# Using Yelp Dataset to Predict Restaurant Closure

By: Charlie Lee

Date: 6/27/2021

# TABLE OF CONTENTS

# ABSTRACT

This report is not affiliated with or endorsed by any organization. The findings and conclusions are for informational purposes only. Regardless of having positive reviews and ratings, and being in good location, many restaurants still go out of business. So what if restaurant closure can be predicted based on the attributes and categories that restaurant owners choose to define their business with? This report covers if a restaurant closure can be predicted prior to starting the business. This report uses the dataset from Yelp which is an online platform which contains enormous amount of data for all kinds of businesses. Data cleaning and wrangling section presents the approach taken to filter the dataset for only restaurants. Exploratory data analysis section displays various types of graphs for open vs. closed restaurants. Inferential statistics section discusses the statistical approach prior to selecting the most important features prior to machine learning. Machine learning section covers how the imbalanced dataset was analyzed, as well as different approaches for attributes and categories. For categories, natural language processing (NLP) is introduced to vectorize the words chosen by the restaurant owners.

# 1. Introduction

## 1.1 Problem Statement

Prior to the COVID-19 pandemic, the restaurant industry projected sales of approximately $899 billion in 2020. In reality, this pandemic impacted the restaurant industry catastrophically, causing the sales to crash down 19.2%, which resulted in $240 billion below the forecasted amount. It is reported that during this time, approximately 110,000 foodservice businesses closed either temporarily or permanently (National Restaurant Association - National Statistics). The downfall of the restaurant industry has also placed many employees out of jobs, as well as creating far fewer jobs, which is reported to be 3.1 million less than expected.

According to the National Restaurant Association's State of the Industry Report, "2021 sales are projected to climb 10.2%, though not nearly enough to recover from the steep hole caused by the pandemic." In such times of needed recovery, the stakeholders (in this report, are identified as current and future restaurant owners, as well as its investors) search for ways to get their business back up on its feet.

Upserve reports that 90% of guests check out a restaurant online before eating there, and 33% of people read other customers' reviews before selecting a place to eat. I, for one, am guilty of belonging to both these groups. I tend to stay away from restaurants with poor reviews and star ratings, and I strongly believe that I'm not the only individual to do so. These statistics show the importance of online platforms such as Yelp and how they can affect customers' decisions to visit restaurants or not. Per Harvard Business School, an extra star on the Yelp platform can translate into an increase in revenue of between 5% and 9%.

The stakeholders wonder what could cause their restaurant to perform better year over year. Obvious factors that will come to their mind at first are reviews, ratings, and locations. It could be true that these factors contribute very heavily to the restaurant closure, but regardless of having positive reviews, ratings, and being in a good location, some restaurants still tend to go out of business. With the statistics shown above, the stakeholders can question themselves, what if they can reduce the possibility of their restaurant closure based on their input in Yelp platform?

Yelp allows the stakeholders to enter in information about their restaurant's attributes and categories. Attributes contain information such as, but are not limited to: if the restaurant allows take out, has parking (if so, is it garage, street, validated, valet, or other), and/or its ambience (romantic, for kids, casual, and etc.). Categories contain information such as, but are not limited to: restaurants, sushi bars, food trucks, pet stores, tax services, and etc.

## 1.2 Client – Restaurant Investors, Banks

Many business investors do not know enough about the restaurant industry to be confident for the possible investment. Based on the report provided, the investors will get a better idea of what variables to consider before investing. Banks will have a better understanding of the risk when loaning the money to a certain restaurant entrepreneur with their proposal. Restaurant owners and the stakeholders will get a better idea of what variables to consider when dealing with Yelp platform.

## 1.3 Dataset

The dataset is available from https://www.yelp.com/dataset. This dataset was collected by Yelp and is a subset of their full dataset. There are 6 different JSON files within this dataset (only 3 files will be used). These JSON files are mostly in a flat format with some being either single or double nested.

- business.json
    - Contains business data including location data, attributes, and categories.
    - Has a 'column' called 'is_open'. 0 or 1 for closed or open, respectively.
- review.json
    - Contains full review text data including the user_id that wrote the review and the business_id the review is written for.

In-depth documentation of the dataset can be found on Yelp's website (https://www.yelp.com/dataset/documentation/main). The dataset contains over 8 million user reviews, 160,000 businesses, and 1.2 million business attributes like hours, parking, availability, and ambience in 10 metropolitan areas. Currently, the metropolitan areas centered on Montreal,

Calgary, Toronto, Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, Madison, and Cleveland, are included in the dataset. The dataset is approximately 10 gigabytes.

The review count may be different from the actual number of reviews since the review count shows the total number of reviews at the time of data collection. Also, the businesses in the dataset are those that have had at least 3 reviews older than 14 days. The user reviews data only contains those that Yelp recommended at the time of data collection. Following are the examples of the documentation provided by Yelp regarding their business and reviews dataset:

```
{
    // string, 22 character unique string business id
    "business_id": "tnhfDv5Il8EaGSXZGiuQGg",

    // string, the business's name
    "name": "Garaje",

    // string, the full address of the business
    "address": "475 3rd St",

    // string, the city
    "city": "San Francisco",

    // string, 2 character state code, if applicable
    "state": "CA",

    // string, the postal code
    "postal code": "94107",

    // float, latitude
    "latitude": 37.7817529521,

    // float, longitude
    "longitude": -122.39612197,

    // float, star rating, rounded to half-stars
    "stars": 4.5,

    // integer, number of reviews
    "review_count": 1198,

    // integer, 0 or 1 for closed or open, respectively
    "is_open": 1,

    // object, business attributes to values. note: some attribute valu
    "attributes": {
        "RestaurantsTakeOut": true,
        "BusinessParking": {
            "garage": false,
            "street": true,
            "validated": false,
            "lot": false,
            "valet": false
        },
    },

    // an array of strings of business categories
    "categories": [
        "Mexican",
        "Burgers",
        "Gastropubs"
    ],

    // an object of key day to value hours, hours are using a 24hr cloc
    "hours": {
        "Monday": "10:00-21:00",
        "Tuesday": "10:00-21:00",
        "Friday": "10:00-21:00",
        "Wednesday": "10:00-21:00",
        "Thursday": "10:00-21:00",
        "Sunday": "11:00-18:00",
        "Saturday": "10:00-21:00"
    }
}
```

```
{
    // string, 22 character unique review id
    "review_id": "zdSx_SD6obEhz9VrW9uAWA",

    // string, 22 character unique user id, maps to the user in user.js
    "user_id": "Ha3iJu77CxlrFm-vQRs_8g",

    // string, 22 character business id, maps to business in business.j
    "business_id": "tnhfDv5Il8EaGSXZGiuQGg",

    // integer, star rating
    "stars": 4,

    // string, date formatted YYYY-MM-DD
    "date": "2016-03-09",

    // string, the review itself
    "text": "Great place to hang out after work: the prices are decent,

    // integer, number of useful votes received
    "useful": 0,

    // integer, number of funny votes received
    "funny": 0,

    // integer, number of cool votes received
    "cool": 0
}
```

**Fig 1.** Business dataset (left), Reviews dataset (right)

# 2. Data wrangling and cleaning

## 2.1 Nested Json Dataset and Handling Null values

As seen in the below Fig 2, the values were 'hidden' due to the nested Json dataset. It shows how the data looked prior to extracting the true information.

| attributes | categories |
|---|---|
| {'RestaurantsTableService': 'True', 'WiFi': 'u... | Gastropubs, Food, Beer Gardens, Restaurants, B... |
| {'RestaurantsTakeOut': 'True', 'RestaurantsAtt... | Salad, Soup, Sandwiches, Delis, Restaurants, C... |
| {'BusinessAcceptsCreditCards': 'True', 'Restau... | Antiques, Fashion, Used, Vintage & Consignment... |
| {'RestaurantsPriceRange2': '1', 'BusinessAccep... | Beauty & Spas, Hair Salons |
| {'GoodForKids': 'False', 'BusinessParking': '{... | Gyms, Active Life, Interval Training Gyms, Fit... |

**Fig 2**. Nested Json Data

The *attributes* column was extracted into many columns in order to bring out their true values. Upon doing so, there were many null values. These were filled with the word 'Not listed.' Such decision was made because even though the owner had a choice put values such as 'True' or 'False' for certain attributes, he or she did not. There were few columns which contained the value 'None' instead of being null, these were probably being used as a placeholder for null. These 'None' were replaced with the word 'Not listed' as well.

| attributes.validated | attributes.lot | attributes.valet | attributes.touristy | attributes.hipster |
|---|---|---|---|---|
| False | False | False | False | False |
| False | False | False | False | False |
| False | False | False | False | False |
| Not listed | Not listed | Not listed | Not listed | Not listed |
| False | True | False | False | False |

**Fig 3.** Attributes Extracted

For the *categories* column, it was discovered that the list of words were separated by commas (Fig 2). Therefore, instead of handling it like the above method, the decision was made to implement NLP (Natural Language Processing). This is covered more in depth in NLP section of the machine learning.

## 2.2 Filtering for Restaurants

Yelp dataset consisted of other types of businesses than the restaurants (160,585 total businesses). Using the reviews.json file (contains over 8 million user reviews); the filter was applied if the user used the word ‘food’ or ‘restaurant’ in their review. There were over 3.41 million reviews after filtering which translated into 61,328 restaurants.

## 2.3 Bias

Since the null values for all *attribute* columns were filled with ‘Not listed’, it was possible to have columns which contained ‘Not listed’ as its majority value. Therefore, if the column contained more than 85% of ‘Not listed’, it was removed from the dataset. Columns such as: *attributes.BusinessAcceptsBitcoin, attributes.halal, attributes BYOB,* and etc. were dropped.

Also, since using reviews.json logic was applied to filter for restaurants, it is possible that small restaurants which the users have never left a review could have not been selected for the analysis.

# 3. Exploratory Data Analysis

## 3.1 Open vs. Closed Star Ratings

Although it is a known fact that restaurants close regardless of good or bad ratings, below graph attempts to visualize the difference.

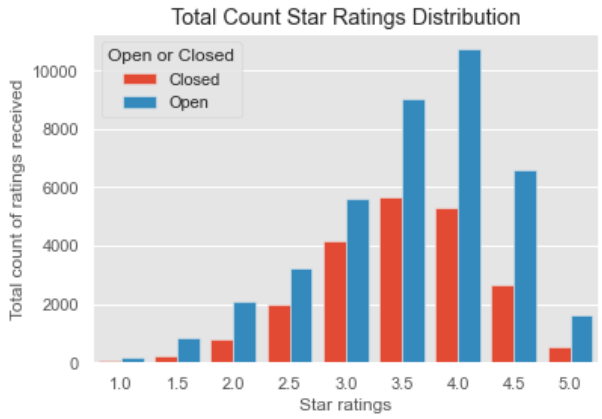



**Fig 4.** Open vs. closed restaurants ratings received (total)

The graph above did not show the comparison clearly due to uneven counts of restaurants in the dataset. Thus, comparing them by total count of ratings was unclear. Below graph visualizes entire star ratings that users left by the percentage to compare if higher star ratings tend to have more open restaurants.

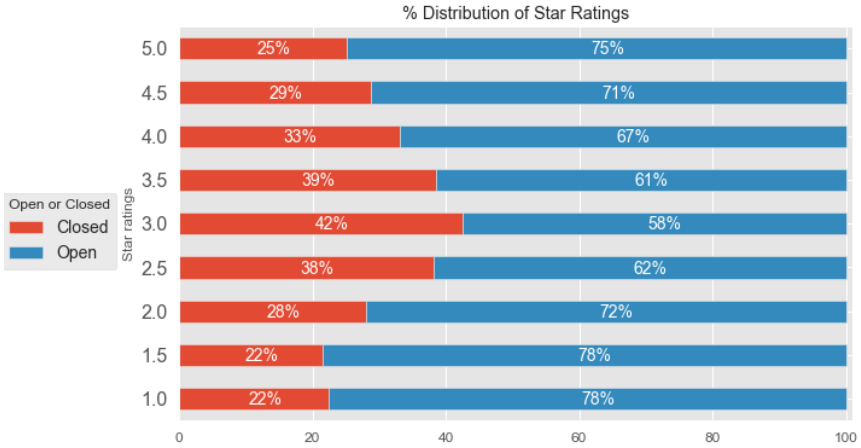



**Fig 5.** Open vs. closed restaurants ratings received (percentage)

## 3.2 Open vs. Closed Review Counts

Just like spectated from above, the same concept was applied to visualize the number of reviews received based on the star ratings and the restaurants' business status.

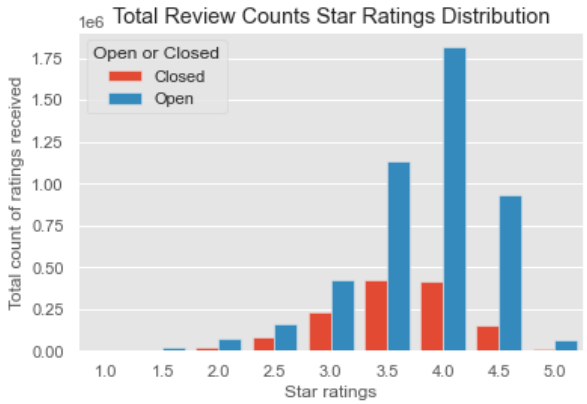



**Fig 6.** Open vs. closed restaurants reviews received (total)

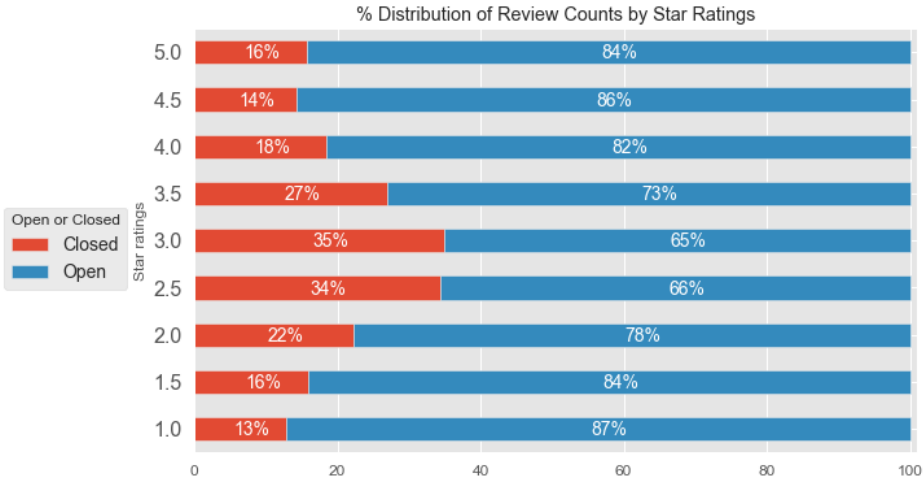



**Fig 7.** Open vs. closed restaurants reviews received (percentage)

## 3.3 Five Largest Restaurant Chains

Per Figure 8 below, the five largest restaurant chains in the dataset were: Starbucks, McDonald's, Subway, Dunkin, and Wendy's.

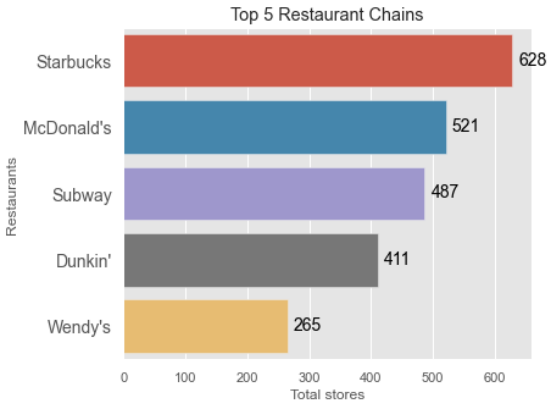



**Fig 8.** 5 Largest restaurant chains

## 3.4 Open vs. Closed Top 10 Appearing Words in '*categories*'

The most common word that both open and closed restaurants used to categorize themselves in the *category* column was 'restaurants', followed by the word 'food'.

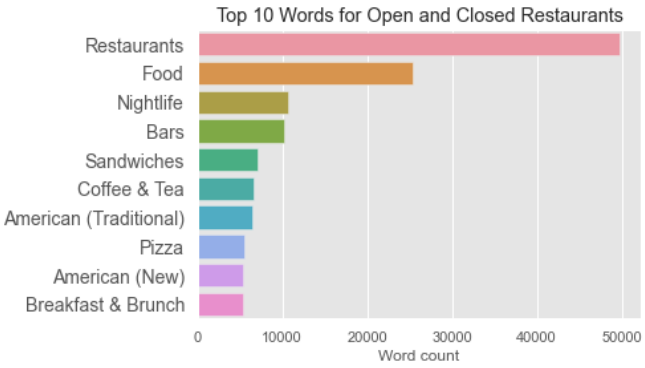



**Fig 9.** Top 10 words in both open and closed restaurants

The top 4 most appearing words for both open and closed restaurants were the same, which were: restaurants, food, nightlife, and bars. As shown below, 'fast food' is present in open restaurants but not present in closed restaurants. Fast food restaurants are most likely to be chains owned by a corporate and are less likely to go out of business.

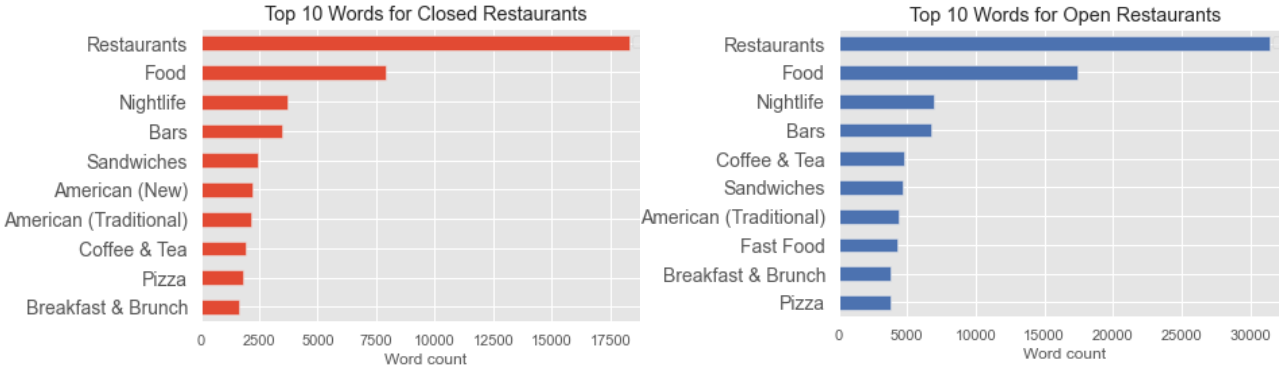



**Fig 10.** Top 10 words in closed restaurants (left),
Top 10 words in open restaurants (right)

Each of the top 10 word counts was divided by the number of open or closed restaurants to visualize the difference. Since these words were chosen by the restaurant owners, we can compare how likely the owner of either open or closed restaurant will use these words.

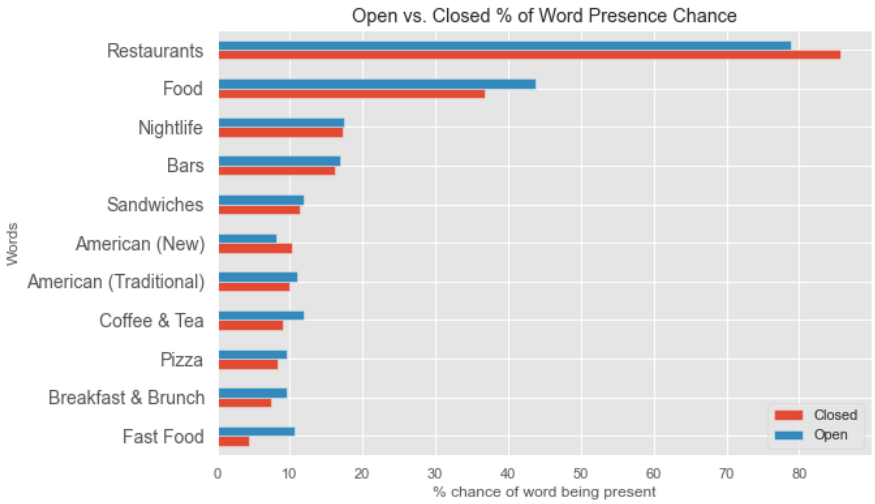



**Fig 11.** Word comparisons for open vs. closed restaurants

The owners of the closed restaurants categorized themselves as ‘restaurants’ more often than the open restaurants. Also, it can be seen that restaurants that categorized themselves as ‘fast food’ and ‘food’ were more likely to stay in business.

## 3.5 Open vs. Closed Attribute Column Value Percentages

The visualizations can be found in the attached Jupyter notebook (Yelp_EDA.ipynb).

# 4. Inferential statistical data analysis

## 4.1 Chi Square Test for Independence

Feature selections were performed using a filter based method, specifically chi-square test for independence. Chi square test for independence was performed between the '*is_open*' and the '*attribute*' columns.

**Chi Square Test for Independence Hypothesis**

$H_0$ : There is no association between the '*is_open*' and the selected attribute

$H_a$ : There is an association between the '*is_open*' and the selected attribute

Null hypothesis was rejected if p-value was less than 0.05 and if chi statistic was greater than the critical value.

**Table 1.** Chi square test for independence

| Attribute | p-value | Rej / Fail | Attribute | p-value | Rej / Fail |
|---|---|---|---|---|---|
| RestaurantsTableService | 0.000 | Reject | Street | 0.000 | Reject |
| WiFi | 0.000 | Reject | Validated | 0.000 | Reject |
| BikeParking | 0.000 | Reject | Lot | 0.000 | Reject |
| BusinessAcceptsCreditCards | 0.000 | Reject | Valet | 0.000 | Reject |
| RestaurantsReservations | 0.000 | Reject | Touristy | 0.000 | Reject |
| WheelchairAccessible | 0.000 | Reject | Hipster | 0.000 | Reject |
| Caters | 0.000 | Reject | Romantic | 0.000 | Reject |
| OutdoorSeating | 0.000 | Reject | Divey | 0.000 | Reject |
| RestaurantsGoodForGroups | 0.000 | Reject | Intimate | 0.000 | Reject |
| HappyHour | 0.000 | Reject | Trendy | 0.000 | Reject |
| RestaurantsPriceRange2 | 0.000 | Reject | Upscale | 0.000 | Reject |
| HasTV | 0.000 | Reject | Classy | 0.000 | Reject |
| Alcohol | 0.000 | Reject | Casual | 0.000 | Reject |
| DogsAllowed | 0.000 | Reject | Dessert | 0.000 | Reject |
| RestaurantsTakeOut | 0.000 | Reject | Latenight | 0.000 | Reject |
| NoiseLevel | 0.000 | Reject | Lunch | 0.000 | Reject |
| RestaurantsAttire | 0.000 | Reject | Dinner | 0.000 | Reject |
| RestaurantsDelivery | 0.000 | Reject | Brunch | 0.000 | Reject |
| GoodForKids | 0.000 | Reject | Breakfast | 0.000 | Reject |
| Garage | 0.000 | Reject | | | |

## 4.2 Post Hoc Test – Bonferroni correction

Comparing multiple classes against each other would means that the error rate of a false positive compound with each test. For example, if the first test is at p-value level 0.05 means there is a 5% chance of a false positive; if there are multiple classes, the test after that would compounding the error with the chance become 10% of a false positive, and so forth. With each subsequent test, the error rate would increase by 5%. Due to this nature, Bonferroni-adjusted method was used

for correcting the p-value. The formula is p/N, where p= the p-value of the original test and N= the number of planned pairwise comparisons. After the post hoc test, the results were still the same as shown in the table 1.

## 4.3 Multicollinearity

Since all features rejected null hypothesis for Chi Square Test for Independence, correlation (spearman) was used to detect possible multicollinearity between the features. LabelEncoding was used the display the correlation between the columns consisting of "true, false, none, or not listed". The initial guess was that the attribute columns such as romantic and intimate will show high correlation. It is assumed here that the restaurant owners would've selected correlating attributes to either True or False.

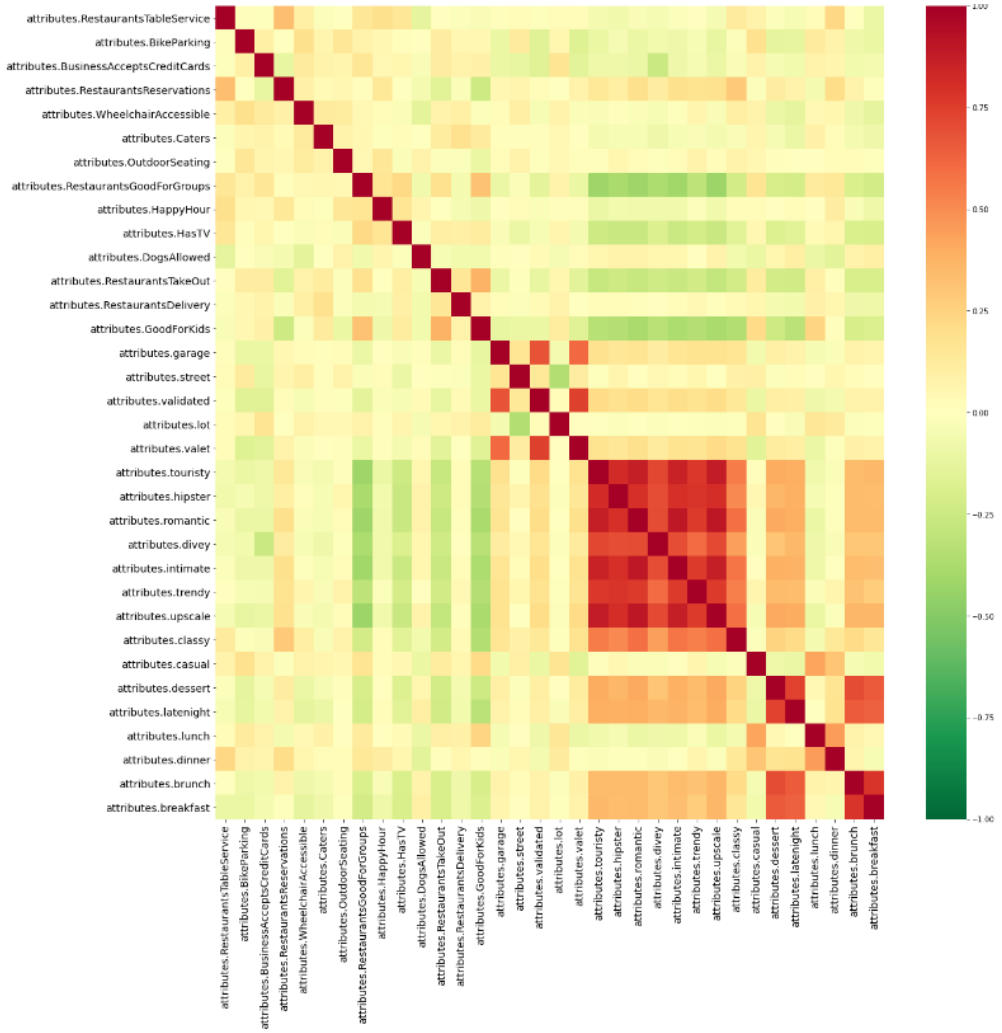



**Fig 12.** Heat correlation plot for attribute features

As shown from above Figure 12, there were strong positive correlations between:

- Brunch and Breakfast
- Intimate and Romantic
- Upscale and Romantic
- Trendy and Hipster
- Etc.

Also, there were weak negative correlations between:

- Late night and Good for Kids
- Classy and Good for Kids
- Upscale and Takeout
- Romantic and Takeout
- Parking Street and Parking Lot
- Etc.

Chances were that if the correlation coefficient was greater than 0.8, multicollinearity was likely to be present. Following attribute features had correlation coefficient that was greater than 0.8: *'attributes.touristy', 'attributes.hipster', 'attributes.romantic', 'attributes.intimate', 'attributes.upscale'*. Using SelectKBest and chi2 as the scoring function, single column was selected ('*attributes.touristy*') and the rest were dropped.

# 5. Machine learning - Attributes

## 5.1 Imbalanced Dataset

Prior to performing the machine learning, it is to be noted that the target variable (is_open) was imbalanced.

```
print('There are {} closed restaurants.'.format(len(df_ml[df_ml.is_open == 0])))
print('There are {} open restaurants.'.format(len(df_ml[df_ml.is_open == 1])))
```

```
There are 21428 closed restaurants.
There are 39900 open restaurants.
```

**Fig 13.** Imbalanced data

As shown in the above figure, there were 21,428 closed (is_open = 0) restaurants and 39,900 open (is_open = 1) restaurants (1:1.86 ratio). Since the data was imbalanced and neither precision nor recall outweighed the other, f-1 score was used to measure the model performance.

## 5.2 Logistic Regression

The baseline model was tested using the raw data with 30% as the test size. The baseline model resulted as shown below:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.63 | 0.69 | 6429 |
| 1 | 0.82 | 0.90 | 0.86 | 11970 |
| accuracy | | | 0.80 | 18399 |
| macro avg | 0.79 | 0.76 | 0.77 | 18399 |
| weighted avg | 0.80 | 0.80 | 0.80 | 18399 |

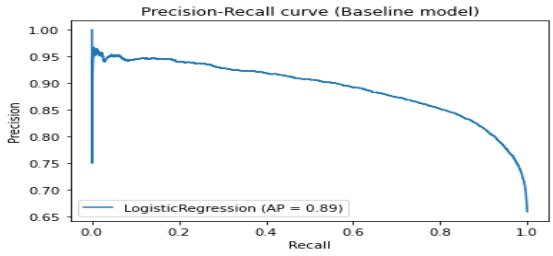



**Fig 14.** Weighted Avg f-1 score (left), Precision-recall curve (right)

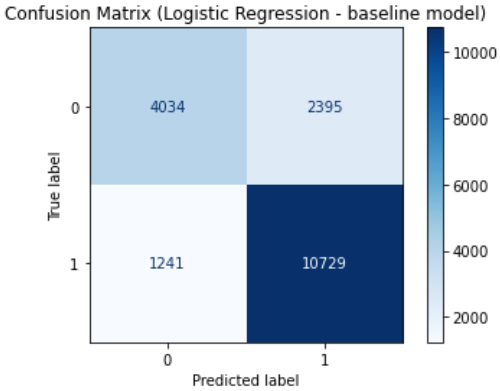



**Fig 15.** Confusion matrix – baseline model

Two different balancing techniques (class weight and upsampling performance) were applied on the baseline model to see if one performs better than the other.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.65 | 0.75 | 0.70 | 6429 |
| 1 | 0.86 | 0.79 | 0.82 | 11970 |
| accuracy | | | 0.77 | 18399 |
| macro avg | 0.75 | 0.77 | 0.76 | 18399 |
| weighted avg | 0.79 | 0.77 | 0.78 | 18399 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.65 | 0.75 | 0.70 | 6429 |
| 1 | 0.86 | 0.79 | 0.82 | 11970 |
| accuracy | | | 0.77 | 18399 |
| macro avg | 0.75 | 0.77 | 0.76 | 18399 |
| weighted avg | 0.79 | 0.77 | 0.78 | 18399 |

**Fig 16.** Class weight applied (left), Upsampling applied (right)

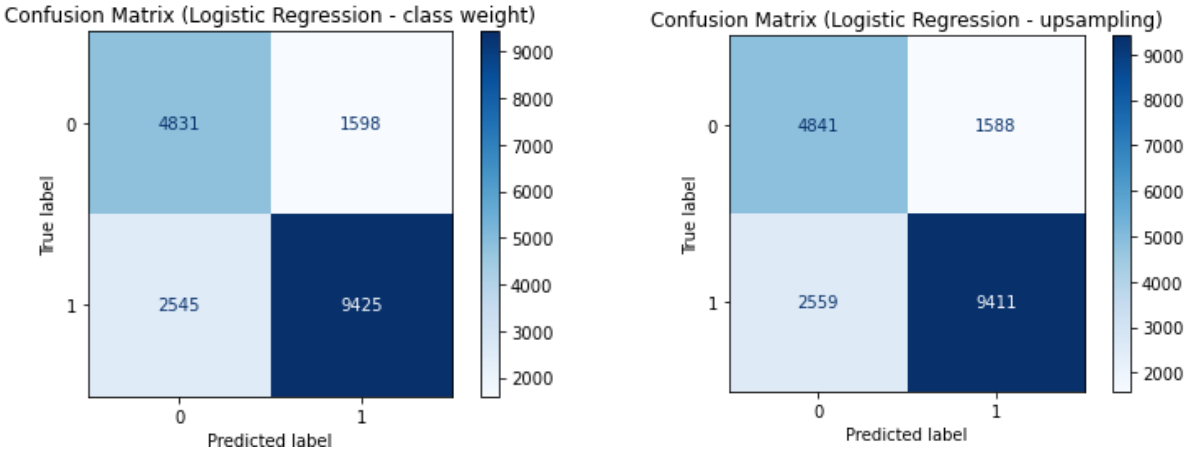
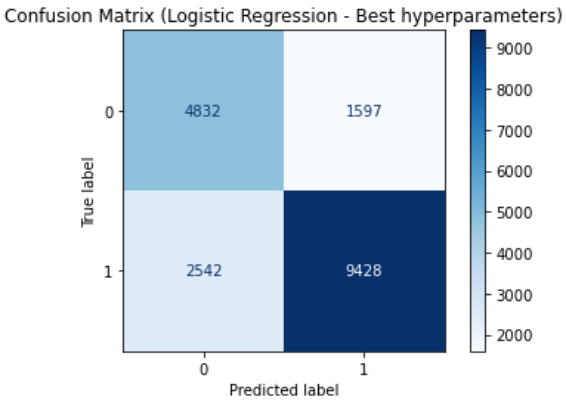



**Fig 17.** Class weight applied (left), Upsampling applied (right)

Both techniques resulted in the same weighted average f-1 scores, but 'class weight' had better average f-1 cross validation score (0.82 vs. 0.77). The model tuning was performed using RandomSearchCV. The overall model performance did not improve. Best weighted average f-1 score was 0.78, while best cross validation score with 5 k-fold was 0.82




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.66 | 0.75 | 0.70 | 6429 |
| 1 | 0.86 | 0.79 | 0.82 | 11970 |
| accuracy | | | 0.78 | 18399 |
| macro avg | 0.76 | 0.77 | 0.76 | 18399 |
| weighted avg | 0.79 | 0.78 | 0.78 | 18399 |

**Fig 18.** Best hyperparameters, class weight applied

## 5.3 Random Forest

The baseline model was tested using the raw data with 30% as the test size. The baseline model resulted as shown below:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.66 | 0.72 | 6429 |
| 1 | 0.83 | 0.91 | 0.87 | 11970 |
| accuracy | | | 0.82 | 18399 |
| macro avg | 0.81 | 0.78 | 0.79 | 18399 |
| weighted avg | 0.82 | 0.82 | 0.82 | 18399 |

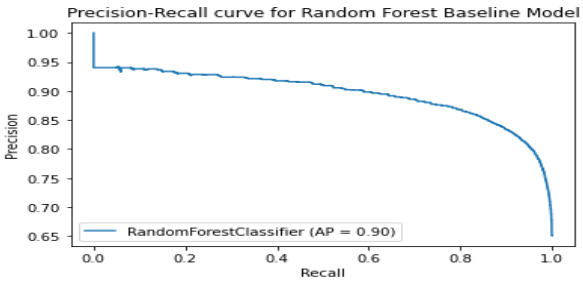



**Fig 19.** Weighted Avg f-1 score (left), Precision-recall curve (right)

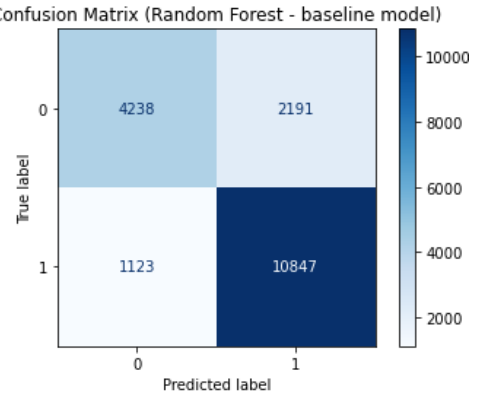



**Fig 20.** Confusion matrix – baseline model

Just like from the logistic regression, two different balancing techniques (class weight and upsampling performance) were applied on the baseline model to see if one performs better than the other.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.67 | 0.72 | 6429 |
| 1 | 0.83 | 0.90 | 0.86 | 11970 |
| accuracy | | | 0.82 | 18399 |
| macro avg | 0.81 | 0.78 | 0.79 | 18399 |
| weighted avg | 0.81 | 0.82 | 0.81 | 18399 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.68 | 0.72 | 6429 |
| 1 | 0.84 | 0.89 | 0.86 | 11970 |
| accuracy | | | 0.81 | 18399 |
| macro avg | 0.80 | 0.78 | 0.79 | 18399 |
| weighted avg | 0.81 | 0.81 | 0.81 | 18399 |

**Fig 21.** Class weight applied (left), Upsampling applied (right)

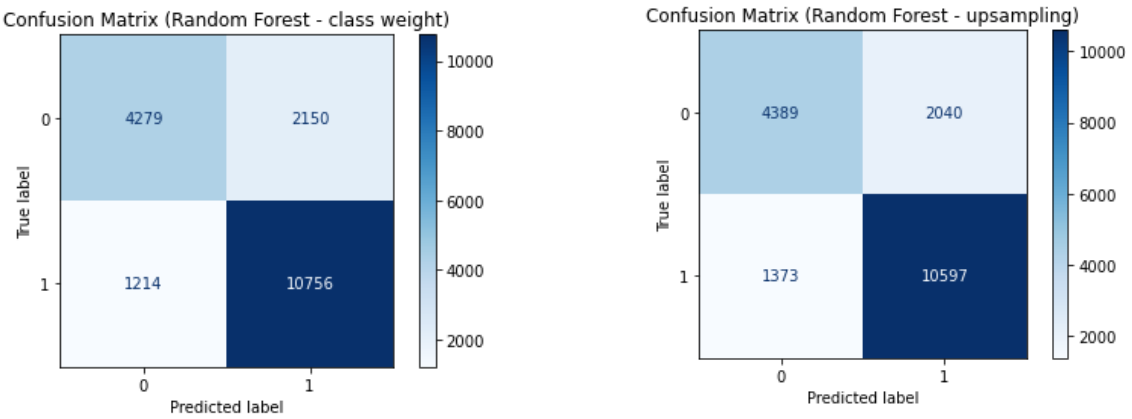
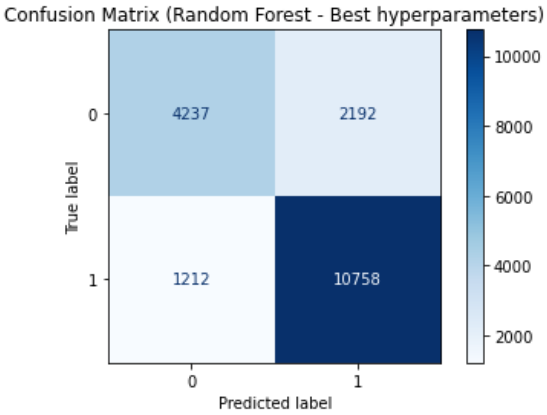



**Fig 22.** Class weight applied (left), Upsampling applied (right)

Both techniques resulted in the same weighted average f-1 scores, but 'upsampling' had better average f-1 cross validation score (0.86 vs. 0.89). The model tuning was performed using RandomSearchCV. The overall model performance did not improve. Best weighted average f-1 score was 0.82, while best cross validation score with 5 k-fold was 0.89.




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.66 | 0.71 | 6429 |
| 1 | 0.83 | 0.90 | 0.86 | 11970 |
| accuracy | | | 0.81 | 18399 |
| macro avg | 0.80 | 0.78 | 0.79 | 18399 |
| weighted avg | 0.81 | 0.81 | 0.81 | 18399 |

**Fig 23.** Best hyperparameters, upsampling applied

## 5.4 XGBoost

The baseline model was tested using the raw data with 30% as the test size. The sample weight was applied using scale_pos_weight parameter. The baseline model resulted as shown below:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.59 | 0.70 | 6429 |
| 1 | 0.81 | 0.95 | 0.88 | 11970 |
| accuracy | | | 0.83 | 18399 |
| macro avg | 0.84 | 0.77 | 0.79 | 18399 |
| weighted avg | 0.83 | 0.83 | 0.82 | 18399 |

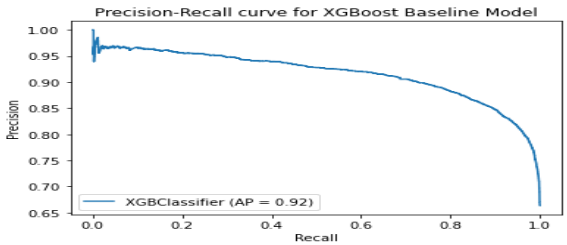



**Fig 24.** Weighted Avg f-1 score (left), Precision-recall curve (right)

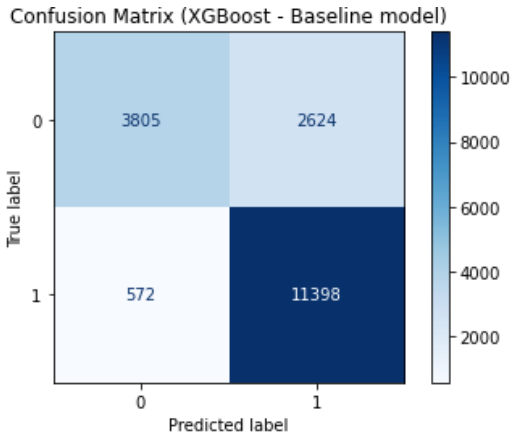
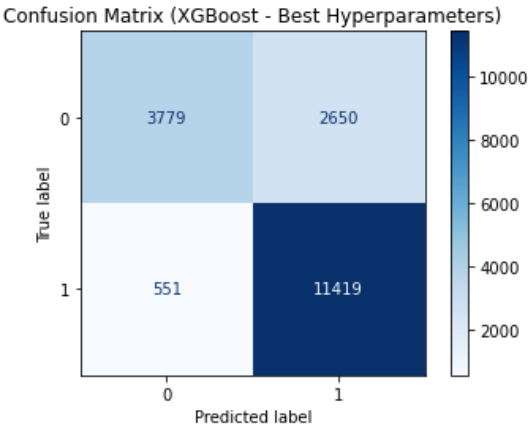



**Fig 25.** Confusion matrix – baseline model

The model tuning was performed using RandomSearchCV. The overall model performance did not improve. Best weighted average f-1 score was 0.82, while best cross validation score with 5 k-fold was 0.87.




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.59 | 0.70 | 6429 |
| 1 | 0.81 | 0.95 | 0.88 | 11970 |
| accuracy | | | 0.83 | 18399 |
| macro avg | 0.84 | 0.77 | 0.79 | 18399 |
| weighted avg | 0.83 | 0.83 | 0.82 | 18399 |

**Fig 26.** Best hyperparameters, scale_pos_weight applied

## 5.5 Deep Learning - Keras

Keras model performed the best with 10 epochs, 5 layers with 128 nodes in each (‘relu’ activation). The model was optimized with ‘adam’ and the loss was set to ‘binary_crossentropy’ . The imbalanced data was counted for using the class_weight parameter. The model resulted in weighted average f1 score of 0.81 and model performance of 0.86.

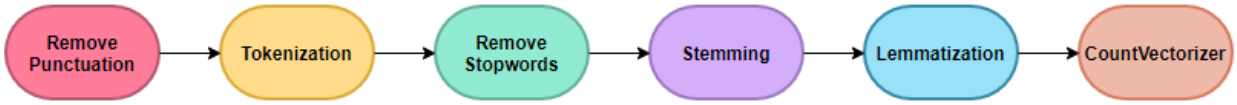
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.75 | 0.68 | 0.72 | 6429 |
| 1 | 0.84 | 0.88 | 0.86 | 11970 |
| accuracy | | | 0.81 | 18399 |
| macro avg | 0.80 | 0.78 | 0.79 | 18399 |
| weighted avg | 0.81 | 0.81 | 0.81 | 18399 |

**Fig 27.** Weighted Avg f-1 score

# 6. Machine learning - Categories

## 6.1 Natural Language Processing (NLP)

The categories column consisted of values separated by commas in each row. The words in the categories section were chosen by the restaurant owners from a list of selections. The NLP was performed by using libraries imported from nltk following the steps below.




For this report, CountVectorizer was used instead of TfidfVectorizer since the frequency of the word was not of interest. For example, 'food' was almost always assisted by another word or words, such as: "Ethnic food" and "Food truck".

After the CountVectorizer, the categories column was expanded to 815 features. To avoid the curse of dimensionality, if sum of column was less than 85% of the feature size, it was dropped.

## 6.2 Bernoulli Naïve Bayes

The baseline model was tested using the raw data with 30% as the test size. Since Naïve Bayes does not have sample weight parameter, the upsampled training data were used.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.42 | 0.67 | 0.52 | 6429 |
| 1 | 0.74 | 0.51 | 0.60 | 11970 |
| accuracy | | | 0.56 | 18399 |
| macro avg | 0.58 | 0.59 | 0.56 | 18399 |
| weighted avg | 0.63 | 0.56 | 0.57 | 18399 |

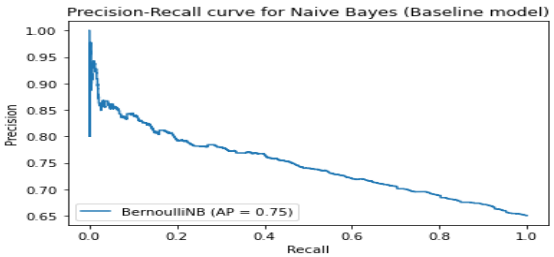



**Fig 27.** Weighted Avg f-1 score (left), Precision-recall curve (right)

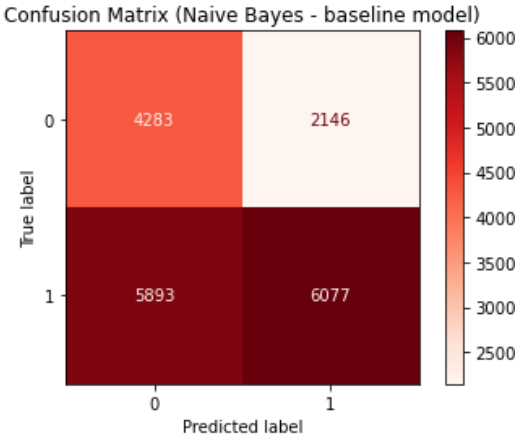
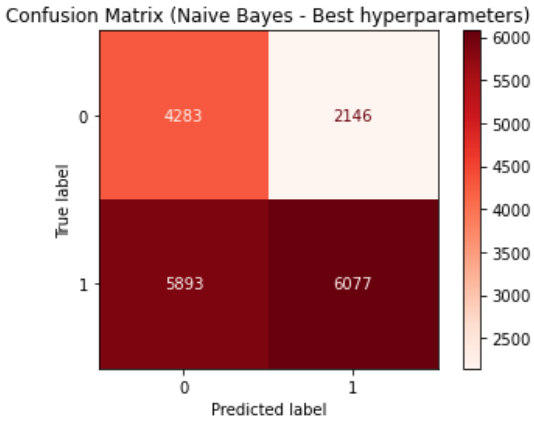



**Fig 28.** Confusion matrix – baseline model

The model tuning was performed using RandomSearchCV. The overall model performance did not improve. Best weighted average f-1 score was 0.57, while best cross validation score with 5 k-fold was 0.56




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.42 | 0.67 | 0.52 | 6429 |
| 1 | 0.74 | 0.51 | 0.60 | 11970 |
| accuracy | | | 0.56 | 18399 |
| macro avg | 0.58 | 0.59 | 0.56 | 18399 |
| weighted avg | 0.63 | 0.56 | 0.57 | 18399 |

**Fig 29.** Best hyperparameters

## 6.3 Logistic Regression

The baseline model was tested using the raw data with 30% as the test size. The baseline model resulted as shown below:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.57 | 0.16 | 0.24 | 6429 |
| 1 | 0.67 | 0.94 | 0.78 | 11970 |
| accuracy | | | 0.66 | 18399 |
| macro avg | 0.62 | 0.55 | 0.51 | 18399 |
| weighted avg | 0.64 | 0.66 | 0.60 | 18399 |

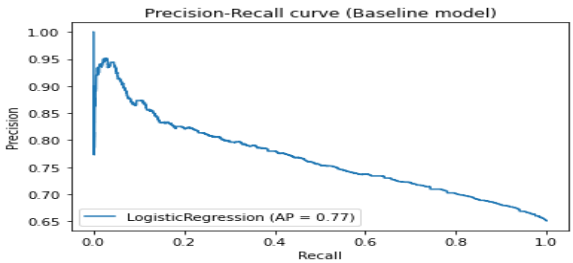



**Fig 30.** Weighted Avg f-1 score (left), Precision-recall curve (right)

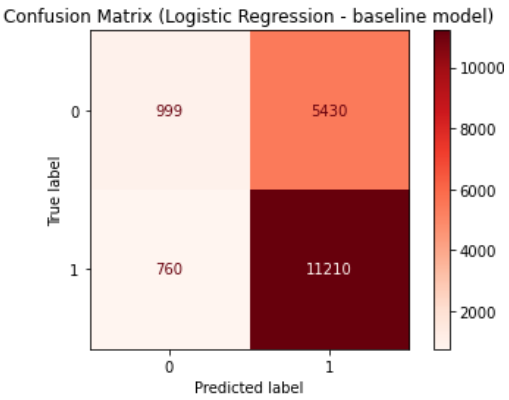



**Fig 31.** Confusion matrix – baseline model

Two different balancing techniques (class weight and upsampling performance) were applied on the baseline model to see if one performs better than the other.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.43 | 0.69 | 0.53 | 6429 |
| 1 | 0.75 | 0.51 | 0.61 | 11970 |
| accuracy | | | 0.57 | 18399 |
| macro avg | 0.59 | 0.60 | 0.57 | 18399 |
| weighted avg | 0.64 | 0.57 | 0.58 | 18399 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.43 | 0.68 | 0.53 | 6429 |
| 1 | 0.75 | 0.51 | 0.61 | 11970 |
| accuracy | | | 0.57 | 18399 |
| macro avg | 0.59 | 0.60 | 0.57 | 18399 |
| weighted avg | 0.64 | 0.57 | 0.58 | 18399 |

**Fig 32.** Class weight applied (left), Upsampling applied (right)

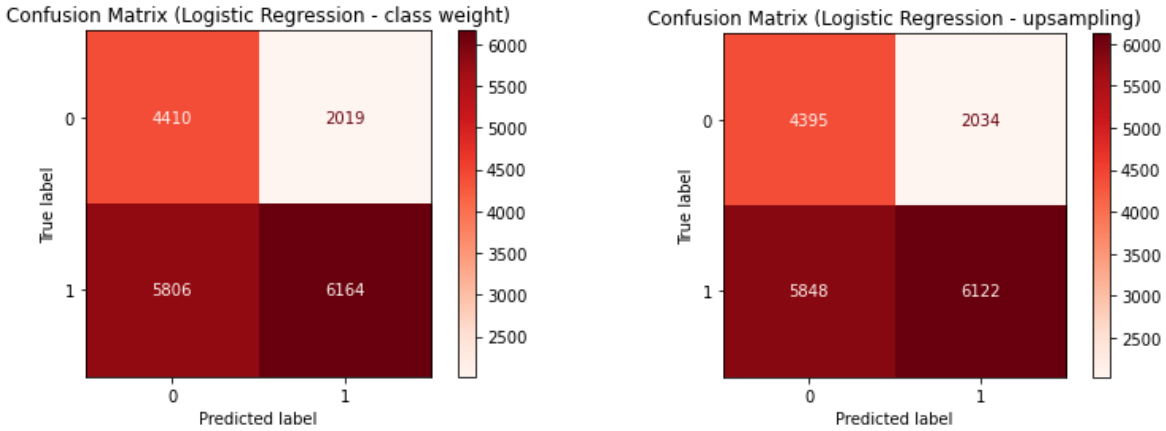
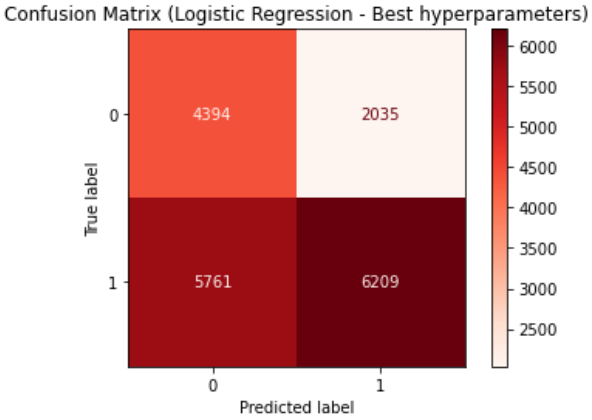



**Fig 33.** Class weight applied (left), Upsampling applied (right)

Both techniques resulted in the same weighted average f-1 scores, but 'class weight' had better average f-1 cross validation score (0.62 vs. 0.56). The model tuning was performed using RandomSearchCV. The overall model performance did not improve. Best weighted average f-1 score was 0.58, while best cross validation score with 5 k-fold was 0.62




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.43 | 0.69 | 0.53 | 21428 |
| 1 | 0.76 | 0.51 | 0.61 | 39900 |
| accuracy | | | 0.58 | 61328 |
| macro avg | 0.60 | 0.60 | 0.57 | 61328 |
| weighted avg | 0.64 | 0.58 | 0.58 | 61328 |

**Fig 34.** Best hyperparameters, class weight applied

## 6.4 Random Forest

The baseline model was tested using the raw data with 30% as the test size. The baseline model resulted as shown below:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.49 | 0.29 | 0.37 | 6429 |
| 1 | 0.69 | 0.84 | 0.75 | 11970 |
| accuracy | | | 0.65 | 18399 |
| macro avg | 0.59 | 0.56 | 0.56 | 18399 |
| weighted avg | 0.62 | 0.65 | 0.62 | 18399 |

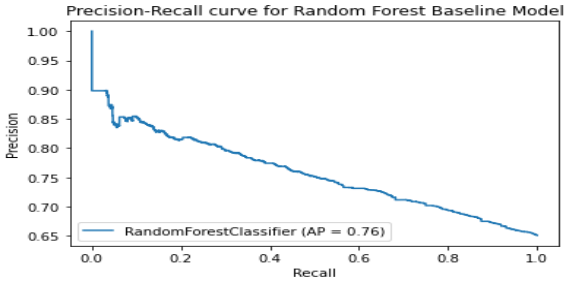



**Fig 35.** Weighted Avg f-1 score (left), Precision-recall curve (right)

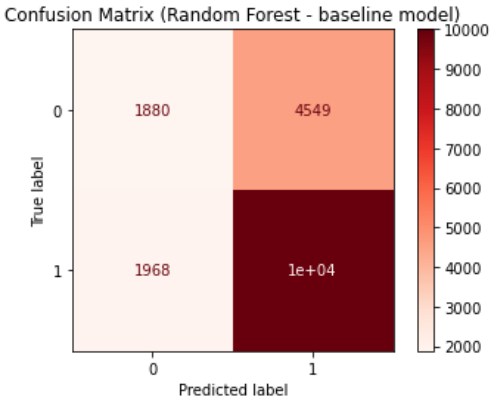



**Fig 36.** Confusion matrix – baseline model

Just like from the logistic regression, two different balancing techniques (class weight and upsampling performance) were applied on the baseline model to see if one performs better than the other.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.44 | 0.61 | 0.51 | 6429 |
| 1 | 0.74 | 0.58 | 0.65 | 11970 |
| accuracy | | | 0.59 | 18399 |
| macro avg | 0.59 | 0.60 | 0.58 | 18399 |
| weighted avg | 0.63 | 0.59 | 0.60 | 18399 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.44 | 0.62 | 0.51 | 6429 |
| 1 | 0.73 | 0.57 | 0.64 | 11970 |
| accuracy | | | 0.59 | 18399 |
| macro avg | 0.59 | 0.59 | 0.58 | 18399 |
| weighted avg | 0.63 | 0.59 | 0.60 | 18399 |

**Fig 37.** Class weight applied (left), Upsampling applied (right)

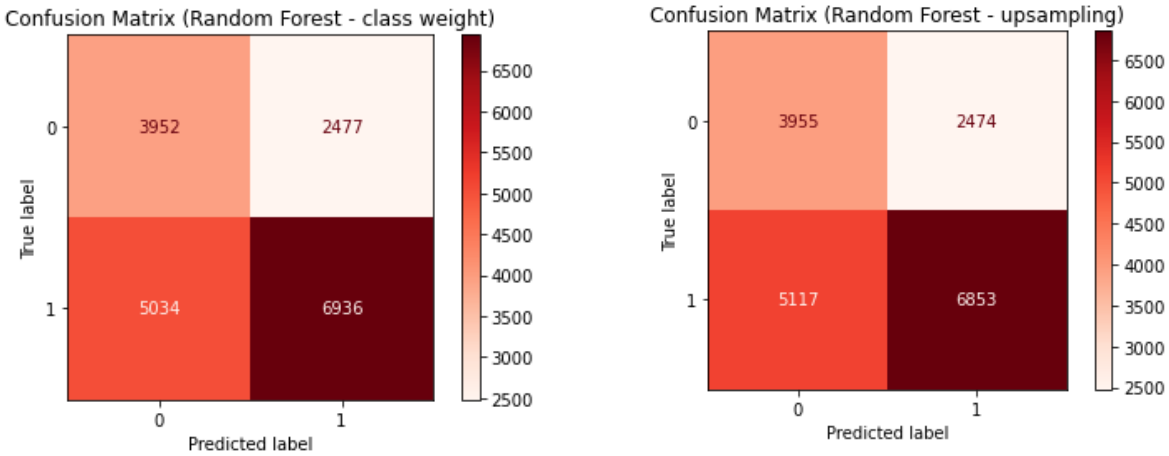
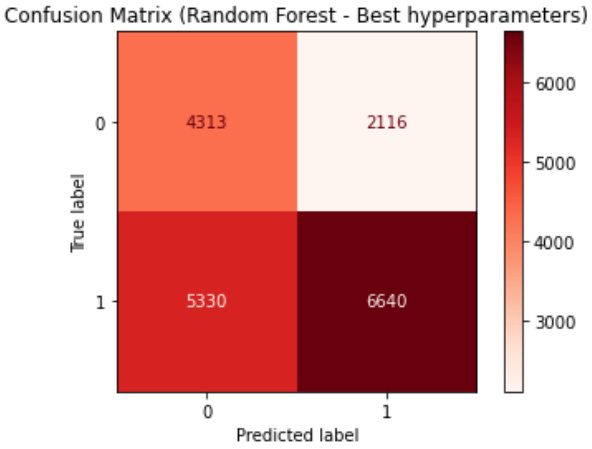



**Fig 38.** Class weight applied (left), Upsampling applied (right)

Both techniques resulted in the same weighted average f-1 scores, but 'class_weight had better average f-1 cross validation score (0.66 vs. 0.64). The model tuning was performed using RandomSearchCV. The overall model performance did not improve. Best weighted average f-1 score was 0.60, while best cross validation score with 5 k-fold was 0.64.




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.45 | 0.67 | 0.54 | 6429 |
| 1 | 0.76 | 0.55 | 0.64 | 11970 |
| accuracy | | | 0.60 | 18399 |
| macro avg | 0.60 | 0.61 | 0.59 | 18399 |
| weighted avg | 0.65 | 0.60 | 0.60 | 18399 |

**Fig 39.** Best hyperparameters, class weight applied

## 6.5 Deep Learning - Keras

Keras model performed the best with 10 epochs, 5 layers with 128 nodes in each (‘relu’ activation). The model was optimized with ‘adam’ and the loss was set to ‘binary_crossentropy’ . The imbalanced data was counted for using the class_weight parameter. The model resulted in weighted average f1 score of 0.59 and model performance of 0.61.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.43 | 0.67 | 0.53 | 6429 |
| 1 | 0.75 | 0.53 | 0.62 | 11970 |
| accuracy | | | 0.58 | 18399 |
| macro avg | 0.59 | 0.60 | 0.57 | 18399 |
| weighted avg | 0.64 | 0.58 | 0.59 | 18399 |

**Fig 40.** Weighted Avg f-1 score