**DSA 5900 Professional Practice**

Department of data Science and Analytics

Spring 2021

**Predicting Business Owners Response on Social Media Using Data Science Approach**

Purushotham Vadde – 113459984

**Supervisor:**

Naveen Kumar,

Assistant Professor of Management Information Systems,

University of Oklahoma

Table of Content

1. Introduction

2. Objectives

3. Preparing Dataset

* Data Collection
* Data Understanding
* Data Preparation
* Data Transformation

4. Methodology

* Random Forest
* Gradient Boosting
* XGBoost
* Oversampling with SMOTE

5. Results and Analysis

6. References.

**1. Introduction:**

Nowadays, most online users rely on online reviews of a business or product before making a decision. Online reviews are playing a central role in decision-making. According to Forbes, 90% of the user will read the reviews of a business to determine the quality, 82% of Yelp users said they typically visit Yelp because they intend to buy a product or service. 74% of users say that the positive reviews make them trust a local business more[1]. The Internet is filled with many reviews, and there are thousands of reviews available for each business. People prefer a business with good reviews compare to the other. All Web applications allow users to write a review of their experience with their respective businesses to know about their service. There are many applications, especially for business reviews, such as Yelp, Google reviews, Facebook, Etc.

The online reviews contain the reviews from customers and business owners' responses for the customer reviews. We can also see the customer reviews are more in number compared to business owner response reviews. The reason for this is that the business owners are not responding to all the customer reviews. They are responding to only a few reviews.

As part of this project, we will develop a Data science approach to determine what causes the business owner to respond to a review about their business or product. We will use the supervised machine learning approach called classification to find the type of reviews for which the Business owner will respond. We are mainly going to use ensemble methods such as Bagging and Boosting algorithms to classify business reviews.

**2. Objective:**

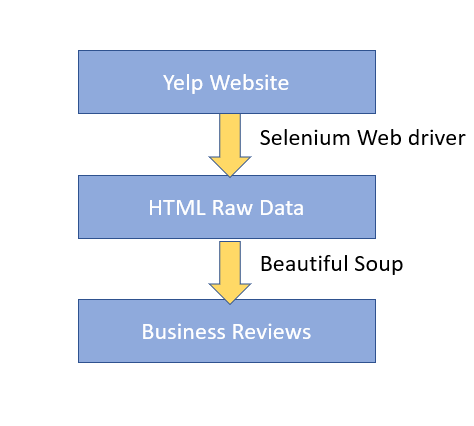
The main goal of this project is to build a Binary Classification machine learning model using ensemble algorithms to classify the online business reviews for which the Business Owners are likely going to respond. Another critical step in this project is data collection, the business reviews for different business categories to be collected from the Yelp website through web scraping. We will also focus on finding the key insights, such as the features that affect business owners to respond to the customer review using descriptive Analytics.

**3. Preparing Dataset**:

**Data Collection:**

In this step, we will collect the business reviews for Seattle City in different categories such as (Auto Repair, Flooring, Handyman, HVAC, Painters, Plumbing, Tires, Transmission Repair) from the yelp web site. We cover a variety of business categories to maintain fairness in Data.

The Data collected through web scraping by using tools such as Selenium Web driver and Beautiful soup.



*Fig 1: Data Collection Architecture*

In the above figure we can the data Collection Architecture using the Selenium Web driver and Beautiful soup.

**Selenium:**

Selenium web driver is a framework used to test web applications. Using the selenium web driver and chrome driver, we can extract the HTML content of the web pages. The main advantage of using the selenium web driver is that the selenium can handle dynamic web content. In comparison, the tools such as urllib cannot manage dynamic web content.

**Beautiful Soup:**

The Beautiful soup is a Python library used to extract data from HTML. The HTML raw data extracted from the Yelp Website parsed using the beautiful soup, and the required Business Reviews data extracted from the HTML page using the Beautiful soup inbuilt functions.

**Challenges in Data Collection:**

While collecting data from the Yelp website for business reviews, we ran into a variety of issues, such as:

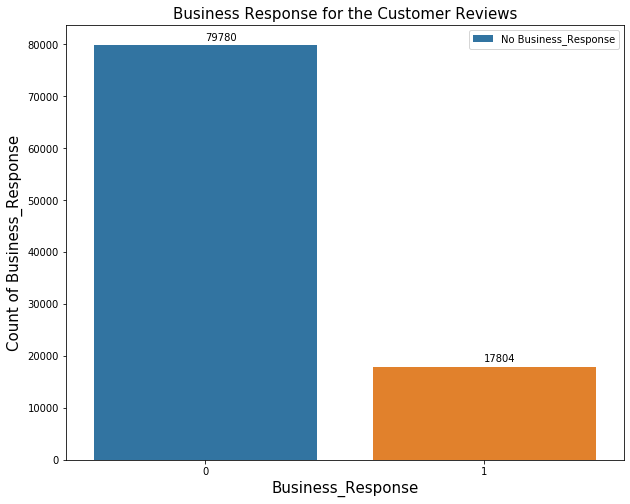
1. Yelp has an API for extracting business reviews, but we can only get a maximum of three random business reviews with it.
2. To collect data from the Yelp website, most of the web content is loaded only after the web page is rendered in the browser; thus, we can't extract the web content using python packages like urllib.
3. The Yelp website will restrict the user from collecting the Business data by repeating the businesses in search results. Also, it will result in a maximum of 240 businesses for any search criteria.
4. The HTML element tags of the Yelp webpage are dynamic and keep changing frequently.

**Best Practices Followed to handle Challenges in Data Collection:**

1. Used Selenium and Beautiful Soup to collect the data from the Yelp website, which can handle the dynamic behavior of web pages.
2. Searched the data by using the zip codes of City instead of searching with City name to extract the maximum business reviews data from the Yelp website.
3. Maintained the HTML tags in the separate JSON file so that no need to change the code files if the HTML tags of the Yelp website vary.

**Data Understanding:**

Our final data set contains data from Seattle City's Business Reviews in eight separate categories. A total of 97584 customer reviews have been Collected. Based on the data we collected, we found that 17804 customer reviews received a business response of 18.24 percent, and 79780 customer reviews received no business response of 81.75 percent.



*Fig 2: Total Reviews VS Business Response*

**Business Category wise Data:**

The Dataset contains data from eight different categories, as shown in the below table, and we can also see the Business Response rate for each category. We observed a more business response rate for the Transmission repair category from the collected data.

Category 0 1 Business Response Rate

-------------------------------------------------------------

AutoRepair 23695 6203 20.74721

Flooring 10696 1833 14.63006

HVAC 8415 1443 14.63786

Handyman 11772 3580 23.31944

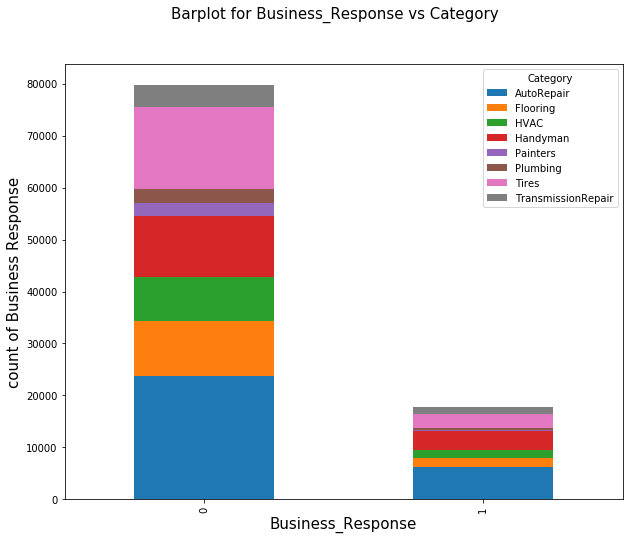
Painters 2448 292 10.65693

Plumbing 2748 356 11.46907

Tires 15818 2671 14.44643

TransmissionRepair 4188 1426 25.40078

*Table 1 : Business Response category wise*



*Fig 3: Bar plot for Business\_Response VS Category*

**Missing and Unwanted Features:**

1. The Business\_Photos\_Count and Customer\_Elite\_Year has more than 90% of data missing, so we dropped this feature from the data set.
2. The Features BusinessResponseBy, BusinessResponeDate, BusinessResponseforReview have the collinearity relationship with the Target variable Business\_Response, so we dropped these features from the dataset.

**Data Preparation:**

The dataset contains data related to business from the Yelp website in three levels, including.

**Business Level Data:**

**Business\_Name:**

The Business\_Name feature represents the name of each business which is a type of category, and there are no null values in the Business\_Name feature.

**Business\_Address:**

The Business\_Address is splitted into 3 features such as.

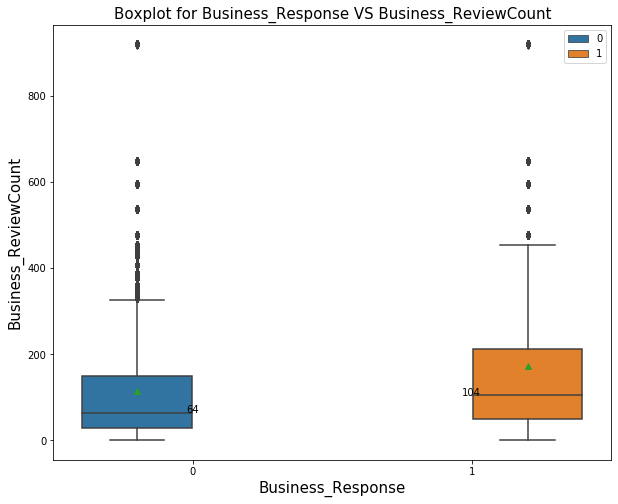
Business\_State: represents the state of business.

Business\_Zipcode : represents the Zip code of Business.

Business\_City : represents the City of Business.

**Business\_ReviewCount:**

We observed that businesses with more reviews count are more likely to respond to customer reviews than businesses with fewer reviews. We can also see that the Mean and Median of business reviews count are high for businesses responding more to customer reviews. From our data, we can see that the Median is 104 reviews for those who are actively responding.

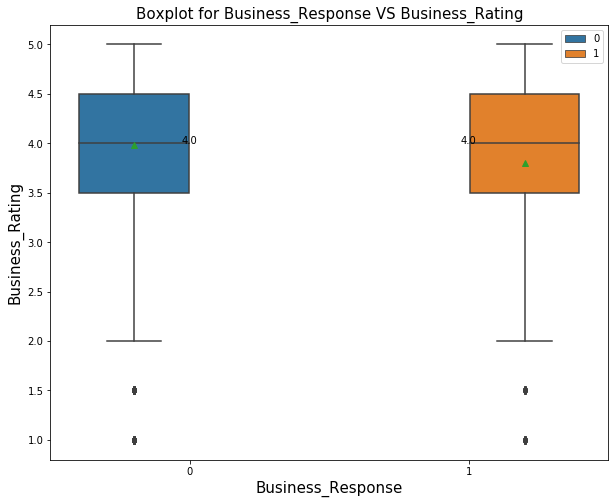


*Fig 4: Business Response VS Business Review Count*

**Business\_Rating:**

The Business Rating feature Represents the rating of businesses. We can see a boxplot for business rating based on the business response below. We observe that the median rating is the same for reviews that received a response and not receive a response.

We also see that the rating means lesser for reviews that got a business response.



*Fig 5: Business Response VS Business Rating*

**Business\_Timings :**

From the Business Timings, we created a feature Operational\_hours representing the total operational hours of each business.

**Business\_Claim\_status:**

The Business\_Claim\_status feature represents whether that business claimed or not. We observed that the business with claimed status is more likely to respond to customer reviews.

**Customer Data**

**Customer\_Name:**

The Customer\_Name feature represents the name of the customer who posted the business review, and we used the gender guesser[2] python package to translate the customer's name to gender. The converted feature Customer\_Gender will have three categories Male, Female, and unknown.

**Customer\_Friends\_count**:

Customer\_Friends\_count feature Represents the count of customer-friends in Yelp profile.

**Customer\_Reviews\_count:**

Customer\_Reviews\_count feature Represents the number of reviews given by the customer in Yelp.

**Customer\_Photos\_count**:

Customer\_Photos\_count feature Represents the number of images a customer has submitted to Yelp.

**Customer\_Elite:**

The Customer\_Elite feature indicates whether the customer is a member of the Yelp Elite group.

**Customer\_Review\_Date:**

The Customer Review Date feature indicates when the customer posted the review. We've created a new feature, Customer\_Review\_Day using the pandas' Timestamp function, which converts a date to a weekday (ex. Monday).

**Customer\_Review\_Uploaded\_Photos :**

Customer\_Review\_Uploaded\_Photos represents the number of photos uploaded by the customer along with the review.

**Customer\_Review\_Useful :**

Customer\_Review\_Useful represents the number of Yelp users who reacted that the customer review is useful.

**Customer\_Review\_Funny :**

Customer\_Review\_Funny represents the number of Yelp users who reacted that the customer review is Funny.

**Customer\_Review\_Cool :**

Customer\_Review\_Cool represents the number of Yelp users who reacted that the customer review is cool.

**Customer\_Rating :**

Customer\_Rating feature represents the rating given by the customer for the business. We can see from our dataset that the business response rate rises as the customer rating score decreases, indicating that customers with lower ratings are more likely to receive a business response.

Customer\_Rating 0 1 Business\_Response Rate

------------------------------------------------------------

1 15114 5124 25.31871

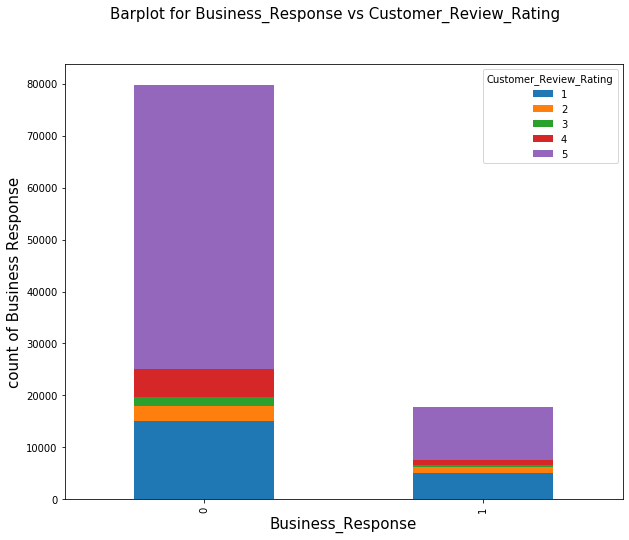
2 2875 1028 26.33871

3 1710 433 20.20532

4 5305 1002 15.88711

5 54776 10217 15.72015

*Table 2: business Response Rate based on Customer Review Rating*



*Fig 6: Business Response VS Customer Review Rating*

**Customer\_Review :**

The Customer\_Review feature represents the review given by the customer for the business. We have a total of 97584 reviews in a dataset. To capture the emotion of the customer, we performed the Sentiment Analysis on the customer reviews.

**Review Sentiment analysis with TextBlob:**

We used Textblob to understand the sentiment of customer reviews. TextBlob is a simple library that supports complex analysis and operations on textual data. The sentiment function in the TextBlob takes the Customer Review as input and returns the two values called Polarity and Subjectivity scores.

**Polarity:** The polarity value lies between the [-1,1], -1 defines the strong Negative sentiment, and 1 defines the strong Positive sentiment.

**Subjectivity:** Subjectivity refers to the amount of personal opinion, emotion, or judgment.

**Sentiment\_Score:**  Sentiment\_Score feature is created based on the polarity score. The Polarity score with greater than zero has set to Positive sentiment reviews. Equal to zero has set to Neutral, and less than zero has set to Negative reviews.

Sentiment\_Score 0 1 Business Response Rate

-------------------------------------------------------------

Negative 8775 2512 22.25569

Neutral 653 171 20.75243

Positive 70352 15121 17.69097

*Table 3: Sentiment Score VS Business Response*

From the above table, we can see that the Business response rate is more for negative reviews. We observed the same behavior in the Customer\_Reviews\_Rating feature, where the Business response rate is high for low rating reviews. So, we can say that we captured the sentiment of customer reviews correctly.

**Customer Review Statistical Features:**

When consumers write negative reviews, they tend to use more words to describe their experience than positive reviews. So, we have created few features based on the statistics of customer review such as

* 1. review\_word\_count
  2. review\_char\_count
  3. review\_sentence\_count

**Data Transformation:**

After the data preparation, we now have a dataset with features that we can use in building the classification model. However, to use features in a machine learning model, they must be in numerical format. Since we have features that include text data, we must convert them to numerical form.

**Label Encoder:** To transform categorical text features into the numerical format, we used the Label encoder from sci-kit learn. The Label encoder takes categorical text data as input and assigns a numerical label to each text category.

**Min-Max Scaler:** The features measured on different scales do not contribute equally to model fitting; thus, the features measured on different scales should be normalized using the min-max scaler.

**4. Methodology:**

We will use the Ensemble algorithms such as Random Forest, Gradient Boost, Ada Boost, and XGBoost algorithms to build the classification model. Ensemble methods are a machine learning technique that combines multiple base models to create one optimal predictive model.

We need to identify the review data for which the business owner is most likely to respond. It is a binary classification problem, and the data set we are working with is a highly imbalanced dataset with an 18% business response rate. For such dataset, the ensemble classification algorithms work well.

**Random Forest:**

Random Forest Classifier consists of many decision trees that operate as an ensemble. Each tree in the random forest spits out a class prediction, and the class with the most votes become our Random Forest model prediction.

**Random Forest hyperparameters[4]:**

A hyperparameter is a model parameter that helps to improve the model's predictive ability. For our model, we used the following parameters.

**max\_depth:**

In a random forest, the max depth parameter represents the tree's depth, from the longest path root node to the last leaf node.

**max\_features:**

The max\_features parameter specifies the maximum number of features that each tree can have. There are many ways to assign max\_features, including:

**Auto:** This will consider all the features that are available in each tree.

**Sqrt**: for each tree, it will take the square root of the total number of features.

**min\_sample\_leaf:**

It represents the minimum number of leaves present in each node before splitting; if the minimum defined samples are not present, the tree will not grow any further, which will help prevent model overfitting.

**min\_samples\_split:**

It tells the tree in a random forest how many observations are needed in each node for it to split. By adjusting the min\_samples\_split feature we can avoid the model over fitting.

**n\_estimators:**

It represents the total number of trees in the model; as the number of trees grows, it increases the model's performance; however, as the number of trees grows, it increases the model's time complexity.

**Performance tuning with GridsearchCV:**

Trying several parameter combinations manually to find the optimal parameter combination that produces the best result is complicated.GridsearchCV can help in the search for the best performing parameters in the parameter space for a particular algorithm and data set. It runs through all the different parameters fed into the parameter grid and produces the best combination of parameters based on a scoring metric of your choice (f1, recall, accuracy).

**Random Forest model Performance:**

By using the GridsearchCV and the scoring metric as f1 we got the best model performance results as below.

precision recall f1-score support

----------------------------------------------------------------

0 0.88 0.96 0.92 15956

1 0.70 0.40 0.51 3561

accuracy 0.86 19517

macro avg 0.79 0.68 0.71 19517

weighted avg 0.85 0.86 0.84 19517

*Table 3: Random Forest Model Performance*

We got the model accuracy as 0.86 and for class 1 we got recall value as 0.40 and f1 score as 0.51.

**Gradient Boosting:**

Gradient Boosting is an ensemble machine learning algorithm that combines the weakest learners to improve prediction accuracy. Any tree T in the model is constructed based on the model's previous tree T-1 results[5]. The tree outcomes that correctly predicted are given a lower weight, while those incorrectly classified are given a higher weight. A new data set is built based on the weights, and predictions are made. This process repeats for many iterations, all trees are given a weight based on their accuracy, and a consolidated result is generated.

**Gradient Boosting Parameters:**

We used the below hyper parameters for tuning the Gradient Boosting model.

**max\_depth:** The max depth parameter represents the tree's depth, from the longest path root node to the last leaf node. The model tends to overfit as the depth of the tree grows.

**n\_estimators:** It represents the total number of trees in the model; as the number of trees grows, it increases the model's performance; however, as the number of trees grows, it increases the model's time complexity.

**min\_samples\_split:** It tells the tree in a random forest how many observations are needed in each node for it to split. By adjusting the min\_samples\_split feature we can avoid the model over fitting.

**learning\_rate:**  The learning rate is the time it takes for an error to be corrected from one tree to the next. The model corrects the prediction error faster as the learning rate increases, and the model corrects the prediction error at a slower rate as the learning rate decreases.

**Gradient Boosting Model Performance:**

We got the following performance results for the best model by using GridsearchCV and cross-fold validation as ten and scoring metric as f1.

precision recall f1-score support

-------------------------------------------------------------

0 0.88 0.95 0.91 15956

1 0.64 0.41 0.50 3561

accuracy 0.85 19517

macro avg 0.76 0.68 0.71 19517

weighted avg 0.84 0.85 0.84 19517

*Table4: Gradient Boosting model Performance*

We got the model accuracy as 0.85 and for class 1 we got the recall value as 0.41 and f1 score as 0.50.

**XGBoost :**

XGBoost has similar behavior to a decision tree in that each tree is split based on a specific range of values in different columns, but unlike decision trees, each node is given a weight. On each iteration a new tree is created, and new node weights are assigned based on similarity score and gain. For each tree, the training examples with the biggest error from the previous tree are given extra attention so that the next tree will optimize more for these training examples, this is the boosting part of the algorithm[6].

Tuning Parameters of XGBoost:

**colsample\_bytree:**  The number of columns that must be considered in the model is represented by this parameter. The range of values is 0 to 1. If colsample\_bytree = 0.5, half of the columns are considered when constructing a tree.

**max\_depth:** The max depth parameter represents the tree's depth, from the longest path root node to the last leaf node. The model tends to overfit as the depth of the tree grows.

**min\_child\_weight:** min\_child\_weight is minimum sum of instance weight needed in a child; The tree won't grow if the tree partition step results in a leaf node with a total of instance weight less than min child weight.

**learning\_rate:**  The learning rate is the time it takes for an error to be corrected from one tree to the next. The model corrects the prediction error faster as the learning rate increases, and the model corrects the prediction error at a slower rate as the learning rate decreases.

**XGBoost model Performance:**

By using the GridsearchCV and the scoring metric as f1 we got the best model performance results as below.

precision recall f1-score support

---------------------------------------------------------------

0 0.88 0.95 0.91 15956

1 0.65 0.44 0.52 3561

accuracy 0.85 19517

macro avg 0.77 0.69 0.72 19517

weighted avg 0.84 0.85 0.84 19517

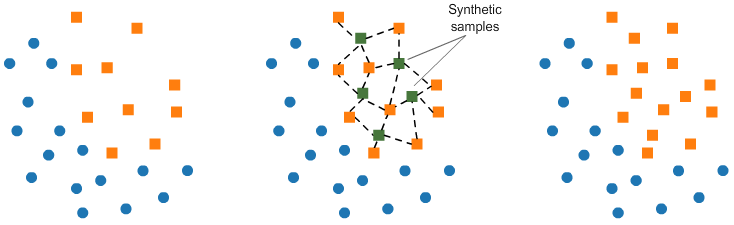
*Table 5: XGBoost Model Performance*

We got the model accuracy as 0.85 and for class 1 we got recall value as 0.44 and f1 score as 0.52.

**Oversampling with SMOTE :**

SMOTE (synthetic Minority Oversampling TEchnique) consists of synthesizing elements for minority class, based on those already exists. It works randomly by picking a point from the minority class and computing the k-nearest neighbors for that point, the synthetic points are added between the chosen point and its neighbors[7].

In the fig:7 we can see that synthetic samples are created for the minority class by using the k- nearest neighbors’ algorithm.



*Fig 7: Oversampling with SMOTE*

**Models Performance with Oversampling SMOTE Data:**

The minority class 1 sample will become equivalent to the majority class after oversampling with smote, 79780 samples for each class.

We obtained the following model results by applying ensemble algorithms to the above oversampled dataset.

Model precision recall f1-score Accuracy

---------------------------------------------------------------

Random Forest 0.93 0.89 0.91 0.91

XGBoost 0.94 0.86 0.90 0.90

Gradient Boosting 0.88 0.79 0.83 0.84

*Table 6: Models performance with Oversampling Smote data.*

Without hyperparameter tuning, we achieved an accuracy of 0.91 for Random Forest, an f1score of 0.91, and a recall of 0.89.

**5.Results and Analysis**

**References**:

1. Ryan Erskine ; 20 Online Reputation Statistics That Every Business Owner Needs To Know ; <https://www.forbes.com/sites/ryanerskine/2017/09/19/20-online-reputation-statistics-that-every-business-owner-needs-to-know/?sh=ace73c4cc5c9>
2. Gender-guesser ; <https://pypi.org/project/gender-guesser/>
3. Parthvi shah; Sentiment Analysis using TextBlob ; <https://towardsdatascience.com/my-absolute-go-to-for-sentiment-analysis-textblob-3ac3a11d524>
4. Sharoon Saxena; A Beginner’s Guide to Random Forest Hyperparameter Tuning; <https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/>
5. AARSHAY JAIN; Complete Machine Learning Guide to Parameter Tuning in Gradient Boosting (GBM) in Python; <https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/>
6. XGBOOST ; <https://www.analyseup.com/python-machine-learning/xgboost-parameter-tuning.html>
7. SMOTE; Resampling strategies for imbalanced datasets; <https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets>