Dauphine | PSL
UNIVERSITÉ PARIS

M1-I2D

Projet Data Science & Machine Learning

Amazon Reviews Analysis

By :
Benamara Abdelkader

Djelid Aymen

Supervisors :
Yger Florian

Borias Nicolas

May 8, 2021

Check our code on (Github)
And our web-application (here)

Contents

Introduction :

Sentiment analysis refers to analyzing an opinion or feelings about something using data like text or images, regarding almost anything. Sentiment analysis helps companies in their decision-making process. For instance, if public sentiment towards a product is not so good, a company may try to modify the product or stop the production altogether in order to avoid any losses.

There are many sources of public sentiment e.g. public interviews, opinion polls,.... etc, In this article, we are interested in the reviews of the products in Amazone Website which is a big platform that contain several's of products from different brands like :Microsoft,Belei,Amazonfresh,revly....etc.we know that there are so many new products are emerging every day in this French website. Therefore, customers need to rely largely on product reviews to make up their minds for better decision making on purchase.However, searching and comparing text reviews can be frustrating for users. Hence we need better numerical ratings system based on the reviews which will make customers purchase decision with ease.

1 Data Gathering :

1.1 Dataset Choice :

We decided to scrape the **Amazon's** website and get user reviews data. So a simple way to define web scraping , is a process of automating the extraction of data in an efficient and fast way. With the help of web scraping, you can extract data from any website, no matter how large is the data, on your computer. In simple terms, web scraping saves you the trouble of manually downloading or copying any data and automates the whole process.
The following figure show us what our code will scrape from a given review.

Figure 1: Scraped Data from an Amazon- Review

In our project we used the library **beautiful-soup** to scrape these reviews from the **Amazon's** website , and to do so we created this **reviews_scrapper** function which will help us to collect get needed and specific information that we're interested in :

```python
# Scrapping Reviews Data
def reviews_scrapper(driver, prod, rev_url, ds):
    driver.get(rev_url)
    rev_soup = BeautifulSoup(driver.page_source, 'html.parser')
    reviews_div = rev_soup.find_all('div', {'class': reviews_div_cls})

    for review in reviews_div:
        review_title = review.find('a', {'data-hook': 'review-title'}).text
        review_rate = review.find('i', {'data-hook': 'review-star-rating'}).text
        review_body = review.find('span', {'data-hook': 'review-body'}).text
        review_help = review.find('span', {'data-hook': 'helpful-vote-statement'}).text
        review_home = review.find('span', {'data-hook': 'review-date'}).text
        rev_result = tuple(map(lambda x:
                               unicodedata.normalize("NFKD", x.strip()),
                               [review_title, review_rate, review_body,
                                review_help, review_home]))
        ds.append(rev_result + (prod,))

    return ds
```

So as shown in the previous function for all reviews in the amazon page we are going to create a tuple with wanted information and for missing data (for example a missing help or missing title we replaced it by an **na-value** which will be handled in coming sections).
After the process of scraping we get this dataset which we stored in a **csv** -file here we have its first 10 rows :

	Rev_Title	Rev_Rate	Rev_Bdy	Rev_Hlp	Rev_Home
0	Génial !	5,0 sur 5 étoiles	J'ai commandé ce produit …	60 personnes ont trouvé c…	Commenté en France le 10 …
1	En panne après moins de 2…	1,0 sur 5 étoiles	Pour un adaptateur de ce …	22 personnes ont trouvé c…	Commenté en France le 13 …
2	Compatible pour Dualshock…	5,0 sur 5 étoiles	Acheté pour branchée la m…	38 personnes ont trouvé c…	Commenté en France le 13 …
3	Dongle Bluetooh parfait	5,0 sur 5 étoiles	J'utilise ce dongle bluet…	25 personnes ont trouvé c…	Commenté en France le 29 …
4	Impeccable	5,0 sur 5 étoiles	Quand je vois certains co…	16 personnes ont trouvé c…	Commenté en France le 7 m…
5	Bonne clé USB Bluetooth	5,0 sur 5 étoiles	Posé sur un ordinateur po…	14 personnes ont trouvé c…	Commenté en France le 20 …
6	Fait son boulot	2,0 sur 5 étoiles	Clé bluethooth servant à …	17 personnes ont trouvé c…	Commenté en France le 11 …
7	Enfin !	5,0 sur 5 étoiles	Enfin une bonne clé , j'u…	9 personnes ont trouvé ce…	Commenté en France le 31 …
8	Super adaptateur USB-Blue…	5,0 sur 5 étoiles	J'ai acheté cet adaptateu…	15 personnes ont trouvé c…	Commenté en France le 6 d…
9	Parfait.	5,0 sur 5 étoiles	Ce dongle remplis bien le…	8 personnes ont trouvé ce…	Commenté en France le 8 a…
10	ne fonctionne pas. erreur…	1,0 sur 5 étoiles	voilà le descriptif :Pilo…	10 personnes ont trouvé c…	Commenté en France le 7 m…

Figure 2: Amazon Reviews Dataset

2 Problem Defintion :

2.1 Tasks to solve :

Given **Amazon's** reviews about several products of different categories and brands for example (In category 'Tech' we can have many different products (Laptops, Smartphones , headphones ,...) The following show how we structured this on a dictionary :

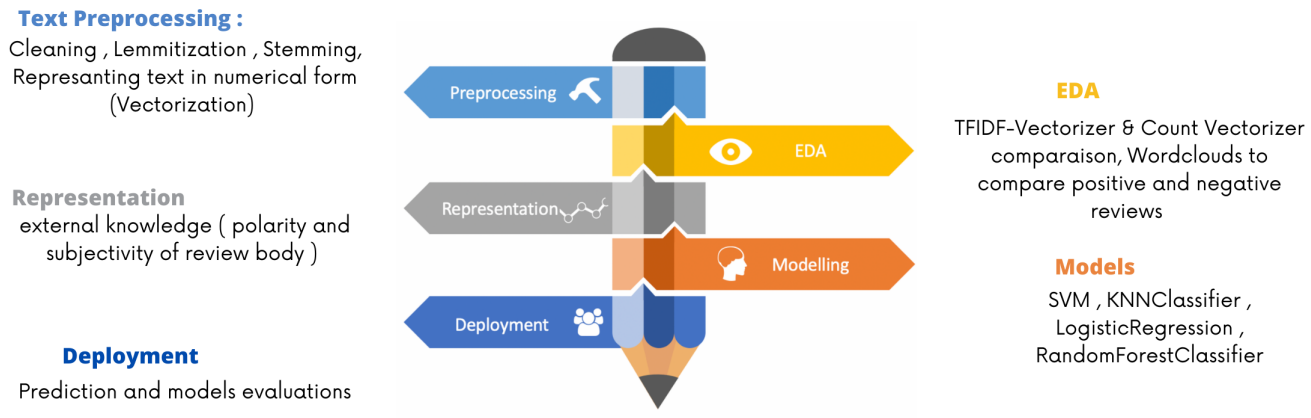
```
'''
    Each Page of reviews contains 10 comments ( So if we took for
     example 100 rev_pages we are talking about 1000 comment )
'''

{
    # Category :
    'informatique':
    # List of products in this category
    [
        {
            'prod_id': 1,
            'prod_url': 'link-to-product-1-page',
            'rev_url': 'link-to-reviews-page',
            'rev_pages': 100,
        },
        {
            'prod_id': 2,
            'prod_url': 'link-to-product-2-page',
            'rev_url': 'link-to-review-page',
            'rev_pages': 'how-many-pages-of-reviews',
        },
    ]
}
```

The task is to predict whether a review contains positive, negative, or neutral sentiment about the product. This is a typical **Supervised Learning** task where given a text string, we have to categorize the text string into predefined categories (3 in our case) We are talking about multi-class classification.
Note : We will be giving this encoding to our 3 classes, by **1** we refer to positive reviews , **-1** negative one's and by **0** neutral reviews.

3 Methodology Description :

The methodolgy for this NLP project (Sentiment Analysis) is as shown in the following scheme :



4 Dataset Analysis :

4.1 Data Pre-processing :

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.
In this section, the following text preprocessing were applied on each row from the dataset.

1. **Rev_Rate :** here is an example of the original scraped data for this column : **(5,0 sur 5 étoiles)** and after cleaning we get just **float(5.0)** for this same row. This transformation is due to the following code

```
# Cleaning Rate review
Rev_Rate = Rev_Rate.str.split(' ').apply(lambda x: float(x[0].replace(',','.')))
```

2. **Rev_Help :** here is an example of the original scraped data for this column : **(60 personnes ont trouvé ça utile)** and after cleaning we get just **int(60)** for this same row. This transformation is due to the following code

```
# Cleaning Hlp Review
Rev_Hlp = Rev_Hlp.apply(lambda x: int(re.sub("[^0-9]","",x)))
```

3. **Rev_Home :** here is an example of the original scraped data for this column : **(Commenté en France le 20 Novembre 2020)** and after cleaning we get just **'France'** for this same row. This transformation is due to the following code

```
# Cleaning Home Review
Rev_Home = Rev_Home.str.split(' ').apply(lambda x: x[2])
```

4. **Rev_Bdy :** This is the most important part of our cleaning process because all sentiment analysis will be based on the review body we can summarize the cleaning steps for the review body with the following function :

```
# Clean Review Body
def clean_review_body(rev):
  # Make text to lowerCase
  clean_rev = rev.lower()
  # Remove ponctuation like .,!? etc
  clean_rev = clean_rev.translate(str.maketrans('','',string.punctuation))
  # Remove words that contain numbers
  clean_rev = re.sub(r'\w*\d\w*','',clean_rev)
  return clean_rev
```

After this previous steps of cleaning we need to drop all rows that are not commented in France (French Analysis only).
Applying this steps to our original dataset produce the following dataset :

	Rev_Title	Rev_Rate	Rev_Bdy	Rev_Hlp	Rev_Home	Prod_ID	Subjectivity	Polarity	Sentiment
0	Génial !	5	jai commandé ce produit p…	60	France	1	0.4125	0.2475	Positive
1	En panne après moins de 2…	1	pour un adaptateur de ce …	22	France	1	0.2220	-0.1068	Negative
2	Compatible pour Dualshock…	5	acheté pour branchée la m…	38	France	1	0.1680	0.1520	Positive
3	Dongle Bluetooh parfait	5	jutilise ce dongle blueto…	25	France	1	0.4450	0.2575	Positive
4	Impeccable	5	quand je vois certains co…	16	France	1	0.1982	0.0750	Positive
5	Bonne clé USB Bluetooth	5	posé sur un ordinateur po…	14	France	1	0.2983	0.3517	Positive
6	Fait son boulot	2	clé bluethooth servant à …	17	France	1	0.4000	-0.1937	Negative
7	Enfin !	5	enfin une bonne clé jutil…	9	France	1	0.3300	0.0588	Positive
8	Super adaptateur USB-Blue…	5	jai acheté cet adaptateur…	15	France	1	0.4180	0.3480	Positive
9	Parfait.	5	ce dongle remplis bien le…	8	France	1	0.1991	0.1955	Positive
10	ne fonctionne pas. erreur…	1	voilà le descriptif pilot…	10	France	1	0.2667	0.0633	Positive

Figure 3: Amazon Reviews pre-processed Dataset

4.1.1 Stemming & Lemmitization :

In this section we have to treat out **Stemming** subject which basically just removes or stems the last few characters of a word, often leading to incorrect meanings and spelling (for example **sel, saler, salière**) we would like to get have just one word in our data (**sel** for example) and this task is done in stemming section, in the other hand **Lemmatization** considers the context and converts the word to its meaningful base form, which is called Lemma. Sometimes, the same word can have multiple different Lemmas. We should identify the Part of Speech (POS) tag for the word in that specific context.
Lemmatization links words with similar meaning to one word. Wordnet and treebank have different tagging systems, so we want to first define a mapping between wordnet tags and POS tags. Then, we lemmatize words using NLTK.
This two tasks are so important to vectorize our data and also to genrate wordcloud in EDA sections.

4.1.2 Representing Text in Numerical Format (Vectorization) :

Statistical algorithms use mathematics to train machine learning models. However, mathematics only work with numbers. To make statistical algorithms work with text, we first have to convert text to numbers. To do so, two main approaches exist **i.e. TF-IDF** and **CountVectorizer** . In this section, we will discuss the **TF-IDF** scheme.
TF computes the classic number of times the word appears in the text, and **IDF** computes the relative importance of this word which depends on how many texts the word can be found. **TF-DF** is the inverse document frequency. It adjusts for the fact that some words appear more frequently in general, like "Je", "Moi", etc. We discard words that appeared in $> 90\%$ of the reviews and words appeared in $< 10\%$ reviews since high appearing words are too common to be meaningful in topics and low appearing words won't have a strong enough signal and might even introduce noise to our model.
Luckily for us, Python's **Scikit-Learn** library contains the **TfidfVectorizer** class that can be used to convert text features into **TF-IDF** feature vectors. The following script performs this :

```
stopwords = stopwords.words('french')

def data_to_matrix(data):
  # Building the Victorizer
  stemmer = FrenchStemmer()
  analyzer = TfidfVectorizer().build_analyzer()
  vec = TfidfVectorizer(stop_words = stopwords,
                        analyzer = lambda doc:(stemmer.stem(w) for w in analyzer(doc))
                        ngram_range = (1,3))
  # Build the new matrix dataset
  mx_rev = vec.fit_transform(data.Rev_Bdy)
  df_rev = pd.DataFrame(mx_rev.toarray(),columns = vec.get_feature_names())
  df_rev.index = data.index
  df_rev.insert(0, 'Title', data.Rev_Title)
  return df_rev
```

4.2 Statistical Analysis (EDA) :

4.2.1 Which Ratings got Highest Number of Reviews ?

Customers have written reviews and ratings were given from 1 to 5 for Products they bought from Amazon. The distribution and percentage of ratings vs number of reviews is shown below. Number of reviews for rating 5 were high compared to other ratings. Overall, customers were happy about the products they purchased. About 79% customers gave 5 rating for the products they purchased. Only about 8% customers gave ratings less than 3 stars on their ratings.

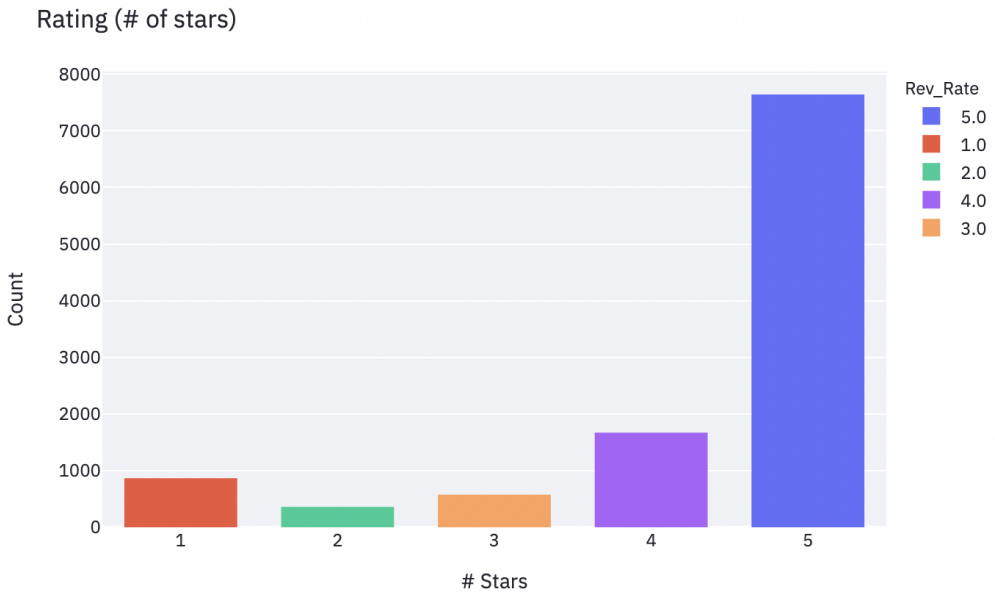


Figure 4: Reviews Rating Distribution

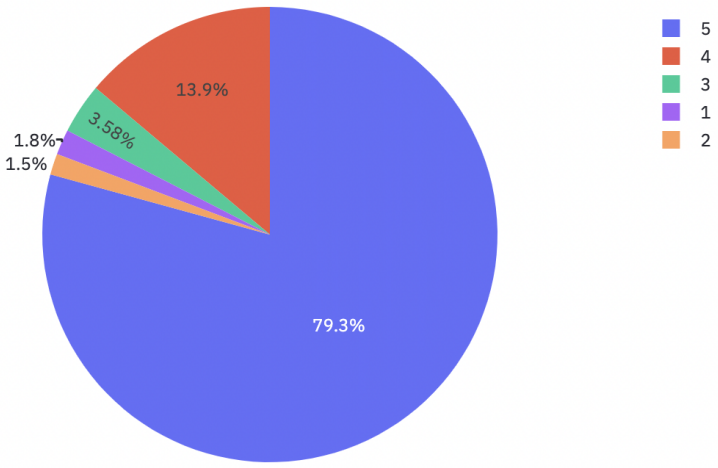


Figure 5: Reviews Rating Percentages

4.2.2 Which Classes got Highest Number of Reviews ?

In this part of our project we are interested in analyzing sentiments of customers from their reviews
A possible solution is to calculate the number of reviews for our 3 classes and the following graph show that we have much more reviews with Positive Sentiment (about 85% from our data) which is clearly dominating the two other classes, but it still logical from the previous ratings distribution where we got **4-5 stars** rating dominating.
It is evident from the output that for almost all the reviews, the majority of the reviews are positive, followed by neutral and negative reviews.

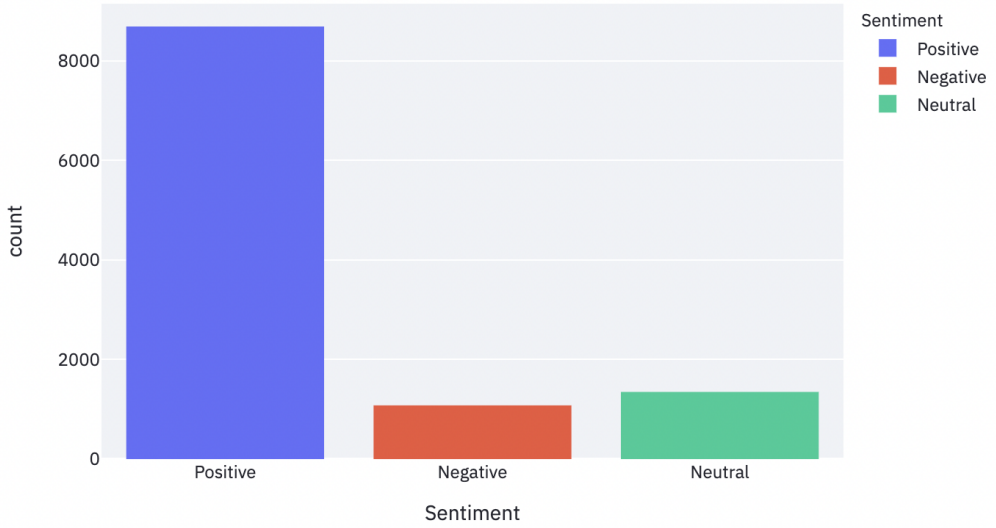


Figure 6: Reviews Sentiments Distribution

Note. The sentiment was generated based on **Polarity** attribute which we got from **TextBlob-fr**

4.2.3 How are Reviews Helpfulness in several products ?

In this section of our project we are interested in analyzing how many people find other reviews helpful for the different products
Our data is categorized in **6** different categories **(Informatique,Cuisine,Sports,Electromenager,Montres,Auto-moto)**
From the output, we can see that the product of ID 7 and 8 are the most helpful products and they are part of catgory **Cuisine**, but also the first category **Informatique** got many helpful tags on its reviews.
The last category compared to other (in terms of helpfulness) is **Auto-moto**.

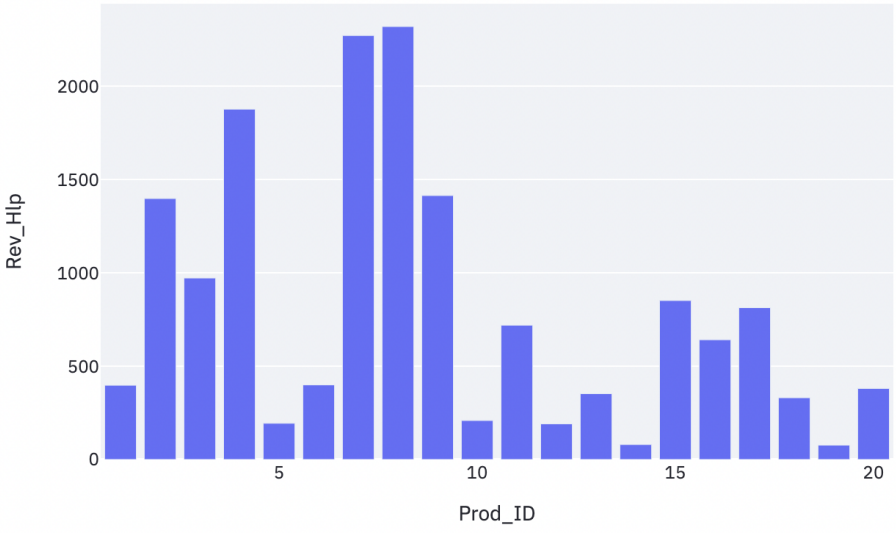


Figure 7: Reviews Helpfulness Distribution

Note. We have respectively **4-4-2-4-4-2** products from the listed categories above.

4.2.4 How are Sentiments Distributed for the Reviews ?

The x-axis shows polarity, and y-axis shows subjectivity.Polarity tells how positive or negative the text is. The subjectivity tells how subjective or opinionated the text is. The green dots that lies on the vertical line are the "neutral" reviews, the red dots on the left are the "negative" reviews, and the blue dots on the right are the "positive" reviews. Bigger dots indicate more subjectivity. We see that positive reviews are more than the negatives.

Figure 8: Reviews Sentiments Distribution

The same scatter graph can be represented in the following histogram in which we confirm the same information that positive comments are those who have a positive polarity and thet are much more than Neutral or Negative reviews which can bbe explained from the previous stats.
We also observe that this data is normally ditributed and between $0.7 - 0.8$ for Polarity we observe a higher value of subjectivity and from the meaning we can say that customers who react subjectivly on comment the have a higher probability to be with a positive sentiment and which completly logical.

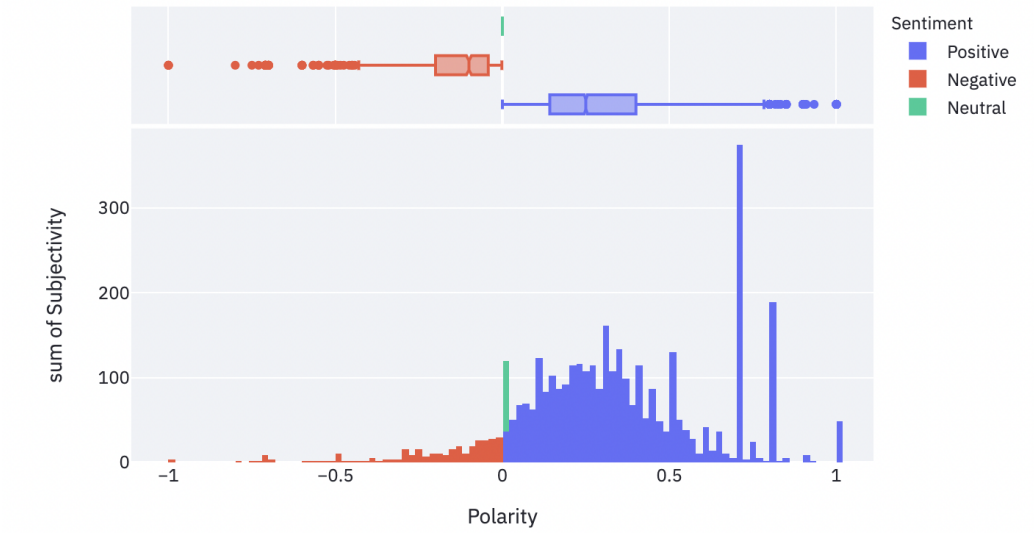


Figure 9: Reviews Snetiment Distribution

4.2.5 Is there any relation between Stars Rating and Polarity ?

In this part we are interested in the relation between polarity and number of stars in reviews rating , we can observe that for all stars ratings values polarity has more appearances between -0.25 and 0.25 and also as much as we have stars in review the polarity is higher which is what we expected and that show that no outliers are in our data, otherwise we would have ratings and Polarity in different order than what we got.

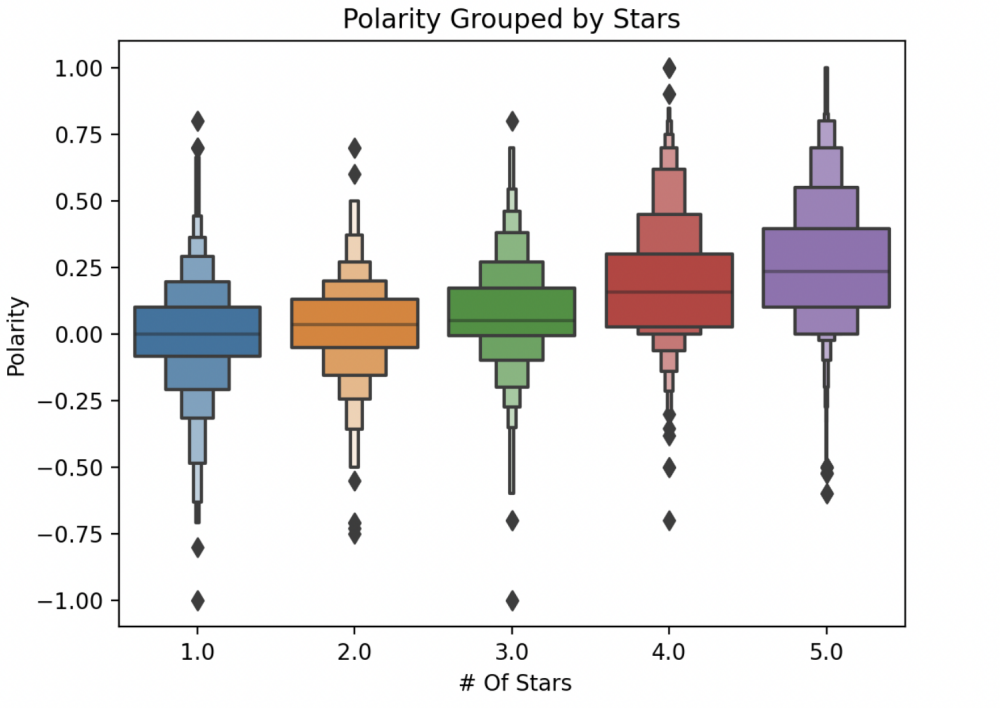
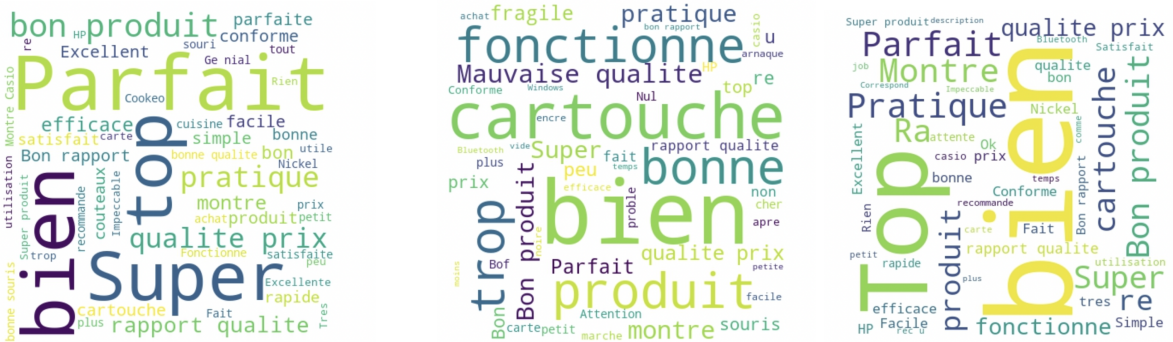


Figure 10: Polarity Grouped By Stars Rating

4.2.6 Which words are used for Positve/Negative & Neutral comments?

We can see that there are some words which are repeated in both Positive and Neutral reviews which mean that In general customers are commenting with the same way (vocabulary) and we see that this step isn't that deducing because we cannot take any conclusion (for example the word **Bien** is present in the three classes which is confusing at the first sight) but it can be explained with the fact of negation (for example many people even with negative opinion can use this word but in a negative way).



(a) Positive Reviews (b) Negative Reviews (c) Neutral Reviews

Figure 11: Word Clouds For Reviews by sentiment

5 Machine Learning Models :

5.1 Setting Classes to Reviews data :

As we said in the beginning of our project , we have a task of classification with 3 classes , to get this classes we used the **TextBlob-fr** Python's library to get a **Sentiment** object which have as attributes **Polarity** and **Subjectivity**, the polarity will help us to set a class for a comment.This task is simply done with an encoder function as follow :

```
def encode_sentiment(sentiment):
    if sentiment == 'Positive' :
      return 1
    elif sentiment == 'Negative' :
      return -1
    else :
      return 0

def decode_sentiment(data):
    if data > 0 :
      return 'Positive'
    elif data < 0 :
      return 'Negative'
    else :
      return 'Neutral'
```

And so with this function we have the labels for our reviews.

```
ds['Polarity'] = ds['Rev_Bdy'].apply(get_polarity)
ds['Sentiment'] = ds['Polarity'].apply(decode_sentiment)
y = ds['Sentiment'].apply(encode_sentiment)
```

5.2 Dividing data into train and test sets :

In the previous section, we converted the data into the numeric form. As the last step before we train our algorithms, we need to divide our data into training and testing sets. The training set will be used to train the algorithm while the test set will be used to evaluate the performance of the machine learning model.
Execute the following code:

```
df = mx_rev.drop('Title',axis = 1)
X_train , X_test , y_train , y_test = train_test_split(df,y,test_size = 0.2)
```

5.3 Training Models :

Once data is split into training and test set, machine learning algorithms can be used to learn from the training data. You can use any machine learning algorithm. However, we will use the SVM, KNN (k-Nearest Neighbors), Logistic Regression and Random Forest algorithms, owing to its ability to act upon non-normalized data.
We will use the help of **Pipelines** to make the life easier !
To define any Model from the following we will use the following code :

```
from sklearn.pipeline import Pipeline
model = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', Classifier()),
])
```

5.3.1 SVM (Support Vector Machine):

5.3.1.1 Training SVM model :

The sklearn.ensemble module contains the **SVC** class that can be used to train the **SVM** Machine Learning model. To do so, we need to call the fit method on our pipeline (**model** object) the SVM class and pass it our training features and labels, as parameters.Look at the following script:

```
from sklearn.pipeline import Pipeline

# Pipline of our model
model = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', SVC(params)),
])

# SVM params to be chosen
params = {
    'kernel' : ['poly','rbf'],
    'C' : [1,2,5,10,20,50,100,200,1000,2000,5000],
    'gamma' : [1e-1,1e-2,1e-3, 1e-4,1e-5],
    'degree' : [2,3],
    'decision_function_shape' : ['ovo','ovr'],

}
# GridSearchCV to get the best params
grid = GridSearchCV(model,param_grid = params, cv = 'number of folds for cross-validation'

model.fit(X_train,y_train)
```

Note . the params object will be detailed in mode evaluation improvment via **GridSearchCV** to get the best parameters

5.3.1.2 Evaluating SVM model :

Once the model has been trained, the last step is to make predictions on the model
Finally, to evaluate the performance of the machine learning models, we can use classification metrics such as a confusion metrix, F1 measure, accuracy, etc.

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report

y_pred = model.predict(X_test)

print(f'accuracy : {accuracy_score(y_pred, y_test)}')
print(classification_report(y_test, y_pred))
```

The output of the script above looks like this :

```
accuracy 0.8837095191364083
              precision    recall  f1-score   support

          -1       0.67      0.45      0.54       209
           0       0.83      0.77      0.80       252
           1       0.91      0.96      0.93      1577

    accuracy                           0.88      2038
   macro avg       0.80      0.73      0.76      2038
weighted avg       0.88      0.88      0.88      2038
```

Figure 12: SVM Evaluation

So As we saw we got 88% accuracy using the **SVM** model we can show below the confusion matrix :

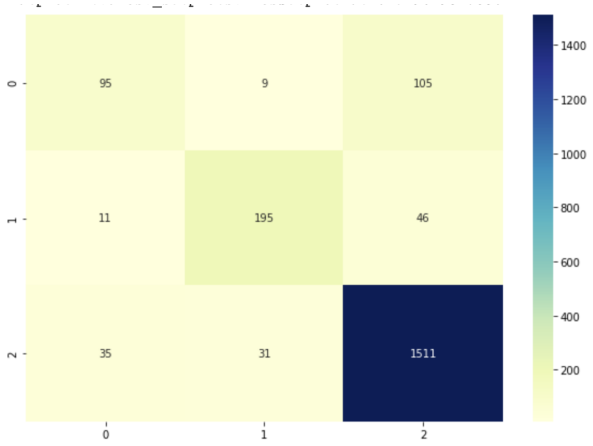


Figure 13: SVM-evaluation Confusion Matrix

From what the confusion matrix and the classification report we can observe that the negative reviews are those which make the accuracy less than expected and that is expected beacause of the amount of negative reviews compared to the postive one's in our dataset.
We also tried to compare the performance of SVM models based on the hyperparmeter **C** and the kernel **rbf** and **linear** the following graph show that for values of **C** out of range 1..500 we have the same performance from rbf and linear kernels but inside this range we observe that the lineat kernel was better.
This doesn't mean that we should choose the **linear** kernel, because when searching for the best parameters using the **GridSearchCv** we got that the best value for $\mathbf{C} = \mathbf{1000}$ which clearly out of the range of dominance that linear kernel show us.

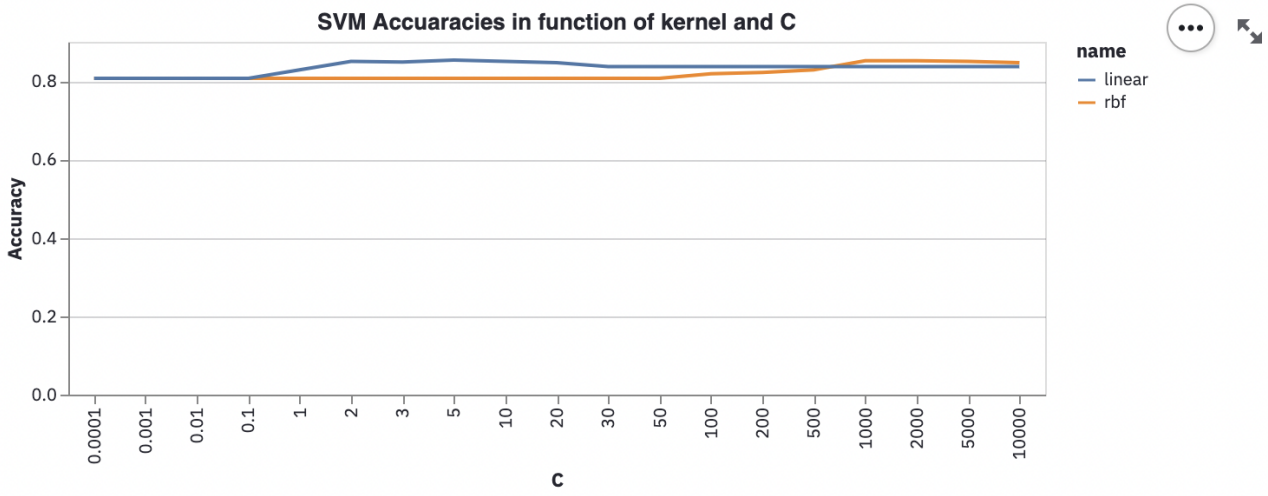


Figure 14: SVM-evaluation Kernel stats

Note . The process of grid search consumed a lot of time because it constructs multiple models for all possible combinations of param values in the param_grid dictionary. So, here we are only tuning two parameters: C and kernel.

5.3.2 k-NN (k-Nearest Neighbors):

5.3.2.1 Training KNN model :

The sklearn.ensemble module contains the **KNeighborsClassifier** class that can be used to train the **KNN** Machine Learning model. To do so, we need to call the fit method on our pipeline (**model** object) the KNeighborsClassifier class and pass it our training features and labels, as parameters.Look at the following script:

```
# KNN params to be chosen
params = {
    'n_neighbors' : [range(1,21)],
    'weights' : ['uniform','distance'],
    'metric' : ['minkowski','manhattan','euclidean','cosine'],
}

# GridSearchCV to get the best params
model = GridSearchCV(model,param_grid = params, cv = 'number of folds for cross-validation')

model.fit(X_train,y_train)
```

Note . the params object will be detailed in mode evaluation improvment via **GridSearchCV** to get the best parameters

5.3.2.2 Evaluating KNN model :

Once the model has been trained, the last step is to make predictions on the model
Finally, to evaluate the performance of the machine learning models, we can use classification metrics such as a confusion metrix, F1 measure, accuracy, etc.

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report

y_pred = model.predict(X_test)

print(f'accuracy : {accuracy_score(y_pred, y_test)}')
print(classification_report(y_test, y_pred))
```

The output of the script above looks like this :

```
accuracy 0.7768496420047732
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00        66
           0       0.43      0.47      0.45       115
           1       0.84      0.91      0.87       657

    accuracy                           0.78       838
   macro avg       0.42      0.46      0.44       838
weighted avg       0.72      0.78      0.75       838
```

Figure 15: KNN Evaluation

So As we saw we got 77% accuracy using the **KNN** model we can show below the confusion matrix :

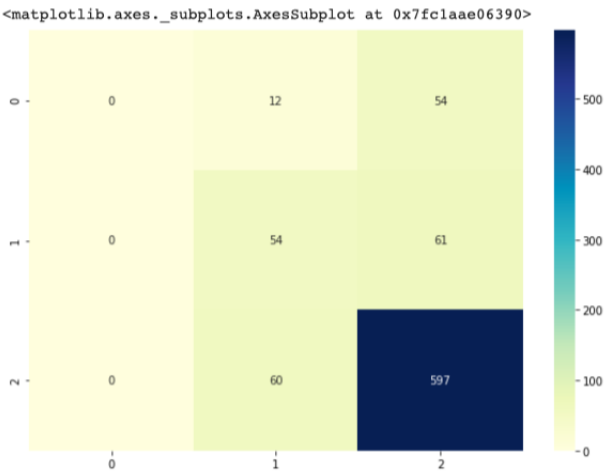


Figure 16: KNN-evaluation Confusion Matrix

From what the confusion matrix and the classification report we can observe that the negative reviews are those which make the accuracy less than expected and that is expected because of the amount of negative reviews compared to the positive one's in our dataset.
We also tried to compare the performance of KNN models based on the hyperparmeter **n_neighbors** the following graph show that for values of **n_neighbors** bigger than 15 are the best for the accuracy but we also observe that they are mostly in constant variation which means there is no need to increase more this parameter than 15.

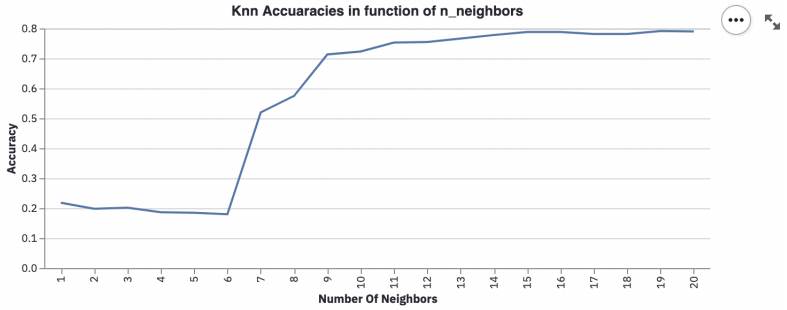


Figure 17: KNN-evaluation neighbors stats

5.3.3 Logistic Regression Model :

5.3.3.1 Training Logistic Regression model :

The sklearn.ensemble module contains the **LogisticRegression** class that can be used to train the **LogisticRegression** Machine Learning model. To do so, we need to call the fit method on our pipeline (**model** object) the LogisticRegression class and pass it our training features and labels, as parameters.Look at the following script:

```
# LogisticRegression params to be chosen
params = {
    'C' : [1,2,5,10,20,50,100,200,1000,2000,5000],
    'tol' : np.linespace(0,1,200),
    'penalty' : ['l1','l2','none'],
}

# GridSearchCV to get the best params
grid = GridSearchCV(model,param_grid = params, cv = '# of folds for cross-validation')

model.fit(X_train,y_train)
```

Note . the params object will be detailed in mode evaluation improvment via **GridSearchCV** to get the best parameters

5.3.3.2 Evaluating Logistic Regression model :

Once the model has been trained, the last step is to make predictions on the model
Finally, to evaluate the performance of the machine learning models, we can use classification metrics such as a confusion metrix, F1 measure, accuracy, etc.

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report

y_pred = model.predict(X_test)

print(f'accuracy : {accuracy_score(y_pred, y_test)}')
print(classification_report(y_test, y_pred))
```

The output of the script above looks like this :

```
accuracy 0.8577036310107949
              precision    recall  f1-score   support

          -1       0.81      0.22      0.35       209
           0       0.88      0.58      0.70       252
           1       0.86      0.99      0.92      1577

    accuracy                           0.86      2038
   macro avg       0.85      0.60      0.66      2038
weighted avg       0.85      0.86      0.83      2038

cross-validation score : 0.8022545167379329
```

Figure 18: Logistic Regression Evaluation

So As we saw we got 85% accuracy using the **Logistic Regression** model we can show below the confusion matrix :

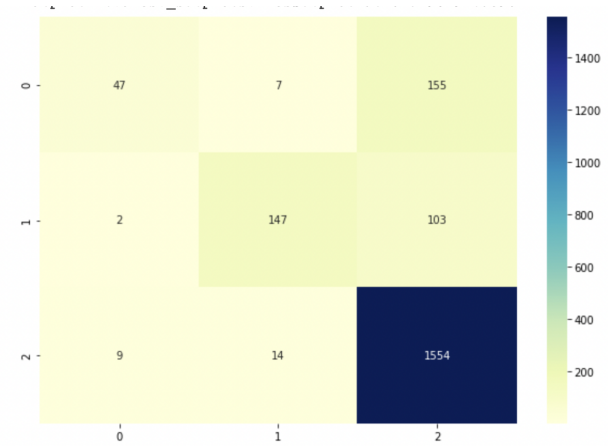


Figure 19: Logistic Regression-evaluation Confusion Matrix

From what the confusion matrix and the classification report we can observe that the negative reviews are those which make the accuracy less than expected and that is expected because of the amount of negative reviews compared to the positive one's in our dataset.
So basically the same issue is present in all trained models which is the amount of negative reviews

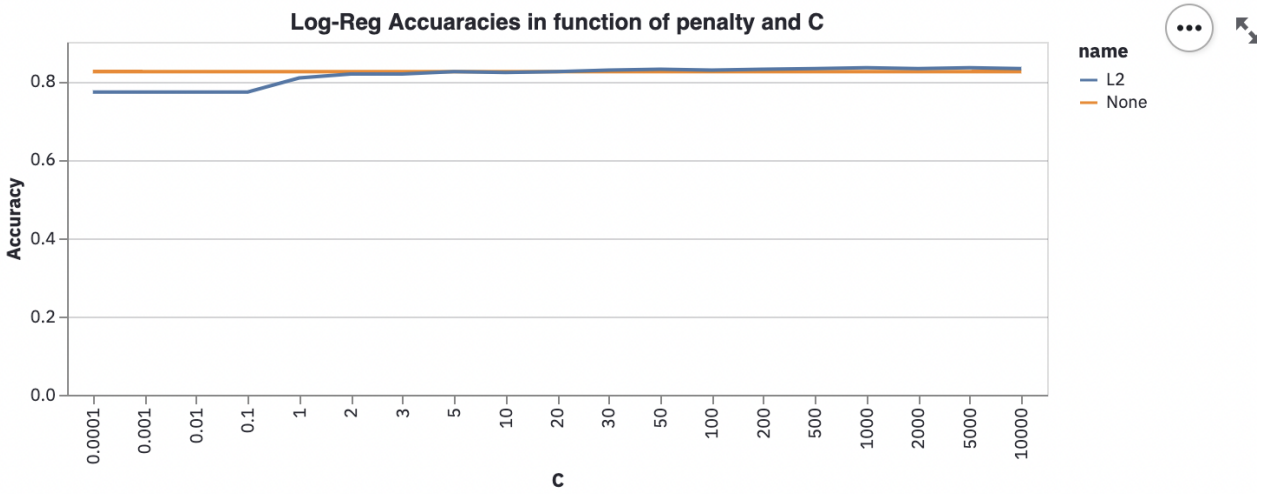


Figure 20: Log-Regression-evaluation Penalty stats

The l2 penalty equal to the square of the magnitude of coefficients. L2 will not yield sparse models and all coefficients are shrunk by the same factor (none are eliminated).
So we tried to compare the performance of Logistic Regression models based on the hyperparmeter **C** and the penalty **l2** and **None** (without penalty) the following graph show that for values of **C** out of range $1e-4$ to 3 we have the same performance between l2 and none penalty but inside this range we observe that the presence of the l2 regularization make accuracy lower for our model.

5.3.4 Random Forest Model:

5.3.4.1 Training RandomForest model :

The sklearn.ensemble module contains the **RandomForestClassifier** class that can be used to train the **RandomForestClassifier** Machine Learning model. To do so, we need to call the fit method on our pipeline (**model** object) the RandomForestClassifier class and pass it our training features and labels, as parameters.Look at the following script:

```
# RandomForestClassifier params to be chosen
params = {
    'n_estimators': [100, 300, 500, 800, 1200],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8,15,25,30],
    'criterion' :['gini', 'entropy'],
    'min_samples_leaf' : [1, 2, 5, 10],
    'min_samples_split' : [2, 5, 10, 15, 100]
}

# GridSearchCV to get the best params
grid = GridSearchCV(model,param_grid = params, cv = '# of folds for cross-validation')

model.fit(X_train,y_train)
```

Note . the params object will be detailed in mode evaluation improvment via **GridSearchCV** to get the best parameters

5.3.4.2 Evaluating RandomForest model :

Once the model has been trained, the last step is to make predictions on the model
Finally, to evaluate the performance of the machine learning models, we can use classification metrics such as a confusion metrix, F1 measure, accuracy, etc.

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report

y_pred = model.predict(X_test)

print(f'accuracy : {accuracy_score(y_pred, y_test)}')
print(classification_report(y_test, y_pred))
```

The output of the script above looks like this :

```
accuracy 0.8412887828162291
              precision    recall  f1-score   support

          -1       1.00      0.01      0.03        78
           0       0.75      0.52      0.61        99
           1       0.85      0.99      0.91       661

    accuracy                           0.84       838
   macro avg       0.87      0.51      0.52       838
weighted avg       0.85      0.84      0.79       838
```

Figure 21: RandomForest Evaluation

So As we saw we got 84% accuracy using the **RandomForest** model we can show below the confusion matrix :

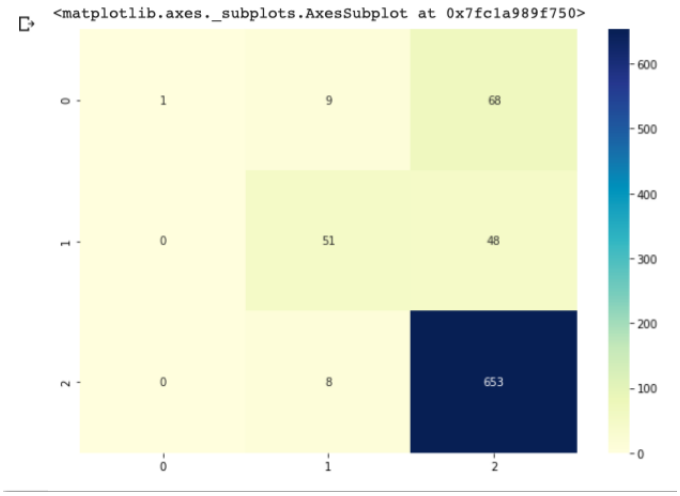
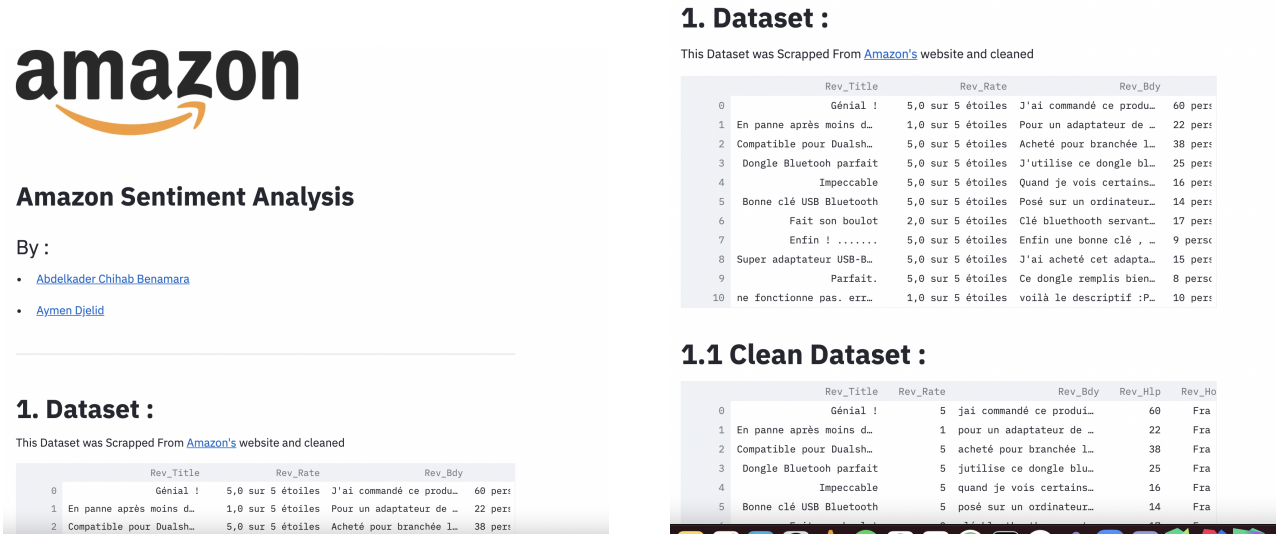


Figure 22: RandomForest-evaluation Confusion Matrix

From what the confusion matrix and the classification report we can observe now another results which are slightly different from the first 3 models , here we have much less errors in precision for the negative reviews which is expected because of the work of a RandomForest and as known RandomForest can do much better but the computations will be more expensive in terms of cpu and time.

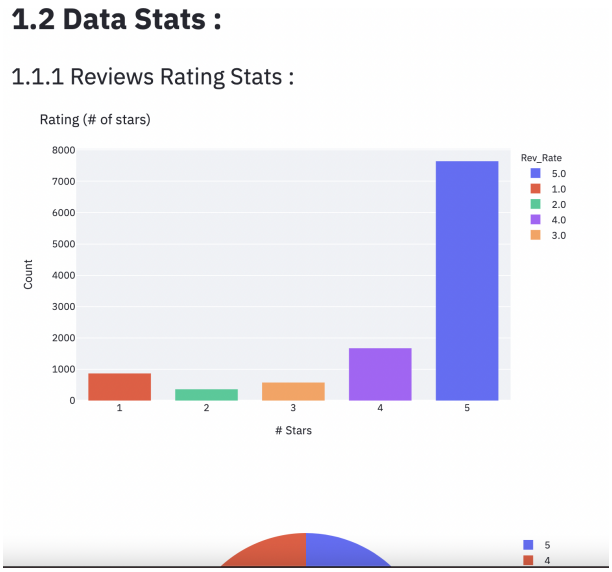
6 Wanna Compare the performance of these models ?

Yes ! , we have a solution for you. We decided to develop a Python web-application using the **Streamlit** API so it make it simple and friendly to train the model you want on our dataset and change the parameters as you want.
We also managed to represent the dataset firstly and after cleaning and also all EDA's are there !
Here some screenshots , and don't hesitate to test out this awesome application !
You can check it out (here)

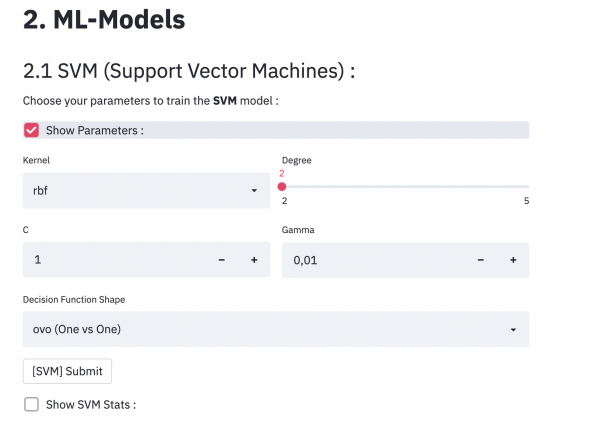


(a) (b)

Figure 23: Some Screenshot from our website

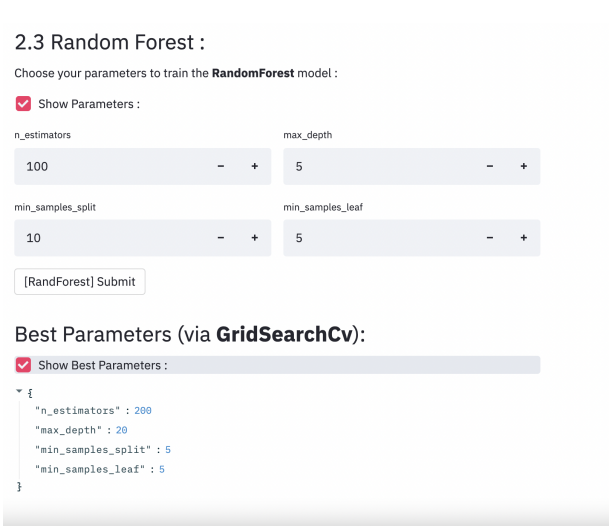


(a)

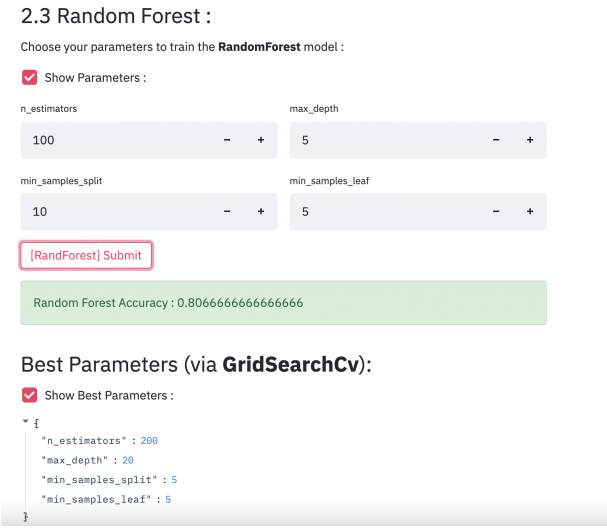


(b)

Figure 24: Some Screenshot from our website



(a)



(b)

Figure 25: Some Screenshot from our website

Conclusions :

Online reviews have become a platform for building trust and influencing consumer buying patterns. With such dependency there is a need to handle such large volume of reviews and present credible reviews before the consumer. Our research is aiming to achieve this by conducting sentiment analysis of Amazon products reviews and classifying the reviews into positive and negative,neutral sentiment.Sentiment analysis is one of the most commonly performed NLP tasks as it helps determine overall public opinion about a certain topic.
We performed an analysis of reviews of amazon products using some Machine Learning Models like SVM (support vector machine),logistic regression ,Knn classifier and Random Forest and we achieved an accuracy of 88% with the SVM classifier which was the best accuracy comparing with the accuracy of the Knn Classifier(77%),logistic regression (85%) and our last model Random forest(84%).