied the sentiment analysis concepts to topics rather than the actual opinions and gleaned how the discussion would follow the sentiment of a topic.

(Pang, 2008) also looked at the user reviews and the possible mistakes that can occur during the data entry when it comes to providing ratings. (Goldberg et. al, 2007, Hopkins et. al, 2007) looked at the opinions of democrat voters and how they felt about the presidential elections. (Bansal et. al., 2008) looked at the long-term aspect of sentiment analysis in the political domain where the voters can get a long-term view of how the politicians act during their tenure. Other projects have as a long-term goal the clarification of politicians' positions. (Jin et. al., 2007) used applied sentiment analysis concepts to detect inappropriate ads. (Cheong et. al., 2011) tackled the problem of finding content related to terrorism. The work looked at the sentiments of civilians and how the twitter data can be harvested to glean such information. More work has been done in the field of twitter harvesting by (Saif et. al., 2012, Agrawal et. al., 2011 and Rosenthal et. al., 2017). Twitter data was also used to build a corpus that could be specific to twitter as explained by (Pak et. al., 2010). Finally, (Cobb et. al., 2013) has applied sentiment analysis to smoking-cessation techniques given certain drug choices.[23]

2.3 Literature Review

- Natural Language Processing, Sentiment Analysis and Clinical Analytics. This paper describes in detail the overview and background of NLP and sentiment analysis. It also discusses the procedural details of all the NLP tasks like Bag-of-words, POS Tagging, etc.
- Predicting Next Shopping Stage Using Google Analytics Data For E-Commerce Applications. This paper proposes a machine learning system which learns the user behavior from multiple previous sessions and predicts useful metrics for the current session.
- An efficient model for sentiment analysis of electronic product reviews in Vietnamese. This paper studies and analyses product reviews in Vietnamese. The final solution is based on Self-attention neural networks.
- Statistical Analysis on E Commerce Reviews, with Sentiment Classification using Bidirectional Recurrent Neural Network. This paper attempts to understand the correlation of different variables in customer reviews on a women clothing e-commerce, and to classify each review whether it recommends the reviewed product or not and whether it consists of positive, negative, or neutral sentiment.

Design of System

This project is divided in two sections i.e.

- (i) Data Preprocessing and EDA
- (ii) Customer reviews and Sentiment Analysis

3.1 Dataset Files

This is quite a vast dataset that comprises of different aspects of the E-Commerce business in Brazil. It includes data about various business existing in the market. It comprises of the data about 100k orders slated from 2016 to 2018. This dataset has been put up on Olist website for public exploration. The choice of language here is obviously Portuguese. While we can easily find a lot of sentiment analysis researches for English, there are barely any works for Portuguese. Portuguese is a unique language and it differs from English in a number of ways. To apply the same techniques that work for English to Portuguese would yield inaccurate results. This has motivated my systematic study and research in sentiment analysis for Portuguese.

Data Source: : <https://www.kaggle.com/olistbr/brazilian-ecommerce>

This data source comprises of files:

- olist_customers_dataset.csv
- olist_geolocation_dataset.csv
- olist_order_items_dataset.csv

- olist_order_payments_dataset.csv
- olist_order_reviews_dataset.csv
- olist_orders_dataset.csv
- olist_products_dataset.csv
- olist_sellers_dataset.csv

3.2 Tools and Technology Used

Jupyter Notebook:

Jupyter Notebook is an open-sourced web-based application which allows to create and share documents containing live code, equations, visualizations, and narrative text. This notebook not only supports Python but also has support for over 40 programming languages. The IDE also includes data cleaning and transformation, numerical simulation, statistical modelling, data visualization, and many others.[24]

Google Colaboratory:

Google provides a research tool, an environment similar to Jupyter Notebook online for researchers to utilize Google's CPU, GPU or TPU powered machines to execute a machine learning code. [26]

NLTK:

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.[25]

Scikit-Learn:

There are several Python libraries which provide solid implementations of a range of machine learning algorithms. One of the best known is Scikit-Learn, a package that provides efficient versions of a large number of common algorithms. Scikit-Learn is characterized by a clean, uniform, and streamlined API, as well as by very useful and complete online documentation. A benefit of this uniformity is that once we understand the basic use and syntax of Scikit-Learn for one type of model, switching to a new model or algorithm is very straightforward.

[27]

Format:

<i>System Elements</i>	<i>Details</i>
Designing Tool	Jupyter Notebooks
Programming Language	Python
Dataset Format	CSV Files

Implementation

4.1 Database

A database is a collection of information that is organized so that it can easily be accessed, managed, and updated. As mentioned above our dataset is about Brazilian E-Commerce which contains all round information regarding customer reviews, Geolocation information, sellers data, orders data, order items data and much more. The data format is csv. It also contains the product ratings, comments and the metadata of the different businesses.

4.1.1 Data Preprocessing

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. This technique is used on the data posted by users which frequently contain missing values, spelling errors and incorrect punctuation. After loading our data, we perform data cleaning on the raw data so that we get proper dataset on which operations can be performed.

- To start with, I loaded the huge dataset with the help of the pandas. As the data format is csv we can interpret the data very

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
olist_order_payments = pd.read_csv('D:\\final sem project\\www\\olist_order_payments_dataset.csv')
olist_order_reviews = pd.read_csv('D:\\final sem project\\www\\olist_order_reviews_dataset.csv')

olist_order_items = pd.read_csv('D:\\final sem project\\www\\olist_order_items_dataset.csv')
olist_order_payments = pd.read_csv('D:\\final sem project\\www\\olist_order_payments_dataset.csv')
olist_order_reviews = pd.read_csv('D:\\final sem project\\www\\olist_order_reviews_dataset.csv')
olist_products = pd.read_csv('D:\\final sem project\\www\\olist_products_dataset.csv')
olist_sellers = pd.read_csv('D:\\final sem project\\www\\olist_sellers_dataset.csv')
```

In [3]: olist_orders.head()

```
Out[3]:
```

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_d
0	e481f51cbdc54678b7cc49136f2d5af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33	2017-10-02 11:07:15	2017-10-04 19:55
1	53c0b2fcb8c7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37	2018-07-26 03:24:27	2018-07-26 14:31
2	47770eb9100c2d0c44946d9c07ec0564	41ce2a54c0b03bf344c3d931a367089	delivered	2018-08-08 08:38:49	2018-08-08 08:55:23	2018-08-08 13:55
3	949d5b44dbf5de918e9c16f97b45f6a	f8b197465ea7920adcdbec7375364d82	delivered	2017-11-18 19:28:06	2017-11-18 19:45:59	2017-11-22 13:35
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8966dbbc4fb7aad2c	delivered	2018-02-13 21:18:39	2018-02-13 22:20:29	2018-02-14 19:46

Figure(4.1)

clearly. This would help in finding the missing data values in it. It would also make easy to look for data values with discrepancies.

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
ght_g', 'product_length_cm', 'product_height_cm', 'product_width_cm']

Sellers: 4 columns
['seller_id', 'seller_zip_code_prefix', 'seller_city', 'seller_state']

In [5]: for name, df in dataframes.items():
        key_count = [col for col in df.columns if 'id' in col or 'code' in col]
        print(f'{name}: {len(key_count)} PKs or FKs')
        print(f'{key_count}\n')
```

Customers: 3 PKs or FKs
['customer_id', 'customer_unique_id', 'customer_zip_code_prefix']

Geolocation: 1 PKs or FKs
['geolocation_zip_code_prefix']

Orders: 2 PKs or FKs
['order_id', 'customer_id']

Items: 4 PKs or FKs
['order_id', 'order_item_id', 'product_id', 'seller_id']

Payments: 1 PKs or FKs
['order_id']

Reviews: 2 PKs or FKs
['review_id', 'order_id']

Products: 1 PKs or FKs
['product_id']

Sellers: 2 PKs or FKs
['seller_id', 'seller_zip_code_prefix']

Figure(4.2)

- Above, we checked how many primary keys and foreign keys are there in each data files. A primary key is used to ensure data in the specific column is unique. You can only set constraints with primary keys, by setting a foreign key to another column which creates a relationship with the column that has the primary key set.
- In the next step, we check in the data files for the null values and we remove them from those data files containing them.

The screenshot shows a Jupyter Notebook with the following code and output:

```
In [7]: #As we can see above there are null data in order, reviews and products datasets.
```

```
In [8]: for name, df in dataframes.items():
         if df.isnull().any().any():
             print(f'Dataset: {name}\n')
             print(f'{df.isnull().sum()}\n')
```

Dataset: Orders

order_id	0
customer_id	0
order_status	0
order_purchase_timestamp	0
order_approved_at	160
order_delivered_carrier_date	1783
order_delivered_customer_date	2965
order_estimated_delivery_date	0
dtype: int64	

Dataset: Reviews

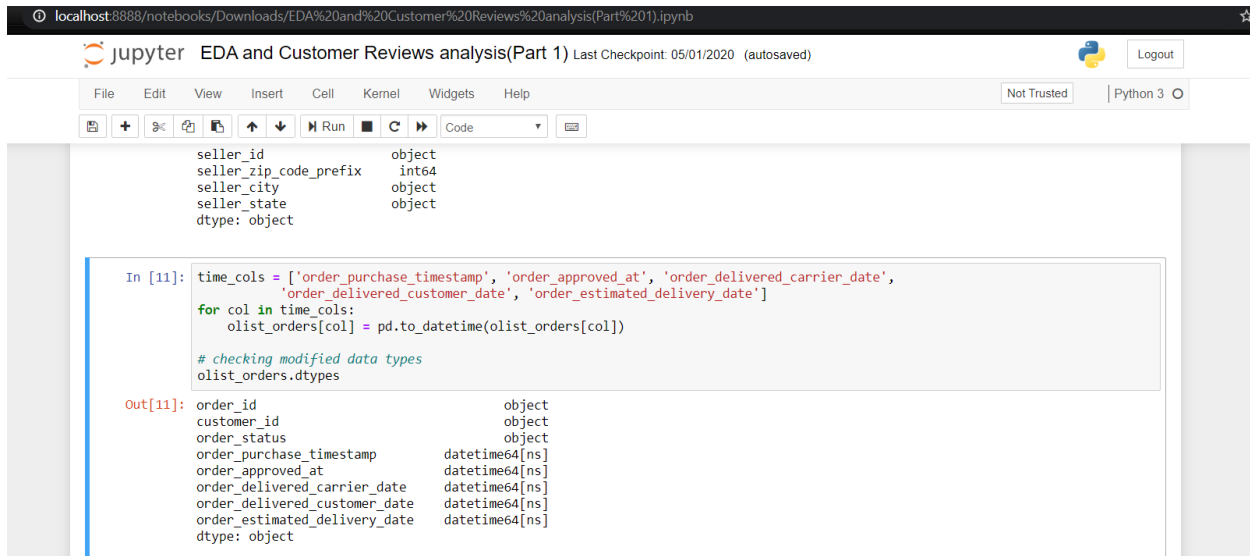
review_id	0
order_id	0
review_score	0
review_comment_title	88285
review_comment_message	58247
review_creation_date	0
review_answer_timestamp	0
dtype: int64	

Dataset: Products

product_id	0
product_category_name	610
product_name_length	610
product_description length	610

Figure(4.3)

- Now we will convert the datatypes of few columns like `order_purchase_timestamp` and `order_delivery_date` to datetime format. This would help us to avoid any further errors regarding the datatypes that would occur during the geolocation analysis.



```
localhost:8888/notebooks/Downloads/EDA%20and%20Customer%20Reviews%20analysis(Part%201).ipynb
jupyter EDA and Customer Reviews analysis(Part 1) Last Checkpoint: 05/01/2020 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3
seller_id      object
seller_zip_code_prefix  int64
seller_city    object
seller_state   object
dtype: object

In [11]: time_cols = ['order_purchase_timestamp', 'order_approved_at', 'order_delivered_carrier_date',
                    'order_delivered_customer_date', 'order_estimated_delivery_date']
         for col in time_cols:
             olist_orders[col] = pd.to_datetime(olist_orders[col])

         # checking modified data types
         olist_orders.dtypes

Out[11]: order_id      object
customer_id  object
order_status  object
order_purchase_timestamp  datetime64[ns]
order_approved_at  datetime64[ns]
order_delivered_carrier_date  datetime64[ns]
order_delivered_customer_date  datetime64[ns]
order_estimated_delivery_date  datetime64[ns]
dtype: object
```

Figure(4.4)

4.2 Data Visualization

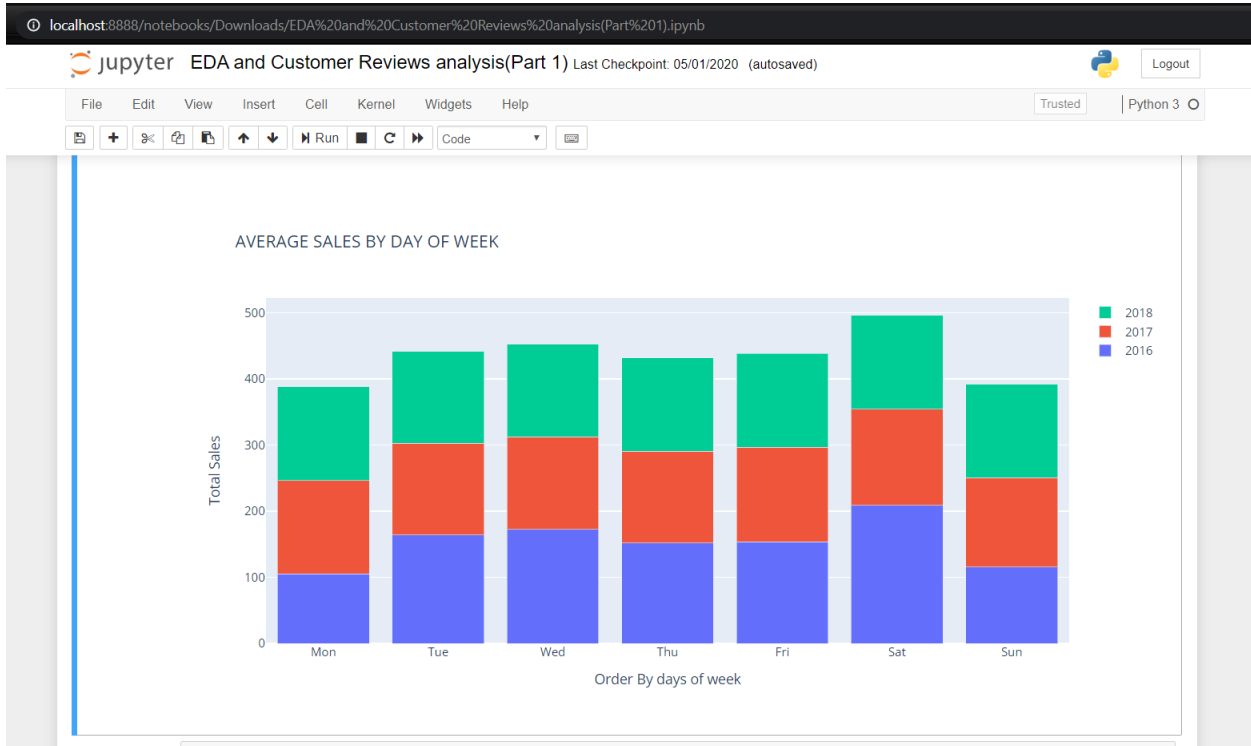
Data Visualization is way to show a complex data in a graphical form which is easy to understand. When one is trying to explore the data or getting acquainted with it then at such a time these plots and graphs are very effective for conveying a clear description of data.

Throughout the process I tried to choose as many different forms of visualizations as possible. As we proceed, I have tried to explain them all along.

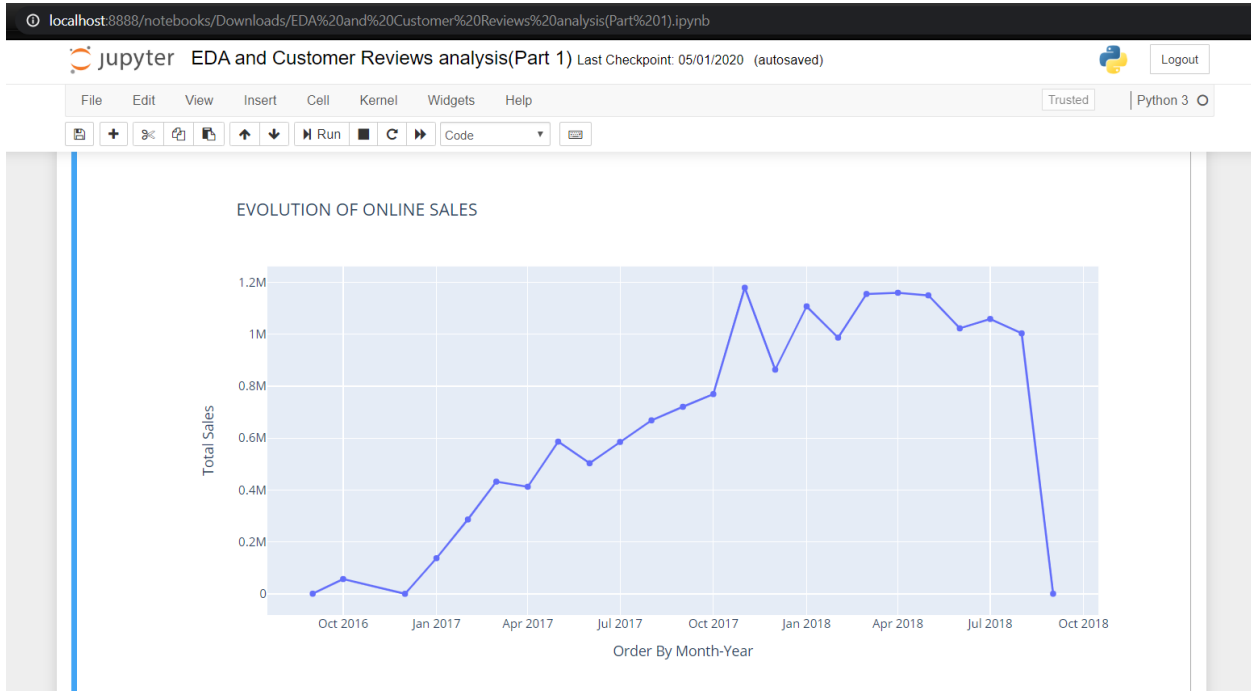
4.2.1 Time Series Analysis

In this section, I tried to include how the business and sales have flourished over the period of time using plotly. It is clear from the visualizations that in the start of the 2016 when business started, the sales were almost negligible.

- To start with, I plotted a graph chart which describes the evolution of online sales. As technology advances people start using more online means than shopping in person.
- The sales recorded at the start of 2016 were negligible as it marks our start of timeline and ends at the end of 2018 as it marks the end of timeline.
- Then we compare the sales of 2017 with sales of 2018 consider last two years using line chart
- Then we compared the sales by days of the week. It is surprising to see that sales were almost similar on all days of week except the Saturday having a little higher sale.



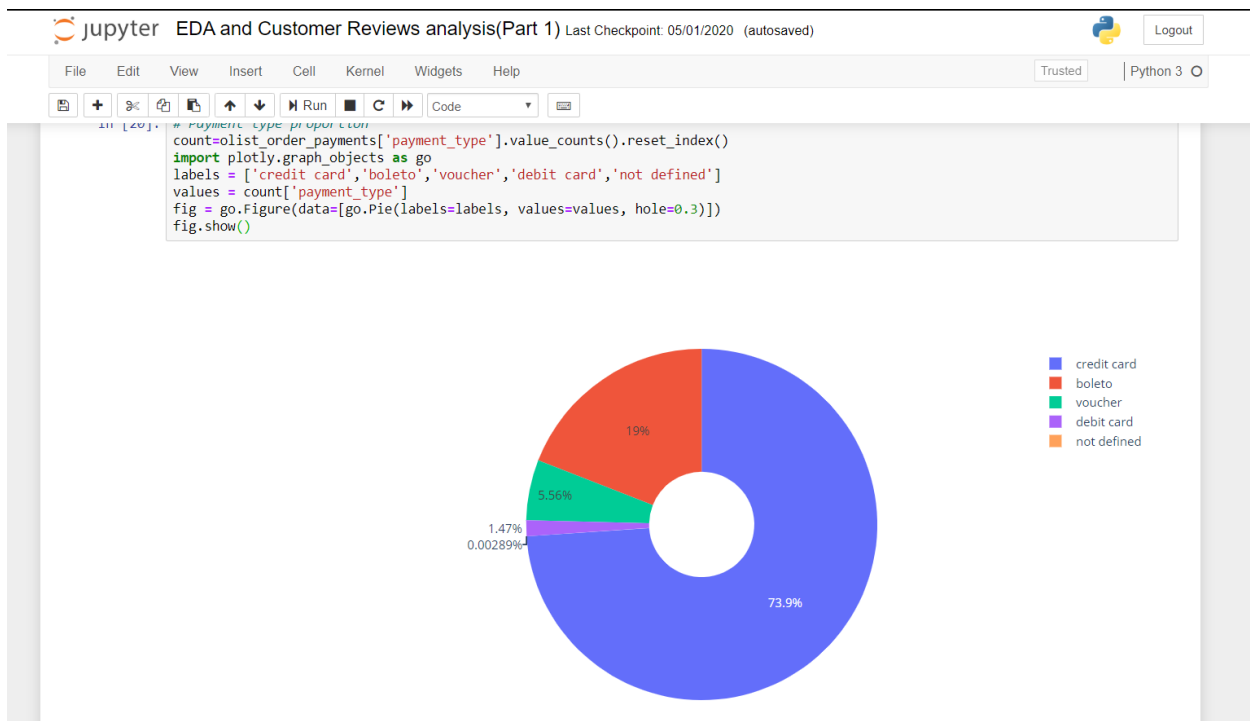
Figure(4.1.1)



Figure(4.1.2)

Payment Type Analysis

- It becomes very important to analyze how the customer choose to pay. From that information we can figure the background of the customers and choose the right target products for particular customer segment.
- We conclude here that people choose to go digital over the period of time and prefer credit cards more over other means of payment.

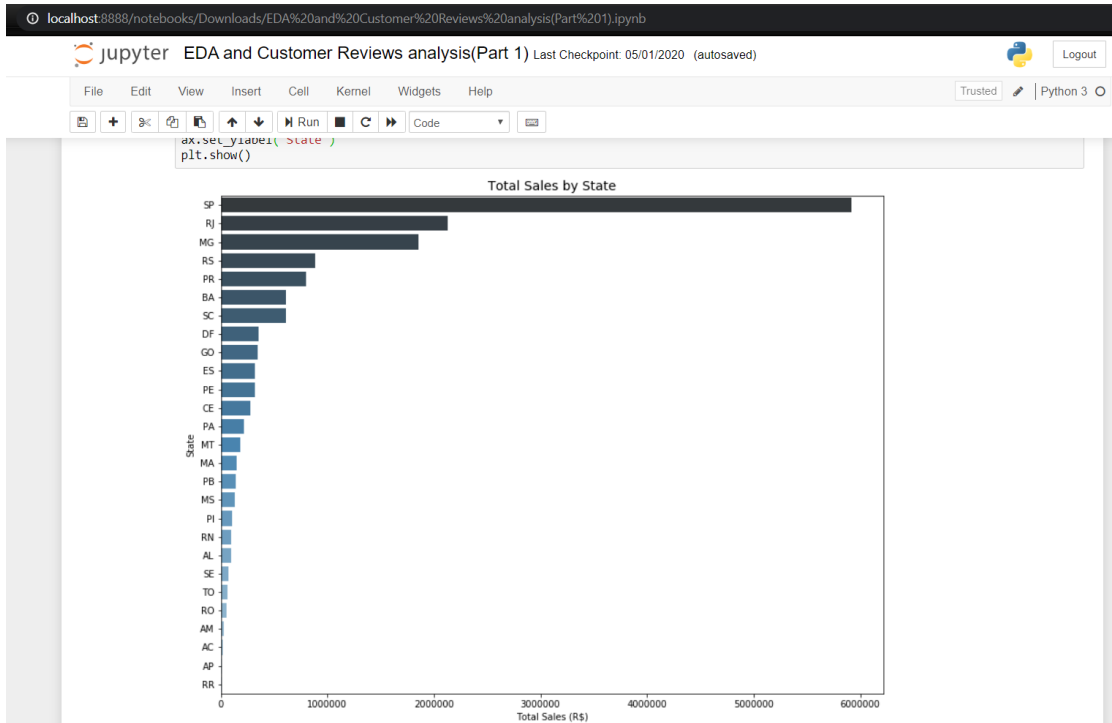


Figure(4.1.3)

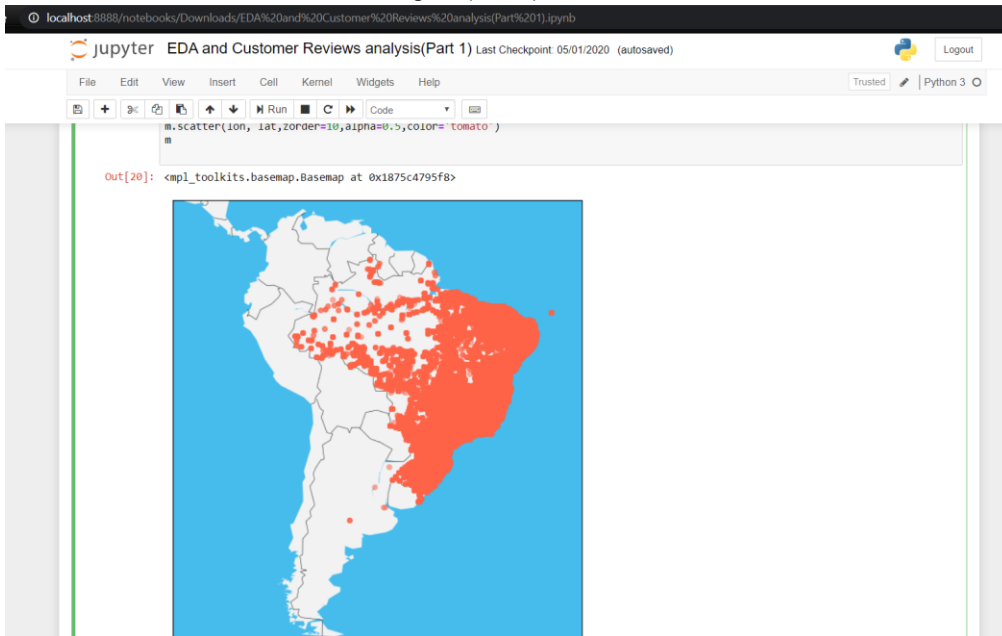
4.2.2 Geospatial data Analysis

The `olist_geolocation.csv` can also be called as geospatial data, i.e data with location information. Such numeric data is used to identify the geographical location of an object. From such spatial data, not only the location but also the length, size and area can be determined. So here I have done some geospatial analysis on the data.

- Firstly, I used bar plot from seaborn library to depict the total sales by States in Brazil. That would make it clear which states have active population that contribute to growing business.
- Secondly I used Basemap toolkit of matplotlib module that is quite challenging to install and to use. It extends matplotlib's functionality by adding geographical projections and some datasets for plotting coast lines and political boundaries, among other things.
- As we can see below the populations from east coast are more active in business contribution than the people living in the western coast.
- From the bar plot we can notice that Sao Paulo has more people than any other states by a big margin. The second is Rio De Janerio.



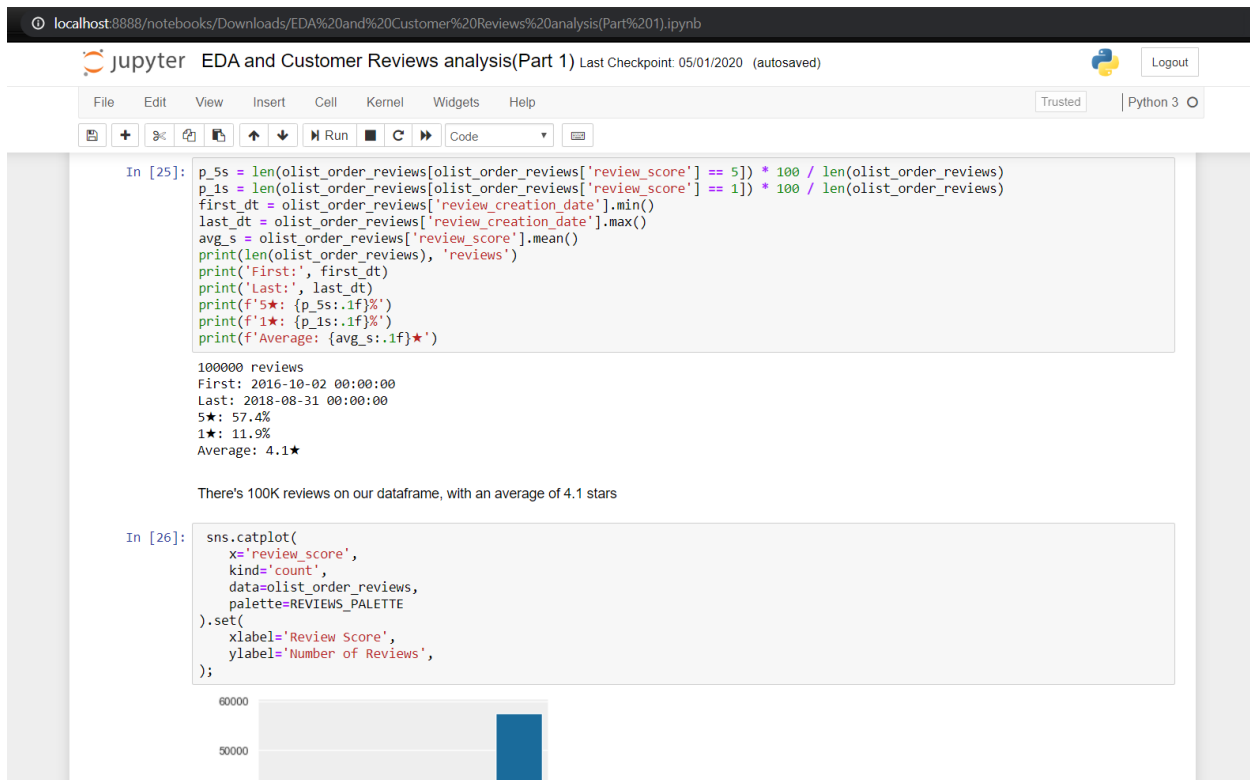
Figure(4.2.1)



Figure(4.2.2)

4.2.3 Review Score Distribution

- In this section we will see how the customers have given the ratings on the products they bought. We will explore their thoughts, sentiments and the rating stars they give along with the comments.
- As we observe that more the customers are happy, the higher ratings they give. Also, there are more negative reviews(1) than the average reviews(2 and 3). There are average reviews of 4.1 stars from 100k reviews in the dataframe. People don't tend to post comments when they are moderately satisfied.



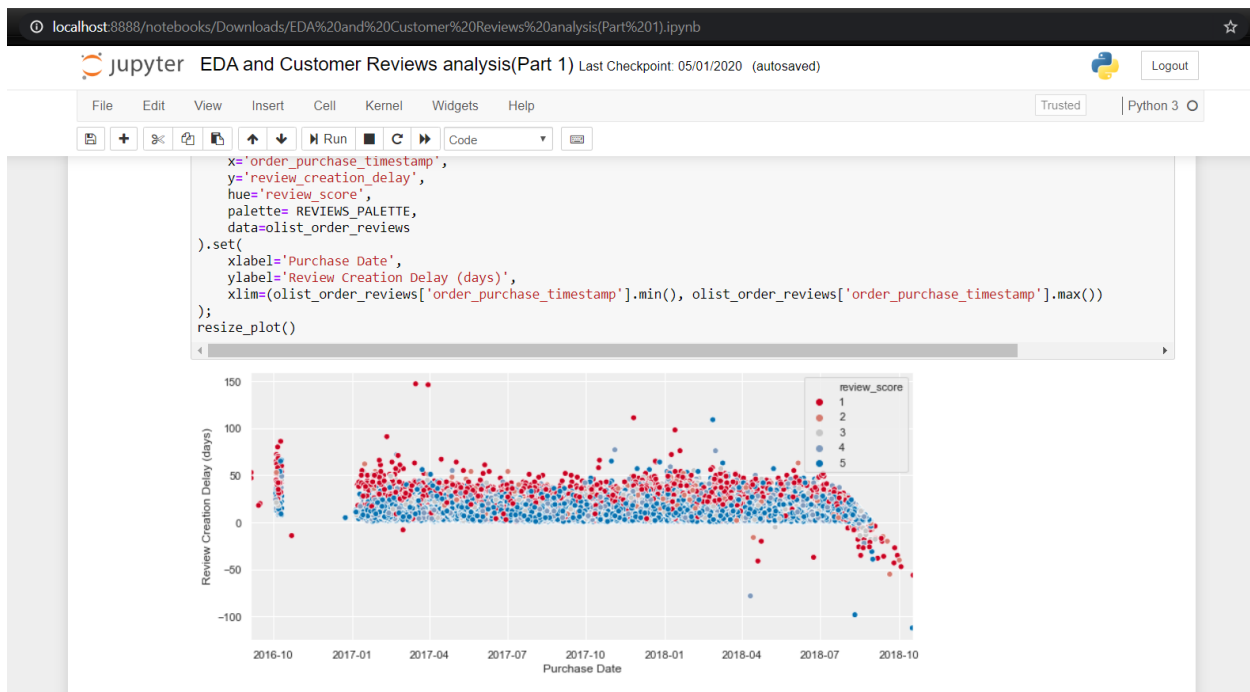
Figure(4.3.1)

4.2.4 Customer Satisfaction Analysis

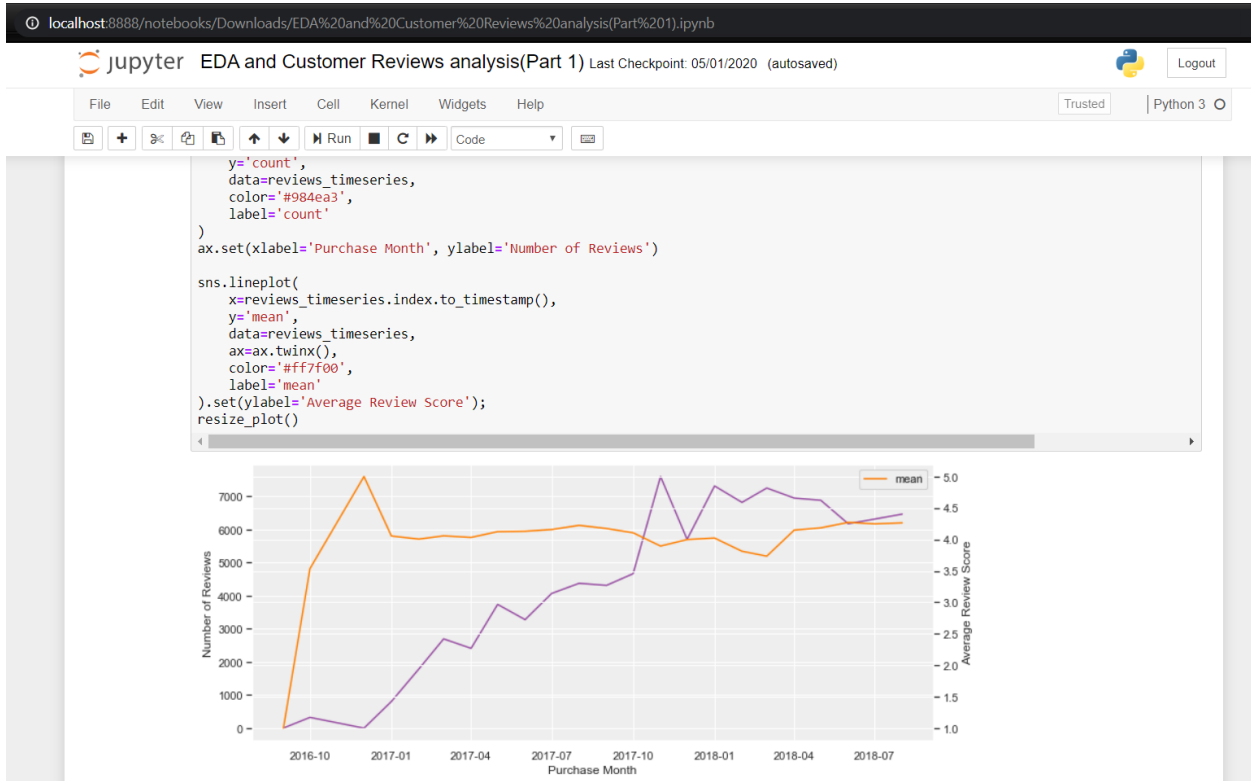
- It is of paramount importance to take care of the satisfaction of the customers. The business cannot flourish without keeping customer satisfaction in the center.
- Mostly, the positive reviews were created the same day they were delivered, but the negative reviews took some time to be posted.
- Generally, it is expected that the reviews are created after the purchases are made. But, surprisingly our dataset has some negative values for review creation delay (reviews posted before the purchase made?). Actually, the products were delivered late after the comments posted.
- For above analysis we used scatter plot. A scatter plot is a data visualization that displays the values of two different variables as points. The data for each point is represented by its horizontal (x) and vertical (y) position on the visualization. Additional variables can be encoded by labels, markers, color, transparency, size (bubbles), and creating 'small multiples' of scatter plots.[5]
- For the next investigation we use line chart. A line chart or line graph is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments. We see the evaluation of review counts with the review score over the period of time. Both the count and the mean score of the reviews increased over time.
- Then we plot a chart which shows response delay compared to the review counts. Here we make an observation that

satisfied customers respond after a while but, the unhappy customers can't control their emotions and respond right away!

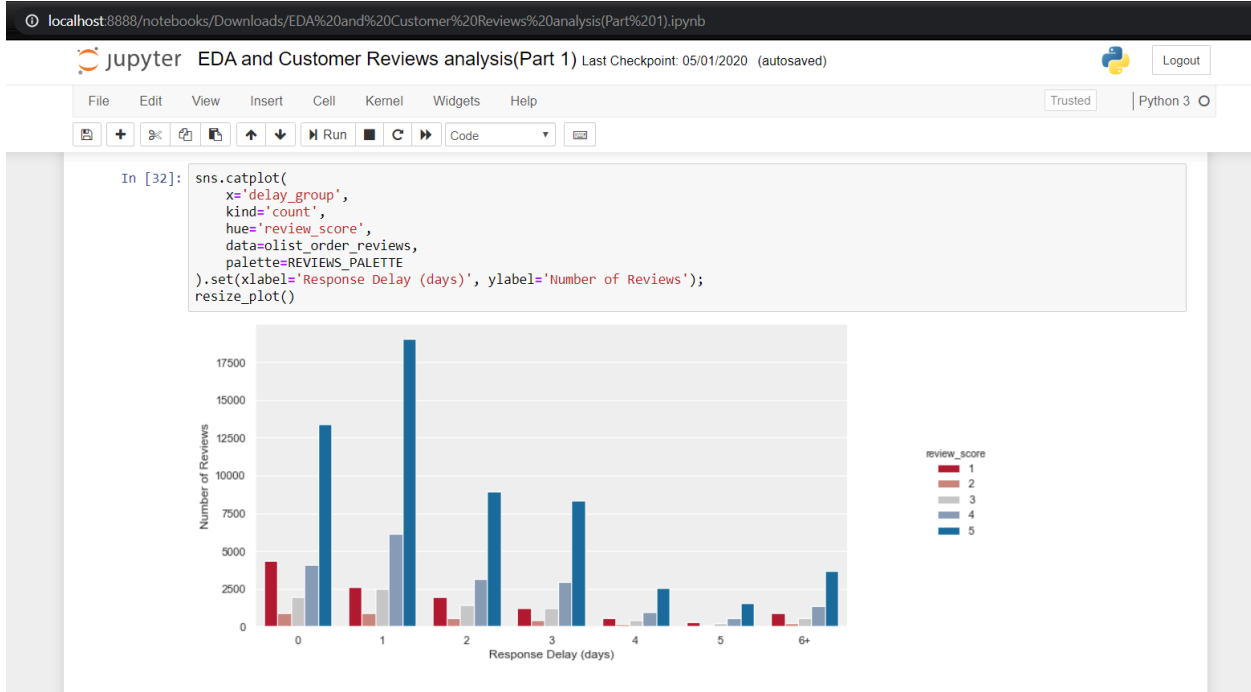
- We also plotted the relationship between the delivery status and the review count to understand better the customer sentiments.



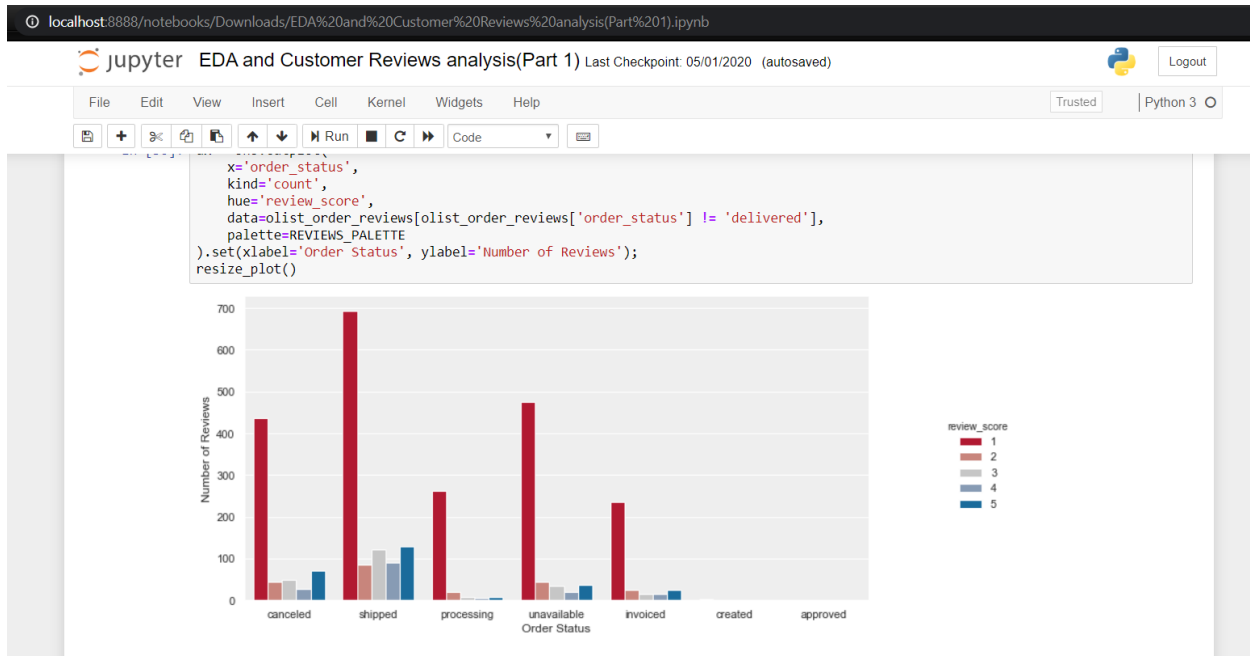
Figure(4.4.1)



Figure(4.4.2)



Figure(4.4.3)

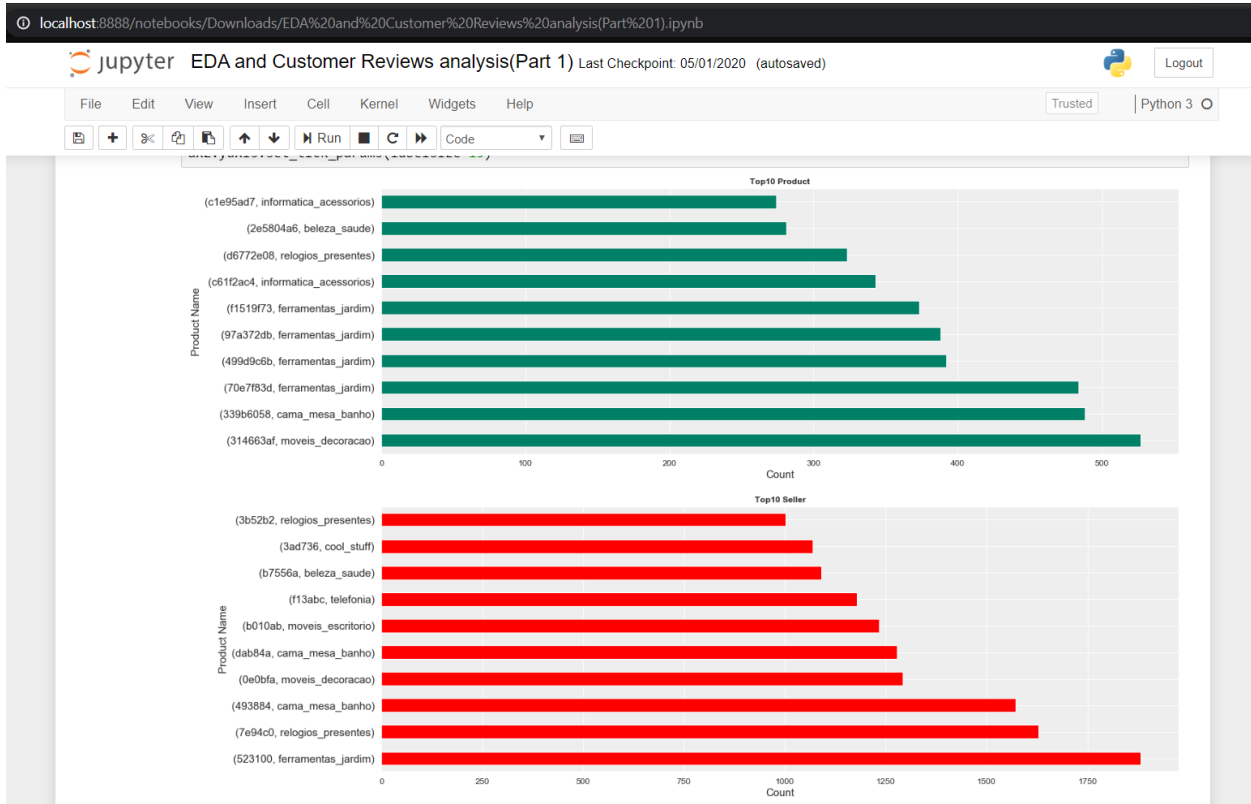


Figure(4.4.4)

4.2.5 Top Products and Sellers

Here we did some preprocessing with the `olist_products` and `olist_sellers` dataset. We had to cut short some ids and only keep some unique characters to make analysis work easy. This would help us to understand how the sellers can focus on customer favorite products to increase the efficiency.

- From the chart, the top 10 products are the from the category 'bed_table_bath'.
- From the seller pie chart, the top selling category is from 'garden tool' one. Then in the last we compare both the top sellers and products to get a clearer view.

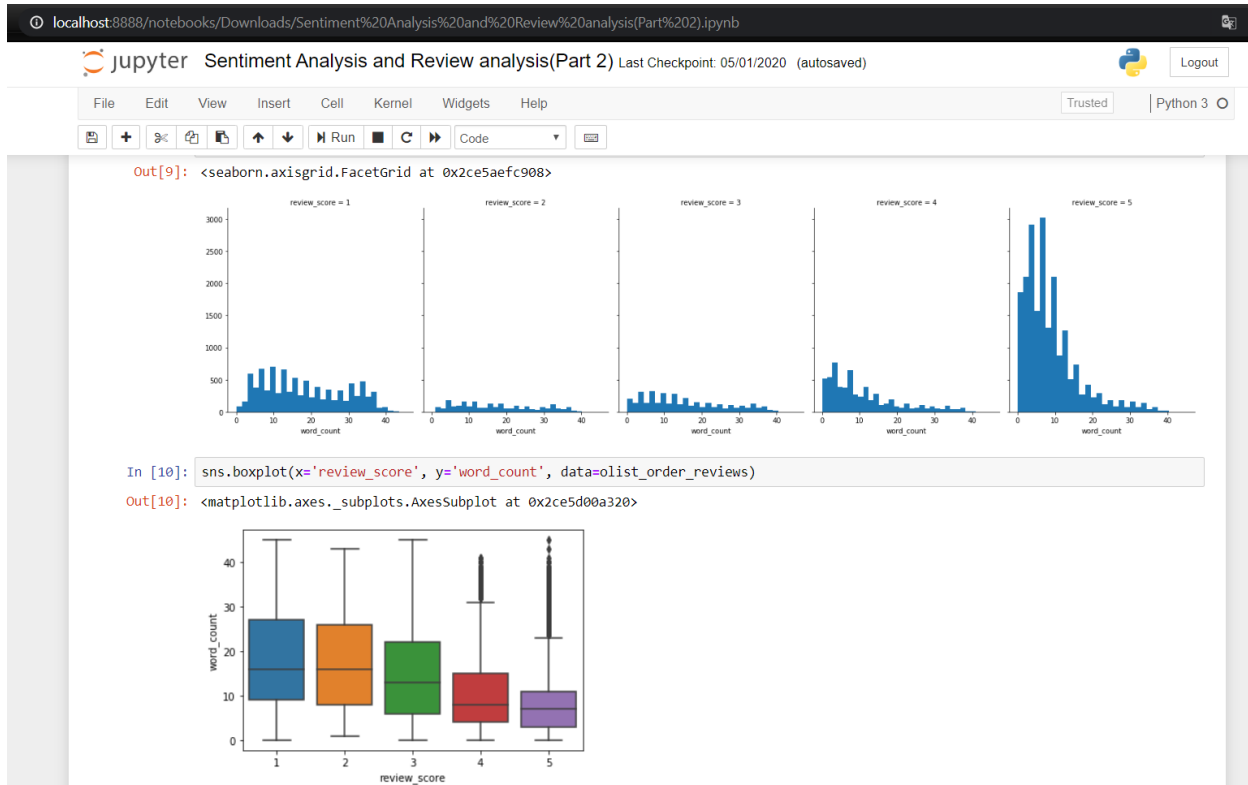


Figure(4.5.1)

4.2.6 Customer Review Analysis

We cannot neglect the prowess of words when we are analyzing the customer reviews. They can convey the emotions so powerfully that no other means can do. For the same reason we need to interpret the words in the reviews as transparently as possible. Portuguese being the language of analysis, adds to the challenge of interpreting it.

- To start with, I pre-processed the data again in the second jupyter notebook. I counted how many words are there per reviews. This would help us understand the customer mentality whether they are satisfied, unsatisfied, happy or angry about the product.
- Then I plotted a grid chart which describes the word counts per each review score. Here we noticed that higher the review score, less words used. People tend to talk more when they are more unsatisfied about the product. That the human nature exactly! This class maps a dataset onto multiple axes arrayed in a grid of rows and columns that correspond to levels of variables in the dataset. The plots it produces are often called “lattice”, “trellis”, or “small-multiple” graphics. [7]
- Then we used box plot system to describe the median value of review scores. A box plot is a method for graphically depicting groups of numerical data through their quartiles. The box extends from the Q1 to Q3 quartile values of the data, with a line at the median (Q2). The whiskers extend from the edges of box to show the range of the data. An important observation here is that lower review score has higher median count than higher review score.



Figure(4.6.1)

- Then after I made one of the important analysis of the customer reviews by using wordcloud. I created a wordcloud containing the summary of all comments in each review scores. I have tried to show below the snippets of reviews for 1 and 5 star ratings.
- A cloud which is filled with lots of words in different sizes, where the sizes of the words represents the frequency, or the importance of each word is known as WordCloud. This tool is generally used for exploring the text data.

- We start with removing the numbers and turning the words in lower cases which would make it easy for our sentiment analysis. We also remove punctuation marks that are unnecessary in the process.
- Thereafter we perform tokenization. Word tokenization is the process of breaking the paragraphs into words.
- Then we removed the stopwords. The nltk package has inbuilt list of stopwords for Portuguese. These words are being filtered out from tokens for further process.
- Then we perform stemming for text normalization. Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language.[8]

The screenshot shows a Jupyter Notebook interface with the following code in a cell:

```

except (TypeError, NameError): # unicode is a default on python 3
    pass
text = unicodedata.normalize('NFD', text)
text = text.encode('ascii', 'ignore')
text = text.decode("utf-8")
return str(text)

In [26]: #Removing numbers
olist_order_reviews.review_comment_message = olist_order_reviews.review_comment_message.str.replace('\d+', ' ')

#Lower cases.
olist_order_reviews.review_comment_message = olist_order_reviews.review_comment_message.apply(lambda x: " ".join(x.lower() for x

#Removing punctuation
olist_order_reviews.review_comment_message = olist_order_reviews.review_comment_message.str.replace('[^\w\s]', ' ')

#Removing stopword
olist_order_reviews.review_comment_message = olist_order_reviews.review_comment_message.apply(lambda x: " ".join(x for x in x.sp

#Removing accentuation
olist_order_reviews.review_comment_message = olist_order_reviews.review_comment_message.apply(strip_accents)

#Tokenize
olist_order_reviews.review_comment_message = olist_order_reviews.apply(lambda row: word_tokenize(row['review_comment_message']),

#Stemming
olist_order_reviews.review_comment_message = olist_order_reviews.review_comment_message.apply(lambda x: " ".join([stemmer.stem(w

```

After preprocessing the reviews, deriving the new word count and median review score.

Figure(4.3.1)

4.3.2 Feature Extraction Phase:

In the text classification we can't directly use the text for our model, therefore we need to convert these texts into some numbers or vectors of numbers. So for this purpose we use bag-of-words (BoW) to extract these features from the text. This BoW is used to convert the text into matrix of occurrence of words within a document and basically concerns whether the given word occurred within the document or not. For this purpose, we use two techniques to check if we get a better score. They are:

- Count Vectorization: This is one of the efficient techniques to put words into our model. Count Vectorization involves counting the number of occurrences each words appears in a document. Python's Sci-kit learn library has a tool called CountVectorizer to accomplish this.[9]
- TF-IDF: TF-IDF stands for *term frequency-inverse document frequency*. It highlights a specific issue which might not be too frequent in our corpus but holds great importance. The TF-IDF value increases proportionally to the number of times a word appears in the document and decreases with the number of documents in the corpus that contain the word. It is composed of 2 sub-parts, which are:[10]
 - Term Frequency (TF)
 - Inverse Document Frequency(IDF)

Term frequency specifies how frequently a term appears in the entire document. It can be thought of as the probability of finding a word within the document.

The inverse document frequency is a measure of whether a term is rare or frequent across the documents in the entire corpus.

4.3.3 Sentiment Classifier

- A Sentiment classifier is learned from the labels that we pass to train our model, usually they are the positive reviews and the negative reviews. The classification in our case is being done using Logistic Regression model.
- in classification, naive Bayes converges quicker but has typically a higher error than logistic regression. On small datasets its preferable to try out Naive Bayes, but as the training set size grows, we likely get better results with logistic regression. This is the reason we chose logistic regression because of our quite a large dataset.^[11]
- We consider here reviews with 1 and 5 scores. Thereby we only consider good and bad reviews. So for convenience, we map scores 2 and 3 to 1 and the score 4 to 5.

4.3.4 Model Building

- For analysing the performance of the model we tend to divide the dataset into training set and a test set. The dataset is being split using the `train_test_split()` function, where three parameters are being passed features, target and the test_set size ,along with this we also use `random_state` so that we can select the records randomly.
- As we can see from the feature extraction phase, there is only small separation between review score 1 and 5. This the reason being, we chose logistic regression as our algorithm. Also, the reviews of both score are seen more separated using TF-IDF than using Count Vectorizer.
- We apply here PCA(Principal Component Analysis) to reduce the size of the feature vectors. It would also help us in better visualization of the words distribution.
- PCA is used to remove the least beneficial features so we have a smaller data set, but without losing too much predictive power. PCA can also improve accuracy if there are chances of overfitting.
- Logistic Regression is one of the useful algorithms for text classification problems. We import the module for Logistic Regression and the use the logistic regression classifier object
- Thereafter, by using the `fit()` function we fit our model on the train set and performed prediction on the test set where I got an accuracy of 89.06% using TF-IDF and 88.53% using Count vectorizer.

- Then using the scatterplot I tried to plot the results of PCA which would clearly show the separation distance between feature vectors.
- At last, I created a visualization where, 20 significant positive and negative review words were plotted using the bar plot; after implementing the model.

The screenshot shows a Jupyter Notebook window titled "Sentiment Analysis and Review analysis(Part 2)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a code editor area. The code in the notebook is as follows:

```
In [39]: from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_count, y_count, test_size = 1/4, random_state = 42)

In [40]: logreg_vector = LogisticRegression()
logreg_vector.fit(X_train, y_train)

C:\Users\mihir\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
FutureWarning)

Out[40]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='warn', n_jobs=None, penalty='l2',
random_state=None, solver='warn', tol=0.0001, verbose=0,
warm_start=False)

In [41]: y_pred = logreg_vector.predict(X_test)

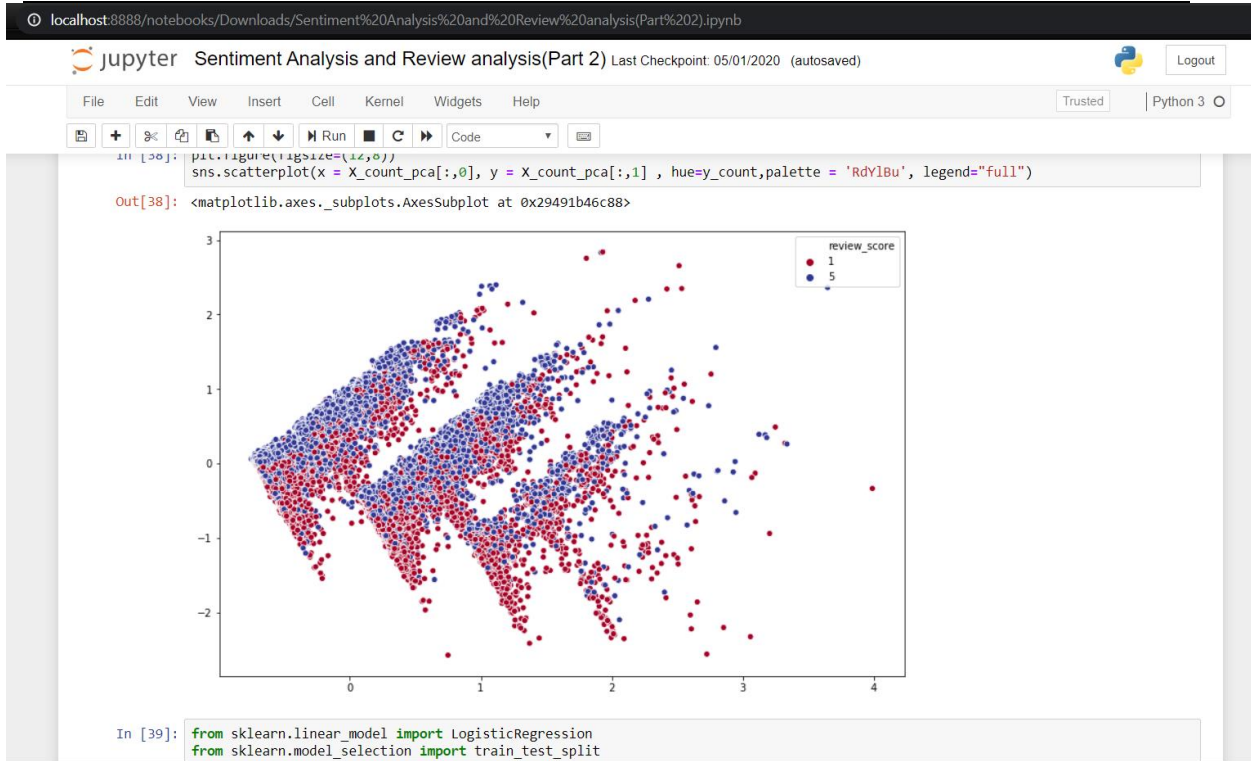
In [42]: score = logreg_vector.score(X_test, y_test)
print(score)
0.8853338442379538

In [43]: from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)

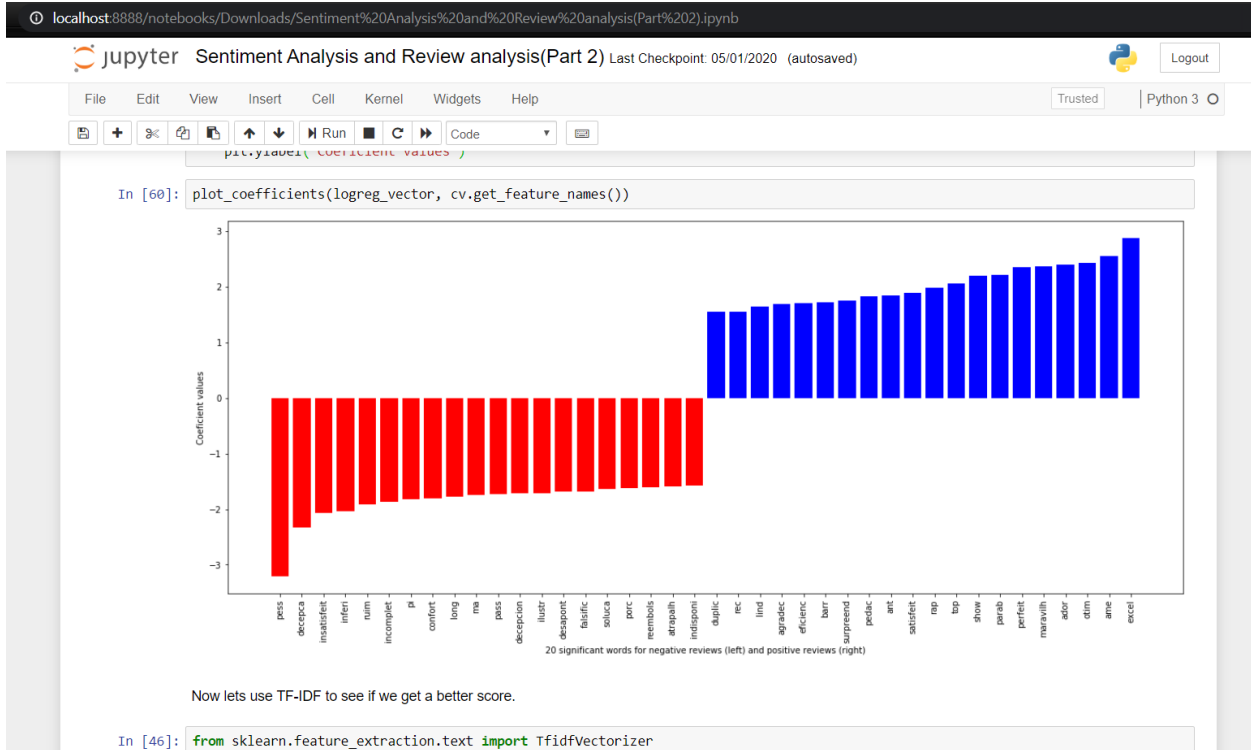
[[3088  692]
 [ 505 6154]]

In [59]: def plot_coefficients(classifier, feature_names, top_features=20):
coef = classifier.coef_.ravel()
```

Figure(4.3.2)



Figure(4.3.3)



Figure(4.3.4)

Conclusion

5.1 Accomplishments and lessons learned

- As a part of this project, I mainly understood the importance of the EDA(Exploration data analysis). Data Visualizations can help to get the better insights of the business which would be difficult to grasp otherwise. I got a deep understanding of the text classification algorithms and requirements of creating an accurate model.
- After performing all the desired tasks, it was concluded that pre-processing tasks are very crucial in achieving the results. The preparation of the dataset undoubtedly required a thorough and specific analysis, requiring research, testing and validation of the most varied, always seeking to improve the results obtained and follow the best practices related to textual analysis. Concepts of Regular Expressions, Stemming, Stop Words removal and other unwanted words were applied.
- From my study and research, I found that Logistic Regression is by far the fastest model and best performing. SVM can be considered but high computational cost would hinder the results.
- However, the performance of these models could be improved by better defining the sentiment of each comment, disengaging it from the critique score as this number often does not reflect the reality of the sentiment shown by the customer. In addition, as the dataset comes from a relationship between user and internet, some slang

and characteristic abbreviations were present, making it difficult to understand the algorithm for proper classification.

5.2 Future Work

I think the customers should be requested to be as transparent as possible in the process of reviewing the products. As a result, the review score doesn't reflect the exact emotions of the customer. After performing extensive exploring and visualizations, I can also include product recommender system. It would be based on both item and user collaborative filtering. Many companies have should start focusing in this area which would help understand the customers behind the screens. The tremendous amounts of data being generated today should be manipulated in a correct way to maximize the profits. The continuing refinement of sentiment analysis technology will increase the technology's accessibility and functionality for small- to medium-sized business owners.

Appendix A

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.pyplot import *
%matplotlib inline
from matplotlib.gridspec import GridSpec
import missingno as msno
import re
from collections import Counter
import string
from PIL import Image
import sys
import time
import sys
import math
import itertools
import geopandas as gpd
import json
from wordcloud import WordCloud
import warnings
warnings.filterwarnings('ignore')

olist_customer = pd.read_csv('D:\\final sem
project\\www\\olist_customers_dataset.csv')
olist_geolocation = pd.read_csv('D:\\final sem
project\\www\\olist_geolocation_dataset.csv')
```

```
olist_orders = pd.read_csv('D:\\final sem  
project\\www\\olist_orders_dataset.csv')
```

```
olist_order_payments = pd.read_csv('D:\\final sem  
project\\www\\olist_order_payments_dataset.csv')  
olist_order_reviews = pd.read_csv('D:\\final sem  
project\\www\\olist_order_reviews_dataset.csv')
```

```
olist_order_items = pd.read_csv('D:\\final sem  
project\\www\\olist_order_items_dataset.csv')  
olist_order_payments = pd.read_csv('D:\\final sem  
project\\www\\olist_order_payments_dataset.csv')  
olist_order_reviews = pd.read_csv('D:\\final sem  
project\\www\\olist_order_reviews_dataset.csv')  
olist_products = pd.read_csv('D:\\final sem  
project\\www\\olist_products_dataset.csv')  
olist_sellers = pd.read_csv('D:\\final sem  
project\\www\\olist_sellers_dataset.csv')
```

```
olist_orders.head()
```

#Data Preprocessing

```
dataframes = {  
    'Customers': olist_customer,  
    'Geolocation': olist_geolocation,  
    'Orders': olist_orders,  
    'Items': olist_order_items,  
    'Payments': olist_order_payments,  
    'Reviews': olist_order_reviews,  
    'Products': olist_products,  
    'Sellers': olist_sellers
```

```
}  
  
for name, df in dataframes.items():  
    print(f'{name}: {len(df.columns)} columns')  
    print(f'{list(df.columns)}\n')  
  
for name, df in dataframes.items():  
    key_count = [col for col in df.columns if '_id' in col or 'code' in col]  
    print(f'{name}: {len(key_count)} PKs or FKs')  
    print(f'{key_count}\n')  
  
for name, df in dataframes.items():  
    print(f'{name:<12}- {df.isnull().any().any()}')  
  
for name, df in dataframes.items():  
    if df.isnull().any().any():  
        print(f'Dataset: {name}\n')  
        print(f'{df.isnull().sum()}\n')  
  
for name, df in dataframes.items():  
    print(f'Dataset: {name}\n')  
    print(f'{df.dtypes}\n')  
  
time_cols = ['order_purchase_timestamp', 'order_approved_at',  
'order_delivered_carrier_date',  
             'order_delivered_customer_date', 'order_estimated_delivery_date']  
for col in time_cols:  
    olist_orders[col] = pd.to_datetime(olist_orders[col])
```

```
olist_orders.dtypes
```

Time Series Analysis

```
olist_orders['order_purchase_year'] = \  
olist_orders['order_purchase_timestamp'].apply(lambda x: x.year)
```

```
olist_orders['order_purchase_month'] = \  
olist_orders['order_purchase_timestamp'].apply(lambda x: x.month)
```

```
olist_orders['order_purchase_dayofweek'] = \  
olist_orders['order_purchase_timestamp'].apply(lambda x: x.dayofweek)
```

```
fig, ax = plt.subplots(figsize=(10, 5))  
ax = sns.countplot(x='order_purchase_year', data=olist_orders,  
palette='Blues_d')
```

```
ncount = len(olist_orders)  
for p in ax.patches:  
    x=p.get_bbox().get_points()[:,0]  
    y=p.get_bbox().get_points()[1,1]  
    ax.annotate('{:.1f}%'.format(100.*y/ncount), (x.mean(), y),  
                ha='center', va='bottom', size=12)
```

```
ax.set_title('Amount of Online Order by Year', size=14)  
ax.set_ylabel('Orders')  
ax.set_xlabel('Year')
```

```
plt.tight_layout()
```

```
plt.show()
```

```
min_order_date = olist_orders['order_purchase_timestamp'].min()
max_order_date = olist_orders['order_purchase_timestamp'].max()
print(f'We have orders from {min_order_date} to {max_order_date}')
```

```
df_orders_items = olist_orders.merge(olist_order_items, on='order_id',
how='inner')
```

```
df_orders_items['total_sales'] = df_orders_items['price'] +
df_orders_items['freight_value']
```

```
df_sales = df_orders_items.groupby(['order_purchase_year',
'order_purchase_month'],
as_index=False).sum()
df_sales = df_sales.loc[:, ['order_purchase_year', 'order_purchase_month',
'total_sales']]
```

```
df_sales_2016 = df_sales[df_sales['order_purchase_year']==2016]
df_sales_2017 = df_sales[df_sales['order_purchase_year']==2017]
df_sales_2018 = df_sales[df_sales['order_purchase_year']==2018]
```

```
df_sales_customer = df_orders_items.merge(olist_customer,
on='customer_id', how='inner')
```

```
df_sales_state = df_sales_customer.groupby(['customer_state'],
as_index=False).sum().iloc[:, np.c_[0, -2, -3]][0]]
```

```
df_sales_state.sort_values(by='total_sales', ascending=False, inplace=True)

fig, ax = plt.subplots(figsize=(13, 10))
sns.barplot(x='total_sales', y='customer_state', data=df_sales_state, ci=None,
            palette='Blues_d')
ax.set_title('Total Sales by State', size=14)
ax.set_xlabel('Total Sales (R$)')
ax.set_ylabel('State')
plt.show()

# Creating a year-month column for correct sorting the data
df_sales['order_purchase_month'] =
df_sales['order_purchase_month'].astype(str).\
apply(lambda x: '0' + x if len(x) == 1 else x)
df_sales['month_year'] = df_sales['order_purchase_year'].astype(str) + '-' + \
df_sales['order_purchase_month'].astype(str)

# Changing the datatype of this new column
df_sales['order_purchase_month'] =
df_sales['order_purchase_month'].astype(int)

import plotly.graph_objects as go

x1=df_sales['month_year']
```

```
y1=df_sales['total_sales']
```

```
fig = go.Figure()
```

```
fig.add_trace(go.Scatter(x=x1, y=y1,mode='lines+markers'))
```

```
fig.update_layout(title_text='EVOLUTION OF ONLINE SALES')
```

```
fig.update_xaxes(title_text='Order By Month-Year')
```

```
fig.update_yaxes(title_text='Total Sales')
```

```
fig.show()
```

```
import plotly.express as px
```

```
df_sales17_18=df_sales[df_sales['order_purchase_year']!=2016]
```

```
fig =px.line(df_sales17_18, x='order_purchase_month', y='total_sales',
```

```
color='order_purchase_year')
```

```
fig.update_layout(title_text='SALES EVOLUTION OVER THE LAST TWO  
YEARS')
```

```
fig.update_xaxes(title_text='Order By Month')
```

```
fig.update_yaxes(title_text='Total Sales')
```

```
fig.update_layout(xaxis = dict(tickmode = 'array',tickvals =
```

```
[0,1,2,3,4,5,6,7,8,9,10,11,12],ticktext =['', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',  
'Jul', 'Aug', 'Sep','Oct', 'Nov', 'Dec']))
```

```
fig.show()
```

```
df_sales_dow_mean = \
```

```
df_orders_items.groupby(['order_purchase_year',
'order_purchase_dayofweek'],as_index=False).mean().iloc[:, np.c_[0, 1, 6][0]]
import plotly.graph_objects as go
```

```
Days=df_sales_dow_mean['order_purchase_dayofweek']
df_dow_2016=df_sales_dow_mean[df_sales_dow_mean['order_purchase_year']==2016]
df_dow_2017=df_sales_dow_mean[df_sales_dow_mean['order_purchase_year']==2017]
df_dow_2018=df_sales_dow_mean[df_sales_dow_mean['order_purchase_year']==2018]
```

```
fig = go.Figure(data=[go.Bar(name='2016', x=Days,
y=df_dow_2016['total_sales']),go.Bar(name='2017', x=Days,
y=df_dow_2017['total_sales']),go.Bar(name='2018', x=Days,
y=df_dow_2018['total_sales'])])
fig.update_layout(barmode='stack')
fig.update_layout(title_text='AVERAGE SALES BY DAY OF WEEK')
fig.update_layout(xaxis = dict(tickmode = 'array',tickvals =
[0,1,2,3,4,5,6],ticktext = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']))
fig.update_xaxes(title_text='Order By days of week')
fig.update_yaxes(title_text='Total Sales')
fig.show()
```

Payment type proportion

```
count=olist_order_payments['payment_type'].value_counts().reset_index()
import plotly.graph_objects as go
labels = ['credit card','boleto','voucher','debit card','not defined']
values = count['payment_type']
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=0.3)])
```

```
fig.show()
```

#geo location analysis

```
import os
import conda
```

```
import mpl_toolkits
mpl_toolkits.__path__.append('/usr/lib/python3.7/dist-packages/mpl_toolkits/')
from mpl_toolkits.basemap import Basemap
```

```
lat = olist_geolocation['geolocation_lat']
lon = olist_geolocation['geolocation_lng']
```

```
plt.figure(figsize=(10,10))
```

```
m = Basemap (llcrnrlat=-55.401805,llcrnrlon=-
92.269176,urcrnrlat=13.884615,urcrnrlon=-27.581676)
m.bluemarble()
m.drawmapboundary(fill_color='#46bcec')
m.fillcontinents(color='#f2f2f2',lake_color='#46bcec')
#m.drawcoastlines()
m.drawcountries()
m.scatter(lon, lat,zorder=10,alpha=0.5,color='tomato')
m
```

#Reviews Analysis

```
sns.set()
```

```
COLOR_5S = '#0571b0'
```

```
COLOR_1S = '#ca0020'
```

```
REVIEWS_PALETTE = sns.color_palette((COLOR_1S, '#d57b6f', '#c6c6c6',  
'#7f9abc', COLOR_5S))
```

```
sns.set_style('darkgrid', {'axes.facecolor': '#eeeeee'})
```

```
resize_plot = lambda: plt.gcf().set_size_inches(12, 5)
```

```
p_5s = len(olist_order_reviews[olist_order_reviews['review_score'] == 5]) *  
100 / len(olist_order_reviews)
```

```
p_1s = len(olist_order_reviews[olist_order_reviews['review_score'] == 1]) *  
100 / len(olist_order_reviews)
```

```
first_dt = olist_order_reviews['review_creation_date'].min()
```

```
last_dt = olist_order_reviews['review_creation_date'].max()
```

```
avg_s = olist_order_reviews['review_score'].mean()
```

```
print(len(olist_order_reviews), 'reviews')
```

```
print('First:', first_dt)
```

```
print('Last:', last_dt)
```

```
print(f'5★: {p_5s:.1f}%')
```

```
print(f'1★: {p_1s:.1f}%')
```

```
print(f'Average: {avg_s:.1f}★')
```

```
sns.catplot(  
    x='review_score',  
    kind='count',
```

```
data=olist_order_reviews,  
palette=REVIEWS_PALETTE  
)  
.set(  
    xlabel='Review Score',  
    ylabel='Number of Reviews',  
)  
);
```

```
olist_order_reviews['review_creation_delay'] =  
(olist_order_reviews['review_creation_date'] -  
olist_order_reviews['order_purchase_timestamp']).dt.days  
sns.scatterplot(  
    x='order_purchase_timestamp',  
    y='review_creation_delay',  
    hue='review_score',  
    palette= REVIEWS_PALETTE,  
    data=olist_order_reviews  
)  
.set(  
    xlabel='Purchase Date',  
    ylabel='Review Creation Delay (days)',  
    xlim=(olist_order_reviews['order_purchase_timestamp'].min(),  
olist_order_reviews['order_purchase_timestamp'].max())  
)  
);  
resize_plot()
```

```
olist_order_reviews['year_month'] =  
olist_order_reviews['order_purchase_timestamp'].dt.to_period('M')  
reviews_timeseries =  
olist_order_reviews[olist_order_reviews['review_creation_delay'] >  
0].groupby('year_month')['review_score'].agg(['count', 'mean'])
```

```
ax = sns.lineplot(
    x=reviews_timeseries.index.to_timestamp(),
    y='count',
    data=reviews_timeseries,
    color='#984ea3',
    label='count'
)
ax.set(xlabel='Purchase Month', ylabel='Number of Reviews')
```

```
sns.lineplot(
    x=reviews_timeseries.index.to_timestamp(),
    y='mean',
    data=reviews_timeseries,
    ax=ax.twinx(),
    color='#ff7f00',
    label='mean'
).set(ylabel='Average Review Score');
resize_plot()
```

```
olist_order_reviews.groupby('order_status')['order_status'].count()
ax = sns.catplot(
    x='order_status',
    kind='count',
    hue='review_score',
    data=olist_order_reviews[olist_order_reviews['order_status'] != 'delivered'],
    palette=REVIEWS_PALETTE
).set(xlabel='Order Status', ylabel='Number of Reviews');
resize_plot()
```

```
olist_order_reviews['delay'] =
(olist_order_reviews['review_answer_timestamp'] -
olist_order_reviews['review_creation_date']).dt.days

bins = [-1, 0, 1, 2, 3, 4, 5, 99999]
labels = ['0', '1', '2', '3', '4', '5', '6+']
olist_order_reviews['delay_group'] = pd.cut(olist_order_reviews['delay'], bins,
labels=labels)
```

Products and sellers info

```
sns.catplot(
    x='delay_group',
    kind='count',
    data=olist_order_reviews,
    palette=sns.color_palette('Greens_r', n_colors=7)
).set(xlabel='Response Delay (days)', ylabel='review_score');

sns.catplot(
    x='delay_group',
    kind='count',
    hue='review_score',
    data=olist_order_reviews,
    palette=REVIEWS_PALETTE
).set(xlabel='Response Delay (days)', ylabel='Number of Reviews');
resize_plot()

total_orders=pd.merge(olist_orders, olist_order_items)
product_orders=pd.merge(total_orders,olist_products, on="product_id")
product_orders.info()

len(product_orders['product_id'].unique())
```

```
len(product_orders['product_id'].str[-8:].unique())
product_orders['product_id_modified']=product_orders['product_id'].str[-8:]
plt.figure(figsize=(20,10))
sns.countplot(x='product_id_modified', data=product_orders,
palette='colorblind',
              order=product_orders['product_id_modified'].value_counts()[:10]\
              .sort_values().index).set_title("Top 10 Products", fontsize=10,
              weight='bold')

product_orders.groupby(["product_category_name"])["product_id_modified"].c
ount().sort_values(ascending=False).head(10)

group_category=
product_orders.groupby(['product_id_modified','product_category_name,'])['pr
oduct_id_modified']\
                .count().sort_values(ascending=False).head(10)

group_category
olist_sellers = pd.read_csv('D:\\final sem
project\\www\\olist_sellers_dataset.csv')
olist_sellers
seller_products = pd.merge(product_orders, olist_sellers, on="seller_id")
seller_products.info()
len(seller_products['seller_id'].unique())
len(seller_products['seller_id'].str[-6:].unique())
seller_products['seller_id_shorten']=seller_products['seller_id'].str[-6:]
plt.figure(figsize=(20,10))
seller_products['seller_id_shorten'].value_counts()[:10].plot.pie(autopct='%1.1f
%%',
                        shadow=True, startangle=90, cmap='tab20')
plt.title("Top 10 Seller",size=14, weight='bold')
seller_category= seller_products.groupby(['seller_id_shorten',
'product_category_name'])\
```

```
['seller_id_shorten'].count().sort_values(ascending=False).head(10)
seller_category
```

```
f, (ax1, ax2) = plt.subplots(2, 1, figsize=(20,15))
group_category.plot.barh(ax=ax1, cmap='summer')
seller_category.plot.barh(ax=ax2, cmap='autumn')
```

```
ax1.set_title('Top10 Product', fontweight='bold')
ax2.set_title('Top10 Seller', fontweight='bold')
```

```
ax1.set_xlabel('Count', fontsize=15)
ax1.set_ylabel('Product Name', fontsize=15)
ax1.xaxis.set_tick_params(labelsize=12)
ax1.yaxis.set_tick_params(labelsize=15)
```

```
ax2.set_xlabel('Count', fontsize=15)
ax2.set_ylabel('Product Name', fontsize=15)
ax2.xaxis.set_tick_params(labelsize=12)
ax2.yaxis.set_tick_params(labelsize=15)
```

#Review Score Distribution

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline

olist_order_reviews.info()
olist_order_reviews.isnull().sum()
```

```
olist_order_reviews =
olist_order_reviews.dropna(subset=['review_comment_message'])
olist_order_reviews['word_count'] =
olist_order_reviews.review_comment_message.apply(lambda x:
len(str(x).split()))
olist_order_reviews.word_count.max()
g = sns.FacetGrid(data=olist_order_reviews, col='review_score',height=5,
aspect=0.8)
before_remove = g.map(plt.hist, 'word_count', bins=30)
before_remove
sns.boxplot(x='review_score', y='word_count', data=olist_order_reviews)
from wordcloud import WordCloud
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize

from nltk.corpus import stopwords
from string import punctuation
from nltk.stem import RSLPStemmer #Stemmer for portugese words.

from nltk.probability import FreqDist
from collections import defaultdict
from heapq import nlargest

stop = stopwords.words('portuguese')
stop.append('nao')

def model (text,n):
    sentences = sent_tokenize(text)
    words = word_tokenize(text.lower())

    stop = set(stopwords.words('portuguese') + list(punctuation))
    sentences_stopwords = [word for word in words if word not in stop]
```

```
frequency = FreqDist(sentences_stopwords)
sentences_important = defaultdict(int)

for i, sentence in enumerate(sentences):
    for word in word_tokenize(sentence.lower()):
        if word in frequency:
            sentences_important[i] += frequency[word]

idx_sentences_important = nlargest(n, sentences_important,
sentences_important.get)
for i in sorted(idx_sentences_important):
    print(sentences[i])

def visualize(label):
    words = ""
    for msg in olist_order_reviews[olist_order_reviews['review_score'] ==
label]['review_comment_message']:
        msg = msg.lower()
        words += msg + ' '
    wordcloud = WordCloud(width=600, height=400).generate(words)
    plt.figure(figsize=(12,8))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.show()
```

#Sentiment Analysis

```
visualize(1)
model(text_review_1,4)
visualize(2)
model(text_review_2,4)
```

```
visualize(3)
model(text_review_3,4)
visualize(4)
model(text_review_4,4)
visualize(5)
model(text_review_5,4)
```

```
#removing numbers
```

```
olist_order_reviews.review_comment_message =
olist_order_reviews.review_comment_message.str.replace('\d+', ' ')
```

```
#lower cases.
```

```
olist_order_reviews.review_comment_message =
olist_order_reviews.review_comment_message.apply(lambda x: "
".join(x.lower() for x in x.split()))
```

```
#Removing punctuation
```

```
olist_order_reviews.review_comment_message =
olist_order_reviews.review_comment_message.str.replace('[^\w\s]', ' ')
```

```
#Removing stopword
```

```
olist_order_reviews.review_comment_message =
olist_order_reviews.review_comment_message.apply(lambda x: " ".join(x for x
in x.split() if x not in stop))
```

```
#removing accentuation
```

```
olist_order_reviews.review_comment_message =
olist_order_reviews.review_comment_message.apply(strip_accents)
```

```
#Tokenmize
```

```
olist_order_reviews.review_comment_message =  
olist_order_reviews.apply(lambda row:  
word_tokenize(row['review_comment_message']), axis=1)
```

#Stemming

```
olist_order_reviews.review_comment_message =  
olist_order_reviews.review_comment_message.apply(lambda x: "  
".join([stemmer.stem(word) for word in x]))
```

```
olist_order_reviews.word_count_new.max()  
g = sns.FacetGrid(data=olist_order_reviews, col='review_score', height=5,  
aspect=0.8)  
g.map(plt.hist, 'word_count_new', bins=30)  
sns.boxplot(x='review_score', y='word_count_new', data=olist_order_reviews)
```

#Model Building

```
order_training = olist_order_reviews  
order_training['review_score'][order_training.review_score == 2] = 1  
order_training['review_score'][order_training.review_score == 3] = 1  
order_training['review_score'][order_training.review_score == 4] = 5
```

```
order_training = order_training[(order_training.review_score == 1) |  
(order_training.review_score == 5)]  
order_training =  
order_training.loc[:,['review_comment_message', 'review_score']]
```

```
from sklearn.feature_extraction.text import CountVectorizer  
cv = CountVectorizer()  
X_count =  
cv.fit_transform(order_training["review_comment_message"]).toarray()  
y_count = order_training.review_score
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_count_pca = pca.fit_transform(X_count)

pca.explained_variance_ratio_
plt.figure(figsize=(12,8))
sns.scatterplot(x = X_count_pca[:,0], y = X_count_pca[:,1] ,
hue=y_count,palette = 'RdYlBu', legend="full")

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_count, y_count, test_size =
1/4, random_state = 42)

logreg_vector = LogisticRegression()
logreg_vector.fit(X_train, y_train)

y_pred = logreg_vector.predict(X_test)
score = logreg_vector.score(X_test, y_test)
print(score)

from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)

def plot_coefficients(classifier, feature_names, top_features=20):
    coef = classifier.coef_.ravel()
    top_positive_coefficients = np.argsort(coef)[-top_features:]
    top_negative_coefficients = np.argsort(coef)[:top_features]
    top_coefficients = np.hstack([top_negative_coefficients,
top_positive_coefficients])
```

```
plt.figure(figsize=(20, 8))
colors = ['red' if c < 0 else 'blue' for c in coef[top_coefficients]]
plt.bar(np.arange(2 * top_features), coef[top_coefficients], color=colors,
align="center")
feature_names = np.array(feature_names)
plt.xticks(np.arange(0, 2 * top_features), feature_names[top_coefficients],
rotation=90)
plt.xlabel("20 significant words for negative reviews (left) and positive
reviews (right)")
plt.ylabel("Coefficient values")

plot_coefficients(logreg_vector, cv.get_feature_names())
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(order_training.review_comment_message)
X_tfidf = X_tfidf.todense()
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_tfidf_pca = pca.fit_transform(X_tfidf)
```

```
pca.explained_variance_ratio_
plt.figure(figsize=(12,8))
sns.scatterplot(x = X_tfidf_pca[:,0], y = X_tfidf_pca[:,1] , hue=y_count, palette
= 'RdYIBu', legend="full")
X_train_tfidf, X_test_tfidf, y_train_tfidf, y_test_tfidf = train_test_split(X_tfidf,
y_count, test_size = 1/4, random_state = 42)
logreg_tfidf = LogisticRegression()
```

```
logreg_tfidf.fit(X_train_tfidf, y_train_tfidf)
y_pred_tfidf = logreg_tfidf.predict(X_test_tfidf)
score = logreg_tfidf.score(X_test_tfidf, y_test_tfidf)
print(score)
cm = metrics.confusion_matrix(y_test_tfidf, y_pred_tfidf)
print(cm)
plot_coefficients(logreg_tfidf, vectorizer.get_feature_names())
```

Appendix B

EDA	Exploratory Data Analysis
NLP	Natural Language Processing
NLTK	Natural Language Processing Tool kit
PCA	Principal Component Analysis
SVM	Support Vector Machines

References

- [1] Machine Learning for Sentiment Analysis
<https://algorithmia.com/blog/using-machine-learning-for-sentiment-analysis-a-deep-dive/>
- [2] Machine Learning
https://en.wikipedia.org/wiki/Machine_learning/
- [3] Sentiment Analysis
<https://monkeylearn.com/sentiment-analysis/>
- [4] Sentiment Analysis of Product Reviews
https://www.researchgate.net/publication/324747522_SENTIMENT_ANALYSIS_OF_PRODUCT_REVIEWS_FOR_E-COMMERCE_RECOMMENDATION/
- [5] Scatter Plot
<https://www.displayr.com/what-is-a-scatter-plot/>
- [6] Line charts in matplotlib
<https://medium.com/@pknerd/data-visualization-in-python-line-graph-in-matplotlib-9dfd0016d180/>
- [7] Facetgrid Description
https://www.google.com/search?rlz=1C1CHBF_enUS861US861&sxsrf=ALeKk038H-cs9iyC9sgQILQ-Np5xummnRg%3A1589663782796&ei=JljAXtqMMMu0tAbYp5uwDQ&q=facetgrid+python+definition&oq=facetgrid+python+definition&gs_lcp=CgZwc3ktYWIQAzIFCAAQzQlyBQgAEM0COgQIABBHOgYIABAHEB46C

[AgAEAcQChAeOgkIABBDEEYQ-
QE6BwgAEBQQhwI6AggAOggIABAIEAcQHIDvhQJYyaICYIm8AmgBcA
B4AIABpAGIAZkLkgEEMTguMZgBAKABAaoBB2d3cy13aXo&sclient=ps
y-ab&ved=0ahUKEwjat9elp7npAhVLGs0KHdjTBtYQ4dUDCAw&uact=5/](https://www.datacamp.com/community/tutorials/stemming-lemmatization-python/)

[8] Stemming and Lemmatization

[https://www.datacamp.com/community/tutorials/stemming-
lemmatization-python/](https://www.datacamp.com/community/tutorials/stemming-lemmatization-python/)

[9] Feature Extraction Techniques

[https://medium.com/@joshsungasong/natural-language-processing-
count-vectorization-and-term-frequency-inverse-document-frequency-
49d2156552c1/](https://medium.com/@joshsungasong/natural-language-processing-count-vectorization-and-term-frequency-inverse-document-frequency-49d2156552c1/)

[10] TF-IDF description

<https://www.geeksforgeeks.org/feature-extraction-techniques-nlp/>

[11] Machine Learning Algorithms comparision

[https://sebastianraschka.com/faq/docs/naive-bayes-vs-logistic-
regression.html/](https://sebastianraschka.com/faq/docs/naive-bayes-vs-logistic-regression.html/)

[12] Future Aspects of Sentiment Analysis

<http://terrificdata.com/2017/04/07/future-sentiment-analysis/>

[13] PCA Decomposition Description

[https://www.quora.com/What-is-the-motivation-around-a-PCA-Is-it-
computational-or-is-it-to-increase-accuracy/](https://www.quora.com/What-is-the-motivation-around-a-PCA-Is-it-computational-or-is-it-to-increase-accuracy/)

[14] [1708.02709] Recent trends in Deep Learning based Sentiment Analysis <https://arxiv.org/abs/1708.02709/>

[15] [1805.03687] Statistical Analysis on E commerce reviews with sentiment classification <https://arxiv.org/abs/1805.03687/>

[16] [1905.12595]Google Analytics for E Commerce Applications
<https://arxiv.org/abs/1905.12595/>

[17] [1910.13162]Sentiment Analysis of electronic products reviews in Vietnamese
<https://arxiv.org/abs/1910.13162/>

[18] Support Vector Machines – Wikipedia
https://en.wikipedia.org/wiki/Support-vector_machine

[19] Why is data Analysis so important in E commerce?
<https://www.christurtonecommerce.com/my-blog/157-data-analysis-ecommerce>

[20] nltk Documentation
<https://www.nltk.org/>

[21] Scikit learn documentation
<https://scikit-learn.org/stable/>

[22] Natural Language Processing – Wikipedia
https://en.wikipedia.org/wiki/Language_model

[23] The evolution of sentiment analysis
<https://arxiv.org/ftp/arxiv/papers/1612/1612.01556.pdf>

[24] Jupyter notebook
https://en.wikipedia.org/wiki/Project_Jupyter

[25] nltk package
<https://www.nltk.org/api/nltk.html>

- [26] Google Colab
<https://colab.research.google.com/notebooks/intro.ipynb>
- [27] Sckit learn documentation
<https://scikit-learn.org/stable/>
- [28] Machine Learning and its applications
https://www.researchgate.net/publication/289980169_An_Overview_of_Machine_Learning_and_its_Applications/
- [29] Sentiment Analysis of Reviews, techniques and Algorithms.
https://www.researchgate.net/publication/339513566_Sentiment_Analysis_in_E-Commerce_A_Review_on_The_Techniques_and_Algorithms/
- [30] Sentiment Analysis of product reviews for recommendations
https://www.researchgate.net/publication/324747522_SENTIMENT_ANALYSIS_OF_PRODUCT_REVIEWS_FOR_E-COMMERCE_RECOMMENDATION/
- [31] Sentiment Analysis, NLP and Clinical Analytics
<https://arxiv.org/ftp/arxiv/papers/1902/1902.00679.pdf>
- [32] Study and Comparision on Sentiment Analysis for online products
https://www.academia.edu/33436353/A_Study_and_Comparison_on_Sentiment_Analysis_for_the_Products_Available_in_E-Commerce/

