Predicting Unsatisfied Guests from TripAdvisor Reviews

Author: Albane Colmenares
Date: January 4th, 2024

Table of Content

1. Overview

This notebook examines reviews about hotels from guests who stayed there and predicts whether the sentiment of reviews is positive or negative.
The organization of this notebook follows the CRoss Industry Standard Process for Data Mining (CRISP-DM) is a process model that serves as the base for a data science process.

2. Business Understanding

The increasing significance of online reputation and social media in the hospitality industry has become integral to a hotel's operational and financial success. The report (https://ecornell-impact.cornell.edu/the-impact-of-social-media-on-lodging-performance/) from Cornell's Center for Hospitality Research highlights the rising trend of consumers relying on online reviews during hotel bookings. The study reveals the link between a hotel's online reputation and its pricing, occupancy rates and revenue performance.

Recognizing the challenges faced by hotels in managing online reputation, a **new branch of TripAdvisor** is now dedicated to supporting and advising hotels on how to **improve their online reputation**. The branch launched a new B2B product identifying **unsatisfied guests in real time**

with the goal to improve their experience before they leave

with the goal to improve their experience before they leave.
Our first client is a worldwide hotel chain: Hyatt. The intended users are corporate directors of guest experience.

Through real-time guest reviews and a sophisticated algorithm, the platform alerts general managers about guests requiring immediate attention. The goal is to enable proactive measures to enhance satisfaction. This innovative tool, backed by an analysis of over 20,000 real guest reviews, also offers tailored recommendations for improvement upon implementation, aligning with the goal of turning detractors (https://www.datasciencecentral.com/what-is-a-good-net-promoter-score-for-the-hotel-resort-industry/) into promoters.

The Task: Shaping Hyatt's strategy to increase their online reputation

Objectives

1. Short-term: improve guest experience
    - thanks to our customized recommendations targeting areas that historically caused guest complaints
2. Long-term: reduce the share of negative reviews
    - by identifying unsatisfied guests through real-time reviews, aiming at improving their experience before they leave

3. Data Understanding

- **Data Source**

The data is hosted on Kaggle (https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews) and is provided by *LARXEL*.

In order to access the file, access the data source through the link provided and click on the `download` button at the top right corner. This will download a zipped folder. The below code unzips it upon loading the data.

The dataset contains about twenty thousand reviews collected from a hotel's TripAdvisor page. Each review is assigned a rating from 1 to 5, 1 being the lowest (negative review), 5 being the highest (positive review).

- **Features**

Prior to preprocessing, the columns are:

- `Review` : the actual review's record
- `Rating` : a number from 1 to 5, given by the guest at the time of the review

- **Target**

The target will be identifying negative reviews. It will be created based on the `Rating` column.

The target will be identifying negative reviews. It will be created based on the `Rating` column, and will be named `Sentiment`. Based on a given set of reviews, I will predict if the review's sentiment was negative. To be more precise, it will be identified as `detractors`. This term is

- **Loading the data**

In [1]:
```
# Importing the necessary packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import seaborn as sns
import warnings
from sklearn.model_selection import train_test_split
import nltk

%matplotlib inline
```

- **Conda Environment**

Before starting any steps, this notebook runs on Conda environment. Ensure the below steps are followed for you to run it locally:

1. Install conda
   - Download it from this link (https://www.anaconda.com/download)
2. Create a Conda Virtual Environment
   - MacOS: run conda env create -f mac_environment.yml
   - Windows: run conda env create -f win_environment.yml
3. Activate the Conda Virtual Environment
   - Type `conda activate learn-env`
   - To confirm it worked, type `conda info --envs` and confirm an asterisk `*` is next to the learn-env environment
4. Troubleshoot
   - If you see a warning message, you may need to update your version of Conda.
   - In this casem run `conda update -n base conda`

- **Kaggle API**

Before opening the file, a Kaggle API needs to be created in order to access Kaggle dataset directly. All you need to do is creating a username and key from your own profile in Kaggle.

If you do not have a Kaggle account, create an account here (https://www.kaggle.com/account/login?phase=startRegisterTab&returnUrl=%2F_).
Then, navigate to your profile by clicking on the top right corner on the goose icon.
From here, select settings
Under **API**, click `Create New Token`. This will download on .json file on your computer.
Open the .json file and replace the `username` and `key` information indicated on the cell below.

More information on how to access the Kaggle API can be found here

The cell below accesses the API. Make sure you change `username` and `key` with the above instructions!
Also delete your username and key when saving and pushing your project to GitHub.

In [ ]:
```
# Setting up the Kaggle API

import json
import os
from pathlib import Path

username = enter_your_username
key = enter_your_key

# your api key
api_key = {
'username':username ,
'key':key}

# uses pathlib Path
kaggle_path = Path('/root/.kaggle')
os.makedirs(kaggle_path, exist_ok=True)

# opens file and dumps python dict to json object
with open (kaggle_path/'kaggle.json', 'w') as handl:
    json.dump(api_key,handl)

os.chmod(kaggle_path/'kaggle.json', 600)
```

In [ ]:
```
# Download the dataset from Kaggle
!kaggle datasets download -d andrewmvd/trip-advisor-hotel-reviews
```

In [ ]:
```
# Create a folder data
!mkdir data
# And store the zipped file in it
!mv trip-advisor-hotel-reviews.zip data/trip-advisor-hotel-reviews.zip
```

Now you can open the file. The text file is encoded using Latin-1 encoding - and is open as is.
Several encodings were tried to ensure the right one matched: utf-8, utf-16, ascii for example.

In [2]:
```
# Loading dataset and saving it as raw_df
raw_df = pd.read_csv('data/trip-advisor-hotel-reviews.zip', compression="zip", er
```

In [3]:
```
# Inspecting the first 5 rows of the DataFrame
raw_df.head()
```

Out[3]:

	Review	Rating
0	nice hotel expensive parking got good deal sta...	4
1	ok nothing special charge diamond member hilto...	2
2	nice rooms not 4* experience hotel monaco seat...	3
3	unique, great stay, wonderful time hotel monac...	5
4	great stay great stay, went seahawk game aweso...	5

In [4]:
```
# Printing the number of rows and columns in the dataset
print(f'The dataset has {len(raw_df)} rows and {len(raw_df.columns)} columns.' )
```

```
The dataset has 20491 rows and 2 columns.
```

In [5]:
```
# Inspecting the number of reviews referring to each rating
raw_df['Rating'].value_counts()
```

Out[5]:
```
5    9054
4    6039
3    2184
2    1793
1    1421
Name: Rating, dtype: int64
```

In [6]:
```
# Inspecting the number of reviews referring to each rating
raw_df['Rating'].value_counts(normalize=True)
```

Out[6]:
```
5    0.441853
4    0.294715
3    0.106583
2    0.087502
1    0.069348
Name: Rating, dtype: float64
```

There is a clear majority of reviews with a rating of 5: over 44% of them while all other ratings are inequally distributed. The dataset is highly imbalanced. I will review whether grouping them as a binary classification can address the issue.

4. Data Preparation

4: 1- Data Cleaning

For a better readability of the reviews, the column width will be increased.

This link (https://www.tripadvisor.com/ShowTopic-g1-i12105-k11476502

This link (https://www.tripadvisor.com/ShowTopic-g1-i12105-k11476502-What_is_the_maximum_character_limit_on_a_review-Tripadvisor_Support.html) from TripAdvisor

In [7]:
```
# Increasing column width
pd.set_option('max_colwidth', 20000)
```

In [8]:
```
# Inspecting the first 5 rows of the DataFrame now that there no longer is a limi
raw_df.head()
```

Out[8]:

	Review	Rating
0	nice hotel expensive parking got good deal stay hotel anniversary, arrived late evening took advice previous reviews did valet parking, check quick easy, little disappointed non-existent view room room clean nice size, bed comfortable woke stiff neck high pillows, not soundproof like heard music room night morning loud bangs doors opening closing hear people talking hallway, maybe just noisy neighbors, aveda bath products nice, did not goldfish stay nice touch taken advantage staying longer, location great walking distance shopping, overall nice experience having pay 40 parking night,	4
1	ok nothing special charge diamond member hilton decided chain shot 20th anniversary seattle, start booked suite paid extra website description not, suite bedroom bathroom standard hotel room, took printed reservation desk showed said things like tv couch ect desk clerk told oh mixed suites description kimpton website sorry free breakfast, got kidding, embassy suits sitting room bathroom bedroom unlike kimpton calls suite, 5 day stay offer correct false advertising, send kimpton preferred guest website email asking failure provide suite advertised website reservation description furnished hard copy reservation printout website desk manager duty did not reply solution, send email trip guest survey did not follow email mail, guess tell concerned guest.the staff ranged indifferent not helpful, asked desk good breakfast spots neighborhood hood told no hotels, gee best breakfast spots seattle 1/2 block away convenient hotel does not know exist, arrived late night 11 pm inside run bellman busy chating cell phone help bags.prior arrival emailed hotel inform 20th anniversary half really picky wanted make sure good, got nice email saying like deliver bottle champagne chocolate covered strawberries room arrival celebrate, told	2

Despite increasing the column width, some reviews are not finished. This was verified inside the raw data, which confirms the reviews were not provided entirely.

4: 1- a) Copying the raw data

In order to keep the raw data as it is in case it needs to be accessed in the future, an exact copy of the dataset will be made. The new DataFrame `df` contains the same columns:

- `Review`
- `Rating`

In [9]:
```
# Making a copy of the raw DataFrame to modify it
df = raw_df.copy()
```

In [10]:
```
# Verifying the changes applied
df.head()
```

Out[10]:

	Review	Rating
0	nice hotel expensive parking got good deal stay hotel anniversary, arrived late evening took advice previous reviews did valet parking, check quick easy, little disappointed non-existent view room room clean nice size, bed comfortable woke stiff neck high pillows, not soundproof like heard music room night morning loud bangs doors opening closing hear people talking hallway, maybe just noisy neighbors, aveda bath products nice, did not goldfish stay nice touch taken advantage staying longer, location great walking distance shopping, overall nice experience having pay 40 parking night,	4
1	ok nothing special charge diamond member hilton decided chain shot 20th anniversary seattle, start booked suite paid extra website description not, suite bedroom bathroom standard hotel room, took printed reservation desk showed said things like tv couch ect desk clerk told oh mixed suites description kimpton website sorry free breakfast, got kidding, embassy suits sitting room bathroom bedroom unlike kimpton calls suite, 5 day stay offer correct false advertising, send kimpton preferred guest website email asking failure provide suite advertised website reservation description furnished hard copy reservation printout website desk manager duty did not reply solution, send email trip guest survey did not follow email mail, guess tell concerned guest.the staff ranged indifferent not helpful, asked desk good breakfast spots neighborhood hood told no hotels, gee best breakfast spots seattle 1/2 block away convenient hotel does not know exist, arrived late night 11 pm inside run bellman busy chating cell phone help bags.prior arrival emailed hotel inform 20th anniversary half really picky wanted make sure good, got nice email saying like deliver bottle champagne chocolate covered strawberries room arrival celebrate, told	2

4: 1- b) Missing data

In the next section, the missing values are inspected.
None of the two columns have null data, no modification is necessary.

In [11]:
```
# Looking for missing values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20491 entries, 0 to 20490
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Review  20491 non-null  object
 1   Rating  20491 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 320.3+ KB
```

4: 1- c) Handling duplicates

No duplicate rows were identified. No changes are necessary.

In [12]:
```
# How many rows were duplicates
print(str(len(df[df.duplicated()])) + f' duplicate rows were identified.')
```

```
0 duplicate rows were identified.
```

In [13]:
```
# Verifying there are no duplicate rows
df[df.duplicated()]
```

Out[13]:

	Review	Rating

4: 1- d) Categorizing the ratings into a binary classification

- **Sentiment**

Five sentiment categories are described, which could be grouped in two: detractors and not_detractors.
Let's first review this assumption that the rating of 5 corresponds to a high positive review.

In [14]:
```
# Filtering on the first 5 reviews with a rating of 5
df[df['Rating'] == 5].head()
```

Out[14]:

	Review	Rating
3	unique, great stay, wonderful time hotel monaco, location excellent short stroll main downtown shopping area, pet friendly room showed no signs animal hair smells, monaco suite sleeping area big striped curtains pulled closed nice touch felt cosy, goldfish named brandi enjoyed, did n't partake free wine coffee/tea service lobby thought great feature, great staff friendly, free wireless internet hotel worked suite 2 laptops, decor lovely eclectic mix pattens color palatte, animal print bathrobes feel like rock stars, nice did n't look like sterile chain hotel hotel personality excellent stay,	5
4	great stay great stay, went seahawk game awesome, downfall view building did n't complain, room huge staff helpful, booked hotels website seahawk package, no charge parking got voucher taxi, problem taxi driver did n't want accept voucher barely spoke english, funny thing speak arabic called started making comments girlfriend cell phone buddy, took second realize just said fact speak language face priceless, ass told, said large city, told head doorman issue called cab company promply answer did n't, apologized offered pay taxi, bucks 2 miles stadium, game plan taxi return going humpin, great walk did n't mind, right christmas wonderful lights, homeless stowed away building entrances leave, police presence not greatest area stadium, activities 7 blocks pike street waterfront great coffee shops way, hotel mantained foyer awesome, wine tasting available evening, best dog, taking st. bernard time family, safes hotel located service desk room, bathroom huge jetted tub huge, funny house keeping walked girlfriend getting dressed, did n't hear knock doing turn service, screamed girlfriend screams hit floor laughing, started talking spanish worked, place recommend price, check online deals just good not better, besite contains deals vouchers travel websites n't tell,	5
5	love monaco staff husband stayed hotel crazy weekend attending memorial service best friend husband celebrating 12th wedding anniversary, talk mixed emotions, booked suite hotel monte carlos, loaned beautiful fan-tanned goldfish named joliet weekend visited dogs worked desk human companions, room decorated nicely couch used pillows, l'occitane bath amenities welcome sight, room quiet peaceful, wireless internet access wonderful server went morning leaving problems printing boarding passes, afternoon reception serves oenophile-satisfying wine australia scrumptious cookies, restaurant closed renovation stay finally ate food good drinks better, word caution restaurant larger person not sit booths wo n't fit, 5'6 125 lbs husband 5'9 175. table smack-against stomach couple inches space mighty uncomfortable patron larger pregnant, bad design opinion place decorated funky welcoming way metal wood handblown glass light fixtures expect seattle capital glass art industry, definitely stay reason,	5
6	cozy stay rainy city, husband spent 7 nights monaco early january 2008. business trip chance come ride.we booked monte carlo suite proved comfortable longish stay, room 905 located street building, street noise not problem view interesting rooms building look dank alley midsection large office building, suite comfortable plenty room spread, bathroom attractive squeaky clean small comparison generous proportions sitting sleeping areas, lots comfortable seating options good lighting plenty storage clothing luggage, hotel staff friendly efficient, housekeeping staff did great job pleasant, requests responded quickly.the location quite good, easy walk pike street market seattle art museum notch shopping dining options.a positive experience,	5
8	hotel stayed hotel monaco cruise, rooms generous decorated uniquely, hotel remodeled pacific bell building charm sturdiness, everytime walked bell men felt like coming home, secure, great single travelers, location fabulous, walk things pike market space needle.little grocery/drug store block away, today green, bravo, 1 double bed room room bed couch separated curtain, snoring mom slept curtain, great food nearby,	5

At first glance, the reviews indicate how great the stay was, how comfortable the rooms were and provide compliments about the building, decor and neighborhood. This confirms 5 star reviews are positive.

I will inspect the 1 star review to confirm that these are negative before finally creating the sentiment column.

In [15]:
```
# Filtering on the first 5 reviews with a rating of 5
df[df['Rating'] == 1].head()
```

Out[15]:

	Review	Rating
15	horrible customer service hotel stay february 3rd 4th 2007my friend picked hotel monaco appealing website online package included champagne late checkout 3 free valet gift spa weekend, friend checked room hours earlier came later, pulled valet young man just stood, asked valet open said, pull bags didn__Ã□_Ã©_ offer help, got garment bag suitcase came car key room number says not valet, car park car street pull, left key working asked valet park car gets, went room fine bottle champagne oil lotion gift spa, dressed went came got bed noticed blood drops pillows sheets pillows, disgusted just unbelievable, called desk sent somebody 20 minutes later, swapped sheets left apologizing, sunday morning called desk speak management sheets aggravated rude, apparently no manager kind supervisor weekend wait monday morning, young man spoke said cover food adding person changed sheets said fresh blood rude tone, checkout 3pm package booked, 12 1:30 staff maids tried walk room opening door apologizing closing, people called saying check 12 remind package, finally packed things went downstairs check, quickly signed paper took, way took closer look room, unfortunately covered food offered charged valet, called desk ask charges lady answered snapped saying aware problem experienced monday like told earlier, life treated like hotel, not sure hotel constantly problems lucky ones stay recommend anybody know,	1
32	noise airconditioner-a standard, arranged stay travel agency unfortunately warwick seattle hotel dissapointment trip, 3 night stay warwick changed 3 rooms, starting minute stay hotel personnel didn`t make feel like guest like intruder, reluctant help solve complaints, hotel right downtown 5 minutes really good restaurants like good thing, availability room offered 2nd floor window	1

Indeed, the first word of the first 1 star review indicate how *horrible* the customer service was. The ranking from 1 to 5 is confirmed to be from low to high.

- **NPS Score**

The Net Promoter Score (NPS) is a metric used to measure customer satisfaction and loyalty with a hotel, (or more generally a product, service, or company). It is based on the question: "How likely is it that you would recommend our *hotel*? The responses can be measured on a scale from 0 to 10 or in our case: 1 to 5.

(https://textexpander.com/blog/how-to-calculate-nps)

According to the 5 point NPS Scale (https://www.surveysensum.com/blog/11-point-and-5-point-nps-scale#:~:text=5%2Dpoint%20NPS%20Scale,-Similar%20to%20the&text=The%20respondents%20on%20the%205,why%20it%20is%20not%20rec

1. Promoters rate 5 star reviews
2. Passive customers rate 4 stars
3. Detractors rate from 1 to 3 star reviews

This indicates that the customers who rank a hotel from 1 to 3 star reviews may really damage a hotel's online reputation and negatively influence other guests in their willingness to book your hotel.

NPS is calculated as:

- the percentage of promoters (guests who highly recommend the hotel: 5 star reviews)
- **minus** the percentage of detractors (guests who do not recommend the hotel: 1 to 3 star reviews)

As a consequence, I will categorize the guests who gave 1 to 3 star reviews as `detractors` and the others as `not_detractors`.

- **NPS Calculation**

As a starting point the NPS for this dataset is calculated.

In [16]:
```
# Reminding the current percentage for each star reviews given
df['Rating'].value_counts(normalize=True)

# Storing values for promoters and detractors
promoters = df['Rating'].value_counts(normalize=True).get(5, 0)
detractors = df['Rating'].value_counts(normalize=True).get(1, 0) + df['Rating'].v

nps_score = promoters - detractors
```

In [17]:
```
# Calculating the current NPS score
nps_score = promoters - detractors
print(f'Your NPS Score is {nps_score:.2%}')
```

```
Your NPS Score is 17.84%
```

If a hotel was to decrease the number of detractors by only 10%, the new NPS score for your hotel chain would be:

In [18]:
```
# Calculating the potential percentage of detractors
potential_detractors = detractors * 0.9
```

In [19]:
```
# Calculating the potential NPS Score
potential_nps_score = promoters - potential_detractors
print(f'Your potential NPS Score could be {potential_nps_score:.2%}')
```

```
Your potential NPS Score could be 20.48%
```

In [20]:
```
# Calculating the growth in NPS Score after reducing detractors by 10%
growth_in_nps = (potential_nps_score - nps_score ) / nps_score
print(f'By decreasing only the percentage of detractors in your hotels by only 10
```

```
By decreasing only the percentage of detractors in your hotels by only 10%, you
r online reputation score - or NPS could increase by 14.76%!
```

I will create this new category under the column `Sentiment` .

In [21]:
```
# Creating the column sentiment
df['Sentiment'] = df['Rating'].apply(lambda x: 'detractors' if x in [1, 2, 3] els
```

In [22]:
```
# Verifying the column was created
df.head()
```

Out[22]:

	Review	Rating	Sentiment
0	nice hotel expensive parking got good deal stay hotel anniversary, arrived late evening took advice previous reviews did valet parking, check quick easy, little disappointed non-existent view room room clean nice size, bed comfortable woke stiff neck high pillows, not soundproof like heard music room night morning loud bangs doors opening closing hear people talking hallway, maybe just noisy neighbors, aveda bath products nice, did not goldfish stay nice touch taken advantage staying longer, location great walking distance shopping, overall nice experience having pay 40 parking night,	4	not_detractors
1	ok nothing special charge diamond member hilton decided chain shot 20th anniversary seattle, start booked suite paid extra website description not, suite bedroom bathroom standard hotel room, took printed reservation desk showed said things like tv couch ect desk clerk told oh mixed suites description kimpton website sorry free breakfast, got kidding, embassy suits sitting room bathroom bedroom unlike kimpton calls suite, 5 day stay offer correct false advertising, send kimpton preferred guest website email asking failure provide suite advertised website reservation description furnished hard copy reservation printout website desk manager duty did not reply solution, send email trip guest survey did not follow email mail, guess tell concerned guest.the staff ranged indifferent not helpful, asked desk good breakfast spots neighborhood hood told no hotels, gee best breakfast spots seattle 1/2 block away convenient hotel does not know exist, arrived late night 11 pm inside run bellman busy chating cell phone help bags.prior arrival emailed hotel inform 20th anniversary half really picky wanted make sure good, got nice email saying like deliver bottle champagne chocolate covered strawberries room arrival celebrate, told needed foam pillows, arrival no champagne strawberries no foam pillows great room view alley high rise building good not better housekeeping staff cleaner room property, impressed left morning shopping room got short trips 2 hours, beds comfortable.not good ac-heat control 4 x 4 inch screen bring green shine directly eyes night, light sensitive tape controls.this not 4 start hotel clean business hotel super high rates, better chain hotels seattle,	2	detractors
2	nice rooms not 4* experience hotel monaco seattle good hotel n't 4* level.positives large bathroom mediterranean suite comfortable bed pillowsattentive housekeeping staffnegatives ac unit malfunctioned stay desk disorganized, missed 3 separate wakeup calls, concierge busy hard touch, did n't provide guidance special requests.tv hard use ipod sound dock suite non functioning. decided book mediterranean suite 3 night weekend stay 1st choice rest party filled, comparison w spent 45 night larger square footage room great soaking tub whirlpool jets nice shower.before stay hotel arrange car service price 53 tip reasonable driver waiting arrival.checkin easy downside room picked 2 person jacuzi tub no bath accessories salts bubble bath did n't stay, night got 12/1a checked voucher bottle champagne nice gesture fish waiting room, impression room huge open space felt room big, tv far away bed chore change channel, ipod dock broken disappointing.in morning way asked desk check thermostat said 65f 74 2 degrees warm try cover face night bright blue light kept, got room night no, 1st drop desk, called maintainence came look thermostat told play settings happy digital box wo n't work, asked wakeup 10am morning did n't happen, called later 6pm nap wakeup forgot, 10am wakeup morning yep forgotten.the bathroom facilities great room surprised room sold whirlpool bath tub n't bath amenities, great relax water jets going,	3	detractors
3	unique, great stay, wonderful time hotel monaco, location excellent short stroll main downtown shopping area, pet friendly room showed no signs animal hair smells, monaco suite sleeping area big striped curtains pulled closed nice touch felt cosy, goldfish named brandi enjoyed, did n't partake free wine coffee/tea service lobby thought great feature, great staff friendly, free wireless internet hotel worked suite 2 laptops, decor lovely eclectic mix pattens color palatte, animal print bathrobes feel like rock stars, nice did n't look like sterile chain hotel hotel personality excellent stay,	5	not_detractors

	Review	Rating	Sentiment
4	great stay great stay, went seahawk game awesome, downfall view building did n't complain, room huge staff helpful, booked hotels website seahawk package, no charge parking got voucher taxi, problem taxi driver did n't want accept voucher barely spoke english, funny thing speak arabic called started making comments girlfriend cell phone buddy, took second realize just said fact speak language face priceless, ass told, said large city, told head doorman issue called cab company promply answer did n't, apologized offered pay taxi, bucks 2 miles stadium, game plan taxi return going humpin, great walk did n't mind, right christmas wonderful lights, homeless stowed away building entrances leave, police presence not greatest area stadium, activities 7 blocks pike street waterfront great coffee shops way, hotel mantained foyer awesome, wine tasting available evening, best dog, taking st. bernard time family, safes hotel located service desk room, bathroom huge jetted tub huge, funny house keeping walked girlfriend getting dressed, did n't hear knock doing turn service, screamed girlfriend screams hit floor laughing, started talking spanish worked, place recommend price, check online deals just good not better, besite contains deals vouchers travel websites n't tell,	5	not_detractors

In [23]:
```
# Inspecting the new value_counts for the Sentiment category
df['Sentiment'].value_counts()
```

Out[23]:
```
not_detractors    15093
detractors         5398
Name: Sentiment, dtype: int64
```

In [24]:
```
# And in percentages
# Inspecting the new value_counts for the Sentiment category
df['Sentiment'].value_counts(normalize=True)
```

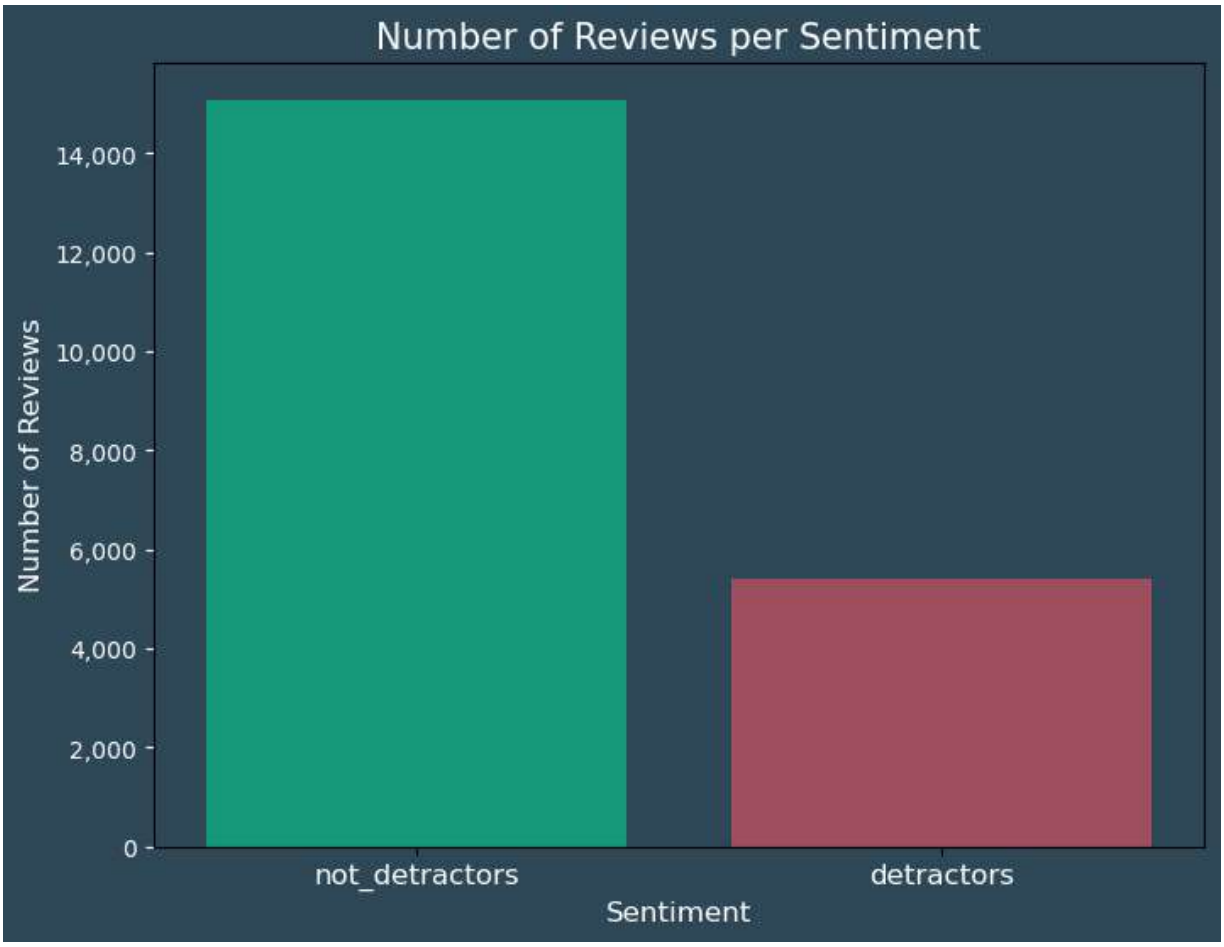
Out[24]:
```
not_detractors    0.736567
detractors        0.263433
Name: Sentiment, dtype: float64
```

The grouping also started addressing the class imbalance. Nevertheless, the dataset remains highly imbalanced.

I will visually represent the split of number of reviews.

In [ ]:
```
# Creating a folder images to store the saved images in them
!mkdir images
```

In [25]:
```
from matplotlib.ticker import FuncFormatter
# Creating a function to plot the number of reviews by sentiment

# Defining custom colors
custom_colors = ['#00AF87', '#AA4255']

def plot_sentiment_count(df, x, label_rotation):
    # Defining the figure
    fig, ax = plt.subplots(figsize=(8,6))

    # Setting facecolor
    fig.set_facecolor('#2F4858')
    ax.set_facecolor('#2F4858')

    # Plotting the count of reviews
    sns.countplot(data=df, x=x, order=df[x].value_counts().index, palette=custom_

    # Defining the labels and titles
    ax.set_xlabel(xlabel = 'Sentiment', fontsize=12, color='white')
    ax.set_ylabel(ylabel = 'Number of Reviews', fontsize=12, color='white')
    ax.set_title(f'Number of Reviews per Sentiment', fontsize=15, color='white')
    ax.tick_params(axis='y', colors='white')

    # Recording the sentiment values to use them as labels
    sentiment_labels = df[x].unique()

    # Setting the x-axis tick labels to the sentiment labels
    ax.set_xticks(range(len(sentiment_labels)))
    ax.set_xticklabels(labels=sentiment_labels, rotation=label_rotation, fontsize


    # Formatting the y-axis labels to show thousands
    ax.yaxis.set_major_formatter(FuncFormatter(lambda x, pos: '{:,}'.format(int(x

     # Saving the plot as a PNG with a transparent background
    plt.savefig('images/reviews_per_sentiment.png')

    # Showing the plot
    plt.show()


plot_sentiment_count(df, 'Sentiment', 0)
```



4: 1- e) Performing a Train-Test Split

The dataset is being divided into two separate subsets: a training set, and a testing (or validation) set. The validation set will allow to assess the performance of the model.

Two parameters are assigned when dividing the dataset:

- random_state=42
  - setting a random seed of 42 ensures that the data split is reproducible
- stratify=y
  - stratified sampling ensures the class distribution is maintained in both sets to address potential class imbalance issues

In [26]:
```
# Splitting df into X and y
X = df.drop('Sentiment', axis=1)
y = df['Sentiment']
```

In [27]:
```
# Performing a train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, strati
```

In [28]:
```
# Inspecting the X_train data
X_train.head()
```

Out[28]:

	Review	Rating
1282	gem, pleasantly surprised accomondations helpful attentive staff need, breakfast good cookies, room lovely clean comfortable, room fronting bush street bit noisy did not away enjoyment nice boutique hotel, definitely stay time,	4
10661	loved fita wife spent nights hotel fita march 2008. hotel staff fantastic helpful pleasant, recommend hotel traveling amsterdam.the hotel located minute walk van gogh rijks museums 10 minute walk leidsplein variety stores shops restaurants located.there tram stop directly hotel, easy access entire city, did walk fita way center city 30 minutes.beautiful accesible friendly definately travel fita wonderful amsterdam,	5
17908	great modern hotel fantastic price stayed just night beginning april, alot reviews say bad area really did n't, literally 100m away nice redeveloped area 10min walk beach.me partner arrived late barcelona decided walk hotel bus station order barings, quite long walk 50mins roughly really great thing order sights tourists dont usually, arrived check quick painless room really really nice, fantastic modern design little extras want, order barcelona partner opted bus turistic, stop not far hotel goes virtually, worth short visit.i wish spent longer hotel, wish swimming pool roof open, overall fantastic hotel price paid room dont distance centre transport options,	5
11685	great experiance, stayed family golden tulip week summer, rooms somewhat small beds incredibly comfortable bathrooms good size, staff really friendly hotel clean food restaurant fantastic not bit expensive, 10 minute train ride puts amsterdam easy use, took bar nights drinks coffee great accomodating large group people, not greatest neighborhood 5 minute walk takes shops, stay,	4
9059	excellent location hotel improved, things location hotel excellent, plenty bars shops restaurants activities nearby 20 minutes walk centre amsterdam red light district interested having look, lobby area nice staff friendly checked quickly approaching room worried area hotel older described poor condition shabby, carpets dirty walls paper hanging really bad smell stale smoke drink food onions entered room group consisted married couple rooms given abysmal, dirty smelly definately not 4 star standard, hotel needs refurbishment.. sooner better, bed quarter size double just room 2 barely floor foot bed walk sides furniture old shabby badly damaged looked like selected junk shop, actually felt quite insulted given room appalling standard absolutely no way earth prepared spend night expect complained, told receptionist not room disgusting insulted fact clearly expected sleep, way apology offered executive room newer hotel discounted rate 25 euro night room apparently usually 45 euros extra room night, vast improvement expect standard, decent sized bed king matching furniture tidy bathroom nice decor plenty room, bright airy clean, looked fairly new given room place not cause complain rated 4 star hotel, newer hotel lovely pleasant bar area, staff friendly overall hard pressed better location.so note book make sure not given room old building disappointed,	3

In [29]:
```
# Inspecting the y_train data
y_train.head()
```

Out[29]:
```
1282     not_detractors
10661    not_detractors
17908    not_detractors
11685    not_detractors
9059         detractors
Name: Sentiment, dtype: object
```

- **Distribution of Target**

In [30]:
```
# Inspecting the distribution in the train data
y_train.value_counts(normalize=True)
```

Out[30]:
```
not_detractors    0.736596
detractors        0.263404
Name: Sentiment, dtype: float64
```

In [31]:
```
# Inspecting the distribution in the test data
y_test.value_counts(normalize=True)
```

Out[31]:
```
not_detractors    0.736483
detractors        0.263517
Name: Sentiment, dtype: float64
```

4: 1- f) Encoding Target

In order to make predictions useable in calculations, I will encode the target:

- 0 will replace `not_detractors`
- 1 will replace `detractors`

In [32]:
```
# Adding in labels for filtering before making any encoding
X_train['label'] = [y_train[val] for val in X_train.index]
X_train[:3]
```

Out[32]:

	Review	Rating	label
1282	gem, pleasantly surprised accomondations helpful attentive staff need, breakfast good cookies, room lovely clean comfortable, room fronting bush street bit noisy did not away enjoyment nice boutique hotel, definitely stay time,	4	not_detractors
10661	loved fita wife spent nights hotel fita march 2008. hotel staff fantastic helpful pleasant, recommend hotel traveling amsterdam.the hotel located minute walk van gogh rijks museums 10 minute walk leidsplein variety stores shops restaurants located.there tram stop directly hotel, easy access entire city, did walk fita way center city 30 minutes.beautiful accesible friendly definately travel fita wonderful amsterdam,	5	not_detractors
17908	great modern hotel fantastic price stayed just night beginning april, alot reviews say bad area really did n't, literally 100m away nice redeveloped area 10min walk beach.me partner arrived late barcelona decided walk hotel bus station order barings, quite long walk 50mins roughly really great thing order sights tourists dont usually, arrived check quick painless room really really nice, fantastic modern design little extras want, order barcelona partner opted bus turistic, stop not far hotel goes virtually, worth short visit.i wish spent longer hotel, wish swimming pool roof open, overall fantastic hotel price paid room dont distance centre transport options,	5	not_detractors

In [33]:
```
# Adding in labels for filtering before making any encoding
X_test['label'] = [y_test[val] for val in X_test.index]
X_test[:3]
```

Out[33]:

	Review	Rating	label
5372	great location good hotel 1night stay recommend hotel want just night accommodation ahead day flight, sheraton stars mentioned architecture n't good hotel, stay night try solution night stay choose directly connected airport, pretty expensive cola 0,2l juices 4 eur price minibar pizza 12 eur ok person problems finding signs sheraton, problems big frankfurt airport.actually easy hotel airport directly connected airport terminal 1.from departure hall upstairs 2 pedestrian bridges whichconnect terminal hotel long distance trains 1 hall b near lufthansa 1 hall b c near american airlines follow signs hotel fernbahnhof long distance trains,	4	not_detractors
16041	good airport hotel booked hotel early departures, nights room comfortable air-conditioning awful, quiet open windows, non-smoking floor smoking allowed restaurants common areas, arriving hotel shuttle available departing odd hour pay 30 taxi no taxi wants 5-minute trip, offer cab driver 10 did agree pay going rate hotel car 30, check-in problem odd hours staff speak english, hotel restaurants awful, chopsticks chinese taste good chinese food u.s. greasy, good wine list course problem smokers, good espresso desert bar outside chopsticks, best aspect hotel excellent travel desk manned efficient friendly lindsay day, ask eat town, english excellent, helped hotel merchant sold damaged item,	2	detractors
15225	mandarin oriental good stayed mandarin oriental way australia europe, visit hotel frankly difficult convince ourselevs stay singapore, stayed club rooms time service provided club guests second none, club rooms need little updating clean modern expect class hotel, location superb access marina square mall subsequently malls linked, prices risen significantly recent years watch special deals bonus free nights, highly recommended,	5	not_detractors

In [34]:
```
# Inspecting y_train
print('Before')
print(y_train.value_counts())
print(y_train.value_counts(normalize=True))

# Encoding y_train
y_train = y_train.apply(lambda x: 0 if x == 'not_detractors' else 1)

# Verifying it correctly applied
print('After')
print(y_train.value_counts())
print(y_train.value_counts(normalize=True))
```

```
Before
not_detractors    11320
detractors         4048
Name: Sentiment, dtype: int64
not_detractors    0.736596
detractors        0.263404
Name: Sentiment, dtype: float64
After
0    11320
1     4048
Name: Sentiment, dtype: int64
0    0.736596
1    0.263404
Name: Sentiment, dtype: float64
```

In [35]:
```
# Inspecting y_test
print('Before')
print(y_test.value_counts())
print(y_test.value_counts(normalize=True))

# Encoding y_test
y_test = y_test.apply(lambda x: 0 if x == 'not_detractors' else 1)

# Verifying it correctly applied
print('After')
print(y_test.value_counts())
print(y_test.value_counts(normalize=True))
```

```
Before
not_detractors    3773
detractors        1350
Name: Sentiment, dtype: int64
not_detractors    0.736483
detractors        0.263517
Name: Sentiment, dtype: float64
After
0    3773
1    1350
Name: Sentiment, dtype: int64
0    0.736483
1    0.263517
Name: Sentiment, dtype: float64
```

- **Visually Inspecting Features on train and test samples**

In [36]:
```
X_test.head()
```

Out[36]:

	Review	Rating	label
5372	great location good hotel 1night stay recommend hotel want just night accommodation ahead day flight, sheraton stars mentioned architecture n't good hotel, stay night try solution night stay choose directly connected airport, pretty expensive cola 0,2l juices 4 eur price minibar pizza 12 eur ok person problems finding signs sheraton, problems big frankfurt airport.actually easy hotel airport directly connected airport terminal 1.from departure hall upstairs 2 pedestrian bridges whichconnect terminal hotel long distance trains 1 hall b near lufthansa 1 hall b c near american airlines follow signs hotel fernbahnhof long distance trains,	4	not_detractors
16041	good airport hotel booked hotel early departures, nights room comfortable air-conditioning awful, quiet open windows, non-smoking floor smoking allowed restaurants common areas, arriving hotel shuttle available departing odd hour pay 30 taxi no taxi wants 5-minute trip, offer cab driver 10 did agree pay going rate hotel car 30, check-in problem odd hours staff speak english, hotel restaurants awful, chopsticks chinese taste good chinese food u.s. greasy, good wine list course problem smokers, good espresso desert bar outside chopsticks, best aspect hotel excellent travel desk manned efficient friendly lindsay day, ask eat town, english excellent, helped hotel merchant sold damaged item,	2	detractors
15225	mandarin oriental good stayed mandarin oriental way australia europe, visit hotel frankly difficult convince ourselevs stay singapore, stayed club rooms time service provided club guests second none, club rooms need little updating clean modern expect class hotel, location superb access marina square mall subsequently malls linked, prices risen significantly recent years watch special deals bonus free nights, highly recommended,	5	not_detractors
18749	great little hotel close forbidden city great little hotel ideal location, minutes walking forbidden city, located traditional beijing hutong neighborhood recommend going early morning late night walk neighborhood sense locals live daily lives, flying shanghai beijing beijing airport got shut heavy rains flight canceled, thought hotel charged night no proper cancellation, arrived morning told not charge missed night knew control.sunny yu speaks good english help tours great wall places interested.my room bit small clean, overall great hotel grateful generously not charging missed night,	5	not_detractors
8348	anther great stay husband stayed 9 nights july 2008. second stay not disappointed, location hotel fantastic rooms nicely appointment great sea view, short walk sitges centre quick taxi ride, make quiet beaches good restaurants 5-mins walking distance hotel.this hotel definitely gets busier weekends spanish tourists coming weekend break uncomfortable, hotel let slightly minor irritations, breakfast superb meals poor, snack bar food terrible getting decent lunch hard, surprising considering brilliant restaurants area, needs improved, irritation lack uk tv channels, clearly does n't sitges watch tv day catch movies 9 days good, no uk channels cnn no movie ordering service, hotel clearly make money introducing service.all excellent choice sitges definitely return,	4	not_detractors

In [37]:
```
# Making a sample of 3 records to display the full text of each
train_sample = X_train.sample(3, random_state=22)
train_sample['target'] = [y_train[val] for val in train_sample.index]
train_sample.style.set_properties(**{'text-align': 'left'})
```

Out[37]:

	Review	Rating	label	target
5452	great location room able walk to/from train station no problems blocks away, location hotel excellent, kept map times easily able navigate way stopping ask directions time needed, people friendly, did morning guided walk florence recommend doing helped berings efficient way big sights david accademia gallery worth seeing, took bus fiezole nice view florence michelangelo piazza far arno river worth short bus ride, del vechio bridge fun spent time exploring river, wonderful atmosphere day night felt safe time ended darker alleys accident no problems, food expensive nice finds outdoor market dress leather coat.my daughter loved fashion major, hotel adequate clean n't beat location, great hotel dominican republic booked trip paradisus punta cana march 2006 stayed hotel early september, websites reviewed prior booking trip described paradisus excited time finally came, say overall hotel lived expectation 1, weather perfect entire time island, did n't rain not not hot expected, temperature doable felt comfortable normal island wear shorts skirts tank tops bathing suits course, mosquitoes got day big red bites legs rest trip did n't really itch, obviously recommend bring bug spray.2, rooms room nice, did n't major problems bugs ants nothing major, worked room	3	detractors	1

In [38]:
```
# Making a sample of 3 records to display the full text of each
test_sample = X_test.sample(3, random_state=22)
test_sample['target'] = [y_test[val] for val in test_sample.index]
test_sample.style.set_properties(**{'text-align': 'left'})
```

Out[38]:

	Review	Rating	label	target
6365	muse, stayed weekend june 6-9. property surpassed expectations, hot weekend new york temperatures 90 daily, room sanctuary quiet cool comfortable, air conditioner worked perfectly quiet easy set digital temperature control.beds comfy great sheets pillows hard leave morning, weekend did not hear construction noises, 3rd floor no real view no street noise, bathroom huge ample supply luxurious towels, shower worked great good water pressue plenty hot water, rooms spotless, flat screen tv ipod dock nice touches, staff cordial helpful check check, wine hour 5pm 6pm real treat free coffee mornings, usually stay algonquin visits new york sold dates took chance muse, lucky not better choice.put place list not disappointed,	5	not_detractors	0
330	no customer care booked travelocity, flight cancelled snow ice delayed day westin refused refund 1 2 nights, 200+ night hotel bit greedy, not mention view n't great air did n't work barely slept night, expect minimum air work room.still waiting hear travelocity not refund told left town refused got, happily ate cost rental car extra day.we wo n't dealing time soon,	1	detractors	1
16243	great stay jesus berenguer lintag stayed ramada 4 days, service good staff courteous helpful room amenities 4 star level, enjoyed stay wish say behalf wife daughter thank hotel staff, mabuhay,	4	not_detractors	0

4: 2- Data Preprocessing & Exploratory Analysis

The dataset is being divided into two separate subsets: a training set, and a testing (or validation) set. The validation set will allow to assess the performance of the model.

Two parameters are assigned when dividing the dataset:

- random_state=42
  - setting a random seed of 42 ensures that the data split is reproducible
- stratify=y
  - stratified sampling ensures the class distribution is maintained in both sets to address potential class imbalance issues

In order to preprocess the reviews, the following transformations were performed:

- **Standardizing case**
  This step was verified as it is important to ensure text is uniform and consistent. Nevertheless, no extra step was taken as text was already saved as lower case.
- **Character Encoding**
  This was done to ensure consistent representation of text by transforming non-text characters into a normalized format, which will facilitate proper text processing, analysis, and model training.
- **Tokenizing**
  Tokens of words were created. This was done with the `RegexpTokenizer` package from nltk.tokenize. This step will facilitate the conversion of word into a suitable form for analysis and modeling.
- **Stopwords**
  To focus on the data's theme, English stopwords were removed. Manual additions were made in this text's context, such as adding industry specific word (i.e. "hotel", "stay")
- **Lemmatize**
  The `WordNetLemmatizer` package from nltk.stem.wordnet was used to reduce words to their base form, allowing a more accurate analysis
- **Frequency Distribution**
  The `FreqDist` package was used to review in a dictionary-like output, the words and their frequencies
- **WordCloud**
  The words' frequencies were represented visually thanks to the `WordCloud` package
- **Bigrams**
  Bigrams were drawn to have a better understanding of the themes thanks to the `collocations` package and its BigramAssocMeasures
- **Mutual Information Scores**
  Bigrams that occur more than 5 times were examined through `mutual information scores`

Before any transformation is done, a copy of the review column will be done so the original one can always be accessed.

In [39]:
```
# Duplicating the column review
X_train['original_review'] = X_train['Review']
```

In [40]:
```
# Verifying the new column was correctly created
X_train[:1]
```

Out[40]:

	Review	Rating	label	original_review
1282	gem, pleasantly surprised accomondations helpful attentive staff need, breakfast good cookies, room lovely clean comfortable, room fronting bush street bit noisy did not away enjoyment nice boutique hotel, definitely stay time,	4	not_detractors	gem, pleasantly surprised accomondations helpful attentive staff need, breakfast good cookies, room lovely clean comfortable, room fronting bush street bit noisy did not away enjoyment nice boutique hotel, definitely stay time,

4: 2- a) Standardizing Case

I will glance at the first sample of review to get an idea of whether I need to standardize case. This step is important to ensure text is standardized, to prevent models from treating words with different cases as different ones

In [41]:
```
# Isolating the first review into windows_sample
sample_review = train_sample.iloc[1]['Review']
sample_review
```

Out[41]:
```
"great hotel dominican republic booked trip paradisus punta cana march 2006 sta
yed hotel early september, websites reviewed prior booking trip described parad
isus excited time finally came, say overall hotel lived expectation 1, weather
perfect entire time island, did n't rain not not hot expected, temperature doab
le felt comfortable normal island wear shorts skirts tank tops bathing suits co
urse, mosquitoes got day big red bites legs rest trip did n't really itch, obvi
ously recommend bring bug spray.2, rooms room nice, did n't major problems bugs
ants nothing major, worked room nice large, bathroom probably needs facelift i
t.3, hotel grounds amazing big, literally tram hotel, lucky room right pool are
a great location, food pool beach right, just tram lobby area, grounds perfecti
on, cute little huts benches sit walkways nice just relax dinner.4, food though
t good, inclusives jamaica twice traveled parts world definitely rate food para
disus, remember dominican republic not new york city quality n't really compara
ble, lunch buffet amazing, lobster want, n't believe, boyfriend probably ate 5
lobsters day lunch, wonderful, favorite place italian food japanese food, boyfr
iend steak house brazilian food, el romanitico wonderful atmosphere went twice
thought food better, 5. service did n't speak english thought, prepared spend p
atient time explaining things special requests, thought service better grounds
beach pool area restaurants good, 6. excursions did day trip island called sama
ra, took 10 person plane area 40 minute trip bus tour horse riding trip mountai
n hiked waterfall followed horses interested lunch freshly butchered chickens r
ice boat trip island just coast buses plane ride, just 250 person, cheap got, c
ompany called flying tours, 7. tipping tipped people thought deserved, simple,
servers bartenders room keepers got money day, took 100 ones spent 60 people da
ys helps plan, did n't think service better tipped did n't tip just did anyway
s.8, advice n't forget sunscreen camera bug spray need, hotel charges fortune t
imes, walk beach direction run beachside tourist shops items priced reasonably,
dominican republic known unique inca art, like said previous years stayed beach
es negril jamaica, make comparison definitely choose negril dominican republic,
hotel jamaica smaller manageable service outstanding food excellent beach nice
r, said recommend paradisus place dominican republic, culture want experience p
aradisus excellent choice, enjoy goes quick,  "
```

- **Lower case**

Changing to lower case is not necessary, it looks like it was already done. I will verify whether this is the case.

In [42]:
```
# Checking that all reviews are in lowercase
def is_lowercase(df_column):
    is_lower = df_column.str.islower().all()
    print('The assumption that all reviews are in lowercase is '+ str(is_lower))

is_lowercase(X_train['Review'])
```

```
The assumption that all reviews are in lowercase is False
```

It seemed clear that all the text was lowercase though. Let's filter on the reviews that were not, out of curiosity and extra verification.

In [43]:
```
# Creating a sample to filter on reviews that were not in lowercase
special_characters_sample = X_train[~X_train['Review'].str.islower()][:2]
other_special_characters_sample = X_train[~X_train['Review'].str.islower()][10:12
special_characters_sample
```

Out[43]:

	Review	Rating	label	original_review
	time riu year group travelling fifth trip dominican republic visit punta cana, travel understanding i__Ã□_Ã©Â□ not staying ritz someplace like don__Ã□_Ã©_ expect resort provide service, fun regardless little quirks do.check inafter 30 minute bus ride resort arrived resort, check good, 10 people bus resort 7 group, emailed week asking pool view room upgrade pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews i__Ã□_Ã©Â□e read, group mainly 3 rooms 1 person away			time riu year group travelling fifth trip dominican republic visit punta cana, travel understanding i__Ã□_Ã©Â□ not staying ritz someplace like don__Ã□_Ã©_ expect resort provide service, fun regardless little quirks do.check inafter 30 minute bus ride resort arrived resort, check good, 10 people bus resort 7 group, emailed week asking pool view room upgrade pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews i__Ã□_Ã©Â□e read, group mainly 3 rooms 1 person away

The uppercase text is actually characters that are incorrectly translated. I opened the file with the latin-1 encoding, let's try utf-8 as encoding the review if this resolves this.

In [44]:
```
# Saving the index of the special_characters_sample to verify them in the future
spec_char_indices = special_characters_sample.index
```

In [45]:
```
# Loading dataset and saving it as raw_df
utf8_df = pd.read_csv('data/trip-advisor-hotel-reviews.zip', compression="zip", e
```

In [46]:
```
# Checking that all reviews are in lowercase
is_lowercase(utf8_df['Review'])
```

```
The assumption that all reviews are in lowercase is False
```

In [47]:
```
# Inspecting 2 reviews that caused the reviews not to be lowercase
utf8_df.loc[spec_char_indices]
```

6829	pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews i__Ç_é□e read, group mainly 3 rooms 1 person away group room issue, rooms 2116 2117 2118.there 1 double bed 1 single bed rooms, impression naiboa 1 type room turns rooms queen/double mixes time we__Ç_éö try type room.we no problems water pressure hot water supply, power went couple times minute, 30 minutes afternoon long hassle.as mentioned air conditioning hotel tends weak non-existent rooms, rooms ceiling fan keeps room comfortable asking 4 times a/c gave up.i prepared solid beds weren__Ç_é_ nearly hard expected, understand rest group beds softer, said no problems sleeping, day sun drinks it__Ç_é_ easy sleep point concern room keys, mentioned room 2116 key opened 2117 2118 2119. knew rooms handy times big security risk, key engraved room 2170 i__Ç_é□ not sure work door not, point keys 2117 2118 2119 did not open door able open couple own.foodi think hotel offered widest selection breakfast items resort i__Ç_é□e, granted selection problem getting morning started, think best thing french-toast style croissants yummy, breakfast terrace morning nice touch.lunch normally eaten beach pizzeria beach bar burgers pizza fries pasta hot dogs, roasted chicken good swiss chalet, didn__Ç_é_ try main buffet lunch don__Ç_é_ know selection.supper different theme night buffet good, did italian la carte 3 times steak house caribbean restaurant, book a-la cartes need talk buffet manager night pick reservation slip breakfast day, snacks time couldn__Ç_é_ eat, pool bar beach bar pizzeria food served 2, managed 24 hour food bamboo 5 minutes it__Ç_é_ end supper buffet 2 seating__Ç_é_, day arrive buffet manager book seating stay, assigned specific table stay, men wear pants shirt sleeves admitted saw people turned away.barsthere bars visit wish, pool bar beach bar lobby bar night pizzeria bar	4

Whether I open the file with encoding `utf-8` or `latin-1`, I get the same results. I went in and checked the data source: it does contain these special characters from the source.

4: 2- b) Character Encoding

I will then opt to keep the initial encoding of `df` and remove the special characters manually. The next section will encode characters which are coming up so they don't affect modeling.

In [48]:
```
# Reminding the previously created special_characters_reviews
special_characters_sample
```

Out[48]:

	Review	Rating	label	original_review
6829	time riu year group travelling fifth trip dominican republic visit punta cana, travel understanding i__Ã□_Ã©Â□ not staying ritz someplace like don__Ã□_Ã©_ expect resort provide service, fun regardless little quirks do.check inafter 30 minute bus ride resort arrived resort, check good, 10 people bus resort 7 group, emailed week asking pool view room upgrade pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews i__Ã□_Ã©Â□e read, group mainly 3 rooms 1 person away group room issue, rooms 2116 2117 2118.there 1 double bed 1 single bed rooms, impression naiboa 1 type room turns rooms queen/double mixes time we__Ã□_Ã©Ã¶ try type room.we no problems water pressure hot water supply, power went couple times minute, 30 minutes afternoon long hassle.as mentioned air conditioning hotel tends weak non-existent rooms, rooms ceiling fan keeps room comfortable asking 4 times a/c gave up.i prepared solid beds weren__Ã□_Ã©_ nearly hard expected, understand rest group beds softer, said no problems sleeping, day sun drinks it__Ã□_Ã©_ easy sleep point concern room keys, mentioned room 2116 key opened 2117 2118 2119. knew rooms handy times big security risk, key engraved room 2170 i__Ã□_Ã©Â□ not sure work door not, point keys 2117 2118 2119 did not open door able open couple own.foodi think hotel offered widest selection breakfast items resort i__Ã□_Ã©Â□e, granted selection problem getting morning started, think best thing french-toast style croissants yummy, breakfast terrace morning nice touch.lunch normally eaten beach pizzeria beach bar burgers pizza fries pasta hot dogs, roasted chicken good swiss chalet, didn__Ã□_Ã©_ try main buffet lunch don__Ã□_Ã©_ know selection.supper different theme night buffet good, did italian la carte 3 times steak house caribbean restaurant, book a-la cartes need talk buffet manager night pick reservation slip breakfast day, snacks time couldn__Ã□_Ã©_ eat, pool bar beach bar pizzeria food served 2, managed 24 hour food bamboo 5 minutes it__Ã□_Ã©_ end supper buffet 2 seating__Ã□_Ã©_, day arrive buffet	4	not_detractors	time riu year group travelling fifth trip dominican republic visit punta cana, travel understanding i__Ã□_Ã©Â□ not staying ritz someplace like don__Ã□_Ã©_ expect resort provide service, fun regardless little quirks do.check inafter 30 minute bus ride resort arrived resort, check good, 10 people bus resort 7 group, emailed week asking pool view room upgrade pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews i__Ã□_Ã©Â□e read, group mainly 3 rooms 1 person away group room issue, rooms 2116 2117 2118.there 1 double bed 1 single bed rooms, impression naiboa 1 type room turns rooms queen/double mixes time we__Ã□_Ã©Ã¶ try type room.we no problems water pressure hot water supply, power went couple times minute, 30 minutes afternoon long hassle.as mentioned air conditioning hotel tends weak non-existent rooms, rooms ceiling fan keeps room comfortable asking 4 times a/c gave up.i prepared solid beds weren__Ã□_Ã©_ nearly hard expected, understand rest group beds softer, said no problems sleeping, day sun drinks it__Ã□_Ã©_ easy sleep point concern room keys, mentioned room 2116 key opened 2117 2118 2119. knew rooms handy times big security risk, key engraved room 2170 i__Ã□_Ã©Â□ not sure work door not, point keys 2117 2118 2119 did not open door able open couple own.foodi think hotel offered widest selection breakfast items resort i__Ã□_Ã©Â□e, granted selection problem getting morning started, think best thing french-toast style croissants yummy, breakfast terrace morning nice touch.lunch normally eaten beach pizzeria beach bar burgers pizza fries pasta hot dogs, roasted chicken good swiss chalet, didn__Ã□_Ã©_ try main buffet lunch don__Ã□_Ã©_ know selection.supper different theme night buffet good, did italian la carte 3 times steak house caribbean restaurant, book a-la cartes need talk buffet manager night pick reservation slip breakfast day, snacks time couldn__Ã□_Ã©_ eat, pool bar beach bar pizzeria food served 2, managed 24 hour food bamboo 5 minutes it__Ã□_Ã©_ end supper buffet 2 seating__Ã□_Ã©_, day arrive buffet

	Review	Rating	label	original_review
3320	manager book seating stay, assigned specific table stay, men wear pants shirt sleeves admitted saw people turned away.barsthere bars visit wish, pool bar beach bar lobby bar night pizzeria bar bamboo bar swim bar taino bamboo plus didn__Ã□_Ã©_, bar went good drinks.the beach time punta cana pleased sand basically white, beach clean expect, need watch walk smokers just putting butts sand chose, ashtrays help habit.we spent 1 day beach, chairs want sit problem finding available ones use second week quite crowded paid beach workers reserve group 8 loungers easy couple trying 8 heard, poolwe naiboa pool got ones access happy beach, naiboa__Ã□_Ã©_ pool nice wasn__Ã□_Ã©_ bad chairs morning, spend day beach want pool late afternoon hard time getting chairs quite then.entertainmentwe 2 shows, audience participation couple__Ã□ funny cheesy, michael jackson foot work awesome, song selection better movement like watch real deal.each thursday night caraval riu caribbean street, bars set street vendors, highlight quick parade street gets guests involved.activitiesthe animation team activities going day come beach spot hotel guests remind events pool party, didn__Ã□_Ã©_ partake did hold stretching sessions water volleyball aqua-fitness water polo, caribe hilton piÃ«Ã□_a coladas stayed caribe hilton nights june 2005. apprehensive bad reviews arrived staff checked right away asked upgrade gave, stayed renovated room new tower great beautiful ocean view, beach clean order hotel serve right, slide water lot fun, concierge really helpful came recommending restaurants old san juan, oh quite easy using cabs, reviewers mentioned cab rides expensive live dc really inexpensive 6 town, overall great experience lovely hotel,	4	not_detractors	manager book seating stay, assigned specific table stay, men wear pants shirt sleeves admitted saw people turned away.barsthere bars visit wish, pool bar beach bar lobby bar night pizzeria bar bamboo bar swim bar taino bamboo plus didn__Ã□_Ã©_, bar went good drinks.the beach time punta cana pleased sand basically white, beach clean expect, need watch walk smokers just putting butts sand chose, ashtrays help habit.we spent 1 day beach, chairs want sit problem finding available ones use second week quite crowded paid beach workers reserve group 8 loungers easy couple trying 8 heard, poolwe naiboa pool got ones access happy beach, naiboa__Ã□_Ã©_ pool nice wasn__Ã□_Ã©_ bad chairs morning, spend day beach want pool late afternoon hard time getting chairs quite then.entertainmentwe 2 shows, audience participation couple__Ã□ funny cheesy, michael jackson foot work awesome, song selection better movement like watch real deal.each thursday night caraval riu caribbean street, bars set street vendors, highlight quick parade street gets guests involved.activitiesthe animation team activities going day come beach spot hotel guests remind events pool party, didn__Ã□_Ã©_ partake did hold stretching sessions water volleyball aqua-fitness water polo, caribe hilton piÃ«Ã□_a coladas stayed caribe hilton nights june 2005. apprehensive bad reviews arrived staff checked right away asked upgrade gave, stayed renovated room new tower great beautiful ocean view, beach clean order hotel serve right, slide water lot fun, concierge really helpful came recommending restaurants old san juan, oh quite easy using cabs, reviewers mentioned cab rides expensive live dc really inexpensive 6 town, overall great experience lovely hotel,

I will try to encode these characters by normalizing the Unicode string using NFKD (Normalization Form Compatibility Decompostition). After the string is normalized, they are encoded to ASCII and any characters that cannot be represented will be ignored, and then it is encoded back to latin-1.

In [49]:
```
# Importing the relevant package
import unicodedata

# Applying the lambda function to normalize the characters.
special_characters_sample['Review'] = special_characters_sample['Review'].apply(
```

In [50]:
```
# Reviewing the modified review
special_characters_sample
```

Out[50]:

	Review	Rating	label	original_review
	time riu year group travelling fifth trip dominican republic visit punta cana, travel understanding i__A_AA not staying ritz someplace like don__A_A_ expect resort provide service, fun regardless little quirks do.check inafter 30 minute bus ride resort arrived resort, check good, 10 people bus resort 7 group, emailed week asking pool view room upgrade pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews i__A_AAe read, group mainly 3 rooms 1 person away			time riu year group travelling fifth trip dominican republic visit punta cana, travel understanding i__Ã□_Ã©Â□ not staying ritz someplace like don__Ã□_Ã©_ expect resort provide service, fun regardless little quirks do.check inafter 30 minute bus ride resort arrived resort, check good, 10 people bus resort 7 group, emailed week asking pool view room upgrade pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews i__Ã□_Ã©Â□e read, group mainly 3 rooms 1 person away

This helped. Now, our special characters are replaced by __A_A_ . I will remove them from the reviews on the next cell.

In [51]:
```
special_characters_sample['Review'] = special_characters_sample['Review'].str.rep
special_characters_sample['Review'] = special_characters_sample['Review'].str.rep

# Verifying it removed it
special_characters_sample
```

Out[51]:

	Review	Rating	label	original_review
6829	time riu year group travelling fifth trip dominican republic visit punta cana, travel understanding i not staying ritz someplace like don expect resort provide service, fun regardless little quirks do.check inafter 30 minute bus ride resort arrived resort, check good, 10 people bus resort 7 group, emailed week asking pool view room upgrade pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews ie read, group mainly 3 rooms 1 person away group room issue, rooms 2116 2117 2118.there 1 double bed 1 single bed rooms, impression naiboa 1 type room turns rooms queen/double mixes time we try type room.we no problems water pressure hot water supply, power went couple times minute, 30 minutes afternoon long hassle.as mentioned air conditioning hotel tends weak non-existent rooms, rooms ceiling fan keeps room comfortable asking 4 times a/c gave up.i prepared solid beds weren nearly hard expected, understand rest group beds softer, said no problems sleeping, day sun drinks it easy sleep point concern room keys, mentioned room 2116 key opened 2117 2118 2119. knew rooms handy times big security risk, key engraved room 2170 i not sure work door not, point keys 2117 2118 2119 did not open door able open couple own.foodi think hotel offered widest selection breakfast items resort ie, granted selection problem getting morning started, think best thing french-toast style croissants yummy, breakfast terrace morning nice touch.lunch normally eaten beach pizzeria beach bar burgers pizza fries pasta hot dogs, roasted chicken good swiss chalet, didn try main buffet lunch don know selection.supper different theme night buffet good, did italian la carte 3 times steak house caribbean restaurant, book a-la cartes need talk buffet manager night pick reservation slip breakfast day, snacks time couldn eat, pool bar beach bar pizzeria food served 2, managed 24 hour food bamboo 5 minutes it end supper buffet 2 seating, day arrive buffet manager book seating stay, assigned specific table stay, men wear pants shirt sleeves admitted saw people turned away.barsthere bars visit wish, pool bar beach bar lobby bar night	4	not_detractors	time riu year group travelling fifth trip dominican republic visit punta cana, travel understanding i__Ã□_Ã©Â□ not staying ritz someplace like don__Ã□_Ã©_ expect resort provide service, fun regardless little quirks do.check inafter 30 minute bus ride resort arrived resort, check good, 10 people bus resort 7 group, emailed week asking pool view room upgrade pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews i__Ã□_Ã©Â□e read, group mainly 3 rooms 1 person away group room issue, rooms 2116 2117 2118.there 1 double bed 1 single bed rooms, impression naiboa 1 type room turns rooms queen/double mixes time we__Ã□_Ã©Ã¶ try type room.we no problems water pressure hot water supply, power went couple times minute, 30 minutes afternoon long hassle.as mentioned air conditioning hotel tends weak non-existent rooms, rooms ceiling fan keeps room comfortable asking 4 times a/c gave up.i prepared solid beds weren__Ã□_Ã©_ nearly hard expected, understand rest group beds softer, said no problems sleeping, day sun drinks it__Ã□_Ã©_ easy sleep point concern room keys, mentioned room 2116 key opened 2117 2118 2119. knew rooms handy times big security risk, key engraved room 2170 i__Ã□_Ã©Â□ not sure work door not, point keys 2117 2118 2119 did not open door able open couple own.foodi think hotel offered widest selection breakfast items resort i__Ã□_Ã©Â□e, granted selection problem getting morning started, think best thing french-toast style croissants yummy, breakfast terrace morning nice touch.lunch normally eaten beach pizzeria beach bar burgers pizza fries pasta hot dogs, roasted chicken good swiss chalet, didn__Ã□_Ã©_ try main buffet lunch don__Ã□_Ã©_ know selection.supper different theme night buffet good, did italian la carte 3 times steak house caribbean restaurant, book a-la cartes need talk buffet manager night pick reservation slip breakfast day, snacks time couldn__Ã□_Ã©_ eat, pool bar beach bar pizzeria food served 2, managed 24 hour food bamboo 5 minutes it__Ã□_Ã©_ end supper buffet 2 seating__Ã□_Ã©_, day arrive buffet

	Review	Rating	label	original_review
	pizzeria bar bamboo bar swim bar taino bamboo plus didn, bar went good drinks.the beach time punta cana pleased sand basically white, beach clean expect, need watch walk smokers just putting butts sand chose, ashtrays help habit.we spent 1 day beach, chairs want sit problem finding available ones use second week quite crowded paid beach workers reserve group 8 loungers easy couple trying 8 heard, poolwe naiboa pool got ones access happy beach, naiboa pool nice wasn bad chairs morning, spend day beach want pool late afternoon hard time getting chairs quite then.entertainmentwe 2 shows, audience participation couple__A funny cheesy, michael jackson foot work awesome, song selection better movement like watch real deal.each thursday night caraval riu caribbean street, bars set street vendors, highlight quick parade street gets guests involved.activitiesthe animation team activities going day come beach spot hotel guests remind events pool party, didn partake did hold stretching sessions water volleyball aqua-fitness water polo,			manager book seating stay, assigned specific table stay, men wear pants shirt sleeves admitted saw people turned away.barsthere bars visit wish, pool bar beach bar lobby bar night pizzeria bar bamboo bar swim bar taino bamboo plus didn__Ã□_Ã©_, bar went good drinks.the beach time punta cana pleased sand basically white, beach clean expect, need watch walk smokers just putting butts sand chose, ashtrays help habit.we spent 1 day beach, chairs want sit problem finding available ones use second week quite crowded paid beach workers reserve group 8 loungers easy couple trying 8 heard, poolwe naiboa pool got ones access happy beach, naiboa__Ã□_Ã©_ pool nice wasn__Ã□_Ã©_ bad chairs morning, spend day beach want pool late afternoon hard time getting chairs quite then.entertainmentwe 2 shows, audience participation couple__Ã□ funny cheesy, michael jackson foot work awesome, song selection better movement like watch real deal.each thursday night caraval riu caribbean street, bars set street vendors, highlight quick parade street gets guests involved.activitiesthe animation team activities going day come beach spot hotel guests remind events pool party, didn__Ã□_Ã©_ partake did hold stretching sessions water volleyball aqua-fitness water polo,
3320	caribe hilton piAA_a coladas stayed caribe hilton nights june 2005. apprehensive bad reviews arrived staff checked right away asked upgrade gave, stayed renovated room new tower great beautiful ocean view, beach clean order hotel serve right, slide water lot fun, concierge really helpful came recommending restaurants old san juan, oh quite easy using cabs, reviewers mentioned cab rides expensive live dc really inexpensive 6 town, overall great experience lovely hotel,	4	not_detractors	caribe hilton piÃ«Ã□_a coladas stayed caribe hilton nights june 2005. apprehensive bad reviews arrived staff checked right away asked upgrade gave, stayed renovated room new tower great beautiful ocean view, beach clean order hotel serve right, slide water lot fun, concierge really helpful came recommending restaurants old san juan, oh quite easy using cabs, reviewers mentioned cab rides expensive live dc really inexpensive 6 town, overall great experience lovely hotel,

Great, this resolved it. I will save the steps into a function and will apply it to the full train data

In [52]:
```
# Creating the function character_encoding to handle them

def character_encoding(df_column):
    # Importing the relevant package
    import unicodedata

    # Normalizing characters
    df_column = df_column.apply(lambda x: unicodedata.normalize('NFKD', x). encoc

    # Generalizing to list of characters to remove
    characters_to_remove = ['__A_AA', '__A_A_', '__AA', '__A', '_Aow__', 'AAA', '

    # Removing the characters from text
    for char in characters_to_remove:
        df_column = df_column.str.replace(char, '')

    # Returning the df_column
    return df_column
```

In [53]:
```
# Testing the function
character_encoding(other_special_characters_sample['Review'])
```

Out[53]:
```
4875
wow oh wow, partner booked surprise, stayed fabulous room poster bed wonderfull
y luxurious feel, think tudor suite, wished winter lit sat snug area, terry but
ler great guide hotel local area, entertaining informative, went bar pre-dinner
drinks daniel barman welcoming stephan offered champagne champagne day day mile
stone, just fantastic lovely nibbles happy stayed bar night, dinner cheneston w
onderful experience lovely setting, felt important special not just partner tre
ated, zana lovely treated like royalty did staff, meal truly treat, started end
ive gruyere pancetta terrine delicious main course, instead went pot roast dove
r sole world, unfortunately n't manage dessert did steal mouthful simon souffl
oh goodness, thomas sure excellent wines course convinced cognac coffee lounge,
used, breakfast calm quality food service near perfect, can thank staff making
short break memorable special ca n't loved milestone ca n't wait return just ar
range girls night dinner, sure stephan thomas daniel make great night,
19458    best vacation spotever best vacation, went large group, imagine travel
ing large group cumbersome relief, need speak little spanish listen carefully e
nglish it totally doable repetition, went way accommodate, eager make stay exce
lente, tips, n't expecting seasons know surroundings, 3rd world country, beds k
inda firm cater europeans americans little flexible days, change linens everyda
y refresh liquor room days, food water restaurants safe, n't believe things wri
tten, just got week ago not sick, food fresh lacks salt prefer, totally purifie
d gallon water stocked fridge, drinking doing drink water, rarely saw individua
l bottles water, ice make drinks water serve purified n't believe hype, beach t
otally open safe night, careful waves undertow night, fierce, love juan carlos
froggy, hotel, total sweethearts, went club pacha, club rocked, site shuttle no
cabs leave safety compound, oh yeah did mention pools heated poolside bar, best
thing swimming cold ocean, shopping site not expensive considering it thing vac
ation pay, not sure place conducive kiddos available booze alcohol all-inclusiv
e thing winner, not mention stay riu wristband gives access area hotels, make s
ure breakfast sign dinner italian restaurant brazilian steakhouse days, fabulou
s, sign 9-12pm dinner night, try sea bass italian place brazilian restaurant ba
con wrapped bananas, sounds gross trust die, brazilian spot there lot meat litt
le veggies prepared, tipped lot staff great, couple dollars long way, aware sta
ff flirts, smooth totally caught, watch, riu punta cana, best vaca say hi georg
ina jefe eugenio definitely possibly yearly,
Name: Review, dtype: object
```

I could now apply it to our full X_train['Review'] column directly, as I kept an unmodified copy as the original. I later defined a function combining all preprocessing steps, so the next cells will be commented out.

In [54]:
```
# Reviewing the other sample
# other_special_characters_sample
```

In [55]:
```
# Calling the function on full X_train
# X_train['Review'] = character_encoding(X_train['Review'])
```

In [56]:
```
# X_train[~X_train['Review'].str.islower()]
```

In [57]:
```
# Verifying that words containing 'a' lowercase were not removed
# assert len(X_train[X_train['Review'].str.contains('stay')]) > 0
```

In [58]:
```
# Verifying that our sample reviews are correctly handled now
# X_train.loc[spec_char_indices]
```

In [59]:
```
# Verifying that I correctly have lowercase everywhere
# is_lowercase(X_train['Review'])
```

This now resolved it and gave me the result I hoped for. I can move on to the next step.

4: 2- c) Tokenizing

Tokenizing text data is one of the fundamental data cleaning steps to further convert words into a suitable form for analysis and modeling.

In [60]:
```
# Reviewing one of our sample of 2 reviews
special_characters_sample
```

Out[60]:

	Review	Rating	label	original_review
6829	time riu year group travelling fifth trip dominican republic visit punta cana, travel understanding i not staying ritz someplace like don expect resort provide service, fun regardless little quirks do.check inafter 30 minute bus ride resort arrived resort, check good, 10 people bus resort 7 group, emailed week asking pool view room upgrade pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews ie read, group mainly 3 rooms 1 person away group room issue, rooms 2116 2117 2118.there 1 double bed 1 single bed rooms, impression naiboa 1 type room turns rooms queen/double mixes time we try type room.we no problems water pressure hot water supply, power went couple times minute, 30 minutes afternoon long hassle.as mentioned air conditioning hotel tends weak non-existent rooms, rooms ceiling fan keeps room comfortable asking 4 times a/c gave up.i prepared solid beds weren nearly hard expected, understand rest group beds softer, said no problems sleeping, day sun drinks it easy sleep point concern room keys, mentioned room 2116 key opened 2117 2118 2119. knew rooms handy times big security risk, key engraved room 2170 i not sure work door not, point keys 2117 2118 2119 did not open door able open couple own.foodi think hotel offered widest selection breakfast items resort ie, granted selection problem getting morning started, think best thing french-toast style croissants yummy, breakfast terrace morning nice touch.lunch normally eaten beach pizzeria beach bar burgers pizza fries pasta hot dogs, roasted chicken good swiss chalet, didn try main buffet lunch don know selection.supper different theme night buffet good, did italian la carte 3 times steak house caribbean restaurant, book a-la cartes need talk buffet manager night pick reservation slip breakfast day, snacks time couldn eat, pool bar beach bar pizzeria food served 2, managed 24 hour food bamboo 5 minutes it end supper buffet 2 seating, day arrive buffet manager book seating stay, assigned specific table stay, men wear pants shirt sleeves admitted saw people turned away.barsthere bars visit wish, pool bar beach bar lobby bar night	4	not_detractors	time riu year group travelling fifth trip dominican republic visit punta cana, travel understanding i__Ã□_Ã©Â□ not staying ritz someplace like don__Ã□_Ã©_ expect resort provide service, fun regardless little quirks do.check inafter 30 minute bus ride resort arrived resort, check good, 10 people bus resort 7 group, emailed week asking pool view room upgrade pleasantly surprised gotten, walked lobby naturally headed reception desk called small table given registration form room keys brought meeting room cocktail drink given general info hotel hotel not tour operator day, keys safe included package given session.roomas mentioned asked pool view room based reviews i__Ã□_Ã©Â□e read, group mainly 3 rooms 1 person away group room issue, rooms 2116 2117 2118.there 1 double bed 1 single bed rooms, impression naiboa 1 type room turns rooms queen/double mixes time we__Ã□_Ã©Ã¶ try type room.we no problems water pressure hot water supply, power went couple times minute, 30 minutes afternoon long hassle.as mentioned air conditioning hotel tends weak non-existent rooms, rooms ceiling fan keeps room comfortable asking 4 times a/c gave up.i prepared solid beds weren__Ã□_Ã©_ nearly hard expected, understand rest group beds softer, said no problems sleeping, day sun drinks it__Ã□_Ã©_ easy sleep point concern room keys, mentioned room 2116 key opened 2117 2118 2119. knew rooms handy times big security risk, key engraved room 2170 i__Ã□_Ã©Â□ not sure work door not, point keys 2117 2118 2119 did not open door able open couple own.foodi think hotel offered widest selection breakfast items resort i__Ã□_Ã©Â□e, granted selection problem getting morning started, think best thing french-toast style croissants yummy, breakfast terrace morning nice touch.lunch normally eaten beach pizzeria beach bar burgers pizza fries pasta hot dogs, roasted chicken good swiss chalet, didn__Ã□_Ã©_ try main buffet lunch don__Ã□_Ã©_ know selection.supper different theme night buffet good, did italian la carte 3 times steak house caribbean restaurant, book a-la cartes need talk buffet manager night pick reservation slip breakfast day, snacks time couldn__Ã□_Ã©_ eat, pool bar beach bar pizzeria food served 2, managed 24 hour food bamboo 5 minutes it__Ã□_Ã©_ end supper buffet 2 seating__Ã□_Ã©_, day arrive buffet

	Review	Rating	label	original_review
	pizzeria bar bamboo bar swim bar taino bamboo plus didn, bar went good drinks.the beach time punta cana pleased sand basically white, beach clean expect, need watch walk smokers just putting butts sand chose, ashtrays help habit.we spent 1 day beach, chairs want sit problem finding available ones use second week quite crowded paid beach workers reserve group 8 loungers easy couple trying 8 heard, poolwe naiboa pool got ones access happy beach, naiboa pool nice wasn bad chairs morning, spend day beach want pool late afternoon hard time getting chairs quite then.entertainmentwe 2 shows, audience participation couple__A funny cheesy, michael jackson foot work awesome, song selection better movement like watch real deal.each thursday night caraval riu caribbean street, bars set street vendors, highlight quick parade street gets guests involved.activitiesthe animation team activities going day come beach spot hotel guests remind events pool party, didn partake did hold stretching sessions water volleyball aqua-fitness water polo,			manager book seating stay, assigned specific table stay, men wear pants shirt sleeves admitted saw people turned away.barsthere bars visit wish, pool bar beach bar lobby bar night pizzeria bar bamboo bar swim bar taino bamboo plus didn__Ã□_Ã©_, bar went good drinks.the beach time punta cana pleased sand basically white, beach clean expect, need watch walk smokers just putting butts sand chose, ashtrays help habit.we spent 1 day beach, chairs want sit problem finding available ones use second week quite crowded paid beach workers reserve group 8 loungers easy couple trying 8 heard, poolwe naiboa pool got ones access happy beach, naiboa__Ã□_Ã©_ pool nice wasn__Ã□_Ã©_ bad chairs morning, spend day beach want pool late afternoon hard time getting chairs quite then.entertainmentwe 2 shows, audience participation couple__Ã□ funny cheesy, michael jackson foot work awesome, song selection better movement like watch real deal.each thursday night caraval riu caribbean street, bars set street vendors, highlight quick parade street gets guests involved.activitiesthe animation team activities going day come beach spot hotel guests remind events pool party, didn__Ã□_Ã©_ partake did hold stretching sessions water volleyball aqua-fitness water polo,
3320	caribe hilton piAA_a coladas stayed caribe hilton nights june 2005. apprehensive bad reviews arrived staff checked right away asked upgrade gave, stayed renovated room new tower great beautiful ocean view, beach clean order hotel serve right, slide water lot fun, concierge really helpful came recommending restaurants old san juan, oh quite easy using cabs, reviewers mentioned cab rides expensive live dc really inexpensive 6 town, overall great experience lovely hotel,	4	not_detractors	caribe hilton piÃ«Ã□_a coladas stayed caribe hilton nights june 2005. apprehensive bad reviews arrived staff checked right away asked upgrade gave, stayed renovated room new tower great beautiful ocean view, beach clean order hotel serve right, slide water lot fun, concierge really helpful came recommending restaurants old san juan, oh quite easy using cabs, reviewers mentioned cab rides expensive live dc really inexpensive 6 town, overall great experience lovely hotel,

I will use `RegexpTokenizer` from NLTK to create tokens of tow or more consecutive word characters, which include letters, numbers and underscores.

- **Tokenizing Pattern**

The token pattern was defined as follows:

- `r` indicates the string is in Python language
- `(?u)` allows to match Unicode characters
- `\b` determines where the word starts or ends

- \b determines where the word starts or ends
- \w matches any word character
- \w+ matches 1 or more word characters
- \b finally the word boundary ensures the matched pattern ends at a word boundary

In [61]:
```
# Importing RegexpTokenizer
from nltk.tokenize import RegexpTokenizer

token_pattern = r"(?u)\b\w\w+\b"

# Instantiating the tokenizer
tokenizer = RegexpTokenizer(token_pattern)

# Tokenizing the reviews
special_characters_sample['Review'] = special_characters_sample['Review'].apply(
special_characters_sample['Review']
```

Out[61]:
```
6829    [time, riu, year, group, travelling, fifth, trip, dominican, republic,
visit, punta, cana, travel, understanding, not, staying, ritz, someplace, like,
don, expect, resort, provide, service, fun, regardless, little, quirks, do, che
ck, inafter, 30, minute, bus, ride, resort, arrived, resort, check, good, 10, p
eople, bus, resort, group, emailed, week, asking, pool, view, room, upgrade, pl
easantly, surprised, gotten, walked, lobby, naturally, headed, reception, desk,
called, small, table, given, registration, form, room, keys, brought, meeting,
room, cocktail, drink, given, general, info, hotel, hotel, not, tour, operator,
day, keys, safe, included, package, given, session, roomas, mentioned, asked, p
ool, view, room, based, reviews, ie, read, group, ...]
3320
[caribe, hilton, piAA_a, coladas, stayed, caribe, hilton, nights, june, 2005, a
pprehensive, bad, reviews, arrived, staff, checked, right, away, asked, upgrad
e, gave, stayed, renovated, room, new, tower, great, beautiful, ocean, view, be
ach, clean, order, hotel, serve, right, slide, water, lot, fun, concierge, real
ly, helpful, came, recommending, restaurants, old, san, juan, oh, quite, easy,
using, cabs, reviewers, mentioned, cab, rides, expensive, live, dc, really, ine
xpensive, town, overall, great, experience, lovely, hotel]
Name: Review, dtype: object
```

I could now apply it to our full X_train['Review'] column directly, as I kept an unmodified copy as the original.
I later defined a function combining all preprocessing steps, so the next cells will be commented out.

In [62]:
```
# Applying the tokenizer on the full train data
# X_train['Review'] = X_train['Review'].apply(lambda x: tokenizer.tokenize(x))
```

In [63]:
```
# Inspecting it
# X_train.head()
```

4: 2- d) Stopwords

Then, I will be removing stopwords so I can focus on the the text data's theme. It looked like the text was somewhat preprocessed, so I will review whether there is a need to further extract Stopwords

This step will not be applied to the full X_train data, as I will use it separately to test whether it positively impacts our model in the future when I reach this step.

In [64]:
```
# Importing relevant packages
import nltk
nltk.download('stopwords', quiet=True)
from nltk.corpus import stopwords

# Creating list to store stopwords
stopwords_list = stopwords.words('english')
stopwords_list[:5]
```

Out[64]:
```
['i', 'me', 'my', 'myself', 'we']
```

I will continue using our previously created special_characters_sample

In [65]:
```
special_characters_sample
```

	expect, resort, provide, service, fun, regardless, little, quirks, do, check, inafter, 30, minute, bus, ride, resort, arrived, resort, check, good, 10, people, bus, resort, group, emailed, week, asking, pool, view, room, upgrade, pleasantly, surprised, gotten, walked, lobby, naturally, headed, reception, desk, called, small, table, given, registration, form, room, keys, brought, meeting, room, cocktail, drink, given, general, info, hotel, hotel, not, tour,			sleeping, day sun drinks it__Ã□_Ã©_ easy sleep point concern room keys, mentioned room 2116 key opened 2117 2118 2119. knew rooms handy times big security risk, key engraved room 2170 i__Ã□_Ã©Â□ not sure work door not, point keys 2117 2118 2119 did not open door able open couple own.foodi think hotel offered widest selection breakfast items resort i__Ã□_Ã©Â□e, granted selection problem getting morning started, think best thing french-toast style croissants yummy, breakfast terrace morning nice touch.lunch normally eaten
6829		4	not_detractors	beach pizzeria beach bar burgers pizza fries pasta hot dogs, roasted chicken good swiss chalet, didn__Ã□_Ã©_ try main buffet lunch don__Ã□_Ã©_ know selection.supper different theme night buffet good, did italian la carte 3 times steak house caribbean restaurant, book a-la cartes need talk buffet manager night pick reservation slip breakfast day, snacks time couldn__Ã□_Ã©_ eat, pool bar beach bar pizzeria food served 2, managed 24 hour food bamboo 5 minutes it__Ã□_Ã©_ end supper buffet 2 seating__Ã□_Ã©_, day arrive buffet manager book seating stay, assigned specific table stay, men wear pants shirt sleeves admitted saw

In [66]:
```
# Defining function that takes in a list of strings and returns only those that d
def remove_stopwords(token_list, stopwords_list):
    stopwords_removed = [token for token in token_list if token not in stopwords_
    return stopwords_removed
```

In [67]:
```
# Testing it on the tokens example
tokens_example = special_characters_sample.iloc[0]['Review']
print("Length with stopwords: ", len(tokens_example))

tokens_example_without_stopwords = remove_stopwords(tokens_example, stopwords_lis
print("Length with stopwords: ", len(tokens_example_without_stopwords))
```

```
Length with stopwords:  526
Length with stopwords:  494
```

Despite the assumption, stopwords did exist and could be removed. The length of the review was not highly impacted so the main meaning is less at risk of being impacted.

In [68]:
```
# Reviewing token example review
tokens_example_without_stopwords[:5]
```

Out[68]:
```
['time', 'riu', 'year', 'group', 'travelling']
```

I will review during the exploratory analysis whether some words need to be manually added to the list.

4: 2- e) Lemmatize

The WordNetLemmatizer package from nltk.stem.wordnet was used to reduce words to their base form, allowing a more accurate analysis.
It first required to be downloaded for Jupyter Notebook. Once the initial download is done, this step was commented out.

This step will not be applied to the full X_train data, as I will use it separately to test whether it positively impacts our model in the future when I reach this step.

In [69]:
```
# Importing relevant package
from nltk.stem.wordnet import WordNetLemmatizer
nltk.download('wordnet')
nltk.download('omw-1.4')

# Instantiating the Lemmatizer
lemmatizer = WordNetLemmatizer()
```

In [70]:
```
# Instantiating the Lemmatizer
def lemmatize_words(token_list):
    lemmatized_tokens = [lemmatizer.lemmatize(token, pos='v') for token in token_
    return lemmatized_tokens
```

In [71]:
```
# Print the tokens_example without stopwords before
print('Before')
print(tokens_example_without_stopwords[:5])
```

```
Before
['time', 'riu', 'year', 'group', 'travelling']
```

In [72]:
```
# Let's review the impact on our token example
print('After')
lemmatize_words(tokens_example_without_stopwords)[:5]
```

```
After
```

Out[72]:
```
['time', 'riu', 'year', 'group', 'travel']
```

The words were lemmatized as intended and reduced to their base form. This can be visible on words: travelling changed into travel.

4: 2- f) Summarizing Review Preprocessing

The previous steps will be gathered into one main function so the above steps can be applied to the entire dataset.

In [73]:
```
# Ensuring the relevant packages are imported
# These were imported individually before but are reminded here if they needed to
# from nltk.tokenize import RegexpTokenizer
# nltk.download('stopwords', quiet=True)
# from nltk.corpus import stopwords
# from nltk.stem.wordnet import WordNetLemmatizer
```

In [74]:
```
# Defining the function
def preprocess_review(df_column):
    # 1. Standardizing case
    df_column = df_column.str.lower()

    # 2. Character Encoding
    # Calling the previously creating the function
    df_column = character_encoding(df_column)

    # 3. Tokenizing
    # Defining the token pattern
    token_pattern = r"(?u)\b\w\w+\b"
    # Instantiating the tokenizer
    tokenizer = RegexpTokenizer(token_pattern)
    # Tokenizing
    # df_column = tokenizer.tokenize(df_column)
    df_column = df_column.apply(lambda x: tokenizer.tokenize(x))

    # 4. Stopwords
    # Creating list to store stopwords
    stopwords_list = stopwords.words('english')

#     Storing words to add to list of stopwords
#     manual_stopwords = []
    # Adding to list of stopwords
#     for word in manual_stopwords:
#         stopwords_list.append(word)

    # Removing stopwords
    # df_column = [token for token in df_column if token not in stopwords_list]
    df_column = df_column.apply(lambda tokens: [token for token in tokens if toke


    # 5. Lemmatizing
    # Instantiating the Lemmatizer
    lemmatizer = WordNetLemmatizer()

    # Lemmatizing words
    # df_column = [lemmatizer.lemmatize(token) for token in df_column]
    df_column = df_column.apply(lambda tokens: [lemmatizer.lemmatize(token, pos='

    #Returning the preprocessed review
    return df_column
```

In [75]:
```
# Verifying the type of X_train['Review']
type(X_train['Review'])
```

Out[75]:
```
pandas.core.series.Series
```

In [76]:
```
# Testing our function on a few reviews
preprocess_review(X_train['Review'].iloc[95:99])
```

Out[76]:
```
9751
[arc, la, rambla, recommed, hotel, august, 2007, stay, hotel, nights, good, thi
ng, say, clean, safe, room, small, shower, curtain, shower, wash, hold, shower,
hand, guy, reception, helpful, rude, enter, hotel, terrible, smell, follow, go,
stairs, room, balcony, great, air, crap, work, properly, group, room, windows,
air, air, better, room, get, older, room, hotel, decorate, part, hotel, ther, p
erfect, location, la, rambla, shop, im, sure, nicer, hotels, strip]
1213
[wonderful, place, celebrate, 20th, anniversary, place, best, really, quiet, st
aff, perfect, love, ability, walk, snack, day, want, wine, cheese, social, eve
n, want, make, 20th, anniversary, special, casablanca, happen, john, kori]
11506
[style, sophistication, arrive, early, hours, morning, hawaii, tire, jet, lag,
12, hour, flight, soon, perk, saw, hotel, fantastic, property, stay, new, york,
club, floor, room, worth, upgrade, huge, extremely, tastefully, furnish, westi
n, heavenly, bed, say, heavenly, heavenly, bath, westin, promote, little, disap
point, bed, nice, hotel, bathroom, nothing, ordinary, go, say, room, clean, sta
ff, generally, helpful, location, superb, mid, town, time, sqaure, 42nd, stree
t, close, especially, show, shop, greyhound, bus, tour, stop, round, corner, wo
rth, money, overall, great, hotel, stylish, stay, city, cities]
7076     [completely, relax, husband, spend, honeymoon, secrets, week, 14th, ri
de, airport, take, hour, 10, minutes, typical, bus, ride, driver, stop, beers,
way, check, room, ready, eat, lunch, return, desk, room, ready, annemarie, mov
e, room, absolutely, lovely, book, corner, suite, large, balcony, move, floor,
2nd, floor, corner, suite, fabulous, flower, fruit, basket, champagne, sash, do
or, read, honeymooners, bldg, center, resort, convenient, restaurants, activiti
es, room, spotless, daily, maid, service, excellant, room, service, food, good,
take, hour, half, food, restaurants, good, spend, day, pool, beach, palapas, ni
ce, break, sun, careful, mamma, wan, na, potent, stuff, loud, music, pool, min
d, average, monthly, salary, 250, course, ...]
Name: Review, dtype: object
```

In [77]:
```
# Applying this to our whole train data
X_train['Review'] = preprocess_review(X_train['Review'])
```

In [78]:
```
# Verifying that I correctly have lowercase strings after character encoding, and
is_lowercase(X_train['Review'])
```

```
The assumption that all reviews are in lowercase is True
```

In [79]:
```
# Now creating a column with of preprocessed reviews without being stored in list
X_train['Review_prep_nolist'] = X_train['Review'].apply(lambda x: ' '.join(x))
```

In [80]:
```
# Inspecting the newly created column
X_train[:2]
```

Out[80]:

	Review	Rating	label	original_review	Review_prep_nolist
1282	[gem, pleasantly, surprise, accomondations, helpful, attentive, staff, need, breakfast, good, cookies, room, lovely, clean, comfortable, room, front, bush, street, bite, noisy, away, enjoyment, nice, boutique, hotel, definitely, stay, time]	4	not_detractors	gem, pleasantly surprised accomondations helpful attentive staff need, breakfast good cookies, room lovely clean comfortable, room fronting bush street bit noisy did not away enjoyment nice boutique hotel, definitely stay time,	gem pleasantly surprise accomondations helpful attentive staff need breakfast good cookies room lovely clean comfortable room front bush street bite noisy away enjoyment nice boutique hotel definitely stay time
10661	[love, fita, wife, spend, nights, hotel, fita, march, 2008, hotel, staff, fantastic, helpful, pleasant, recommend, hotel, travel, amsterdam, hotel, locate, minute, walk, van, gogh, rijks, museums, 10, minute, walk, leidsplein, variety, store, shop, restaurants, locate, tram, stop, directly, hotel, easy, access, entire, city, walk, fita, way, center, city, 30, minutes, beautiful, accesible, friendly, definately, travel, fita, wonderful, amsterdam]	5	not_detractors	loved fita wife spent nights hotel fita march 2008. hotel staff fantastic helpful pleasant, recommend hotel traveling amsterdam.the hotel located minute walk van gogh rijks museums 10 minute walk leidsplein variety stores shops restaurants located.there tram stop directly hotel, easy access entire city, did walk fita way center city 30 minutes.beautiful accesible friendly definately travel fita wonderful amsterdam,	love fita wife spend nights hotel fita march 2008 hotel staff fantastic helpful pleasant recommend hotel travel amsterdam hotel locate minute walk van gogh rijks museums 10 minute walk leidsplein variety store shop restaurants locate tram stop directly hotel easy access entire city walk fita way center city 30 minutes beautiful accesible friendly definately travel fita wonderful amsterdam

- **Preprocessing test data for later use**

In [81]:
```
# Creating a duplicate of the review column
X_test['original_review'] = X_test['Review']
```

In [82]:
```
# Preprocessing reviews to the test data
X_test['Review'] = preprocess_review(X_test['Review'])
```

In [83]:
```
# Now creating a column of preprocessed reviews without being stored in lists, fo
X_test['Review_prep_nolist'] = X_test['Review'].apply(lambda x: ' '.join(x))
```

In [84]:
```
# Verifying that I correctly have lowercase strings after character encoding, and
is_lowercase(X_test['Review'])
```

```
The assumption that all reviews are in lowercase is True
```

4: 2- g) Frequency Distributions

A frequency distribution is a data structure which can be compared to a list displaying how often a piece of data - or a word appears.

In order to do this, I will use the `FreqDist` package. It allows us to pass in a single list of words. It then produces a dictionary-like output of those words and their frequencies.

I will visualize the top 10 words to evaluate further what cleaning needs to be done.

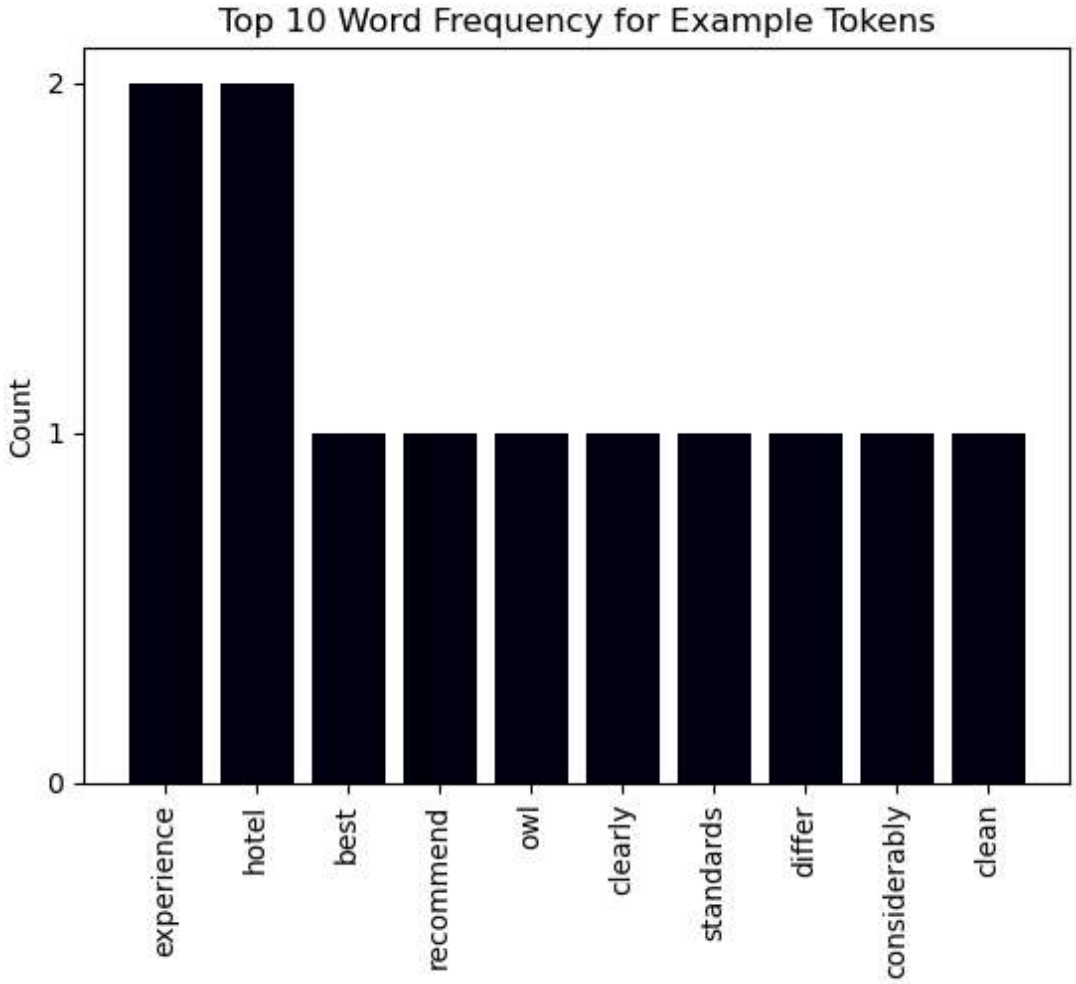
In [85]:
```
# Importing the relevant package: FreqDist
from nltk import FreqDist
```

- **FreqDist**

In [86]:
```
# Creating an example of Frequency distribution for 1 review
example_freq_dist = FreqDist(X_train.iloc[100]['Review'][:20])
example_freq_dist
```

Out[86]:
```
FreqDist({'experience': 2, 'hotel': 2, 'best': 1, 'recommend': 1, 'owl': 1, 'cl
early': 1, 'standards': 1, 'differ': 1, 'considerably': 1, 'clean': 1, ...})
```

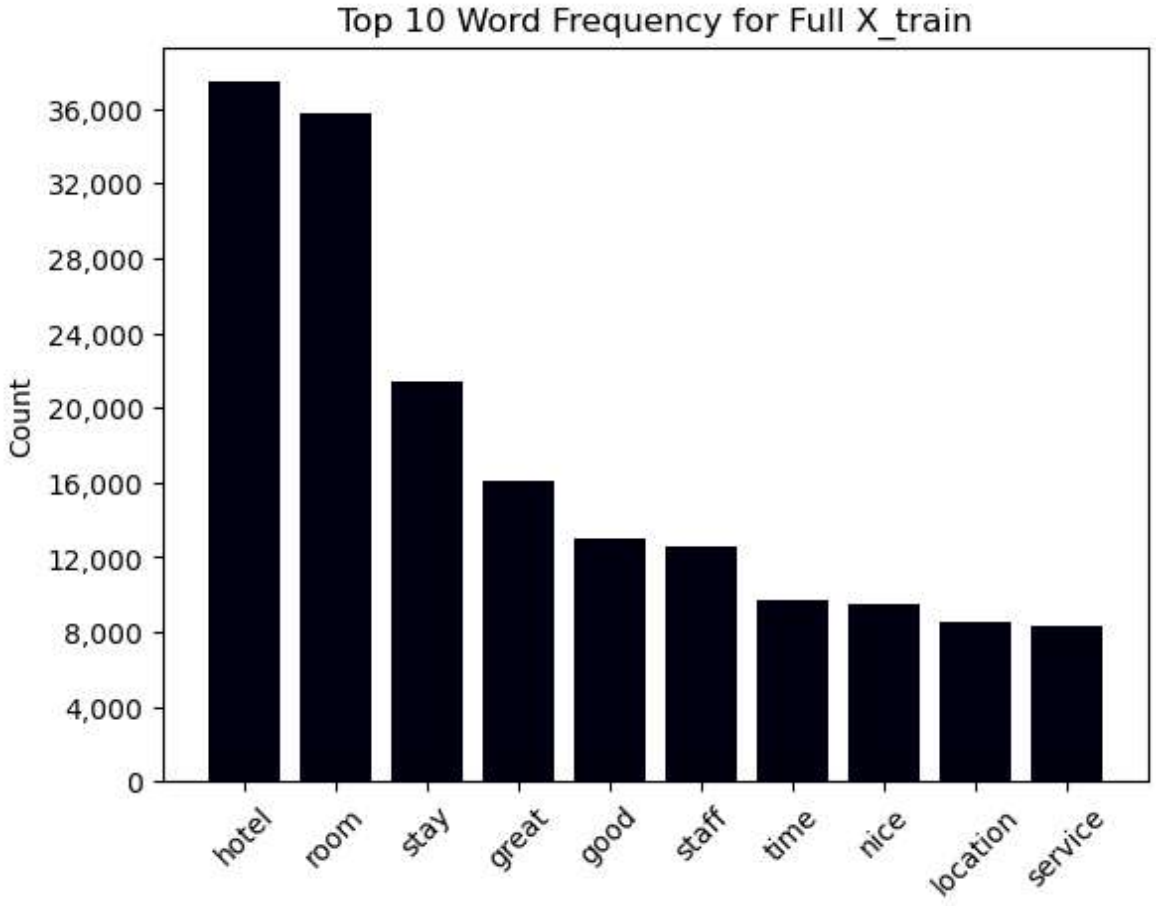
In [87]:
```
# Importing the relevant package for top number of words
from matplotlib.ticker import MaxNLocator

# Creating a function to visualize the top 10 words

def visualize_top_10(freq_dist, title, rotation):
#     extracting data for graph
    top_10 = list(zip(*freq_dist.most_common(10)))
    tokens = top_10[0]
    counts = top_10[1]

#     Setting up graph and plotting data
    fig, ax = plt.subplots()
    ax.bar(tokens, counts, color='#00000F')
#     ax.set_facecolor('#ffffff')

#     Customizing plot appearance
    ax.set_title(title)
    ax.set_ylabel('Count')

    # Formatting the y-axis labels to show thousands
    ax.yaxis.set_major_locator(MaxNLocator(integer=True))
    ax.yaxis.set_major_formatter(FuncFormatter(lambda x, pos: '{:,}'.format(int(x


    ax.tick_params(axis='x', rotation=rotation)

visualize_top_10(example_freq_dist, "Top 10 Word Frequency for Example Tokens", 9
```



- **FreqDist on the Full DataSet**

In order to calculate the count of words, they need to be stored into a list. To do so, I will `explode` the dataset.

In [88]:
```
# Creating a frequency distribution for X_train
train_freq_dist = FreqDist(X_train['Review'].explode())

# Plotting the top 10 tokens
visualize_top_10(train_freq_dist, 'Top 10 Word Frequency for Full X_train', 45)
```



In [89]:
```
# Inspecting the most common 20 words
train_freq_dist.most_common(20)
```

Out[89]:
```
[('hotel', 37427),
 ('room', 35770),
 ('stay', 21391),
 ('great', 16054),
 ('good', 13046),
 ('staff', 12527),
 ('time', 9687),
 ('nice', 9459),
 ('location', 8529),
 ('service', 8308),
 ('clean', 8215),
 ('beach', 7747),
 ('walk', 7731),
 ('breakfast', 7539),
 ('night', 7498),
 ('day', 7410),
 ('place', 7377),
 ('like', 7102),
 ('food', 7061),
 ('resort', 6750)]
```

I will also subdivide this by category (detractors/not_detractors) to see if it makes a difference:

In [90]:
```
# # Adding in labels for filtering
# X_train['label'] = [y_train[val] for val in X_train.index]
```

In [91]:
```
X_train[:3]
```

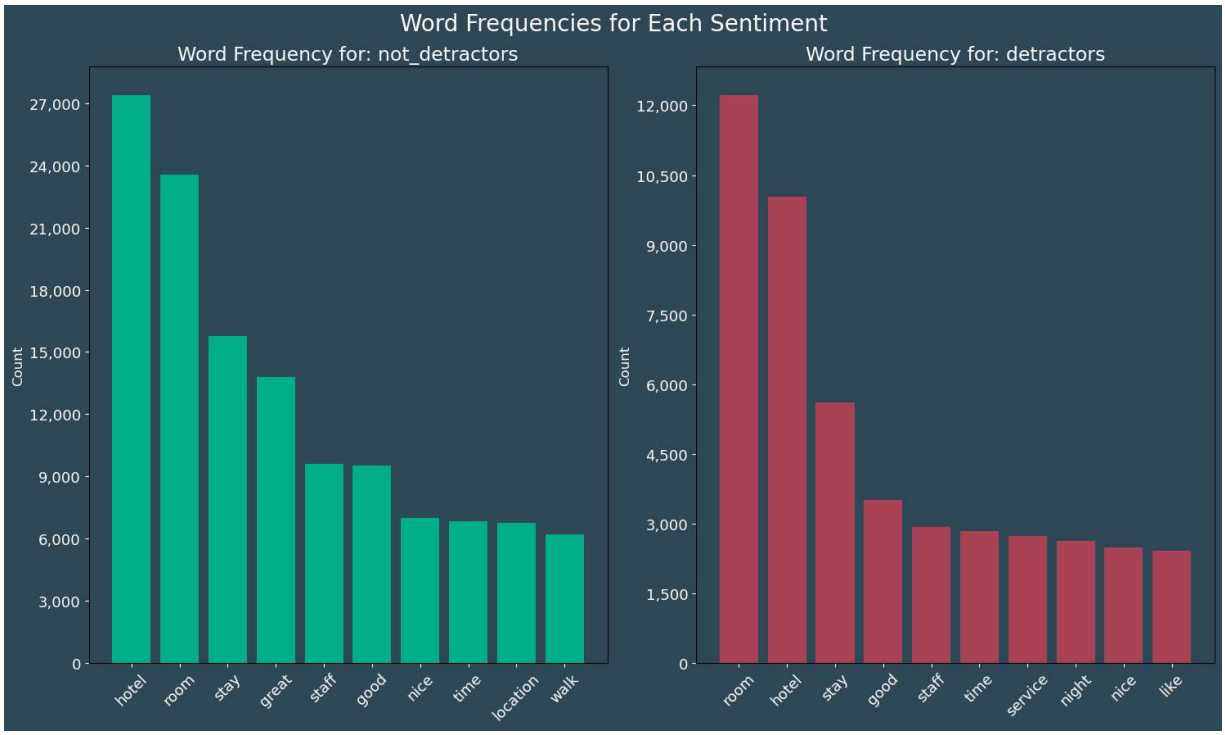
Out[91]:

	Review	Rating	label	original_review	Review_prep_nolist
1282	[gem, pleasantly, surprise, accomondations, helpful, attentive, staff, need, breakfast, good, cookies, room, lovely, clean, comfortable, room, front, bush, street, bite, noisy, away, enjoyment, nice, boutique, hotel, definitely, stay, time]	4	not_detractors	gem, pleasantly surprised accomondations helpful attentive staff need, breakfast good cookies, room lovely clean comfortable, room fronting bush street bit noisy did not away enjoyment nice boutique hotel, definitely stay time,	gem pleasantly surprise accomondations helpful attentive staff need breakfast good cookies room lovely clean comfortable room front bush street bite noisy away enjoyment nice boutique hotel definitely stay time
10661	[love, fita, wife, spend, nights, hotel, fita, march, 2008, hotel, staff, fantastic, helpful, pleasant, recommend, hotel, travel, amsterdam, hotel, locate, minute, walk, van, gogh, rijks, museums, 10, minute, walk, leidsplein, variety, store, shop, restaurants, locate, tram, stop, directly, hotel, easy, access, entire, city, walk, fita, way, center, city, 30, minutes, beautiful, accesible, friendly, definately, travel, fita, wonderful, amsterdam]	5	not_detractors	loved fita wife spent nights hotel fita march 2008. hotel staff fantastic helpful pleasant, recommend hotel traveling amsterdam.the hotel located minute walk van gogh rijks museums 10 minute walk leidsplein variety stores shops restaurants located.there tram stop directly hotel, easy access entire city, did walk fita way center city 30 minutes.beautiful accesible friendly definately travel fita wonderful amsterdam,	love fita wife spend nights hotel fita march 2008 hotel staff fantastic helpful pleasant recommend hotel travel amsterdam hotel locate minute walk van gogh rijks museums 10 minute walk leidsplein variety store shop restaurants locate tram stop directly hotel easy access entire city walk fita way center city 30 minutes beautiful accesible friendly definately travel fita wonderful amsterdam

	Review	Rating	label	original_review	Review_prep_nolist
17908	[great, modern, hotel, fantastic, price, stay, night, begin, april, alot, review, say, bad, area, really, literally, 100m, away, nice, redevelop, area, 10min, walk, beach, partner, arrive, late, barcelona, decide, walk, hotel, bus, station, order, bar, quite, long, walk, 50mins, roughly, really, great, thing, order, sight, tourists, dont, usually, arrive, check, quick, painless, room, really, really, nice, fantastic, modern, design, little, extras, want, order, barcelona, partner, opt, bus, turistic, stop, far, hotel, go, virtually, worth, short, visit, wish, spend, longer, hotel, wish, swim, pool, roof, open, overall, fantastic, hotel, price, pay, room, dont, distance, centre, transport, options]	5	not_detractors	great modern hotel fantastic price stayed just night beginning april, alot reviews say bad area really did n't, literally 100m away nice redeveloped area 10min walk beach.me partner arrived late barcelona decided walk hotel bus station order barings, quite long walk 50mins roughly really great thing order sights tourists dont usually, arrived check quick painless room really really nice, fantastic modern design little extras want, order barcelona partner opted bus turistic, stop not far hotel goes virtually, worth short visit.i wish spent longer hotel, wish swimming pool roof open, overall fantastic hotel price paid room dont distance centre transport options,	great modern hotel fantastic price stay night begin april alot review say bad area really literally 100m away nice redevelop area 10min walk beach partner arrive late barcelona decide walk hotel bus station order bar quite long walk 50mins roughly really great thing order sight tourists dont usually arrive check quick painless room really really nice fantastic modern design little extras want order barcelona partner opt bus turistic stop far hotel go virtually worth short visit wish spend longer hotel wish swim pool roof open overall fantastic hotel price pay room dont distance centre transport options

In [92]:
```
# Reminding custom colors
# Defining custom colors
# custom_colors = ['#00AF87', '#DEB887'] #green, gold
custom_colors = ['#00AF87', '#AA4255'] # #green, dark red

# custom_colors = ['#00AF87', '#00000F']
```

In [93]:
```
# Defining funcrion to plot 2 visualizations

# Creating two columns
def two_subplits():
    fig = plt.figure(figsize=(15, 9))
    fig.set_tight_layout(True)
    fig.set_facecolor('#2F4858')
    gs = fig.add_gridspec(1, 2)

    ax1 = fig.add_subplot(gs[0, 0]) #row 0, col 0
    ax2 = fig.add_subplot(gs[0, 1]) #row 0, col 1
    return fig, [ax1, ax2]

# Plotting the graph
def plot_distribution_by_sentiment(X_version, column, axes, rotation):
    for index, category in enumerate(X_version['label'].unique()):
        # Calculating frequency distribution for this subset
        all_words = X_version[X_version['label'] == category][column].explode()
        freq_dist = FreqDist(all_words)
        top_10 = list(zip(*freq_dist.most_common(10)))
        tokens = top_10[0]
        counts = top_10[1]


        # Setting up a plot
        ax = axes[index]
        ax.bar(tokens, counts, color=custom_colors[index])

        # Setting background color
        ax.set_facecolor('#2F4858')

        # Customizing plot appearance
        title = "Word Frequency for:"
        ax.set_title(f"{title} {category}", fontsize=17, color='white')
        ax.set_ylabel("Count", fontsize=12, color='white')

        # Setting tick color
        ax.tick_params(axis='x', colors='white', labelsize=13)
        ax.tick_params(axis='y', colors='white', labelsize=13)

        # Formatting the y-axis labels to show thousands
        ax.yaxis.set_major_locator(MaxNLocator(integer=True))
        ax.yaxis.set_major_formatter(FuncFormatter(lambda x, pos: '{:,}'.format(
        ax.tick_params(axis='x', rotation=rotation)


fig, axes = two_subplits()
plot_distribution_by_sentiment(X_train, 'Review', axes, 45)
fig.suptitle('Word Frequencies for Each Sentiment', fontsize=20, color='white')

 # Saving the plot as a PNG with a transparent background
plt.savefig('images/word_freq.png', transparent=True)

plt.show()
```



This is helpful to review them as a graph. I will define the subsets in a preliminary step to be able to review the frequency distribution without the need of a graph.

In [94]:
```
# Frequency distribution
# Defining subset prior. Here: detractors, notdetractors

notdetractors_reviews = X_train[X_train['label'] == 'not_detractors']
detractors_reviews = X_train[X_train['label'] == 'detractors']

# Defining the function to return the most frequently mentioned words
def freq_distr(subset, most_common):
    exploded_subset = subset['Review'].explode()
    subset_freq = FreqDist(exploded_subset)
    return subset_freq.most_common(most_common)
```

In [95]:
```
# Getting frequency distribution for top 20 strings of detractors
freq_distr(detractors_reviews, 20)
```

Out[95]:
```
[('room', 12221),
 ('hotel', 10028),
 ('stay', 5608),
 ('good', 3507),
 ('staff', 2920),
 ('time', 2840),
 ('service', 2722),
 ('night', 2630),
 ('nice', 2479),
 ('like', 2406),
 ('resort', 2362),
 ('day', 2335),
 ('get', 2330),
 ('go', 2310),
 ('food', 2286),
 ('beach', 2286),
 ('great', 2272),
 ('place', 2081),
 ('clean', 2068),
 ('say', 2046)]
```

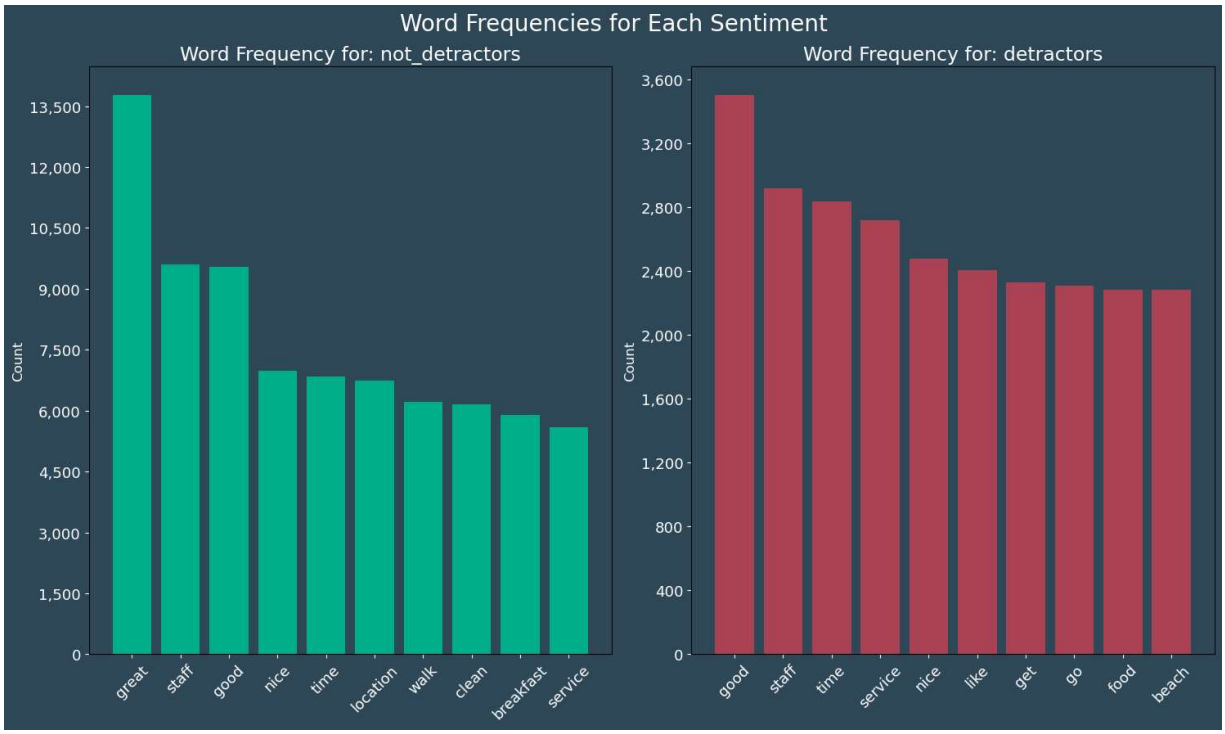
Now reviewing the first 20 words for reviews coming from detractors help us confirm that the first top 5 words are as represented in both categories. They are most common words to refer to a hotel stay. They should be removed for clearer analysis.

In [96]:
```
# # Applying this to our whole dataset
# X_train['Review'] = X_train['Review'].apply(lambda x: preprocess_review(x))
```

In [97]:
```
# Including these new words to a new list of stopwords
new_stopwords = ['hotel', 'room', 'night', 'day', 'stay', 'resort', 'place']
```

In [98]:
```
# Calling the previously defined function to remove stopwords
X_train['Review'] = X_train['Review'].apply(lambda x: remove_stopwords(x, new_sto
```

In [99]:
```
# Calling the function again to review the new split of top 10 words for each ser

fig, axes = two_subplits()
plot_distribution_by_sentiment(X_train, 'Review', axes, 45)
fig.suptitle('Word Frequencies for Each Sentiment', fontsize=20, color='white')

 # Saving the plot as a PNG with a transparent background
plt.savefig('images/word_freq.png')

plt.show()
```



I will filter on reviews that include specific words such as 'time' to review if they should be considered stopwords.

In [100]:
```
# Defining the word to search
word_search = 'time'

# Printing the results
X_train[X_train['Review_prep_nolist'].str.contains(word_search)][:3]

# Another way of searching directly in the column 'Review' in case several words
# word_researched = 'time'
# filtered_on_word = X_train[X_train['Review'].apply(lambda review_list: any(word
# filtered_on_word
```

	[hat, family, husband, children, disable, june, 23, 30, 2007, ok, time, hot, water, time, contact, service, desk, advise, respond, attitude, staff, work, look, happy, reflect, tremendously, service, food, terrible, sick, end, hosiptal, beg, doctor, discharge, early, children, claim, sick, imagination, rep, suggest, eat, bread, week, nice,	husband 3 children disabled, stayed resort june 23-30 2007. room ok times did not hot water, time contacted service desk advise respond attitude, staff working resort looked not happy reflected tremendously service, food terrible, sick ended hosiptal beg doctor discharge early children, hotel claimed no sick imagination, rep suggested eat bread week nice suggestion, hotel aware hospital stay ambulance pick hotel doctor office not equipped handle bacterial infection, no hotel contacted doing	hat resort family husband children disable stay resort june 23 30 2007 room ok time hot water time contact service desk advise respond attitude staff work resort look happy reflect tremendously service food terrible sick end hosiptal beg doctor discharge early children hotel claim sick imagination rep suggest eat bread week nice suggestion hotel aware hospital stay ambulance pick hotel doctor office equip handle bacterial infection hotel contact

Time is so commonly used to describe eihter category: such as 'next time' to describe a 'times where water did not work'. I will not include it in the list of stopwords to add. Instead I will try to understand through visualizations and bigrams.

4: 2- h) WordCloud

I will now visually represent the most frequently mentioned words, without representing them in a bar graph.
Word clouds visually represent the frequency of words in a given text, with more frequently occurring words displayed in larger font size. This allows a quick and intuitive overview of the words occuring most.

In [101]:
```
# Installing wordcloud
!pip install --trusted-host pypi.org --trusted-host pypi.python.org --trusted-hos
```

In [102]:
```
# Importing relevant packages
from matplotlib.colors import LinearSegmentedColormap

# Defining a colormap that interpolates between the two defined colors

custom_colors_detractors = ['#BFA5A7', '#AA4255']
# custom_colors_detractors = ['#E5BEC4', '#AA4255', '#F3E8EA']
custom_colors_notdetractors =  ['#EEFDF7', '#619C87', '#00AF87'] #blue-green, #tr

n_bins = 5

# Creating detractors colormap
cmap_detractors = LinearSegmentedColormap.from_list("cmap_detractors", custom_col

# Creating not_detractors colormap
cmap_notdetractors = LinearSegmentedColormap.from_list("cmap_notdetractors", cust
```

In [103]:
```
# Importing relevant packages
from wordcloud import WordCloud

# Concatenate all Review into a single string
all_reviews = ' '.join(X_train['Review'].apply(lambda x: ' '.join(map(str, x))))

# Defining the function to plot wordclouds
def wordcloud_graph(text):

    # Generate a word cloud
    wordcloud = WordCloud(width=800, height=400, background_color='#2F4858', colo

    # Display the generated word cloud using matplotlib
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.show()
```

In [104]:
```
# Calling the function on all reviews
wordcloud_graph(all_reviews)
```

Place, resort and time are really stand out among some of the most talked about topics in all reviews. However it is difficult to distinguish whether these reviews indicate anything mentioned by detractors or not, so I will divide them into 2 categories:

- detractors
- not_detractors

I will verify that our X_train includes the label to be able to divide the dataset into 2 categories.

In [105]:
```
# Ensuring the 'label' is part of X_train
X_train.columns
```

Out[105]:
```
Index(['Review', 'Rating', 'label', 'original_review', 'Review_prep_nolist'], d
type='object')
```

In [106]:
```
# Importing relevant packages

# Concatenating detractors and not_detractors reviews separately
detractors_reviews = ' '.join(X_train[X_train['label'] == 'detractors']['Review']
not_detractors_reviews = ' '.join(X_train[X_train['label'] == 'not_detractors']['

# Defining the function to plot word clouds
def wordcloud_graph(text, title, colormap, background_color):
    # Generating a word cloud
    wordcloud = WordCloud(width=400, height=200, background_color=background_colo


    # Displaying the generated word cloud using matplotlib
    fig = plt.figure(figsize=(10, 5))
    fig.set_facecolor('#2F4858')
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(title, color='white', fontsize=15, pad=25)
    plt.show()

# Plotting detractors reviews word cloud
wordcloud_graph(detractors_reviews, 'Detractors Reviews Word Cloud', cmap_detract
```

In [107]:
```
# Plotting not_detractors reviews word cloud
wordcloud_graph(not_detractors_reviews, 'Not detractors Reviews Word Cloud', cmap
```

While the wordcloud for not_detractors seem to standout more, the ones from detractors look like they are a bit harder to distinguish due to the number of times "time" is mentioned. I will try removing it from this extract, along with say and go to better understand the themes.

In [108]:
```
# Making a copy of the list of words
detractors_reviews_modified = detractors_reviews
```

In [109]:
```
# Defining the strings to remove
strings_to_remove = ['time', 'say', 'go', 'nice', 'great', 'od', 'fo']
```

In [110]:
```
# Removing them
for string in strings_to_remove:
    detractors_reviews_modified = detractors_reviews_modified.replace(string, '')
# detractors_reviews_modified
```

In [111]:
```
# Plotting detractors reviews word cloud
wordcloud_graph(detractors_reviews_modified, 'Detractors Reviews Word Cloud', cma
plt.savefig('images/wordcloud_detractors.png')
```

<Figure size 640x480 with 0 Axes>

In [112]:
```
# Some words were researched individually as well to ensure
X_train[X_train['Review_prep_nolist'].str.contains(' service ')][:1]
```

Out[112]:

	Review	Rating	label	original_review	Review_prep_nolist
6730	[hat, family, husband, children, disable, june, 23, 30, 2007, ok, time, hot, water, time, contact, service, desk, advise, respond, attitude, staff, work, look, happy, reflect, tremendously, service, food, terrible, sick, end, hosiptal, beg, doctor, discharge, early, children, claim, sick, imagination, rep, suggest, eat, bread, week, nice, suggestion, aware, hospital, ambulance, pick, doctor, office, equip, handle, bacterial, infection, contact, actual, fact, treat, garbage, kid, club, kid, club, children, active, love, swim, play, sport, interact, children, want, kid, club, nothing, kid, kid, sing, songs, circle, break, heart, children, fun, years, away, children, travel, family, cheap, especially, take, years, put, away, money, little, ...]	1	detractors	hated resort family 5. husband 3 children disabled, stayed resort june 23-30 2007. room ok times did not hot water, time contacted service desk advise respond attitude, staff working resort looked not happy reflected tremendously service, food terrible, sick ended hosiptal beg doctor discharge early children, hotel claimed no sick imagination, rep suggested eat bread week nice suggestion, hotel aware hospital stay ambulance pick hotel doctor office not equipped handle bacterial infection, no hotel contacted doing actual fact treated garbage, kids club, kids club, children active love swim play sports interact children did not want kids club, did nothing, kids night, kids, night sang songs circle, broke heart children not having fun, years away children, having travel family not cheap, especially takes years putting away money little little order enjoy week vacation, disappointed hotel not recommend unless young going bunch girls guys, like types travelers, lived dr travelled dr 15 times treated poorly did, ignored posting site warning figured people picky, realize hotels dr just not worth,	hat resort family husband children disable stay resort june 23 30 2007 room ok time hot water time contact service desk advise respond attitude staff work resort look happy reflect tremendously service food terrible sick end hosiptal beg doctor discharge early children hotel claim sick imagination rep suggest eat bread week nice suggestion hotel aware hospital stay ambulance pick hotel doctor office equip handle bacterial infection hotel contact actual fact treat garbage kid club kid club children active love swim play sport interact children want kid club nothing kid night kid night sing songs circle break heart children fun years away children travel family cheap especially take years put away money little little order enjoy week vacation disappoint hotel recommend unless young go bunch girls guy like type travelers live dr travel dr 15 time treat poorly ignore post site warn figure people picky realize hotels dr worth

Themes that come up the most for the detractors' reviews are:

- the overall service and staff friendliness:
  - service
  - staff
  - people
- the view from the rooms
- the decor
  - look
- cleanliness
  - clean
- problem solving
  - problem

- problem
- little

Other smaller negative items:

- Restaurant
- Bathroom

The mention of 'pay' is indicated. When contrasted with the 'value' mentioned in reviews coming from non-detractors, this indicates guests are okay with paying higher rates as long as they feel like it represents a good value for money.

As initial recommendations, in order to reduce the number of reviews from detractors:
1. Provide training sessions to enhance the friendliness of the staff
2. Re-define cleanliness standards
3. Revise the aesthetics of the lobby and rooms — appearance is crucial
4. Cultivate a problem-solving environment for staff

Coming up with these themes required extra research though through individual words, so I will try to use bigrams to make the themes more obvious.

4: 2- h) Bigrams

Bigrams

- **All Reviews**

In [113]:
```
# Importing relevant package
from nltk.collocations import *

# Storing nltk.collocations.BigramAssocMeasures into variable
bigram_measures = nltk.collocations.BigramAssocMeasures()
```

I will define a function to review bigrams and be able to filter in the future by the category I need.

In [114]:
```
# Defining a function to review bigrams
def bigram_review(text, top_n):
    # Creating a finder and passing it the words of reviews summarized as 1 list
    text_finder = BigramCollocationFinder.from_words(text.sum())
    text_scored = text_finder.score_ngrams(bigram_measures.raw_freq)
    return text_scored[: top_n]
```

I am starting with the top 20 bigrams overall.

In [115]:
```
# Calling the function to review the top 20 bigrams overall
bigram_review(X_train['Review'], 20)
```

Out[115]:
```
[(('great', 'location'), 0.001195786887061),
 (('staff', 'friendly'), 0.0011807859762914177),
 (('punta', 'cana'), 0.0008879110517424272),
 (('walk', 'distance'), 0.000804334548883325),
 (('friendly', 'helpful'), 0.0007857619926924135),
 (('highly', 'recommend'), 0.0007250440205298179),
 (('staff', 'helpful'), 0.0005964647853619684),
 (('minute', 'walk'), 0.0005750349128339935),
 (('read', 'review'), 0.000535032484115107),
 (('location', 'great'), 0.0005036020044074104),
 (('air', 'condition'), 0.0005000303589860813),
 (('make', 'sure'), 0.0004978873717332838),
 (('san', 'juan'), 0.0004943157263119546),
 (('great', 'time'), 0.0004857437773007647),
 (('good', 'value'), 0.0004285974505594 9826),
 (('good', 'location'), 0.0004228828178853716),
 (('breakfast', 'buffet'), 0.00038716636367208007),
 (('food', 'good'), 0.0003850233764192826),
 (('new', 'york'), 0.000379308743745156),
 (('value', 'money'), 0.0003757370983238268)]
```

This started providing better insights. In addition to what was identified for detractors, it can be identified here that what is most appreciated by guests are:

- Great location, close to key attractions
- Friendly and helpful staff
- Good value for money would be felt

In [116]:
```
# Creating subsets by detractors or not detractor reviews
X_train_detractors = X_train[X_train['label'] == 'detractors']
X_train_notdetractors = X_train[X_train['label'] != 'detractors']
```

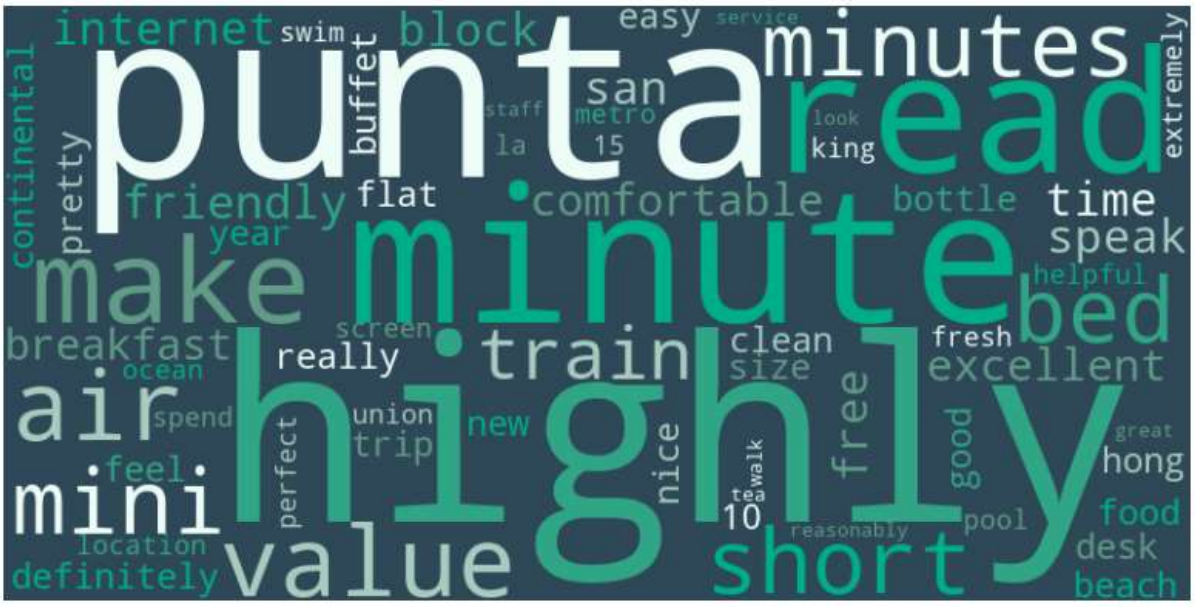
In [117]:
```
# Calling the bigram function on detractorss
detractors_bigrams_100 = bigram_review(X_train_detractors['Review'], 100)

# Calling the bigram function on not detractorss
notdetractors_bigrams_100 = bigram_review(X_train_notdetractors['Review'], 100)
```

In [118]:
```
# Creating a function to review bigrams in WordClouds
def wordcloud_bigrams(bigram_list, title, colormap, background_color):
    # Converting bigrams to a dictionary for WordCloud
    wordcloud_data = {bigram[0]: freq for bigram, freq in bigram_list}

    # Creating WordClouds
    wordcloud = WordCloud(width=800, height=400, background_color=background_colo

    # Plotting WordCloud
    plt.figure(figsize=(10,5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(title, fontsize=15, pad=20)
    plt.axis('off')
    plt.show()
```

In [119]:
```
# Calling it on the detractors_bigrams_100 list
wordcloud_bigrams(detractors_bigrams_100, 'Detractors Reviews: Bigrams', cmap_det

# Calling it on the notdetractors_bigrams_100 list
wordcloud_bigrams(notdetractors_bigrams_100, 'Not Detractors Reviews: Bigrams', c
```

Detractors Reviews: Bigrams

Not Detractors Reviews: Bigrams



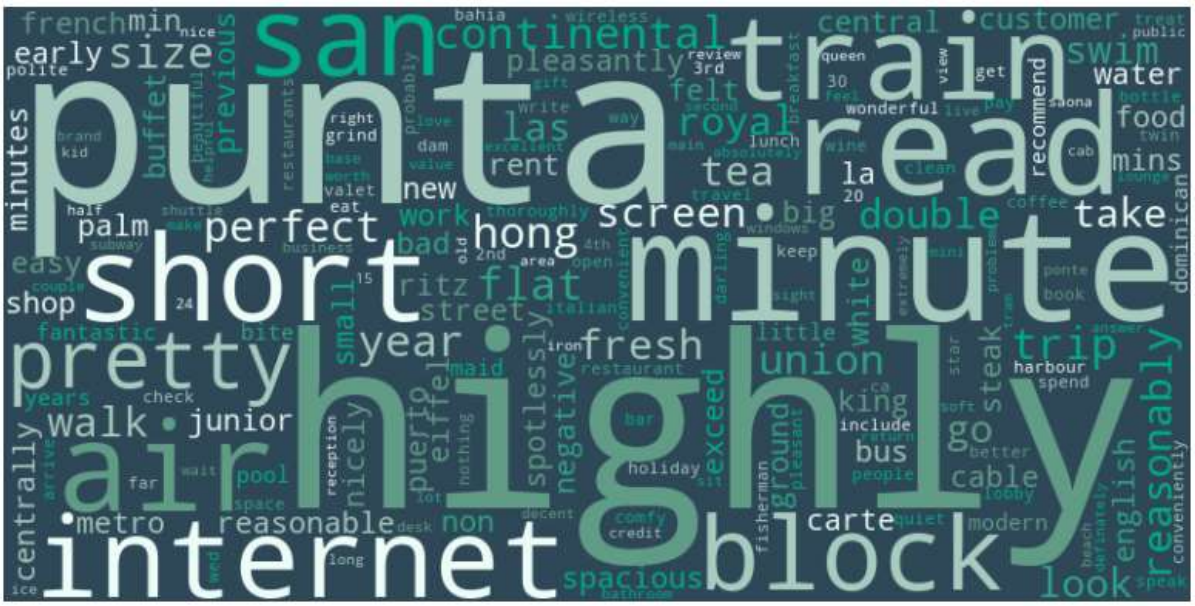
It is not easy to distinguish the difference in detactors vs not detactors reviews when individualizing again the bigrams. What about if I select more bigrams, such as the top 500 for each?

In [120]:
```
# Calling the bigram function on detractors
detractors_bigrams_500 = bigram_review(X_train_detractors['Review'], 500)

# Calling the bigram function on not detractors
notdetractors_bigrams_500 = bigram_review(X_train_notdetractors['Review'], 500)
```

In [121]:
```
# Calling it on the detractor_bigrams_500 list
wordcloud_bigrams(detractors_bigrams_500, 'Detractors Reviews: Bigrams', cmap_det

# Calling it on the notdetractor_bigrams_500 list
wordcloud_bigrams(notdetractors_bigrams_500, 'Not detractors Reviews: Bigrams', c
```

Detractors Reviews: Bigrams

Not detractors Reviews: Bigrams



There are still some similarities among dectractors and not detractors reviews referring to `Punta Cana` and walking. Let's research these specific mention to get a better understanding.

In [122]:
```
# Research walking and punta in detractors_bigrams_500

def research_bigram(word, bigram_list):
    word_researched = [bigram for bigram in bigram_list if word in bigram[0]]
    return word_researched

print(research_bigram('punta', detractors_bigrams_500))
# print(research_bigram('punta', dectractors_bigrams_500))
```

```
[(('punta', 'cana'), 0.00099021799216478)]
```

In [123]:
```
# notdetractors_bigrams_500

# print(research_bigram('walking', notpositive_bigrams_500))
print(research_bigram('punta', notdetractors_bigrams_500))
```

```
[(('punta', 'cana'), 0.0008446452439154541)]
```

The referrence to Punta Cana may only indicate that there is a high number of reviews.

Out of curiosity, I drew bigrams for Punta Cana specifically to get an understanding of why they are

In [124]:
```
# Calling the bigram function on punta
detractors_bigrams_punta = bigram_review(X_train_detractors[X_train_detractors['F

# Calling the bigram function on not detractors
notdetractors_bigrams_punta = bigram_review(X_train_notdetractors[X_train_notdetr
```

In [125]:
```
# Calling it on the detractors list
wordcloud_bigrams(detractors_bigrams_punta, 'Punta Cana Detractors Reviews: Bigra

# Calling it on the notpositive_bigrams_punta list
wordcloud_bigrams(notdetractors_bigrams_punta, 'Punta Cana Not Detractors Reviews
```

Punta Cana Detractors Reviews: Bigrams

Punta Cana Not Detractors Reviews: Bigrams

It looks like Punta Cana is one of the top visited resorts or one of those generating the highest number of reviews which explains why the name comes up so often.

By separating only for Punta Cana the negatives and not negative words, I can get better insights: horseback riding is one of the top activities, gusts appreciate the all inclusive service, and the hotel guests are referring to is the `Bahia Principe` hotel and resort.

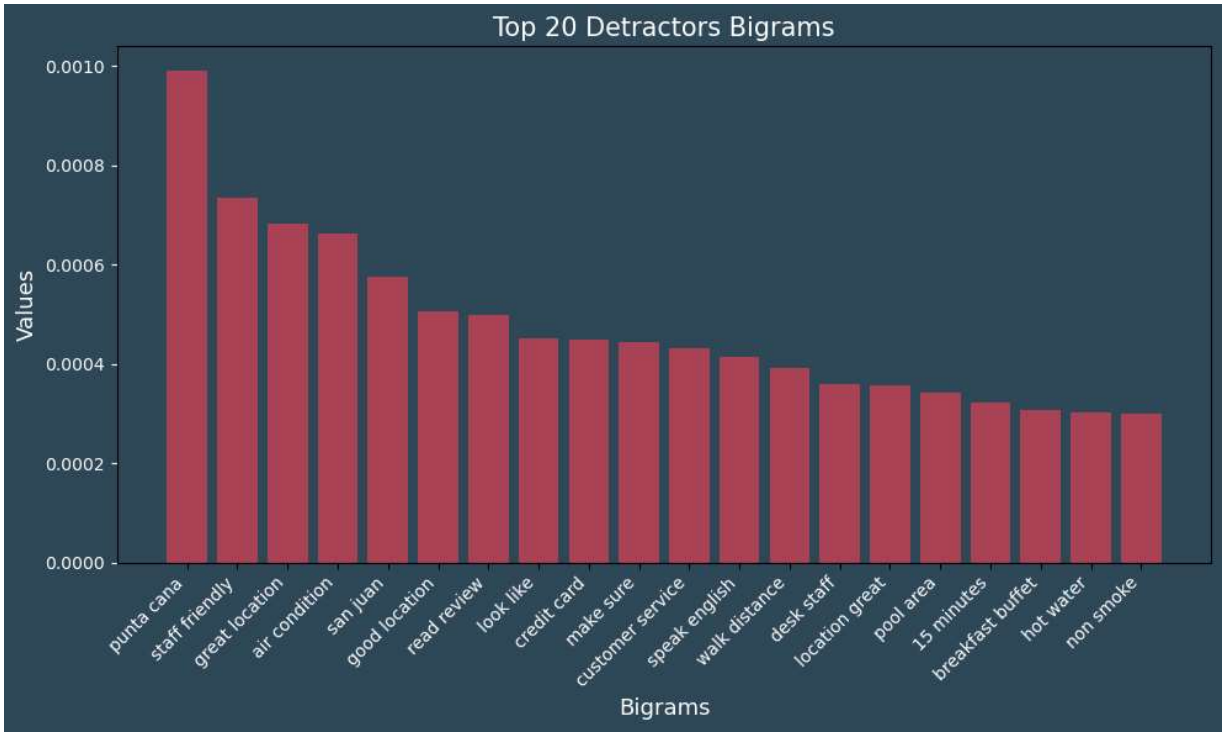
I will now try to visualize the top 20 positive and negative bigrams overall to close on this topic

I will now try to visualize the top 20 positive and negative bigrams overall to close on this topic

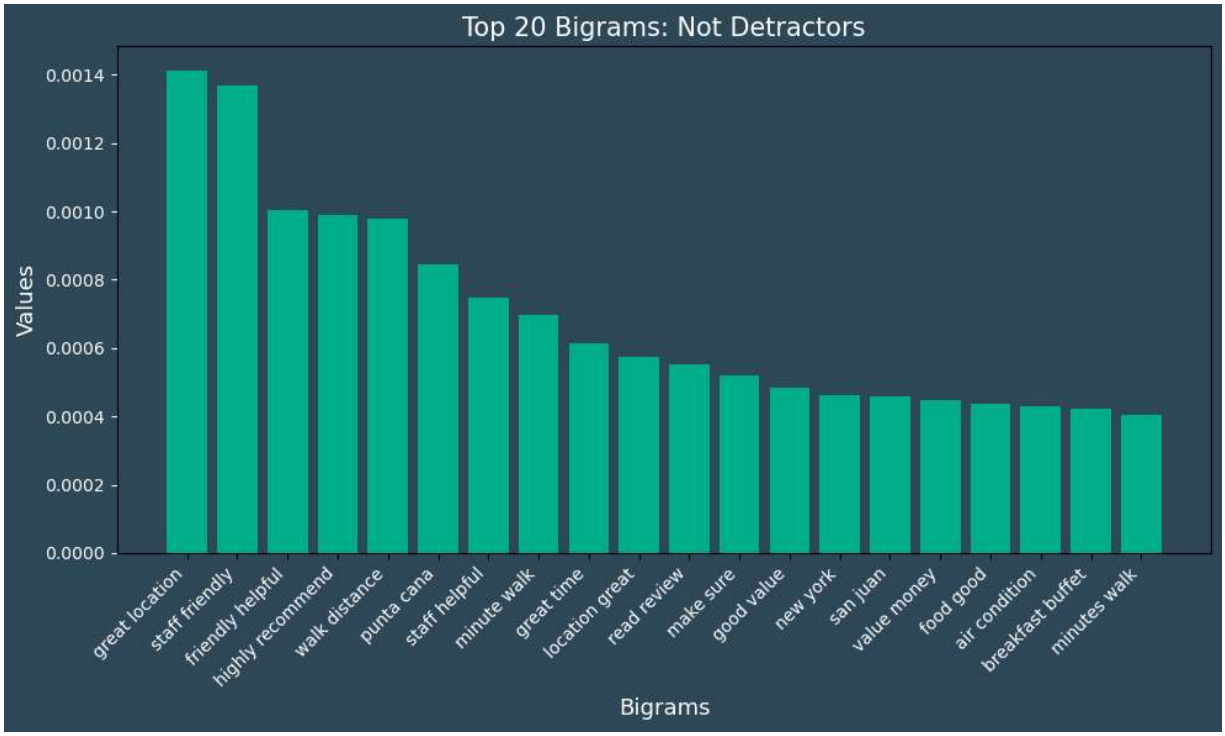
In [126]:
```
# Storing the top 20 bigrams for each sentiment
detractors_bigrams_20 = detractors_bigrams_500[:20]
notdetractors_bigrams_20 = notdetractors_bigrams_500[:20]
```

In [127]:
```
# Combining bigram words into a single string for each sentiment
detractor_bigram_20 = [' '.join(bigram[0]) for bigram in detractors_bigrams_20]
notdetactor_bigram_20 = [' '.join(bigram[0]) for bigram in notdetractors_bigrams_

# Reviewing what the newly created lists look like
# detractor_bigram_20
```

In [128]:
```
# Plotting the detractor bigrams as a bar chart

# Extracting bigrams and values
bigrams, values = zip(*detractors_bigrams_20)

# Sort bigrams and values in descending order based on values
sorted_indices = sorted(range(len(values)), key=lambda k: values[k], reverse=True
sorted_bigrams = [bigrams[i] for i in sorted_indices]
sorted_values = [values[i] for i in sorted_indices]

# Create a bar chart
fig, ax = plt.subplots(figsize=(10, 6))
fig.set_facecolor('#2F4858')
ax.set_facecolor('#2F4858')
ax.bar(range(len(sorted_values)), sorted_values, align='center', color='#AA4255')
ax.set_xticks(range(len(sorted_values)))
ax.set_xticklabels([f'{bigram[0]} {bigram[1]}' for bigram in sorted_bigrams], rot
ax.set_xlabel('Bigrams', color='white', fontsize=13)
ax.set_ylabel('Values', color='white', fontsize=13)
ax.tick_params(axis='y', colors='white')
ax.set_title('Top 20 Detractors Bigrams', color='white', fontsize=15)
plt.tight_layout()
plt.show()
```



In [129]:
```
# Plotting the not_detractor bigrams as a bar chart

# Extracting bigrams and values
bigrams, values = zip(*notdetractors_bigrams_20)

# Sort bigrams and values in descending order based on values
sorted_indices = sorted(range(len(values)), key=lambda k: values[k], reverse=True
sorted_bigrams = [bigrams[i] for i in sorted_indices]
sorted_values = [values[i] for i in sorted_indices]

# Create a bar chart
fig, ax = plt.subplots(figsize=(10, 6))
fig.set_facecolor('#2F4858')
ax.set_facecolor('#2F4858')
ax.bar(range(len(sorted_values)), sorted_values, align='center', color='#00AF87')
ax.set_xticks(range(len(sorted_values)))
ax.set_xticklabels([f'{bigram[0]} {bigram[1]}' for bigram in sorted_bigrams], rot
ax.set_xlabel('Bigrams', color='white', fontsize=13)
ax.set_ylabel('Values', color='white', fontsize=13)
ax.tick_params(axis='y', colors='white')
ax.set_title('Top 20 Bigrams: Not Detractors', color='white', fontsize=15)
plt.tight_layout()
plt.show()
```



In [130]:
```
# Searching through actual reviews to define the themes of some bigrams
# X_train_detractors[X_train_detractors['Review_prep_nolist'].str.contains('credi
```

Despite detractors giving overall negative reviews, they also highlight what they liked in the hotel as a contrast. This is why location is often referred to as something great.

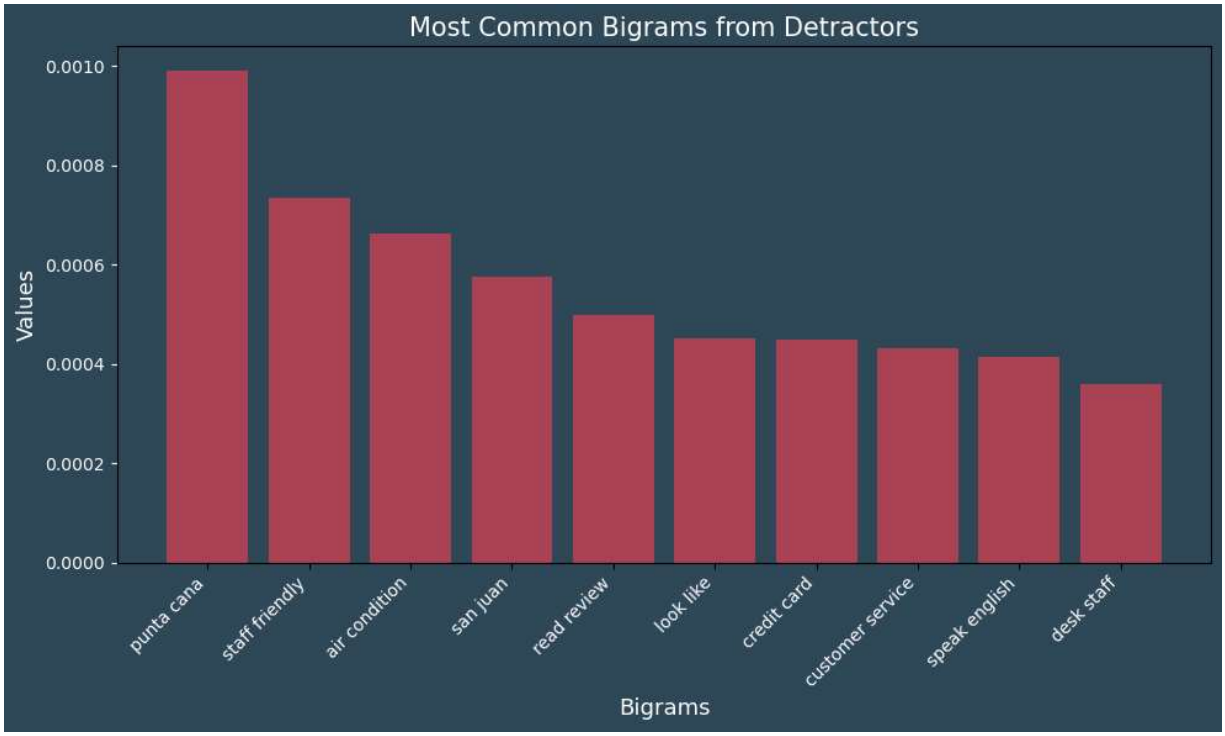
I will remove the references to location, as well as the indications such as 'make sure' so I can clearly display the top 10 themes to focus on.

In [131]:
```
bigrams_to_remove = [('great', 'location'), ('make', 'sure'), ('good', 'location'
```

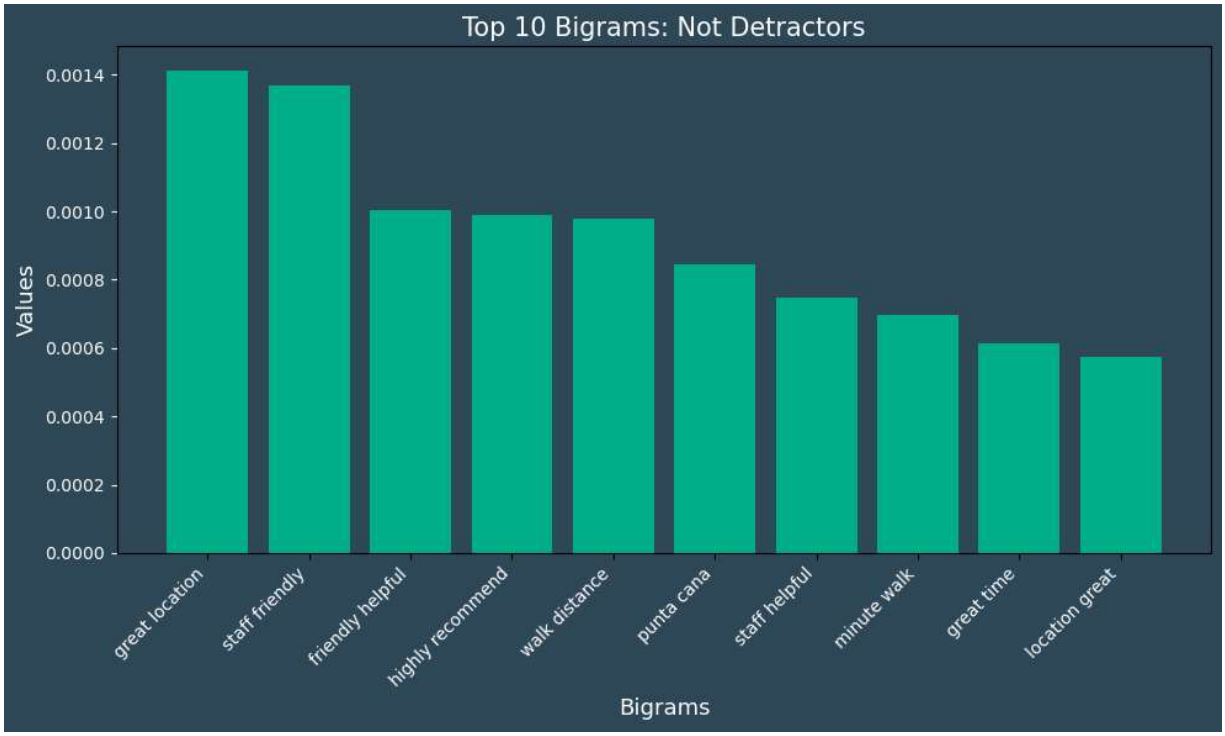
In [132]:
```
# Creating top_10_bigrams if they are not in list of bigrams to remove
detractors_bigrams_10 = [bigram for bigram in detractors_bigrams_500 if bigram[0]
detractors_bigrams_10
```

Out[132]:
```
[(('punta', 'cana'), 0.00099021799216478),
 (('staff', 'friendly'), 0.000735453168937919),
 (('air', 'condition'), 0.0006633499170812604),
 (('san', 'juan'), 0.0005768260148532699),
 (('read', 'review'), 0.0004975124378109452),
 (('look', 'like'), 0.00045184704496839474),
 (('credit', 'card'), 0.00044944360323983943),
 (('customer', 'service'), 0.0004302160694113971),
 (('speak', 'english'), 0.0004133919773115101),
 (('desk', 'staff'), 0.0003581128175547384)]
```

In [133]:
```
# Plotting the detractor bigrams as a bar chart

# Extracting bigrams and values
bigrams, values = zip(*detractors_bigrams_10)

# Sort bigrams and values in descending order based on values
sorted_indices = sorted(range(len(values)), key=lambda k: values[k], reverse=True
sorted_bigrams = [bigrams[i] for i in sorted_indices]
sorted_values = [values[i] for i in sorted_indices]

# Create a bar chart
fig, ax = plt.subplots(figsize=(10, 6))
fig.set_facecolor('#2F4858')
ax.set_facecolor('#2F4858')
ax.bar(range(len(sorted_values)), sorted_values, align='center', color='#AA4255')
ax.set_xticks(range(len(sorted_values)))
ax.set_xticklabels([f'{bigram[0]} {bigram[1]}' for bigram in sorted_bigrams], rot
ax.set_xlabel('Bigrams', color='white', fontsize=13)
ax.set_ylabel('Values', color='white', fontsize=13)
ax.tick_params(axis='y', colors='white')
ax.set_title('Most Common Bigrams from Detractors', color='white', fontsize=15)
plt.tight_layout()

plt.savefig('images/top_10_bigrams.png')
plt.show()
```



In [134]:
```
# Plotting the not_detractor bigrams as a bar chart

# Extracting bigrams and values
bigrams, values = zip(*notdetractors_bigrams_20[:10])

# Sort bigrams and values in descending order based on values
sorted_indices = sorted(range(len(values)), key=lambda k: values[k], reverse=True
sorted_bigrams = [bigrams[i] for i in sorted_indices]
sorted_values = [values[i] for i in sorted_indices]

# Create a bar chart
fig, ax = plt.subplots(figsize=(10, 6))
fig.set_facecolor('#2F4858')
ax.set_facecolor('#2F4858')
ax.bar(range(len(sorted_values)), sorted_values, align='center', color='#00AF87')
ax.set_xticks(range(len(sorted_values)))
ax.set_xticklabels([f'{bigram[0]} {bigram[1]}' for bigram in sorted_bigrams], rot
ax.set_xlabel('Bigrams', color='white', fontsize=13)
ax.set_ylabel('Values', color='white', fontsize=13)
ax.tick_params(axis='y', colors='white')
ax.set_title('Top 10 Bigrams: Not Detractors', color='white', fontsize=15)
plt.tight_layout()
plt.show()
```



Bigrams from detractors indicate that:

- Punta Cana and San Juan are the first destinations that require attention
- Staff friendliness needs to be addressed as identified in wordclouds
- Air conditioning needs to work and remain quiet
- Appearance matters: ensure the hotel is well-maintainted
- Hire staff who can speak English in destinations welcoming English-speaking guests
- Ensure easy payment methods are allowed and work correctly
  Some hotels (i.e. Florence) do not take credit cards. This needs to be addressed nowadays

- Respond to reviews: a lot of detractors mention they read them

4: 2- i) Mutual Information Scores

I will calculate mutual information scores and I will create a frequency filter, so that I only examine the strength of association between pairs of words (bigrams) for detractors or not detractors sentiments.

In [135]:
```
# Defining a function to calculate the mutual information scores
def mutual_info_score(text, n_filter):
    text_pmi_finder = BigramCollocationFinder.from_words(text.sum())
    text_pmi_finder.apply_freq_filter(n_filter)
    text_pmi_scored = text_pmi_finder.score_ngrams(bigram_measures.pmi)
    return text_pmi_scored
```

In [136]:
```
# Calling the function for the top 10 mutual information scores for all reviews
mutual_info_score(X_train['Review'], 5)[:10]
```

Out[136]:
```
[(('ajili', 'mojili'), 18.094979706748987),
 (('krispy', 'kreme'), 18.094979706748987),
 (('desa', 'seni'), 17.83194530091519),
 (('ocho', 'rios'), 17.83194530091519),
 (('ropa', 'vieja'), 17.609552879578743),
 (('altos', 'chavon'), 17.41690780163635),
 (('dai', 'ichi'), 17.41690780163635),
 (('kwai', 'fong'), 17.41690780163635),
 (('ograde', 'oreview'), 17.41690780163635),
 (('piet', 'hein'), 17.41690780163635)]
```

In [137]:
```
# Reminding the previously created subsets by detractors or not detractors review
X_train_detractors = X_train[X_train['label'] == 'detractors']
X_train_notdetractors = X_train[X_train['label'] != 'detractors']
```

In [138]:
```
# Calling the function for the top 10 mutual information scores for positive revi
mutual_info_score(X_train_detractors['Review'], 5)[:10]
```

Out[138]:
```
[(('sha', 'tsui'), 16.344538648678256),
 (('tsim', 'sha'), 16.344538648678256),
 (('degli', 'orafi'), 16.08150424284446),
 (('lo', 'behold'), 16.08150424284446),
 (('rick', 'steves'), 15.859111821508012),
 (('ograde', 'oreview'), 15.666466743565618),
 (('chateau', 'lemoyne'), 15.496541742123306),
 (('postage', 'stamp'), 15.496541742123306),
 (('sagrada', 'familia'), 15.47382166562322),
 (('zona', 'rosa'), 15.444074322229168)]
```

In [139]:
```
# Calling the function for the top 10 mutual information scores for notdetractors
mutual_info_score(X_train_notdetractors['Review'], 5)[:10]
```

Out[139]:
```
[(('kwai', 'fong'), 17.58614342342153),
 (('desa', 'seni'), 17.323109017587733),
 (('smith', 'wollensky'), 17.323109017587733),
 (('piet', 'hein'), 17.10071659625129),
 (('degli', 'orafi'), 16.908071518308894),
 (('pont', 'neuf'), 16.908071518308894),
 (('dai', 'ichi'), 16.90807151830889),
 (('zoologischer', 'garten'), 16.90807151830889),
 (('crabtree', 'evelyn'), 16.73814651686658),
 (('frankfurter', 'allee'), 16.685679096972443)]
```

The mutual information scores in themselves are a bit hearder to drive direct conclusions from them. Additional analyses would be necessary for them to be conclusive.

The groups of words mostly indicate some specific neighborhoods in destinations that were not appreciated by guests, or at the opposite, some destinations and attractions in destinations that were often talked by guests who reviewed the hotels positively.

5. Modeling

I now have an initial idea for recommendations on focus areas to improve guest satistfaction.
My objective is now to:

- Develop a tool to identify unsatisfied customer reviews in real time

Incorrectly identifying a guest as satisfied when they weren't would lead in a detractor posting a review instead of catching them while they are in-house, and having a chance of improving their satisfaction.

As a consequence, `recall` is the main evaluation metric for this model.

As the dataset is a text, it requires a transformation before it can be used for modeling. Like other types of dataset would one-hot encoded, here, the reviews were vectorized, using the common method in natural language processing: `TfidfVectorizer` .

It converts a collection of text documents to a matrix of tf-idf features.

- Term-Frequency
  Measures how often a term (word) appears in a document
- Inverse Document Frequency (IDF)
  Measures the importance of a term in the entire collection of documents.

7 main classification models were explored:

1. Multinomial Naive Bayes
2. K-Nearest Neighbor
3. Decision Tree

3. Decision Tree
4. Random Forest
5. Gradient Boosting
6. AdaBoost
7. XGBoost

Undersampling, stopwords, lemmatize, and hyperparameter tuning were parameters used to tune models. In addition, X_train was split with `stratify` in order to keep the distribution ratios for ech sentiment.

In [140]:
```
# Reminding the natural balance
y_train.value_counts(normalize=True)
```

Out[140]:
```
0    0.736596
1    0.263404
Name: Sentiment, dtype: float64
```

If I were to guess the majority class every time I would get 74% accuracy.

In [141]:
```
# Verifing labels is not part of X_train
X_train.columns
```

Out[141]:
```
Index(['Review', 'Rating', 'label', 'original_review', 'Review_prep_nolist'], d
type='object')
```

In [142]:
```
# Dropping it
# X_train = X_train.drop('label', axis=1)
```

By using only the dataset's natural class balance, and if I guessed the contribution of the majority class every time I would get 74% accuracy. However, if I were to guess that a review was not from a detractor, I would expect only about 26% accuracy.

5. a) Baseline Model with TfidfVectorizer and MultinomialNB

The first baseline model will vectorize the reviews and make predictions using Multinomial Naive Bayes. The first step is to import the vectorizer, instantiate a vectorizer object and fit it on `X_train['original_review']` .

1st iteration: Vanilla Tfidf Vectorizer with Pipeline

1) Fitting and training on train data

In [143]:
```
# Importing the relevant packages
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB

# Define the pipeline steps
tfidf_vectorizer = TfidfVectorizer()
naive_bayes_classifier = MultinomialNB()

# Create the pipeline
base_pipeline = Pipeline([
    ('tfidf', tfidf_vectorizer),
    ('classifier', naive_bayes_classifier)
])

# Fitting the pipeline on X_train['original_review'] and y_train
base_pipeline.fit(X_train['original_review'], y_train)

# Calculating predictions using this model
base_y_pred = base_pipeline.predict(X_test['original_review'])

# Optionally, you can access the individual components of the pipeline:
X_train_vectorized = base_pipeline.named_steps['tfidf'].transform(X_train['origir
baseline_model = base_pipeline.named_steps['classifier']
```

2) Evaluation Metrics

In [144]:
```
# Importing the relevant packages
from sklearn.metrics import recall_score, accuracy_score, f1_score
from sklearn.model_selection import cross_val_score

def evaluation_metrics(y_test, y_pred, model, X, y):
    with warnings.catch_warnings():
            warnings.simplefilter("ignore", category=RuntimeWarning)

    # Calculating and printing accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Accuracy: {accuracy:.4f}')

    # Calculating and printing F1-score
    f1 = f1_score(y_test, y_pred, average='weighted')
    print(f'F1-Score: {f1:.4f}')

    # Calculating and printing recall
    recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
    print(f'Recall: {recall:.4f}')

    # Performing cross-validation and printing the mean accuracy
    cv_scores = cross_val_score(model, X, y, cv=5)
    mean_cv_accuracy = cv_scores.mean()
    print(f'Mean Cross-Validated Accuracy: {mean_cv_accuracy:.4f}')

    return accuracy, f1, recall, mean_cv_accuracy
```

In [145]:
```
# Naming the model and calling the function to evaluate it
baseline_model_name = 'Baseline'

# Calling the function and recording into the defined values
accuracy_base, f1_base,recall_base, cv_base = evaluation_metrics(
    y_test,
    base_y_pred,
    base_pipeline,
    X_train['original_review'],
    y_train)
```

```
Accuracy: 0.7546
F1-Score: 0.6656
Recall: 0.7546
Mean Cross-Validated Accuracy: 0.7507
```

The model shows a reasonable overall accuracy in predicting sentiments from hotel reviews on TripAdvisor. It achieves a good balance between recall and F1-Score, with a mean cross-validated accuracy that suggests stability across different subsets of the data.

Let's verify this using the classification report and confusion matrix.

3) Classification Report

In [146]:
```
# Defining a function to print a classification report

import warnings
from sklearn.metrics import classification_report

def class_calculation(y_test, y_pred):
    # y_preds will be calculated for each model beforehand

    # Generating and printing classification report
    with warnings.catch_warnings():
        warnings.simplefilter("ignore", category=RuntimeWarning)
    class_report = classification_report(y_test, y_pred, zero_division=0, digits=
#     class_report = classification_report(y_test, y_pred, zero_division=0, digit

    # Due to class imbalance, the initial recall for positive returns 0
    # Consequently, warnings need to be handled
    print('Classification Report:\n', class_report)


     # Parsing the classification report to extract recall for 'Detractors'
    lines = class_report.split('\n')
    recall_line = lines[3]
    # Extracting numerical values
    recall_values = [float(value) for value in recall_line.split()[1:]]
    # Storing recall for Detractors
    recall_detractors = recall_values[1]


    return class_report, recall_detractors
```

Because what matters most is recall but for the detractors' side, we will also store this metric.

In [147]:
```
# Calling the function to record the classification report
base_class_report, recall_detractors_base = class_calculation(y_test, base_y_pred
```

```
Classification Report:
                 precision    recall  f1-score   support

Not Detractors    0.75020   0.99973   0.85718      3773
    Detractors    0.98947   0.06963   0.13010      1350

      accuracy                        0.75464      5123
     macro avg    0.86984   0.53468   0.49364      5123
  weighted avg    0.81325   0.75464   0.66558      5123
```

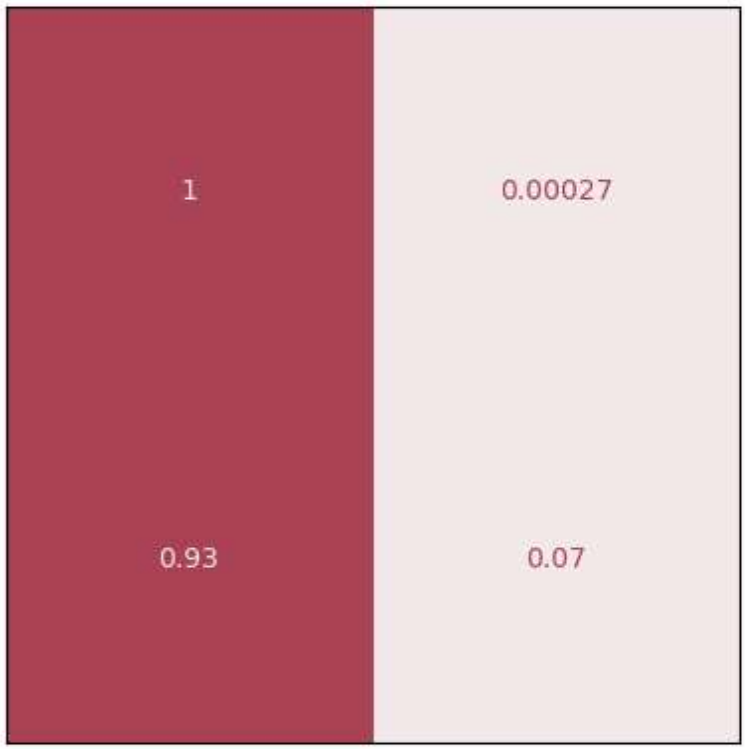
4) Confusion Matrix

In [148]:
```
# Creating a color map
from matplotlib.colors import LinearSegmentedColormap

# Defining a colormap that interpolates between the defined colors
custom_colors_cnf = ['#00AF87', '#00A4A0', '#0097B9', '#0097B9']

custom_colors_detractors = ['#F3E8EA', '#E5BEC4', '#AA4255']
# custom_colors_detractors_reversed = ['#AA4255', '#E5BEC4', '#F3E8EA']

n_bins = 5

# Creating the custom colormap
custom_cmap_cnf = LinearSegmentedColormap.from_list("custom_cmap_cnf", custom_col
```

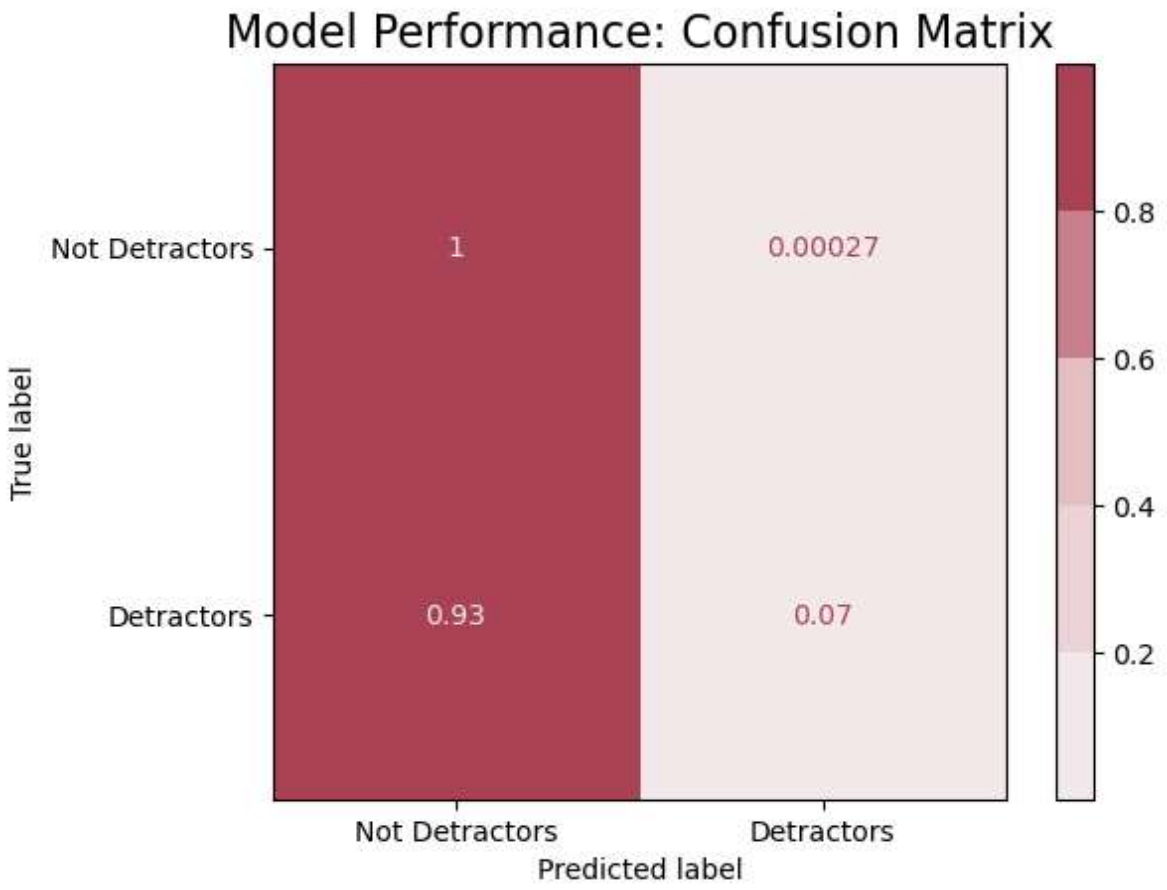
In [149]:
```
# Displaying visually the confusion matrix

# Importing the relevant package
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

def confusion_matrix_display(model, y_test, y_pred):
    # Defining the confusion matrix
    cnf_matrix = confusion_matrix(y_test, y_pred)
#     print(cnf_matrix)

    # Normalizing the confusion matrix
    cnf_matrix_normalized = cnf_matrix.astype('float') / cnf_matrix.sum(axis=1)[

    # Taking in the model's classes as labels
#     disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix_normalized, displ

    # Taking in pre-defined labels
    disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix_normalized, display

    disp.plot(cmap=custom_cmap_cnf)

    plt.title("Model Performance: Confusion Matrix", fontsize=16)

    # Saving the plot as a PNG with a transparent background
#     plt.savefig('images/confusion_matrix.png', transparent=True)

    return cnf_matrix
    plt.show()
```

In [150]:
```
# Creating a copy for the best model, with white values for labels and titles

def best_confusion_matrix_only(model, y_test, y_pred):
    # Defining the confusion matrix
    cnf_matrix = confusion_matrix(y_test, y_pred)

    # Normalizing the confusion matrix
    cnf_matrix_normalized = cnf_matrix.astype('float') / cnf_matrix.sum(axis=1)[
    # Taking in pre-defined labels
    disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix_normalized, displa

    # Customizing the plot
    disp.plot(cmap=custom_cmap_cnf, colorbar=False)  # Set colorbar to False to r

    # Setting the title and labels color to white

    plt.title("Model Performance: Confusion Matrix", fontsize=16, color='white')
    # Setting the color of x-axis and y-axis labels to white

    disp.ax_.xaxis.label.set_color("white")
    disp.ax_.yaxis.label.set_color("white")
    disp.ax_.tick_params(axis='x', colors='white')
    disp.ax_.tick_params(axis='y', colors='white')

    # Saving the plot as a PNG with a transparent background
    plt.savefig('images/best_confusion_matrix.png', transparent=True)

    plt.show()
```

In [151]:
```
# Testing best_confusion_matrix
best_confusion_matrix_only(base_pipeline, y_test, base_y_pred)
```



In [152]:
```
# Calling the function to display the confusion matrix
base_cfn_matrix = confusion_matrix_display(base_pipeline, y_test, base_y_pred)
```



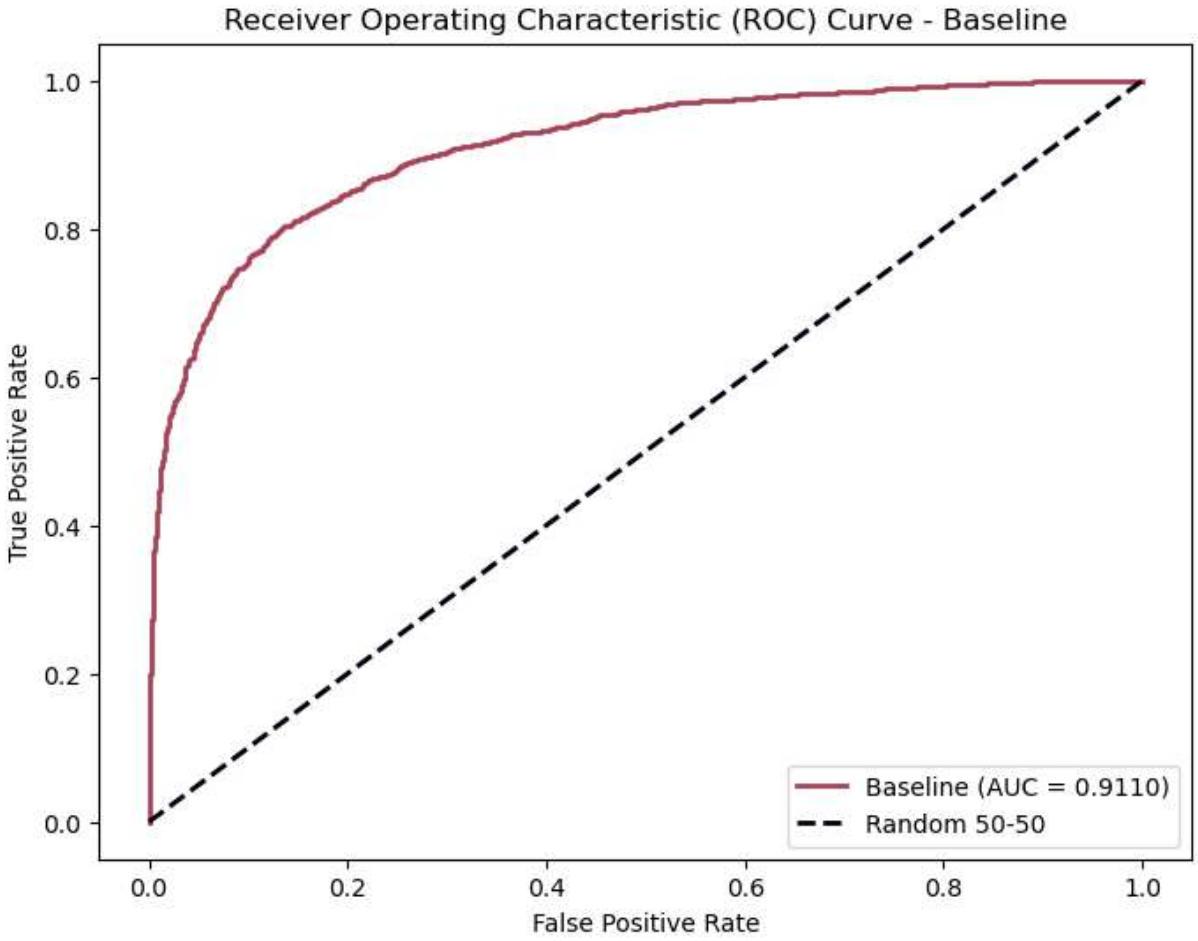
While the overall metrics look decent, the results for each sentiment are not balanced at all and the recall for detractors is the lowest of recorded results: 0.06963. This needs to be addressed through the dataset's balance.

5) ROC Curve and AUC

The Receiver Operating Characteristic (ROC) Curve displays on a graph the False Positive Rate against the True Positive Rate. It is another evaluation metric for classification algorithms combining both described ratios. It is compared to a *worthless accuracy*: a classifier with 50-50 accuracy. It is considered worthless, as it would be comparable to just random guessing.

In [153]:
```
# Importing the relevant packages
from sklearn.metrics import roc_curve, auc


def plot_roc_curve(y_true, y_pred_proba, model_name='Model'):
    # Computing ROC curve and AUC
    fpr, tpr, thresholds = roc_curve(y_true, y_pred_proba)
    roc_auc = auc(fpr, tpr)

    # Plottting ROC curve
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='#AA4255', lw=2, label='{} (AUC = {:.4f})'.format(mo

    # Plotting the 50-50 accuracy line
    plt.plot([0, 1], [0, 1], color='#00000F', lw=2, linestyle='--', label='Randor
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve - {}'.format(model_n
    plt.legend(loc='lower right')
    plt.show()
    return roc_auc
```

In [154]:
```
# Plotting the ROC curve for the baseline model
base_y_pred_proba = base_pipeline.predict_proba(X_test['original_review'])[:, 1]
auc_base = plot_roc_curve(y_test, base_y_pred_proba, model_name=baseline_model_na
```



ROC-AUC a common classification metric for binary classification problems to evaluate the ability of a model to distinguish between the positive and negative classes. It visually represents the trade-off between the true positive rate and the false positive rate at various thresholds. The area under the ROC curve (AUC) provides a single value that summarizes the model's performance. The higher AUC is, the better the model can distinguish between the classes.

In [155]:
```
# Defining a function that provides an interpretation of area under the curve
def auc_interpretation(auc):
    if auc == 1:
        print(f'AUC = {auc:.4f}. The model is a perfect classifier.')
    elif auc >= 0.95:
        print(f'AUC = {auc:.4f}. The model has a predictive power close to perfec
    elif auc > 0.9:
        print(f'AUC = {auc:.4f}. The model has a great predictive power in distir
    elif auc > 0.8:
        print(f'AUC = {auc:.4f}. The model has a good predictive power.')
    elif auc > 0.7:
        print(f'AUC = {auc:.4f}. The model has some predictive power.')
    elif auc == 0.5:
        print(f'AUC = {auc:.4f}. The model does not perform better than random ch
```

In [156]:
```
# Printing the interpretation
auc_interpretation(auc_base)
```

```
AUC = 0.9110. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [157]:
```
# Creating an empty list of metrics to easily add new evaluations as they are dor
modelnames = []
accuracies = []
f1s = []
recalls = []
recalls_detractors = []
cv_accs = []
roc_aucs = []
```

In [158]:
```
# Appending each metric to the lists
modelnames.append(baseline_model_name)
accuracies.append(accuracy_base)
f1s.append(f1_base)
recalls.append(recall_base)
recalls_detractors.append(recall_detractors_base)
cv_accs.append(cv_base)
roc_aucs.append(auc_base)
```

2nd iteration: Addressing class imbalance: undersampling negative reviews

Sampling Strategy: 0.8

While the initial categorization of sentiment following Net Promoter Score classification helped address the dataset imbalance, the number of reviews from non-detractors remained higher than the one from detractors.

As a consequence, the dataset needs to be resampled. More precisely, negative reviews need to be undersampled. I will first try with a sampling strategy of 0.8.

1) Fitting and training on train data

In [159]:
```
# Importing relevant packages
from imblearn.under_sampling import RandomUnderSampler
import imblearn.pipeline
```

I will now include the Random Undersampler to our pipeline so it under samples our majority class.

In [160]:
```
# Defining the pipeline steps
tfidf_vectorizer = TfidfVectorizer()
naive_bayes_classifier = MultinomialNB()

# Including the UnderSampler to the pipeline
rs_pipeline = imblearn.pipeline.Pipeline([
    ('tfidf', tfidf_vectorizer),
    ('us', RandomUnderSampler(sampling_strategy=0.8, random_state=42)),
    ('classifier', naive_bayes_classifier)
])

# Fitting the pipeline on X_train['original_review'] and y_train
rs_pipeline.fit(X_train['original_review'], y_train)

rs_y_pred = rs_pipeline.predict(X_test['original_review'])
```

In [161]:
```
# Verifying the value counts of each category, before undersampling
original_value_counts = y_train.value_counts()
original_value_counts_normalized = y_train.value_counts(normalize=True)
print("Original class distribution:")
print(original_value_counts)
print(original_value_counts_normalized)

# Getting the indices of the resampled data
resampled_indices = rs_pipeline.named_steps['us'].sample_indices_

# Verifying the new value counts of each category, after undersampling
resampled_value_counts = y_train.iloc[resampled_indices].value_counts()
resampled_value_counts_normalized = y_train.iloc[resampled_indices].value_counts(
print("\nClass distribution after undersampling:")
print(resampled_value_counts)
print(resampled_value_counts_normalized)
```

```
Original class distribution:
0    11320
1     4048
Name: Sentiment, dtype: int64
0    0.736596
1    0.263404
Name: Sentiment, dtype: float64

Class distribution after undersampling:
0    5060
1    4048
Name: Sentiment, dtype: int64
0    0.555556
1    0.444444
Name: Sentiment, dtype: float64
```

The non-detractors reviews were randomly undersampled from our model, which now has a higher proportion of detractors reviews and lower not_detractor ones.

I will now run our evaluation metrics to review if the model is providing better predictions.

2) Evaluation Metrics

In [162]:
```
# Naming the model
resampled_model_name = 'Resampled'

# Calling the function and recording into the defined values
accuracy_rs, f1_rs, recall_rs, cv_rs = evaluation_metrics(
    y_test,
    rs_y_pred,
    rs_pipeline,
    X_train['original_review'],
    y_train)
```

```
Accuracy: 0.8842
F1-Score: 0.8784
Recall: 0.8842
Mean Cross-Validated Accuracy: 0.8836
```

In [163]:
```
# Creating a function to print the metrics' comparison
def metrics_comparison(metric_name, old_metric, new_metric):
    if new_metric > old_metric:
        print(f'{metric_name} improved from {old_metric:.4f} to {new_metric:.4f}.
    else:
        print(f'{metric_name} decreased from {old_metric:.4f} to {new_metric:.4f}
```

In [164]:
```
# Comparing the evaluation metrics between baseline and Resampled model
metrics_comparison('Accuracy', accuracy_base, accuracy_rs)
metrics_comparison('F1', f1_base, f1_rs)
metrics_comparison('Recall', recall_base, recall_rs)
metrics_comparison('Cross-validated Accuracy', cv_base, cv_rs)
```

```
Accuracy improved from 0.7546 to 0.8842.
F1 improved from 0.6656 to 0.8784.
Recall improved from 0.7546 to 0.8842.
Cross-validated Accuracy improved from 0.7507 to 0.8836.
```

In [165]:
```
# Printing the overall summary
print('The resampling started to improve the overall scores')
```

```
The resampling started to improve the overall scores
```

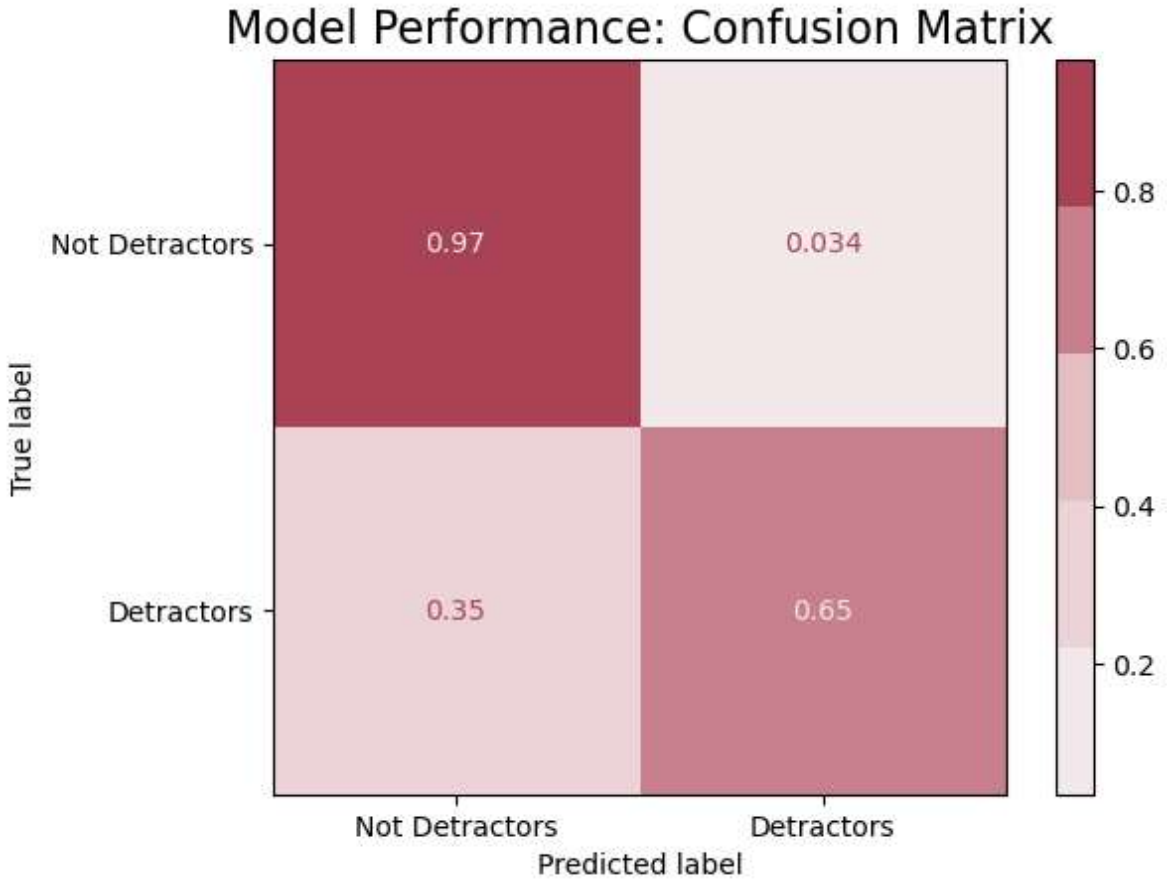
3) Classification Report

In [166]:
```
# Calling confusion report for resampled model
resamp_class_report, recall_detractors_rs = class_calculation(y_test, rs_y_pred)
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.88667   0.96634   0.92479      3773
    Detractors    0.87438   0.65481   0.74884      1350

      accuracy                        0.88425      5123
     macro avg    0.88053   0.81058   0.83681      5123
  weighted avg    0.88343   0.88425   0.87843      5123
```

4) Confusion Matrix

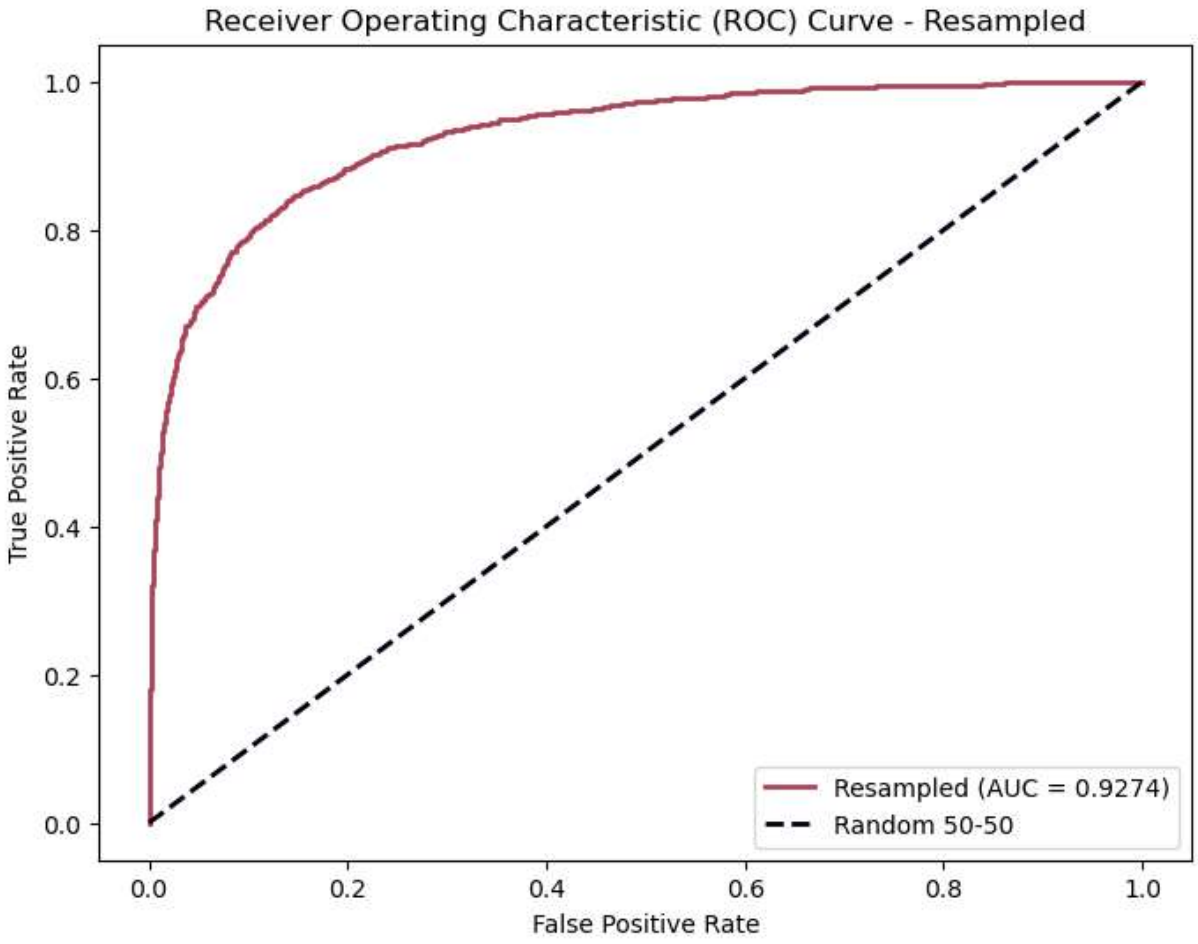
In [167]:
```
# Calling confusion matrix for resampled model
rs_cfn_matrix = confusion_matrix_display(rs_pipeline, y_test, rs_y_pred)
```



Recall not only improved overall, it now has a better result for Detractors specifically, which is what we are aiming for.
The Not Detractors class remains extremely strong.

5) ROC Curve and AUC

In [168]:
```
# Plotting the ROC curve for the baseline model
resamp_y_pred_proba = rs_pipeline.predict_proba(X_test['original_review'])[:, 1]
auc_resampled = plot_roc_curve(y_test, resamp_y_pred_proba, model_name=resampled_
```



In [169]:
```
# Printing the interpretation
auc_interpretation(auc_resampled)
```

```
AUC = 0.9274. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [170]:
```
# Comparing AUC to previous one
metrics_comparison('AUC', auc_base, auc_resampled)
```

```
AUC improved from 0.9110 to 0.9274.
```

AUC increased, which was to be expected, since the majority class was undersampled to give more weight to the minority class.

In [171]:
```
# Appending each metric to the lists
modelnames.append(resampled_model_name)
accuracies.append(accuracy_rs)
f1s.append(f1_rs)
recalls.append(recall_rs)
recalls_detractors.append(recall_detractors_rs)
cv_accs.append(cv_rs)
roc_aucs.append(auc_resampled)
```

3rd iteration: Undersampling negative reviews: Sampling Strategy

Sampling Strategy: 1

I will try to reach the same number of detractors reviews than there are non_detractors reviews. To do so, I will apply a sampling strategy of 1.

1) Fitting and training on train data

In [172]:
```
# Importing relevant packages
from imblearn.under_sampling import RandomUnderSampler
import imblearn.pipeline
```

I will now include the Random Undersampler to our pipeline so it under samples our majority class.

In [173]:
```
# Defining the pipeline steps
tfidf_vectorizer = TfidfVectorizer()
naive_bayes_classifier = MultinomialNB()

# Including the UnderSampler to the pipeline
rs1_pipeline = imblearn.pipeline.Pipeline([
    ('tfidf', tfidf_vectorizer),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', naive_bayes_classifier)
])

# Fitting the pipeline on X_train['original_review'] and y_train
rs1_pipeline.fit(X_train['original_review'], y_train)

rs1_y_pred = rs1_pipeline.predict(X_test['original_review'])
```

In [174]:
```
# Verifying the value counts of each category, before undersampling
original_value_counts = y_train.value_counts()
print("Original class distribution:")
print(original_value_counts)

# Getting the indices of the resampled data
resampled_indices = rs1_pipeline.named_steps['us'].sample_indices_

# Verifying the new value counts of each category, after undersampling
resampled_value_counts = y_train.iloc[resampled_indices].value_counts()
print("\nClass distribution after undersampling:")
print(resampled_value_counts)
```

```
Original class distribution:
0    11320
1     4048
Name: Sentiment, dtype: int64

Class distribution after undersampling:
0    4048
1    4048
Name: Sentiment, dtype: int64
```

The not positive reviews were randomly undersampled from our model, which now has the exact same number of reviews for both sentiments.

I will now run our evaluation metrics to review if the model is providing better predictions.

    2) Evaluation Metrics

In [175]:
```
# Naming the model
resampled1_model_name = 'Sampling 1'

# Calling the function and recording into the defined values
accuracy_rs1, f1_rs1, recall_rs1, cv_rs1 = evaluation_metrics(
    y_test,
    rs1_y_pred,
    rs1_pipeline,
    X_train['original_review'],
    y_train)
```

```
Accuracy: 0.8312
F1-Score: 0.8378
Recall: 0.8312
Mean Cross-Validated Accuracy: 0.8442
```

In [176]:
```
# Comparing the evaluation metrics between baseline and Resampled model
metrics_comparison('Accuracy', accuracy_rs, accuracy_rs1)
metrics_comparison('F1', f1_rs, f1_rs1)
metrics_comparison('Recall', recall_rs, recall_rs1)
metrics_comparison('Cross-validated Accuracy', cv_rs, cv_rs1)
```

```
Accuracy decreased from 0.8842 to 0.8312.
F1 decreased from 0.8784 to 0.8378.
Recall decreased from 0.8842 to 0.8312.
Cross-validated Accuracy decreased from 0.8836 to 0.8442.
```

In [177]:
```
# Printing the overall summary
print('The resampling with sampling strategy of 1 did not improve the overall scc
```

```
The resampling with sampling strategy of 1 did not improve the overall scores
```

3) Classification Report

In [178]:
```
# Calling confusion report for resampled model
resamp1_class_report, recall_detractors_rs1 = class_calculation(y_test, rs1_y_pre
```

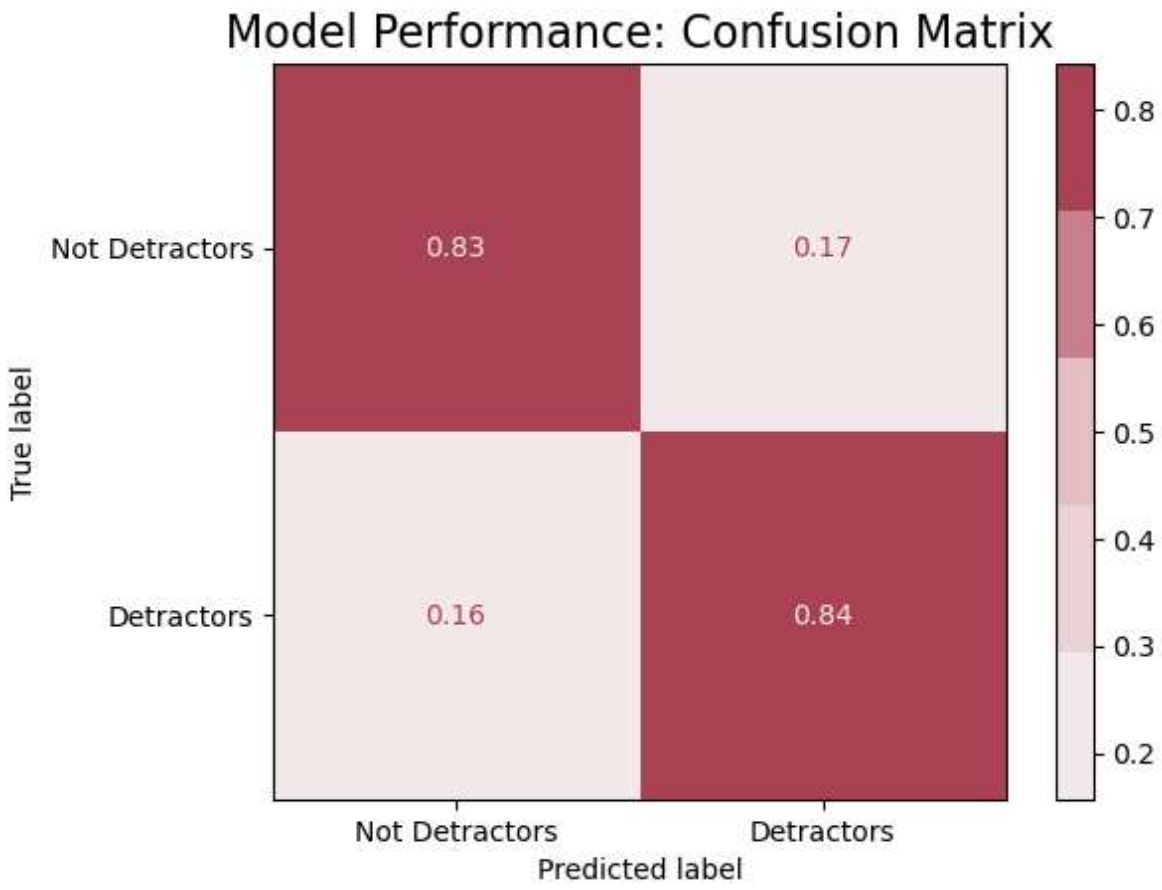
```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.93637   0.82693   0.87825      3773
    Detractors    0.63540   0.84296   0.72461      1350

      accuracy                        0.83115      5123
     macro avg    0.78589   0.83495   0.80143      5123
  weighted avg    0.85706   0.83115   0.83777      5123
```

While the overall scores did not improve, the balance between each class: detractors and not detractors, look much better.

`Recall` for the Detractors class is the highest it was so far.

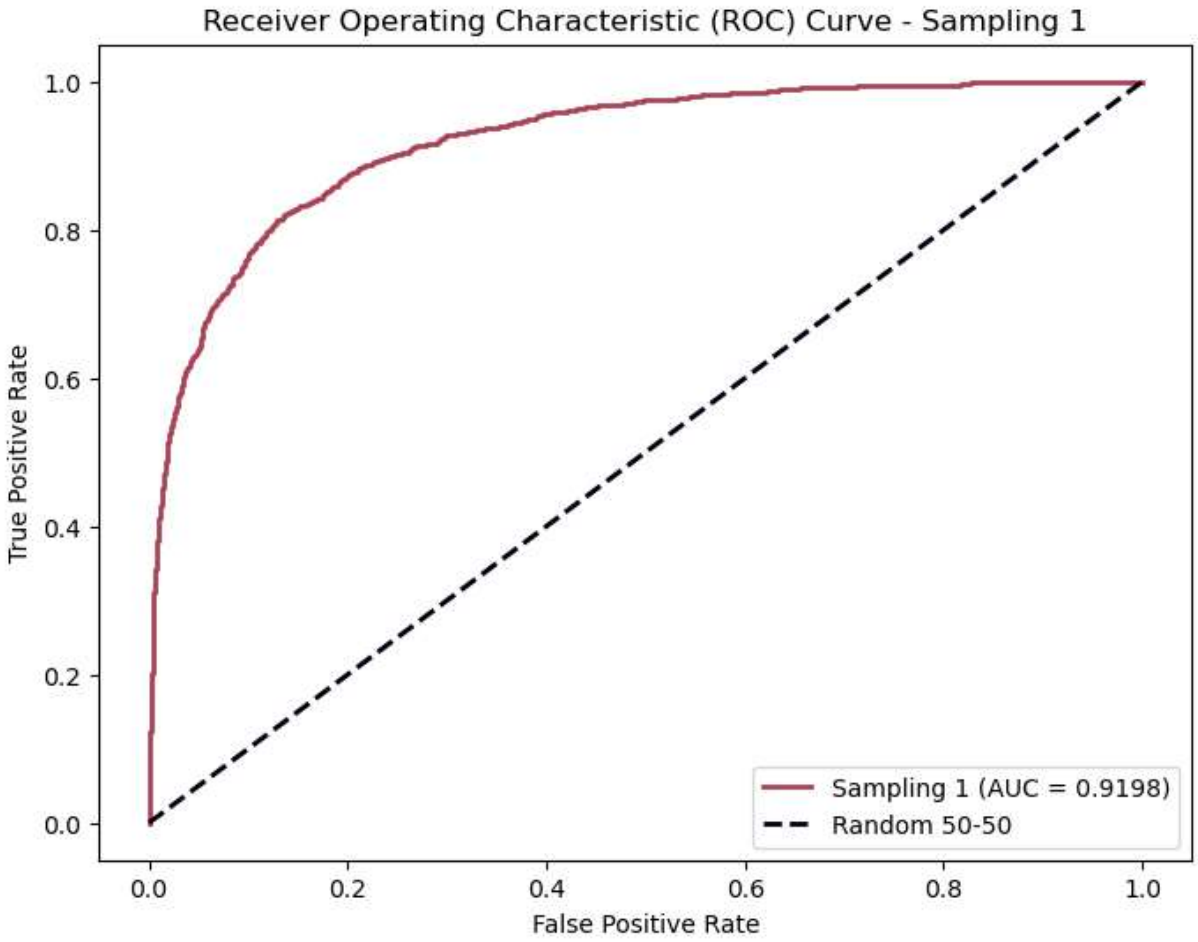
4) Confusion Matrix

In [179]:
```
# Calling confusion matrix for resampled model
rs1_cfn_matrix = confusion_matrix_display(rs1_pipeline, y_test, rs1_y_pred)
```



5) ROC Curve and AUC

In [180]:
```
# Plotting the ROC curve for the baseline model
resamp1_y_pred_proba = rs1_pipeline.predict_proba(X_test['original_review'])[:, 1
auc_resamp1 = plot_roc_curve(y_test, resamp1_y_pred_proba, model_name=resampled1_
```



In [181]:
```
# Printing the interpretation
auc_interpretation(auc_resamp1)
```

```
AUC = 0.9198. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [182]:
```
# Comparing AUC to previous
metrics_comparison('AUC', auc_resampled, auc_resamp1)
```

```
AUC decreased from 0.9274 to 0.9198.
```

There was a slight decrease of the model's ability to differentiate the class. Nevertheless, the score remains very high so the predictive power remains very good.

In [183]:
```
# Appending each metric to the lists
modelnames.append(resampled1_model_name)
accuracies.append(accuracy_rs1)
f1s.append(f1_rs1)
recalls.append(recall_rs1)
recalls_detractors.append(recall_detractors_rs1)
cv_accs.append(cv_rs1)
roc_aucs.append(auc_resamp1)
```

4th iteration: including stopwords

I will now test fitting the vectorizer by removing the stopwords from reviews to review if this can help predictions be more accurate.

1) Fitting and training train data

In [184]:
```
# Defining the pipeline steps, including the stopwords_list created
tfidf_vectorizer = TfidfVectorizer(stop_words=stopwords_list)
naive_bayes_classifier = MultinomialNB()

# Instantiating the pipeline with the undersampler and the new vectorizer
pipeline_nostop = imblearn.pipeline.Pipeline([
    ('tfidf', tfidf_vectorizer),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', naive_bayes_classifier)
])

# Fit the pipeline on X_train['original_review'] and y_train
pipeline_nostop.fit(X_train['original_review'], y_train)

# Generating predictions
nostop_y_pred = pipeline_nostop.predict(X_test['original_review'])
```

2) Evaluation Metrics

In [185]:
```
# Naming the model
nostop_model_name = 'No Stopwords'

# Calling the function and recording into the defined values
accuracy_nostop, f1_nostop, recall_nostop, cv_nostop = evaluation_metrics(
    y_test,
    nostop_y_pred,
    pipeline_nostop,
    X_train['original_review'],
    y_train)
```

```
Accuracy: 0.8319
F1-Score: 0.8382
Recall: 0.8319
Mean Cross-Validated Accuracy: 0.8466
```

In [186]:
```
# Comparing the evaluation metrics between baseline and Resampled model
metrics_comparison('Accuracy', accuracy_rs1, accuracy_nostop)
metrics_comparison('F1', f1_rs1, f1_nostop)
metrics_comparison('Recall', recall_rs1, recall_nostop)
metrics_comparison('Cross-validated Accuracy', cv_rs1, cv_nostop)
```

```
Accuracy improved from 0.8312 to 0.8319.
F1 improved from 0.8378 to 0.8382.
Recall improved from 0.8312 to 0.8319.
Cross-validated Accuracy improved from 0.8442 to 0.8466.
```

In [187]:
```
# Printing the overall summary
print('Removing English stopwords improved all scores overall. I will now verify
```

```
Removing English stopwords improved all scores overall. I will now verify how t
rue this is by class.
```

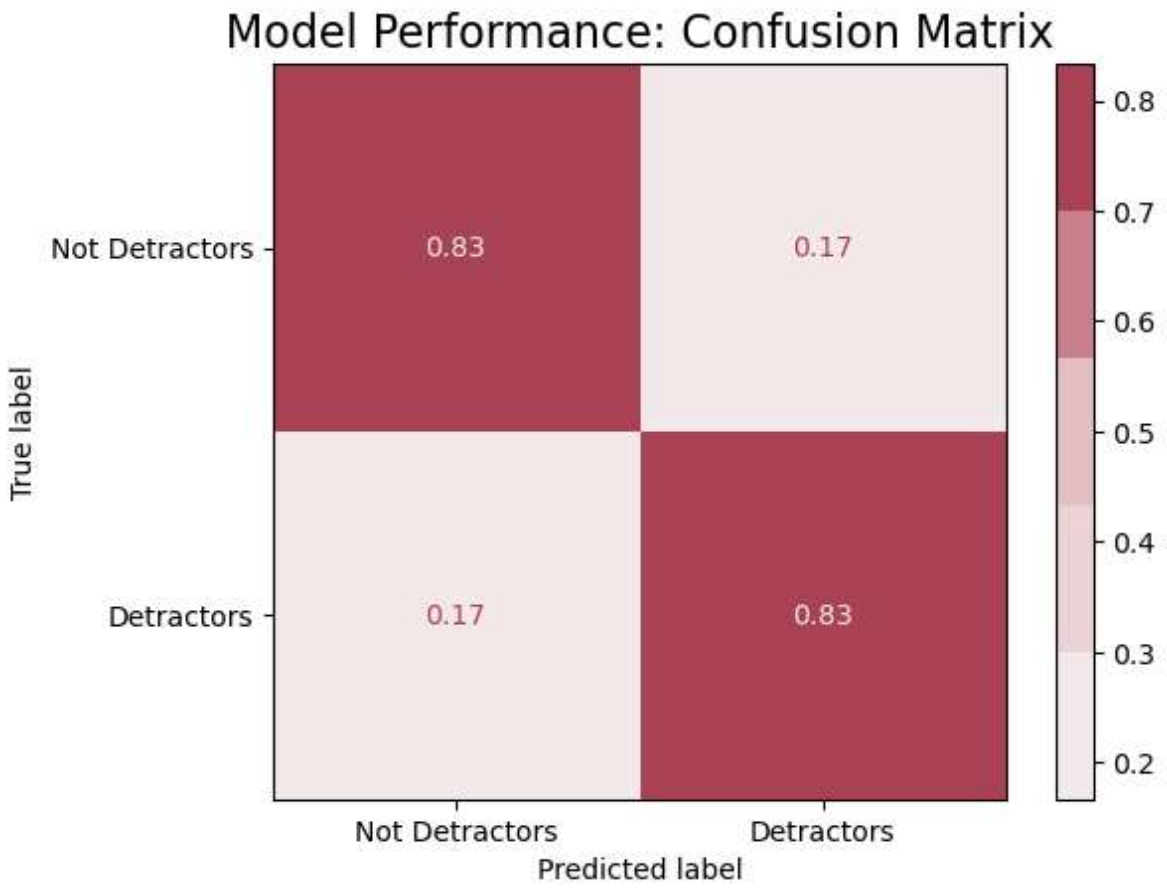
3) Classification Report

In [188]:
```
# Calling the classification function
nostop_class_report, recall_detractors_nostop = class_calculation(y_test, nostop_
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.93308   0.83143   0.87933      3773
    Detractors    0.63884   0.83333   0.72324      1350

      accuracy                        0.83193      5123
     macro avg    0.78596   0.83238   0.80128      5123
  weighted avg    0.85554   0.83193   0.83820      5123
```

4) Confusion Matrix

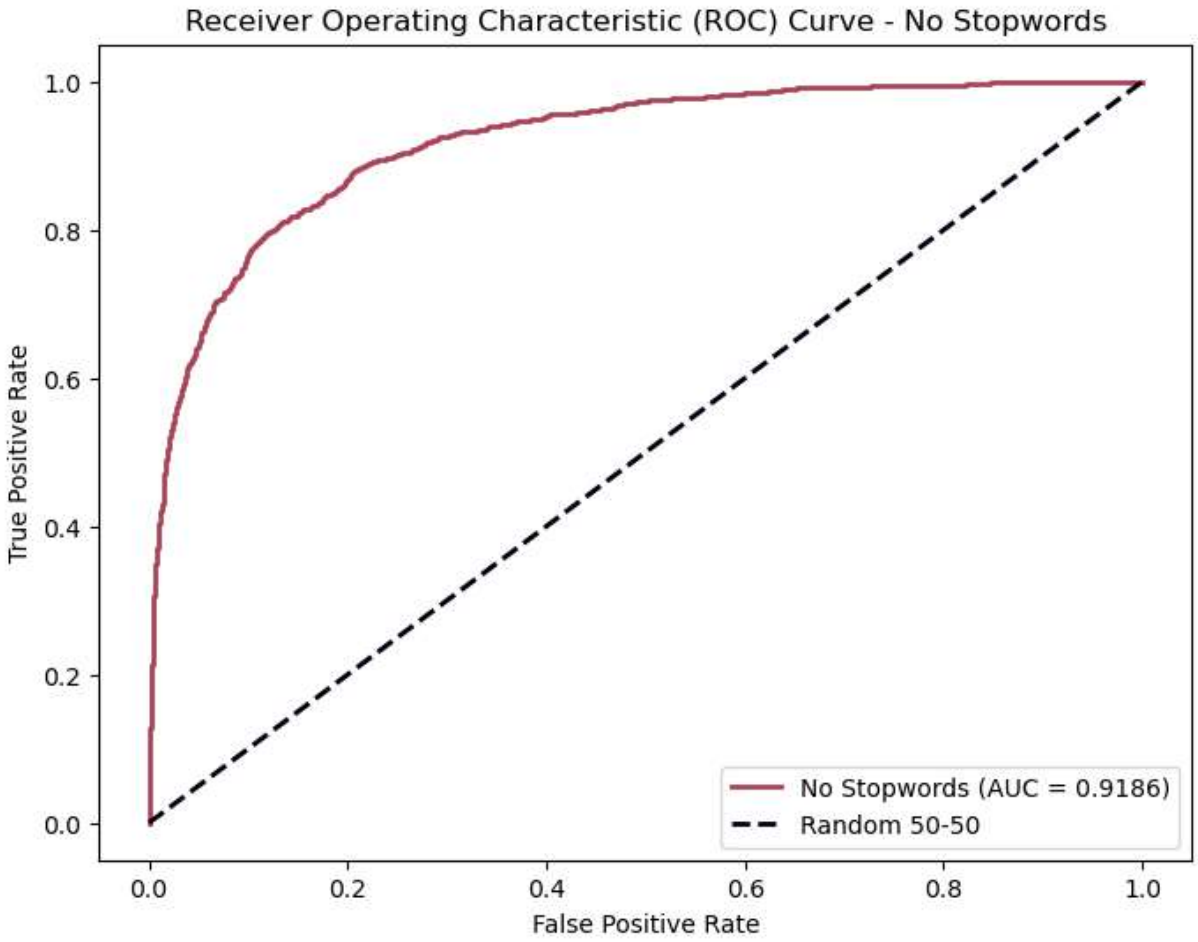
In [189]:
```
# Calling the confusion matrix function
nostop_cfn_matrix = confusion_matrix_display(pipeline_nostop, y_test, nostop_y_pr
```



Recall for detractors slightly decreased, but remains very high while all other metrics also performed really strongly.

5) ROC Curve and AUC

In [190]:
```
# Plotting the ROC curve for the baseline model
nostop_y_pred_proba = pipeline_nostop.predict_proba(X_test['original_review'])[:,
auc_nostop = plot_roc_curve(y_test, nostop_y_pred_proba, model_name=nostop_model_
```



In [191]:
```
# Printing the interpretation
auc_interpretation(auc_nostop)
```

```
AUC = 0.9186. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [192]:
```
metrics_comparison('AUC', auc_resamp1, auc_nostop)
```

```
AUC decreased from 0.9198 to 0.9186.
```

In [193]:
```
# Appending each metric to the lists
modelnames.append(nostop_model_name)
accuracies.append(accuracy_nostop)
f1s.append(f1_nostop)
recalls.append(recall_nostop)
recalls_detractors.append(recall_detractors_nostop)
cv_accs.append(cv_nostop)
roc_aucs.append(auc_nostop)
```

5th iteration: including only the industry specific stopwords

For analysis purposes, I had removed a list of industry specific stopwords which came up often and were as present on the positive than not_positive reviews without adding specific meaning to them.

I will test our model on by including only this list as stopwords.

1) Fitting and training train data

In [194]:
```
# Reminding the list of industry specific stopwords
new_stopwords
```

Out[194]:
```
['hotel', 'room', 'night', 'day', 'stay', 'resort', 'place']
```

In [195]:
```
# Defining the pipeline steps, including the stopwords_list created
tfidf_vectorizer = TfidfVectorizer(stop_words=new_stopwords)
naive_bayes_classifier = MultinomialNB()

# Instantiating the pipeline with the undersampler and the new vectorizer
pipeline_nonewstop = imblearn.pipeline.Pipeline([
    ('tfidf', tfidf_vectorizer),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', naive_bayes_classifier)
])

# Fit the pipeline on X_train['original_review'] and y_train
pipeline_nonewstop.fit(X_train['original_review'], y_train)

# Generating predictions
nonewstop_y_pred = pipeline_nonewstop.predict(X_test['original_review'])
```

2) Evaluation Metrics

In [196]:
```
# Naming the model
nonewstop_model_name = 'Industry Stopwords'

# Calling the function and recording into the defined values
accuracy_nonewstop, f1_nonewstop, recall_nonewstop, cv_nonewstop = evaluation_met
    y_test,
    nonewstop_y_pred,
    pipeline_nonewstop,
    X_train['original_review'],
    y_train)
```

```
Accuracy: 0.8331
F1-Score: 0.8396
Recall: 0.8331
Mean Cross-Validated Accuracy: 0.8460
```

In [197]:
```
# Comparing the evaluation metrics between baseline and Resampled model
metrics_comparison('Accuracy', accuracy_nostop, accuracy_nonewstop)
metrics_comparison('F1', f1_nostop, f1_nonewstop)
metrics_comparison('Recall', recall_nostop, recall_nonewstop)
metrics_comparison('Cross-validated Accuracy', cv_nostop, cv_nonewstop)
```

```
Accuracy improved from 0.8319 to 0.8331.
F1 improved from 0.8382 to 0.8396.
Recall improved from 0.8319 to 0.8331.
Cross-validated Accuracy decreased from 0.8466 to 0.8460.
```

All scores continued increasing, except from cross-validated accuracy. Cross-validated accuracy is an estimate of the model's performance on test data using cross-validation. The decrease is very small, but it indicates the model would perform slightly lower on unseen data.

In [198]:
```
# Printing the overall summary
print('Including Industry stopwords improved scores on training data compared to
```

```
Including Industry stopwords improved scores on training data compared to inclu
ding English stopwords.
```

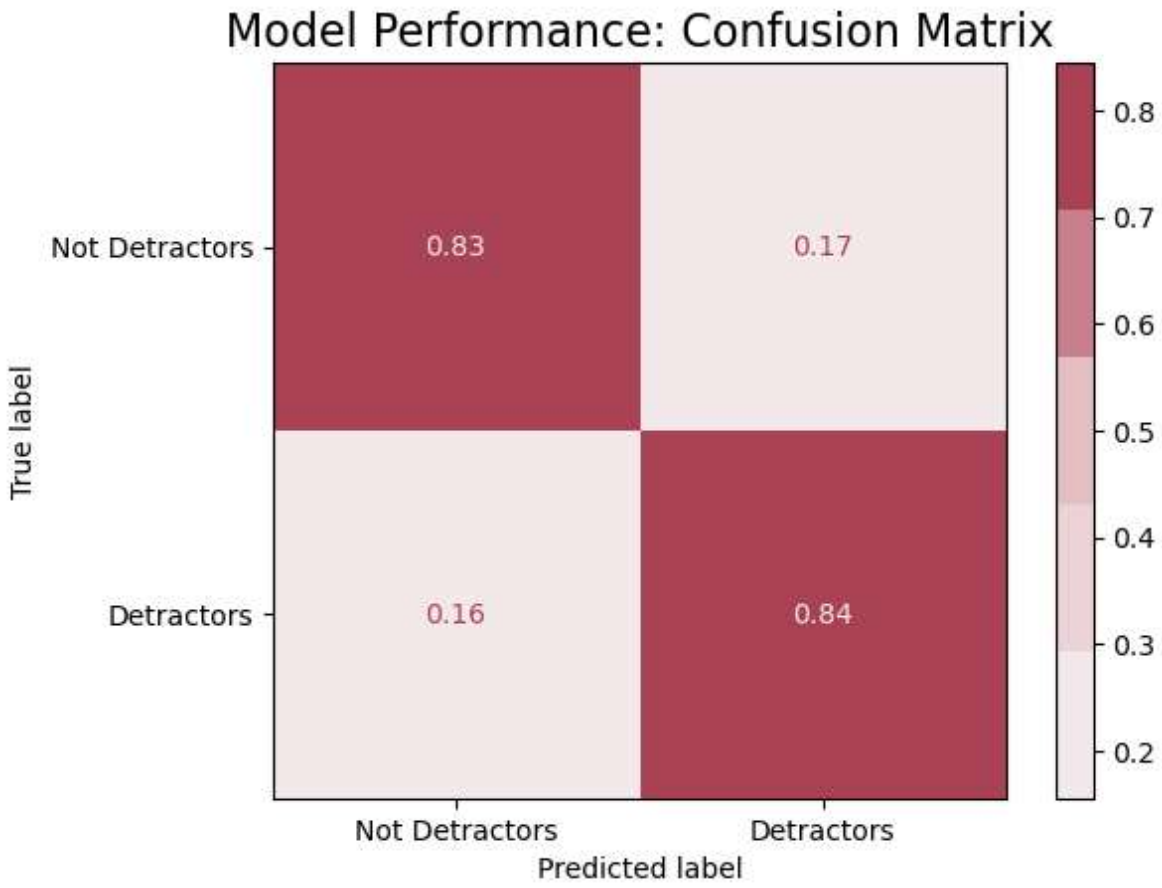
3) Classification Report

In [199]:
```
# Calling the classification function
nonewstop_class_report, recall_detractors_nonewstop = class_calculation(y_test, r
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.93709   0.82905   0.87976      3773
    Detractors    0.63866   0.84444   0.72727      1350

      accuracy                        0.83311      5123
     macro avg    0.78787   0.83675   0.80352      5123
  weighted avg    0.85845   0.83311   0.83958      5123
```

4) Confusion Matrix

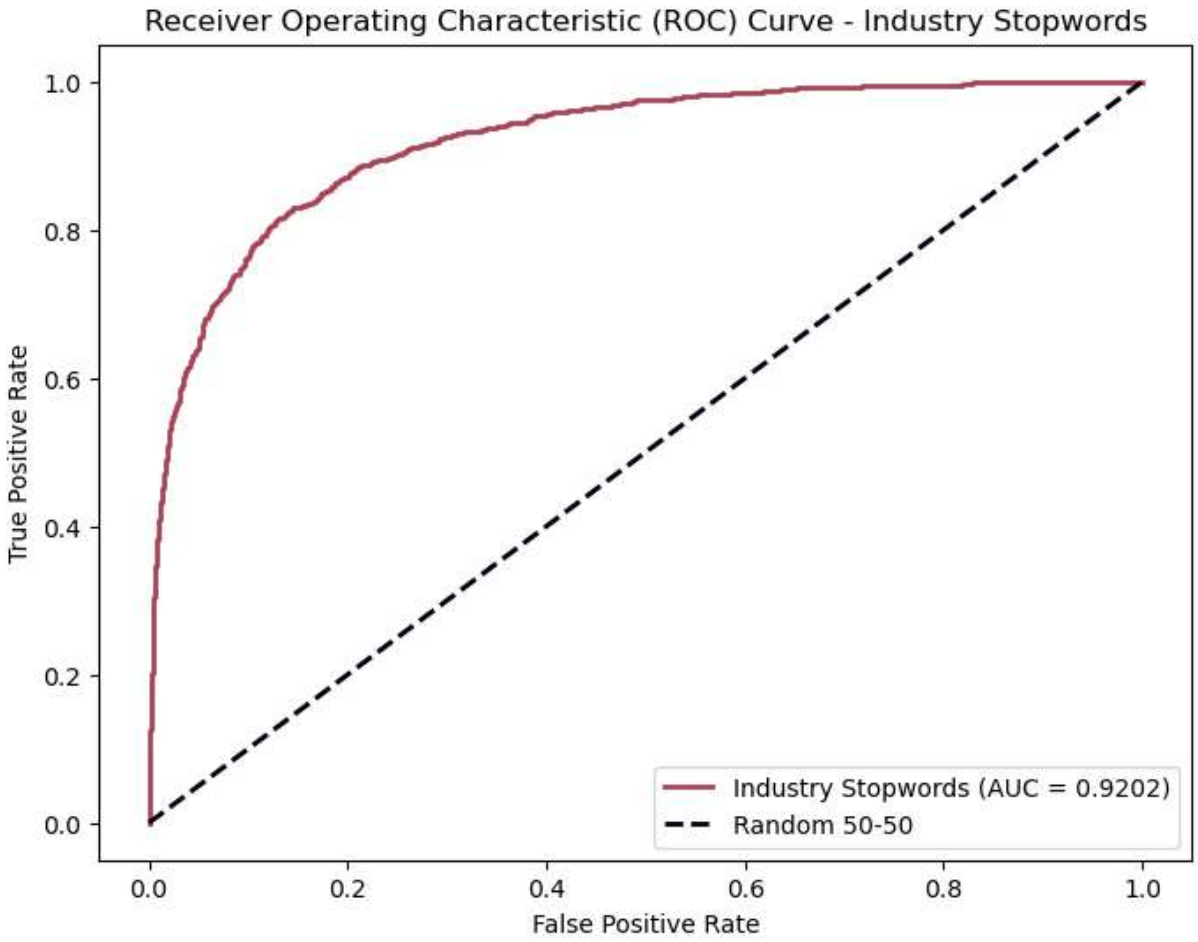
In [200]:
```
# Calling the confusion matrix function
nonewstop_cfn_matrix = confusion_matrix_display(pipeline_nonewstop, y_test, nonew
```



Removing stopwords increased the `recall` for Detractors back to higher results, above 84%.

5) ROC Curve and AUC

In [201]:
```
# Plotting the ROC curve for the baseline model
nonewstop_y_pred_proba = pipeline_nonewstop.predict_proba(X_test['original_review
auc_nonewstop = plot_roc_curve(y_test, nonewstop_y_pred_proba, model_name=nonewst
```



In [202]:
```
# Printing the interpretation
auc_interpretation(auc_nonewstop)
```

```
AUC = 0.9202. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [203]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_nostop, auc_nonewstop)
print()
```

```
Compared to the last run model:
AUC improved from 0.9186 to 0.9202.
```

And the model got better at distinguishing the two classes.

In [204]:
```
# Appending each metric to the lists
modelnames.append(nonewstop_model_name)
accuracies.append(accuracy_nonewstop)
f1s.append(f1_nonewstop)
recalls.append(recall_nonewstop)
recalls_detractors.append(recall_detractors_nonewstop)
cv_accs.append(cv_nonewstop)
roc_aucs.append(auc_nonewstop)
```

6th iteration: adding both English and industry stopwords

I will now test the model performance after excluding the combined list of stopwords: English, and industry-related.

1) Fitting and training train data

I will create a new list of stopwords as a copy of the existing one, so the industry stopwords can be added to it.

In [205]:
```
# Reminding the list of English stopwords
stopwords_list[:3] # is the one including all English stopwords.
```

Out[205]:
```
['i', 'me', 'my']
```

In [206]:
```
# Reminding the list of industry specific stopwords
new_stopwords # includes industry specific stopwords
```

Out[206]:
```
['hotel', 'room', 'night', 'day', 'stay', 'resort', 'place']
```

In [207]:
```
# Creating a new list to combine both stopwords lists
full_stopwords = stopwords_list.copy()

# Adding the new_stopwords to the existing list
full_stopwords.extend(new_stopwords)
```

In [208]:
```
# Verifying the list of new_stopwords were added to the stopwords list
assert full_stopwords[-len(new_stopwords):] == new_stopwords
```

In [209]:
```
# Defining the pipeline steps, including the new stopwords_list created
tfidf_vectorizer = TfidfVectorizer(stop_words=full_stopwords)
naive_bayes_classifier = MultinomialNB()

# Instantiating the pipeline with the undersampler and the new vectorizer
pipeline_fullstop = imblearn.pipeline.Pipeline([
    ('tfidf', tfidf_vectorizer),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', naive_bayes_classifier)
])

# Fit the pipeline on X_train['original_review'] and y_train
pipeline_fullstop.fit(X_train['original_review'], y_train)

# Generating predictions
fullstop_y_pred = pipeline_fullstop.predict(X_test['original_review'])
```

2) Evaluation Metrics

In [210]:
```
# Naming the model
fullstop_model_name = 'Full Stopwords'

# Calling the function and recording into the defined values
accuracy_fullstop, f1_fullstop, recall_fullstop, cv_fullstop = evaluation_metrics
    y_test,
    fullstop_y_pred,
    pipeline_fullstop,
    X_train['original_review'],
    y_train)
```

```
Accuracy: 0.8345
F1-Score: 0.8405
Recall: 0.8345
Mean Cross-Validated Accuracy: 0.8481
```

The accuracy score now increased slightly, however it remains below just guessing the majority class.

In [211]:
```
# Comparing the evaluation metrics between baseline and Resampled model
metrics_comparison('Accuracy', accuracy_nonewstop, accuracy_fullstop)
metrics_comparison('F1', f1_nonewstop, f1_fullstop)
metrics_comparison('Recall', recall_nonewstop, recall_fullstop)
metrics_comparison('Cross-validated Accuracy', cv_nonewstop, cv_fullstop)
```

```
Accuracy improved from 0.8331 to 0.8345.
F1 improved from 0.8396 to 0.8405.
Recall improved from 0.8331 to 0.8345.
Cross-validated Accuracy improved from 0.8460 to 0.8481.
```

In [212]:
```
# Printing the overall summary
print('Including full list of stopwords improved scores.')
```

```
Including full list of stopwords improved scores.
```

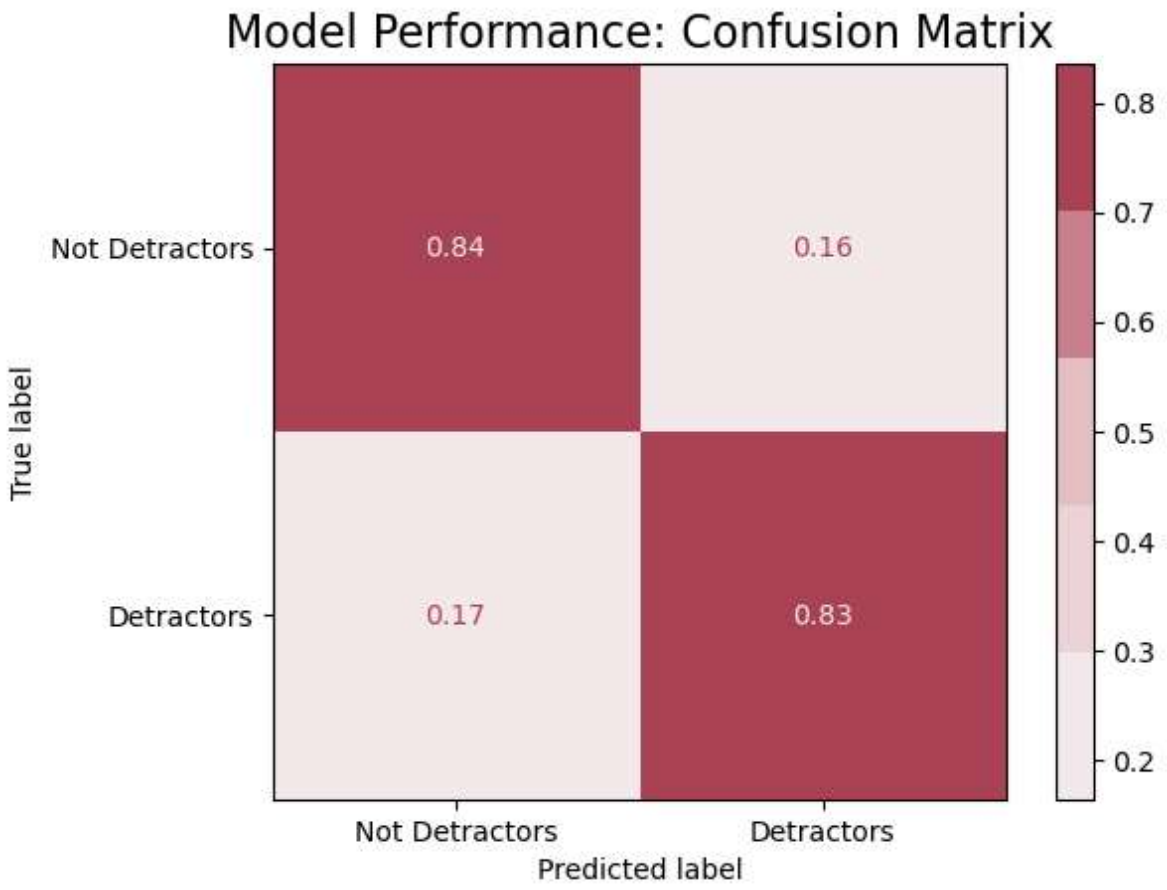
3) Classification Report

In [213]:
```
# Calling the classification function
fullstop_class_report, recall_detractors_fullstop = class_calculation(y_test, ful
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.93308   0.83514   0.88140      3773
    Detractors    0.64376   0.83259   0.72610      1350

      accuracy                        0.83447      5123
     macro avg    0.78842   0.83387   0.80375      5123
  weighted avg    0.85684   0.83447   0.84047      5123
```

4) Confusion Matrix

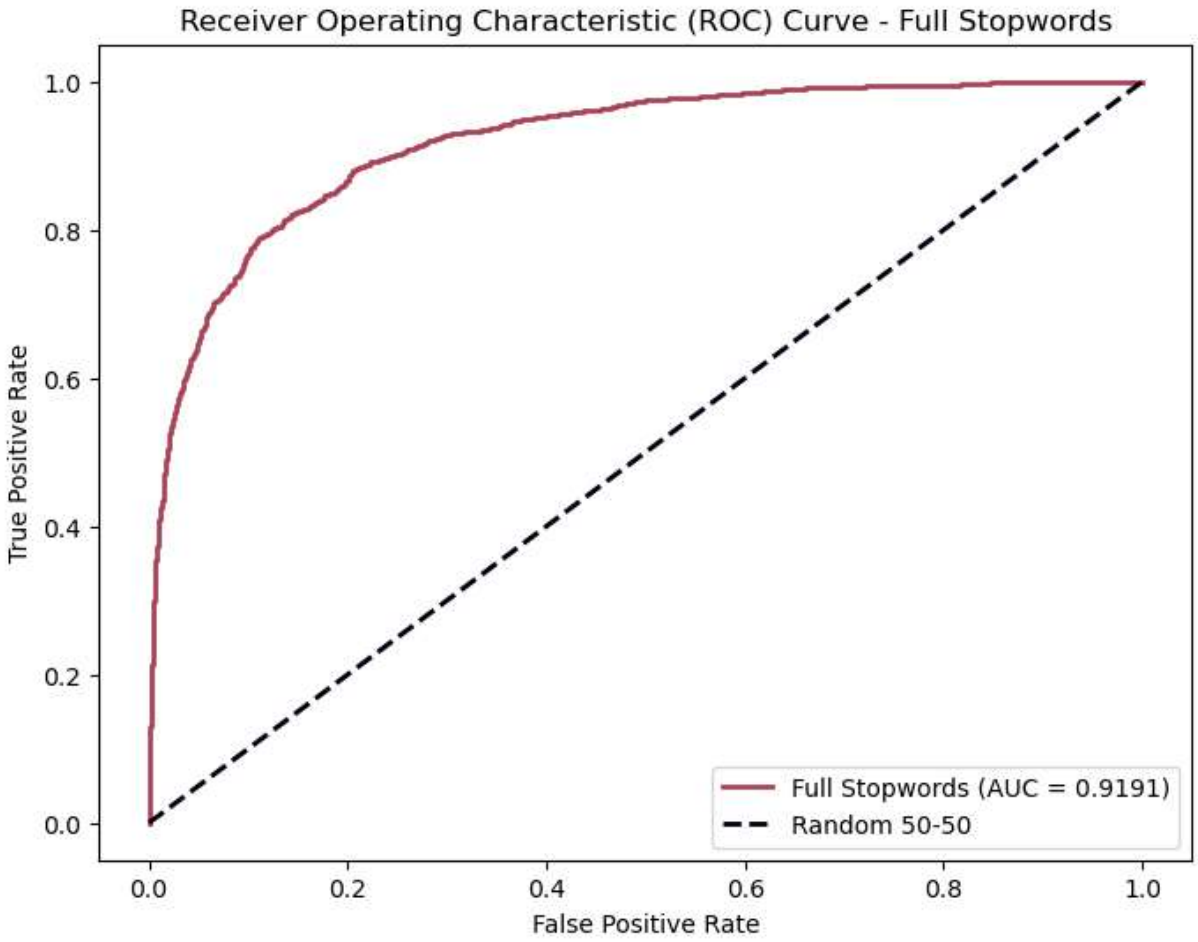
In [214]:
```
# Calling the confusion matrix function
fullstop_cfn_matrix = confusion_matrix_display(pipeline_fullstop, y_test, fullsto
```



Including the full list improved the model but not recall for detractors.

5) ROC Curve and AUC

In [215]:
```
# Plotting the ROC curve for the baseline model
fullstop_y_pred_proba = pipeline_fullstop.predict_proba(X_test['original_review']
auc_fullstop = plot_roc_curve(y_test, fullstop_y_pred_proba, model_name=fullstop_
```



In [216]:
```
# Printing the interpretation
auc_interpretation(auc_fullstop)
```

```
AUC = 0.9191. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [217]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_nonewstop, auc_fullstop)
print()
```

```
Compared to the last run model:
AUC decreased from 0.9202 to 0.9191.
```

In [218]:
```
# Appending each metric to the lists
modelnames.append(fullstop_model_name)
accuracies.append(accuracy_fullstop)
f1s.append(f1_fullstop)
recalls.append(recall_fullstop)
recalls_detractors.append(recall_detractors_fullstop)
cv_accs.append(cv_fullstop)
roc_aucs.append(auc_fullstop)
```

7th iteration: Lemmatized Reviews

Models perform better when `industry stopwords` are excluded from the reviews. Let's now review how it compares when it runs of lemmatized words.

1) Fitting and training train data

For the exploratory analysis, some adjectives were included in this list, however they would be important to help predicting the sentiment of the review. They will be dropped from this list so they are still considered by the model.

In [219]:
```
# Only lemmatizing X_train['lemm_review ']
X_train['lemmatized_review'] = X_train['original_review'].apply(lambda x: tokeni
X_train['lemmatized_review'] = X_train['lemmatized_review'].apply(lambda tokens:
```

In [220]:
```
# Storing outside a list
X_train['lemmatized_review'] = X_train['lemmatized_review'].apply(lambda x: ' '.j
```

In [221]:
```
# Viewing the newly created column
X_train[:3]
```

Out[221]:

	Review	Rating	label	original_review	Review_prep_nolist	lemmatized_re
1282	[gem, pleasantly, surprise, accomondations, helpful, attentive, staff, need, breakfast, good, cookies, lovely, clean, comfortable, front, bush, street, bite, noisy, away, enjoyment, nice, boutique, definitely, time]	4	not_detractors	gem, pleasantly surprised accomondations helpful attentive staff need, breakfast good cookies, room lovely clean comfortable, room fronting bush street bit noisy did not away enjoyment nice boutique hotel, definitely stay time,	gem pleasantly surprise accomondations helpful attentive staff need breakfast good cookies room lovely clean comfortable room front bush street bite noisy away enjoyment nice boutique hotel definitely stay time	gem pleas sur accomonda helpful atte staff need brea good cookies lovely comfortable front bush stree noisy do not enjoyment boutique definitely stay
	[love, fita, wife, spend, nights, fita, march			loved fita wife spent nights hotel fita march		

In [222]:
```
# Applying the same to X_test

X_test['lemmatized_review'] = X_test['original_review'].apply(lambda x: tokenizer
X_test['lemmatized_review'] = X_test['lemmatized_review'].apply(lambda tokens: [
X_test['lemmatized_review'] = X_test['lemmatized_review'].apply(lambda x: ' '.joi
```

In [223]:
```
# Defining the pipeline steps, including full stopwords and on lemmatized
tfidf_vectorizer = TfidfVectorizer(stop_words=new_stopwords)
naive_bayes_classifier = MultinomialNB()

# Instantiating the pipeline with the undersampler and the new vectorizer
pipeline_lemm = imblearn.pipeline.Pipeline([
    ('tfidf', tfidf_vectorizer),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', naive_bayes_classifier)
])

# Fit the pipeline on X_train['original_review'] and y_train
pipeline_lemm.fit(X_train['lemmatized_review'], y_train)

# Generating predictions
lemm_y_pred = pipeline_lemm.predict(X_test['lemmatized_review'])
```

2) Evaluation Metrics

In [224]:
```
# Naming the model
lemm_model_name = 'Lemmatized'

# Calling the function and recording into the defined values
accuracy_lemm, f1_lemm, recall_lemm, cv_lemm = evaluation_metrics(
    y_test,
    lemm_y_pred,
    pipeline_lemm,
    X_train['lemmatized_review'],
    y_train)
```

```
Accuracy: 0.8271
F1-Score: 0.8344
Recall: 0.8271
Mean Cross-Validated Accuracy: 0.8382
```

In [225]:
```
# Comparing the evaluation metrics between undersampled model (best model) and nc
metrics_comparison('Accuracy', accuracy_fullstop, accuracy_lemm)
metrics_comparison('F1', f1_fullstop, f1_lemm)
metrics_comparison('Recall', recall_fullstop, recall_lemm)
metrics_comparison('Cross-validated Accuracy', cv_fullstop, cv_lemm)
```

```
Accuracy decreased from 0.8345 to 0.8271.
F1 decreased from 0.8405 to 0.8344.
Recall decreased from 0.8345 to 0.8271.
Cross-validated Accuracy decreased from 0.8481 to 0.8382.
```

In [226]:
```
# Printing the overall summary
print('Most scores decreased slightly.')
```

```
Most scores decreased slightly.
```

3) Classification Report

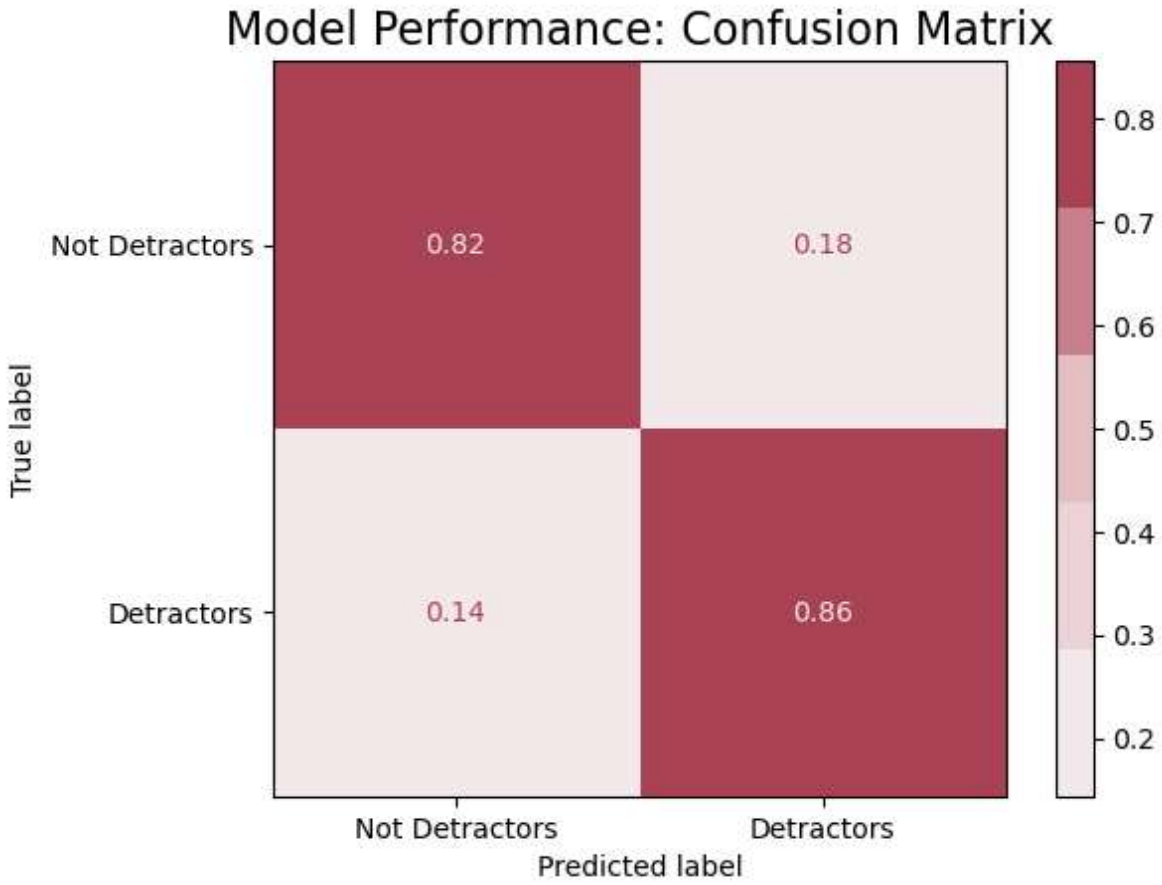
In [227]:
```
# Calling the classification function
lemm_class_report, recall_detractors_lemm = class_calculation(y_test, lemm_y_prec
```

```
Classification Report:
                 precision    recall  f1-score   support

Not Detractors    0.94076   0.81659   0.87429      3773
    Detractors    0.62554   0.85630   0.72295      1350

      accuracy                        0.82705      5123
     macro avg    0.78315   0.83644   0.79862      5123
  weighted avg    0.85770   0.82705   0.83441      5123
```

At first glance, it would seem like the model performance decreased. It actually got the highest results ever recorded for the Detractors class, which is what we are aiming for.

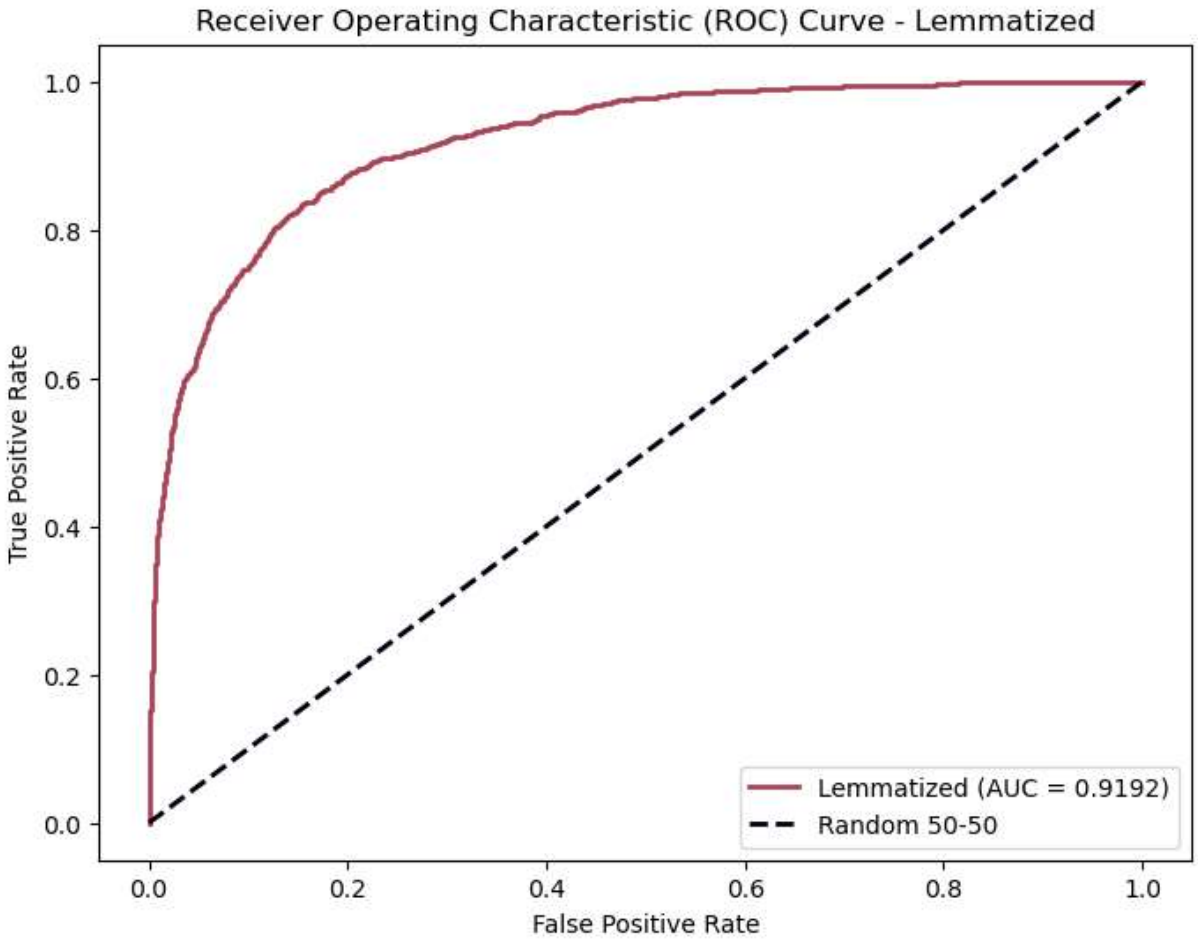
4) Confusion Matrix

In [228]:
```
# Calling the confusion matrix function
lemm_cfn_matrix = confusion_matrix_display(pipeline_lemm, y_test, lemm_y_pred)
```



5) ROC Curve and AUC

In [229]:
```
# Plotting the ROC curve for the baseline model
lemm_y_pred_proba = pipeline_lemm.predict_proba(X_test['lemmatized_review'])[:, 1
auc_lemm = plot_roc_curve(y_test, lemm_y_pred_proba, model_name=lemm_model_name)
```



In [230]:
```
# Printing the interpretation
auc_interpretation(auc_lemm)
```

```
AUC = 0.9192. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [231]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_fullstop, auc_lemm)
print()
```

```
Compared to the last run model:
AUC improved from 0.9191 to 0.9192.
```

In [232]:
```
# Appending each metric to the lists
modelnames.append(lemm_model_name)
accuracies.append(accuracy_lemm)
f1s.append(f1_lemm)
recalls.append(recall_lemm)
recalls_detractors.append(recall_detractors_lemm)
cv_accs.append(cv_lemm)
roc_aucs.append(auc_lemm)
```

Seeing the recall results for the detractors class, it makes sense that the AUC improved compared to the previous model run.

8th iteration: Training on the fully preprocessed reviews

1) Fitting and training on train data

In [233]:
```
# Not applying stopwords to the vectorizer, as they were applied to the preproces
```

In [234]:
```
# Defining the pipeline steps
tfidf_vectorizer = TfidfVectorizer()
naive_bayes_classifier = MultinomialNB()

# Instantiating the pipeline with the undersampler and the new vectorizer
pipeline_prep = imblearn.pipeline.Pipeline([
    ('tfidf', tfidf_vectorizer),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', naive_bayes_classifier)
])

# Fitting the pipeline on X_train and X_test['Review_prep_nolist'] which contains
pipeline_prep.fit(X_train['Review_prep_nolist'], y_train)

# Generating predictions
prep_y_pred = pipeline_prep.predict(X_test['Review_prep_nolist'])
```

2) Evaluation Metrics

In [235]:
```
pp_model_name = 'Preprocessed'

# Calling the function and recording into the defined values
accuracy_pp, f1_pp, recall_pp, cv_pp = evaluation_metrics(
    y_test,
    prep_y_pred,
    pipeline_prep,
    X_train['Review_prep_nolist'],
    y_train)
```

```
Accuracy: 0.8271
F1-Score: 0.8341
Recall: 0.8271
Mean Cross-Validated Accuracy: 0.8404
```

In [236]:
```
# Comparing the evaluation metrics between undersampled model (best model) and nc
metrics_comparison('Accuracy', accuracy_rs1, accuracy_pp)
metrics_comparison('F1', f1_rs1, f1_pp)
metrics_comparison('Recall', recall_rs1, recall_pp)
metrics_comparison('Cross-validated Accuracy', cv_rs1, cv_pp)
```

```
Accuracy decreased from 0.8312 to 0.8271.
F1 decreased from 0.8378 to 0.8341.
Recall decreased from 0.8312 to 0.8271.
Cross-validated Accuracy decreased from 0.8442 to 0.8404.
```

In [237]:
```
# Printing the overall summary
print('All scores decreased on the fully preprocessed reviews.')
```

```
All scores decreased on the fully preprocessed reviews.
```

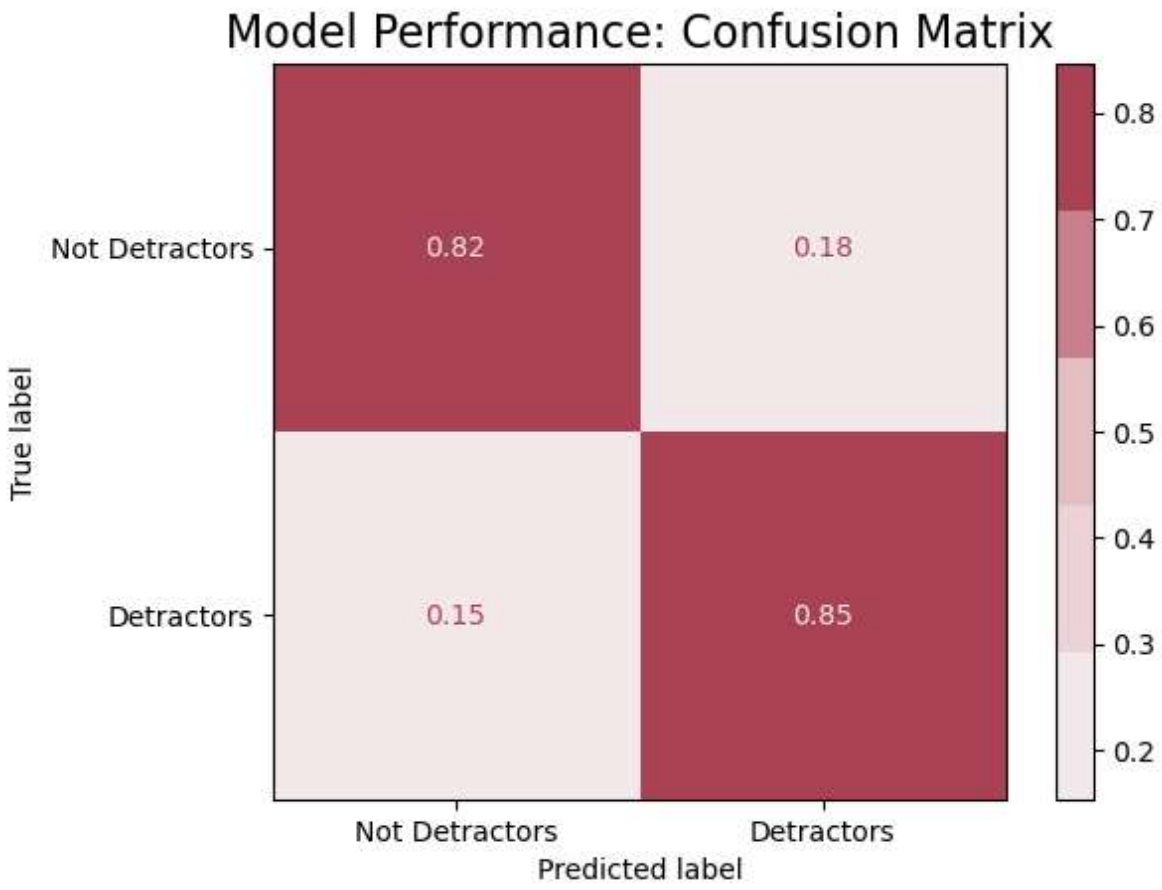
3) Classification Report

In [238]:
```
# Drawing the classification report
prep_class_report, recall_detractors_prep = class_calculation(y_test, prep_y_prec
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.93703   0.82030   0.87479      3773
    Detractors    0.62747   0.84593   0.72050      1350

      accuracy                        0.82705      5123
     macro avg    0.78225   0.83311   0.79765      5123
  weighted avg    0.85545   0.82705   0.83413      5123
```

4) Confusion Matrix

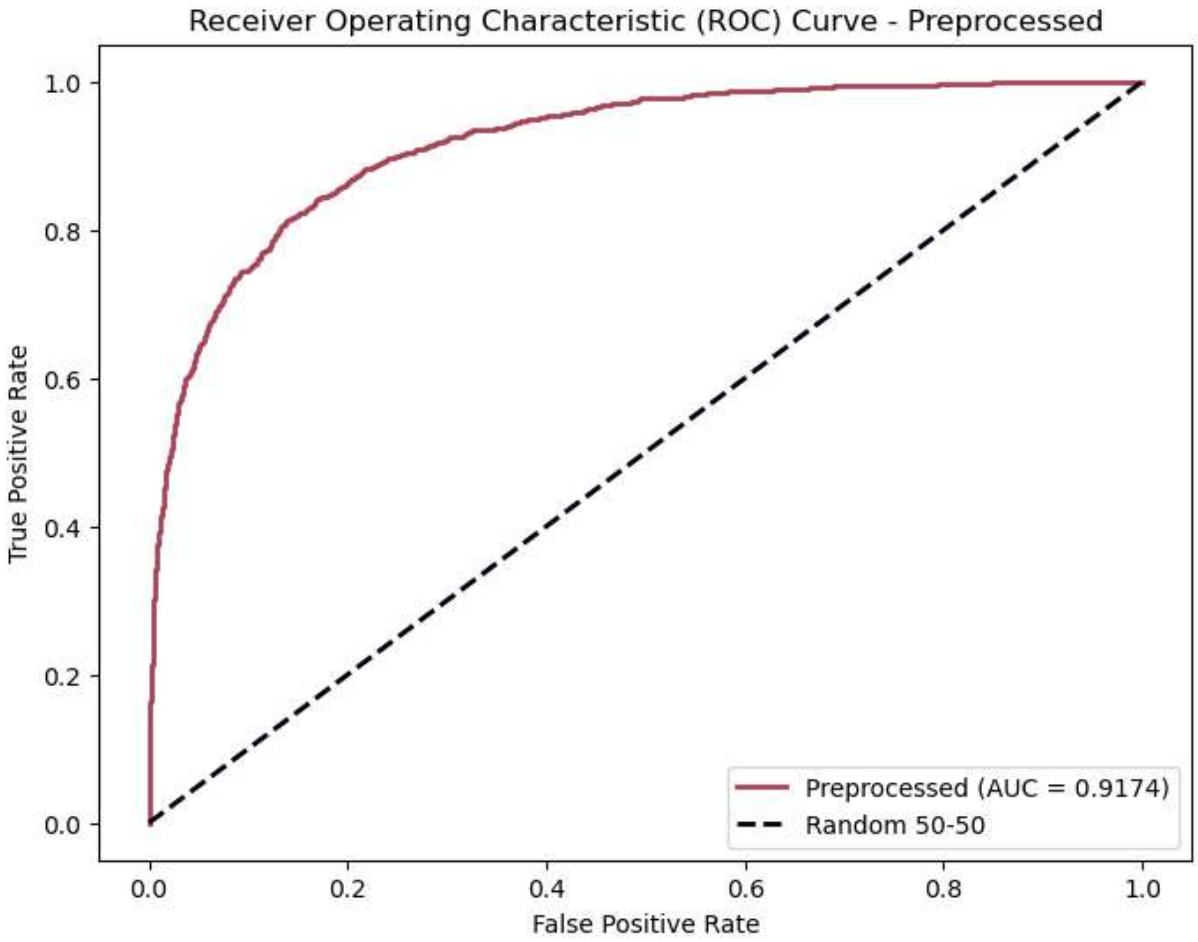
In [239]:
```
# Plotting the confusion matrix
prep_cnf_matrix = confusion_matrix_display(pipeline_prep, y_test, prep_y_pred)
```



This is also reflected on recall for detractors which decreased compared to the one for lemmatized reviews.

5) ROC Curve and AUC

In [240]:
```
# Plotting the ROC curve for the baseline model
prep_y_pred_proba = pipeline_prep.predict_proba(X_test['Review_prep_nolist'])[:,
auc_pp = plot_roc_curve(y_test, prep_y_pred_proba, model_name=pp_model_name)
```



In [241]:
```
# Printing the interpretation
auc_interpretation(auc_pp)
```

```
AUC = 0.9174. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [242]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_lemm, auc_pp)
print()
```

```
Compared to the last run model:
AUC decreased from 0.9192 to 0.9174.
```

And AUC decreased as well. The models should be applied on the lemmatized reviews, not the preprocessed ones.

In [243]:
```
# Appending each metric to the lists
modelnames.append(pp_model_name)
accuracies.append(accuracy_pp)
f1s.append(f1_pp)
recalls.append(recall_pp)
recalls_detractors.append(recall_detractors_prep)
cv_accs.append(cv_pp)
roc_aucs.append(auc_pp)
```

9th iteration: Tuning Tfidf Vectorizer - Hyperparameter tuning

The model performed better when stopwords were removed but worse when applied on the full tokenized reviews. Let's try to use combinatoric grid searching to find the best parameters for the vectorizer.

1) Fitting and training on train data

In [244]:
```
from sklearn.model_selection import GridSearchCV

# Defining the pipeline steps, excluding any manual input of features, with the v
# The pipeline still includes the undersampler to ensure class imbalance is addre
gs_pipeline = imblearn.pipeline.Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', MultinomialNB())
])

# Define the parameter grid to search over
parameters = {
    'tfidf__max_features': [100, None],
    'tfidf__max_df': [0.7, 0.9],
    'tfidf__min_df': [1, 3],
    'tfidf__sublinear_tf': [True, False],
#     'tfidf__ngram_range': [(1, 1), (2, 2)],
    # including the list of stopwords that was defined earlier
    'tfidf__stop_words': [stopwords_list, new_stopwords, full_stopwords],
}
```

In [245]:
```
# Performing grid search for hyperparameter tuning
grid_search = GridSearchCV(gs_pipeline, parameters, cv=5, scoring='recall', error

# Fitting the tuned pipeline on training data
grid_search.fit(X_train['lemmatized_review'], y_train)
# grid_search.fit(X_train['original_review'], y_train)
```

Out[245]:
```
GridSearchCV(cv=5, error_score='raise',
             estimator=Pipeline(steps=[('tfidf', TfidfVectorizer()),
                                       ('us',
                                        RandomUnderSampler(random_state=42,
                                                           sampling_strategy=
1)),
                                       ('classifier', MultinomialNB())]),
             param_grid={'tfidf__max_df': [0.7, 0.9],
                         'tfidf__max_features': [100, None],
                         'tfidf__min_df': [1, 3],
                         'tfidf__stop_words': [['i', 'me', 'my', 'myself', 'w
e',
                                                'our', 'ours', 'ourse...
                                                'itself', ...],
                                               ['hotel', 'room', 'night', 'da
y',
                                                'stay', 'resort', 'place'],
                                               ['i', 'me', 'my', 'myself', 'w
e',
                                                'our', 'ours', 'ourselves',
                                                'you', "you're", "you've",
                                                "you'll", "you'd", 'your',
                                                'yours', 'yourself',
                                                'yourselves', 'he', 'him',
                                                'his', 'himself', 'she',
                                                "she's", 'her', 'hers',
                                                'herself', 'it', "it's", 'its',
                                                'itself', ...]],
                         'tfidf__sublinear_tf': [True, False]},
             scoring='recall')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [246]:
```
# Recording the best parameters and printing them
best_tfidf_params = grid_search.best_params_

# Recording the best estimator as the best_pipeline
best_pipeline = grid_search.best_estimator_

print("Best Parameters:", best_tfidf_params)
```

```
Best Parameters: {'tfidf__max_df': 0.7, 'tfidf__max_features': None, 'tfidf__mi
n_df': 1, 'tfidf__stop_words': ['hotel', 'room', 'night', 'day', 'stay', 'resor
t', 'place'], 'tfidf__sublinear_tf': True}
```

In [247]:
```
# Fitting the best pipeline on training data
best_pipeline.fit(X_train['lemmatized_review'], y_train)

# Fitting the best model on the full training data
best_gs_y_pred = best_pipeline.predict(X_test['lemmatized_review'])
```

2) Evaluation Metrics

In [248]:
```
# Naming the model
gs_model_name = 'NaiveBayes GS'

# Calling the function and recording into the defined values
accuracy_gs, f1_gs, recall_gs, cv_gs = evaluation_metrics(
    y_test,
    best_gs_y_pred,
    best_pipeline,
    X_train['lemmatized_review'],
    y_train)
```

```
Accuracy: 0.8274
F1-Score: 0.8349
Recall: 0.8274
Mean Cross-Validated Accuracy: 0.8370
```

In [249]:
```
# Comparing the evaluation metrics between undersampled model (best model) and no
metrics_comparison('Accuracy', accuracy_fullstop, accuracy_gs)
metrics_comparison('F1', f1_fullstop, f1_gs)
metrics_comparison('Recall', recall_fullstop, recall_gs)
metrics_comparison('Cross-validated Accuracy', cv_fullstop, cv_gs)
```

```
Accuracy decreased from 0.8345 to 0.8274.
F1 decreased from 0.8405 to 0.8349.
Recall decreased from 0.8345 to 0.8274.
Cross-validated Accuracy decreased from 0.8481 to 0.8370.
```

In [250]:
```
# Printing the overall summary
print('Tuning hyperparameters for recall did not increase overall results.')
```

```
Tuning hyperparameters for recall did not increase overall results.
```

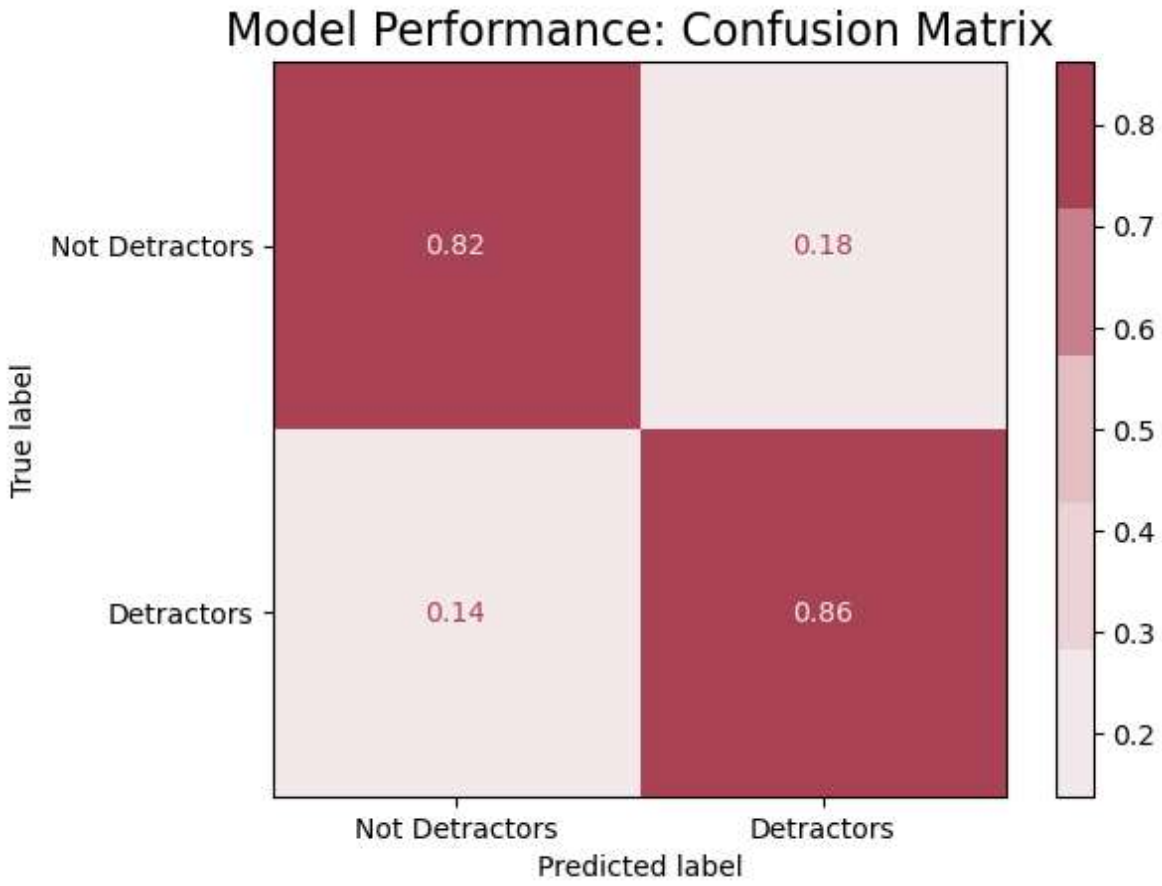
3) Classification Report

In [251]:
```
# Calling the classification report function
tuned_class_report, recall_detractors_gs = class_calculation(y_test, best_gs_y_pr
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.94269   0.81527   0.87436      3773
    Detractors    0.62527   0.86148   0.72461      1350

      accuracy                        0.82744      5123
     macro avg    0.78398   0.83837   0.79949      5123
  weighted avg    0.85904   0.82744   0.83490      5123
```

4) Confusion Matrix

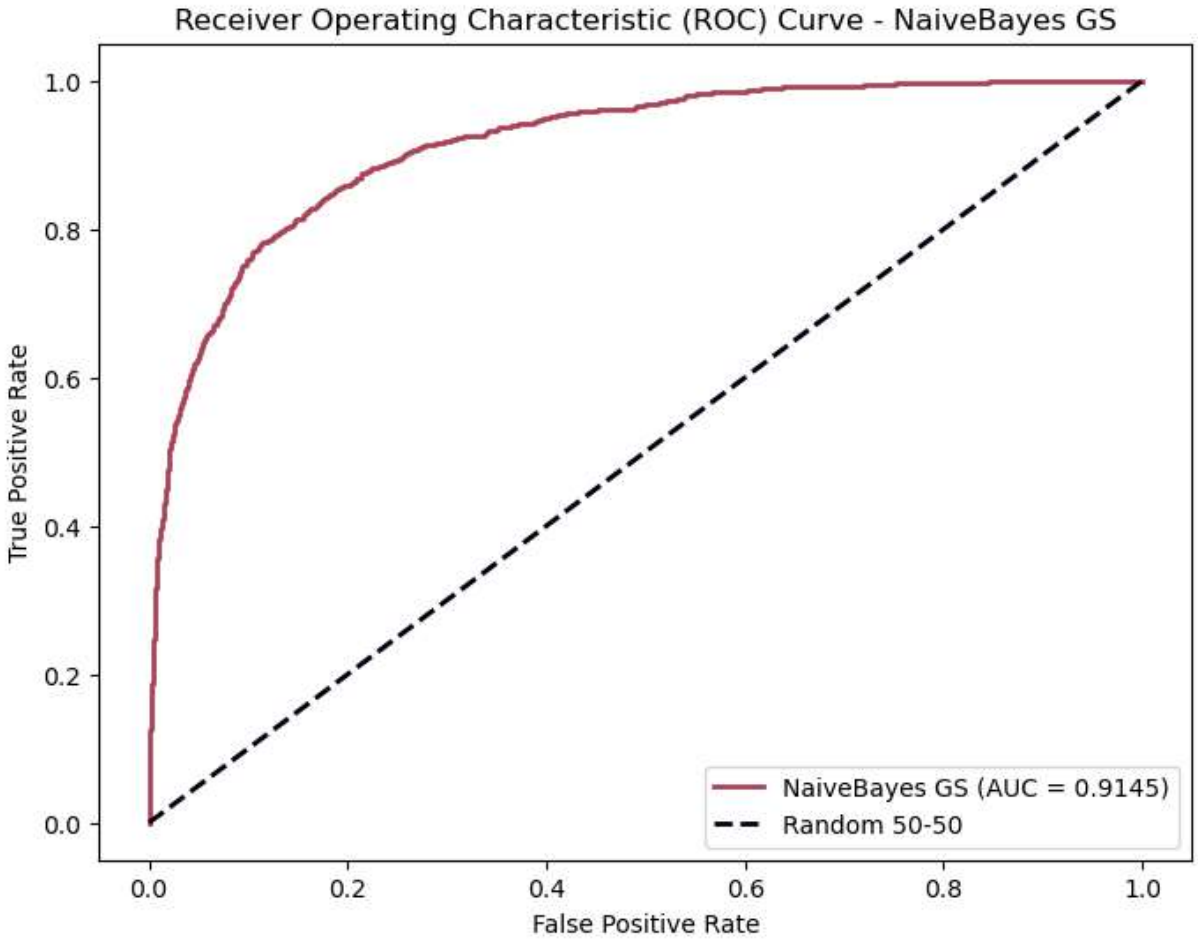
In [252]:
```
# Plotting the confusion matrix
gs_cfn_matrix = confusion_matrix_display(best_pipeline, y_test, best_gs_y_pred)
```



Recall for detractors was the highest ever reached despite lower overall results. This is the best model so far.

5) ROC Curve and AUC

In [253]:
```
# Plotting the ROC curve for the baseline model
gs_y_pred_proba = best_pipeline.predict_proba(X_test['original_review'])[:, 1]
auc_gs = plot_roc_curve(y_test, gs_y_pred_proba, model_name=gs_model_name)
```



In [254]:
```
# Printing the interpretation
auc_interpretation(auc_gs)
```

```
AUC = 0.9145. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [255]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_pp, auc_gs)
print()


# # Interpreting AUC:
# print('Compared to the best model so far: ')
# metrics_comparison('AUC', auc_resamp1, auc_prep)
```

```
Compared to the last run model:
AUC decreased from 0.9174 to 0.9145.
```

Every time the true positive rate decreases for not_detractors, AUC decreases slighlty as well.

In [256]:
```
# Appending each metric to the lists
modelnames.append(gs_model_name)
accuracies.append(accuracy_gs)
f1s.append(f1_gs)
recalls.append(recall_gs)
recalls_detractors.append(recall_detractors_gs)
cv_accs.append(cv_gs)
roc_aucs.append(auc_gs)
```

5. d) TfidfVectorizer and K-Nearest Neighbor

10th iteration: K-Nearest Neighbor Tfidf Vectorizer

```
1) Fitting and training on train data
```

The previous model was a bit computationally expensive. Let's see if the simpler K-Nearest Neighbor classifier would improve on that end. Nevertheless, kNN makes predictions based on what similar cases around it suggest so there is a risk it captures more noise created by the imbalanced dataset, despite the undersampled not_detractors reviews.

Since the grid search model with Naive Bayes recorded the best results, the iterations will be made on this base.

For higher computing performance, the best parameters recorded on the vectorizer with Multinomial Naive Bayes will be kept. Only the classifier will be modified. Let's see if, by using the best TFIDF parameters with another classifier, I can improve further these predictions.

In [257]:
```
# Verifying the best vectorizer parameters
print(best_tfidf_params)
```

```
{'tfidf__max_df': 0.7, 'tfidf__max_features': None, 'tfidf__min_df': 1, 'tfidf_
_stop_words': ['hotel', 'room', 'night', 'day', 'stay', 'resort', 'place'], 'tf
idf__sublinear_tf': True}
```

The parameters were transformed to be used in the vectorizer but they are not needed at this stage.

In [258]:
```
# Creating a function that modifies the parameters, to allow them to be used for
def transform_params(best_params):
    new_best_params = {}
    for key, value in best_params.items():
        # Removing 'tfidf__' from the key
        new_key = key.replace('tfidf__', '')
        new_best_params[new_key] = value
    return new_best_params
```

In [259]:
```
# Calling the newly defined parameters
# new_best_tfidf_params = transform_params(best_tfidf_params)

# # Inspecting them
# print(new_best_tfidf_params)
```

In [260]:
```
from sklearn.neighbors import KNeighborsClassifier

# Defining the pipeline with the fixed tfidf parameters and RandomForestClassifie
knn_pipeline = imblearn.pipeline.Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=new_stopwords)),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', KNeighborsClassifier())
])

# Fitting the pipeline on training data
knn_pipeline.fit(X_train['lemmatized_review'], y_train)

# Making predictions on test data
knn_y_pred = knn_pipeline.predict(X_test['lemmatized_review'])
```

2) Evaluation Metrics

In [261]:
```
# Naming the model and calling the function to evaluate it
knn_model_name = 'kNN'


# Calling the function and recording into the defined values
accuracy_knn, f1_knn, recall_knn, cv_knn = evaluation_metrics(
    y_test,
    knn_y_pred,
    knn_pipeline,
    X_train['lemmatized_review'],
    y_train
)
```

```
Accuracy: 0.7451
F1-Score: 0.7569
Recall: 0.7451
Mean Cross-Validated Accuracy: 0.7564
```

In [262]:
```
# Comparing the evaluation metrics between undersampled model (best model) and nc
metrics_comparison('Accuracy', accuracy_gs, accuracy_knn)
metrics_comparison('F1', f1_gs, f1_knn)
metrics_comparison('recall', recall_gs, recall_knn)
metrics_comparison('Cross-validated Accuracy', cv_gs, cv_knn)
```

```
Accuracy decreased from 0.8274 to 0.7451.
F1 decreased from 0.8349 to 0.7569.
recall decreased from 0.8274 to 0.7451.
Cross-validated Accuracy decreased from 0.8370 to 0.7564.
```

In [263]:
```
# Printing the overall summary
print('The metrics recorded from kNN are so far behind what was recorded, no iter
```

```
The metrics recorded from kNN are so far behind what was recorded, no iteration
s will be built from this model.
```

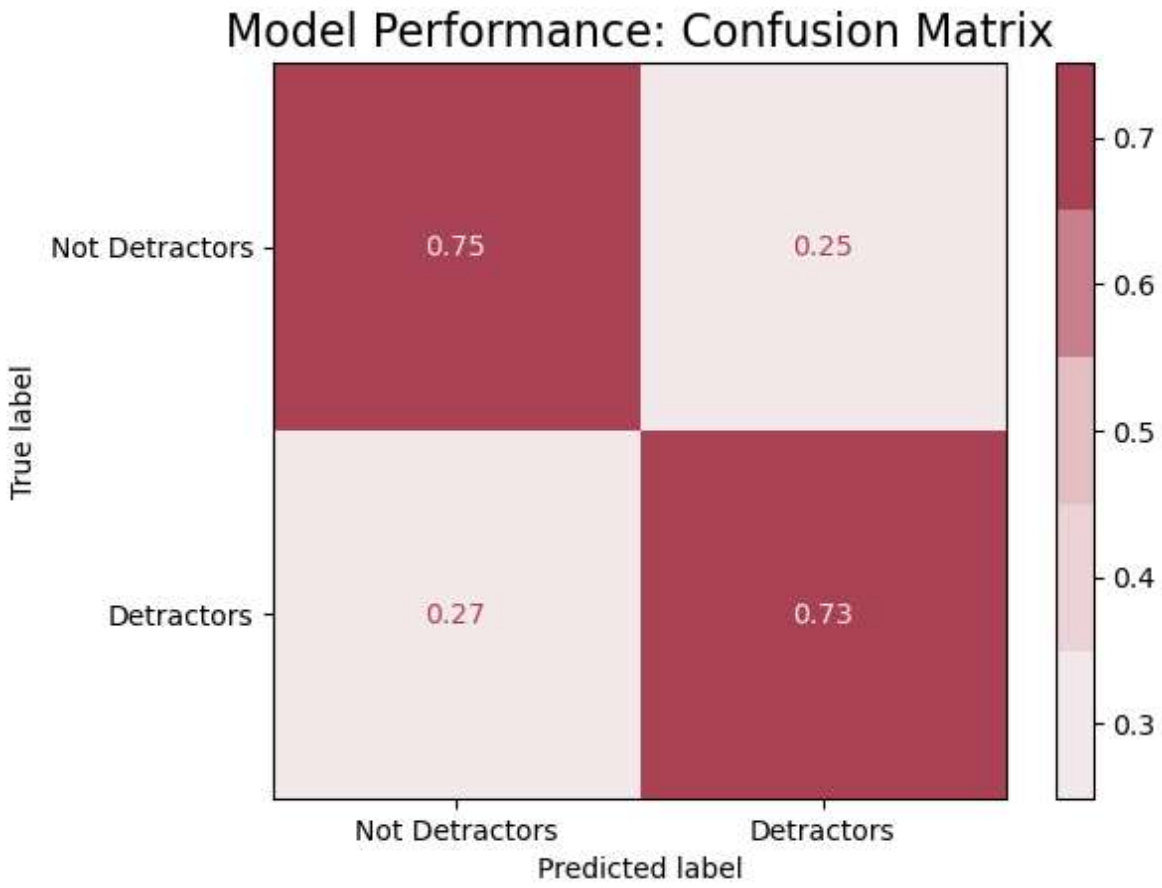
3) Classification Report

In [264]:
```
# Calling the function
knn_class_report, recall_detractors_knn = class_calculation(y_test, knn_y_pred)
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.88535   0.75113   0.81273      3773
    Detractors    0.51145   0.72815   0.60086      1350

      accuracy                        0.74507      5123
     macro avg    0.69840   0.73964   0.70679      5123
  weighted avg    0.78682   0.74507   0.75690      5123
```

4) Confusion Matrix

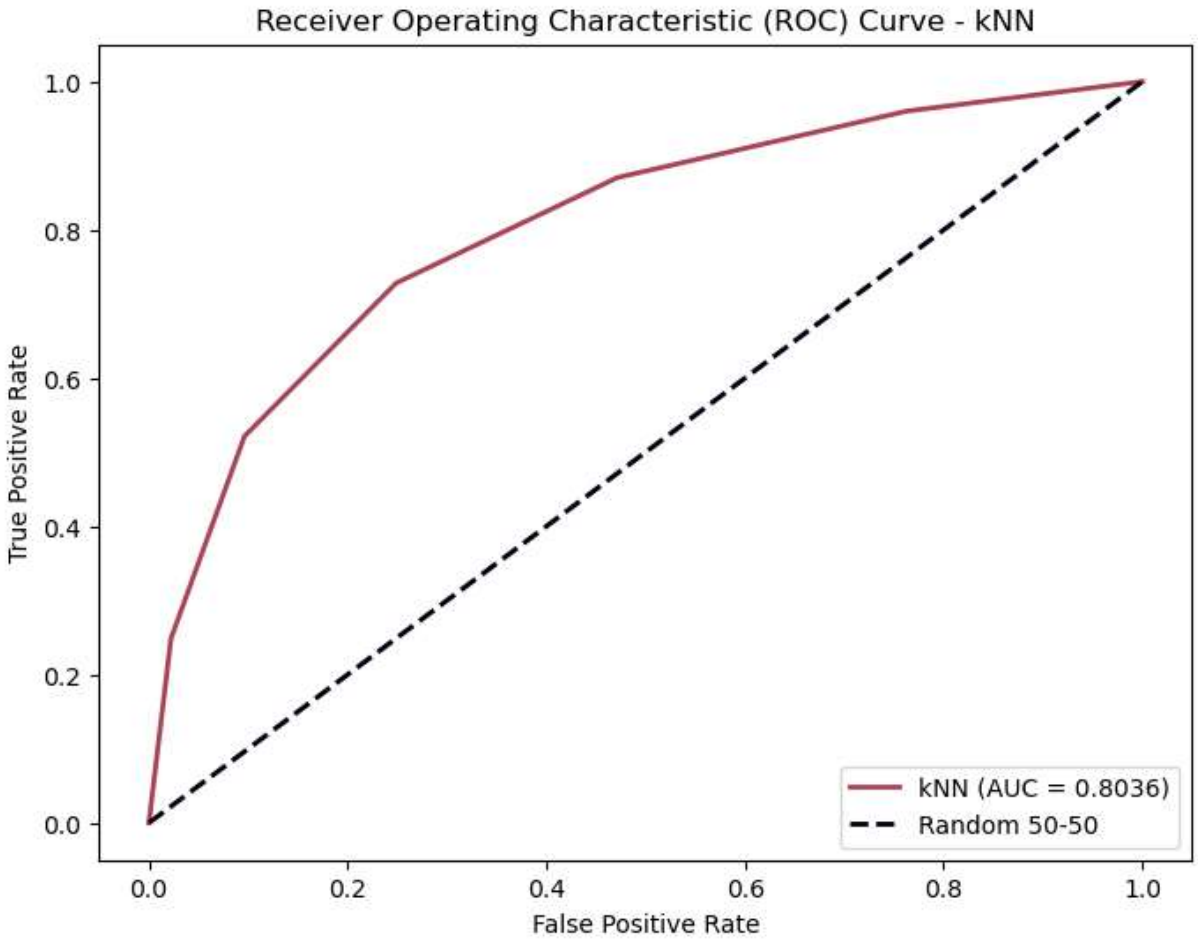
In [265]:
```
# Recording and displaying the confusion matrix
knn_confusion_matrix = confusion_matrix_display(knn_pipeline, y_test, knn_y_pred)
```



K-Nearest Neighbor classifier had a negative impact on the model. There is no point in trying to tune it.

5) ROC Curve and AUC

In [266]:
```
# Plotting the ROC curve for the baseline model
knn_y_pred_proba = knn_pipeline.predict_proba(X_test['lemmatized_review'])[:, 1]
auc_knn = plot_roc_curve(y_test, knn_y_pred_proba, model_name=knn_model_name)
```



In [267]:
```
# Printing the interpretation
auc_interpretation(auc_knn)
```

```
AUC = 0.8036. The model has a good predictive power.
```

In [268]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_gs, auc_knn)
print()
```

```
Compared to the last run model:
AUC decreased from 0.9145 to 0.8036.
```

In [269]:
```
# Appending each metric to the lists
modelnames.append(knn_model_name)
accuracies.append(accuracy_knn)
f1s.append(f1_knn)
recalls.append(recall_knn)
recalls_detractors.append(recall_detractors_knn)
cv_accs.append(cv_knn)
roc_aucs.append(auc_knn)
```

5. c) TfidfVectorizer and Decision Trees

11th iteration: Decision Trees and Tfidf Vectorizer

Decision trees work well for understanding language because they are easy to interpret and handle the nuances in how words relate. They are good at understanding what words matter most and can deal with different types of word data without much difficulty.

1) Fitting and training train data

In [270]:
```
# Importing the relevant packages
from sklearn.tree import DecisionTreeClassifier

# Defining the pipeline with new classifier, but the same best parameters
dt_pipeline = imblearn.pipeline.Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=new_stopwords)),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', DecisionTreeClassifier())
])

# Fitting the pipeline on training data
dt_pipeline.fit(X_train['lemmatized_review'], y_train)

# Making predictions on test data
dt_y_pred = dt_pipeline.predict(X_test['lemmatized_review'])
```

2) Evaluation Metrics

In [271]:
```
# Naming the model
dt_model_name = 'DecisionTree'


# Calling the function and recording into the defined values
accuracy_dt, f1_dt, recall_dt, cv_dt = evaluation_metrics(
    y_test,
    dt_y_pred,
    dt_pipeline,
    X_train['lemmatized_review'],
    y_train)
```

```
Accuracy: 0.7293
F1-Score: 0.7421
Recall: 0.7293
Mean Cross-Validated Accuracy: 0.7244
```

In [272]:
```
# Comparing the evaluation metrics between undersampled model (best model) and nc
metrics_comparison('Accuracy', accuracy_knn, accuracy_dt)
metrics_comparison('F1', f1_knn, f1_dt)
metrics_comparison('Recall', recall_knn, recall_dt)
metrics_comparison('Cross-validated Accuracy', cv_knn, cv_dt)
```

```
Accuracy decreased from 0.7451 to 0.7293.
F1 decreased from 0.7569 to 0.7421.
Recall decreased from 0.7451 to 0.7293.
Cross-validated Accuracy decreased from 0.7564 to 0.7244.
```

In [273]:
```
# Printing the overall summary
print('Decision tree produced results that were even lower than kNN. No iteratior
```

```
Decision tree produced results that were even lower than kNN. No iterations wil
l be built from this.
```

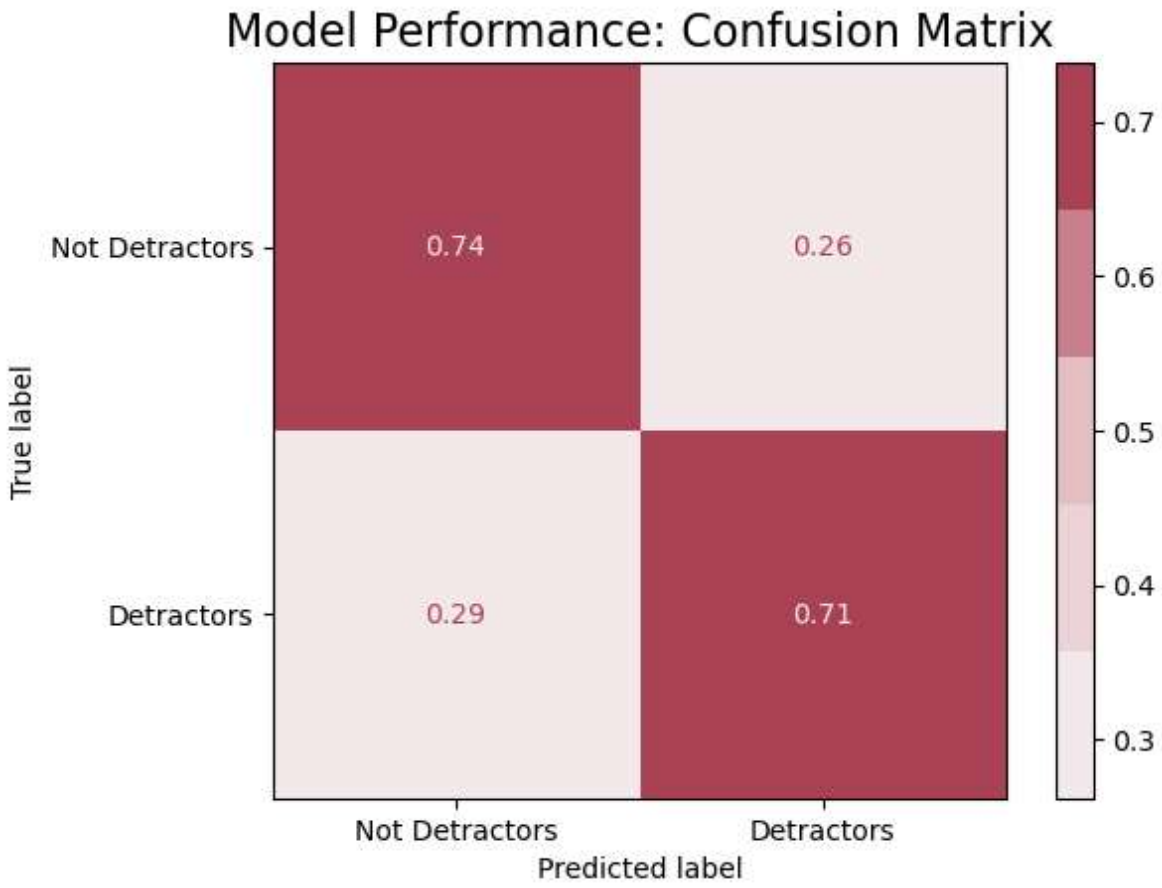
3) Classification Report

In [274]:
```
# Calling the classification report function
class_report_dt, recall_detractors_dt = class_calculation(y_test, dt_y_pred)
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.87492   0.73787   0.80058      3773
    Detractors    0.49047   0.70519   0.57855      1350

      accuracy                        0.72926      5123
     macro avg    0.68270   0.72153   0.68956      5123
  weighted avg    0.77361   0.72926   0.74207      5123
```

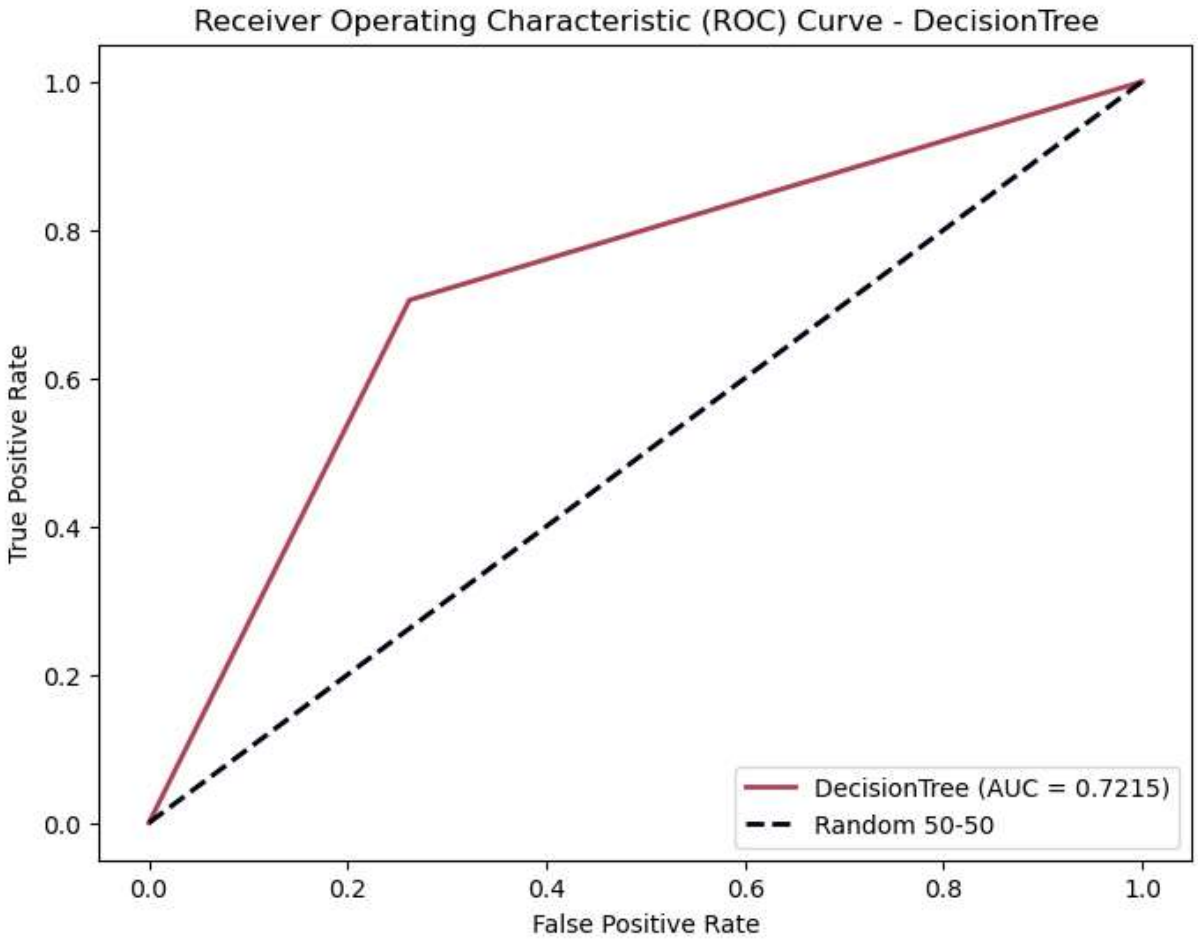
4) Confusion Matrix

In [275]:
```
# Recording and displaying the confusion matrix
dt_cfn_matrix = confusion_matrix_display(dt_pipeline, y_test, dt_y_pred)
```



5) ROC Curve and AUC

In [276]:
```
# Plotting the ROC curve for the baseline model
dt_y_pred_proba = dt_pipeline.predict_proba(X_test['lemmatized_review'])[:, 1]
auc_dt = plot_roc_curve(y_test, dt_y_pred_proba, model_name=dt_model_name)
```



In [277]:
```
# Printing the interpretation
auc_interpretation(auc_dt)
```

```
AUC = 0.7215. The model has some predictive power.
```

In [278]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_knn, auc_dt)
print()


# # Interpreting AUC:
# print('Compared to the best model so far: ')
# metrics_comparison('AUC', auc_resamp1, auc_prep)
```

```
Compared to the last run model:
AUC decreased from 0.8036 to 0.7215.
```

In [279]:
```
# Appending each metric to the lists
modelnames.append(dt_model_name)
accuracies.append(accuracy_dt)
f1s.append(f1_dt)
recalls.append(recall_dt)
recalls_detractors.append(recall_detractors_dt)
cv_accs.append(cv_dt)
roc_aucs.append(auc_dt)
```

5. d) TfidfVectorizer and Random Forest

12th iteration: RandomForestClassifierTuning Tfidf Vectorizer

1) Fitting and training on train data

Random Forest classifiers can be thought of as an extension of multiple decision trees working together together to understand language text.Let's see if, by using the industry stopwords and lemmatized reviews with another classifier, I can improve further these predictions.

In [280]:
```
# Importing the relevant package
from sklearn.ensemble import RandomForestClassifier

# Defining the pipeline with the fixed tfidf parameters and RandomForestClassifie
rf_pipeline = imblearn.pipeline.Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=new_stopwords)),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', RandomForestClassifier())
])

# Fitting the pipeline on training data
rf_pipeline.fit(X_train['lemmatized_review'], y_train)

# Making predictions on test data
rf_y_pred = rf_pipeline.predict(X_test['lemmatized_review'])
```

2) Evaluation Metrics

In [281]:
```
# Naming the model and calling the function to evaluate it
rf_model_name = 'RandomForest'


# Calling the function and recording into the defined values
accuracy_rf, f1_rf, recall_rf, cv_rf = evaluation_metrics(
    y_test,
    rf_y_pred,
    rf_pipeline,
    X_train['lemmatized_review'],
    y_train
)
```

```
Accuracy: 0.8366
F1-Score: 0.8421
Recall: 0.8366
Mean Cross-Validated Accuracy: 0.8507
```

In [282]:
```
# Comparing the evaluation metrics between the last model and the new one
metrics_comparison('Accuracy', accuracy_dt, accuracy_rf)
metrics_comparison('F1', f1_dt, f1_rf)
metrics_comparison('Recall', recall_dt, recall_rf)
metrics_comparison('Cross-validated Accuracy', cv_dt, cv_rf)
```

```
Accuracy improved from 0.7293 to 0.8366.
F1 improved from 0.7421 to 0.8421.
Recall improved from 0.7293 to 0.8366.
Cross-validated Accuracy improved from 0.7244 to 0.8507.
```

In [283]:
```
# Printing the overall summary
print('Random Forest finally started to improve results at similar levels than Na
```

```
Random Forest finally started to improve results at similar levels than NaiveBa
yes.
```

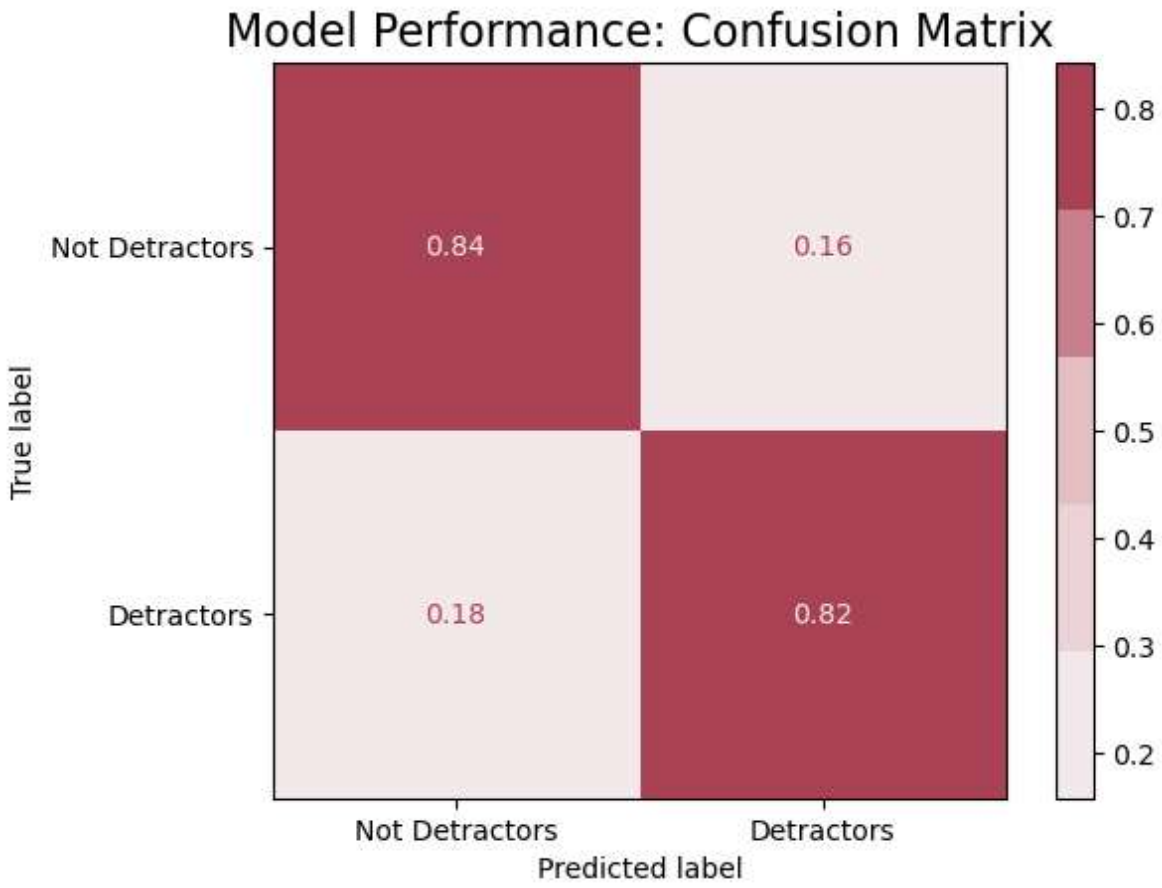
3) Classification Report

In [284]:
```
# Calling the classification report function
rf_class_report, recall_detractors_rf = class_calculation(y_test, rf_y_pred)
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.92949   0.84204   0.88360      3773
    Detractors    0.65044   0.82148   0.72602      1350

      accuracy                        0.83662      5123
     macro avg    0.78997   0.83176   0.80481      5123
  weighted avg    0.85596   0.83662   0.84208      5123
```

4) Confusion Matrix

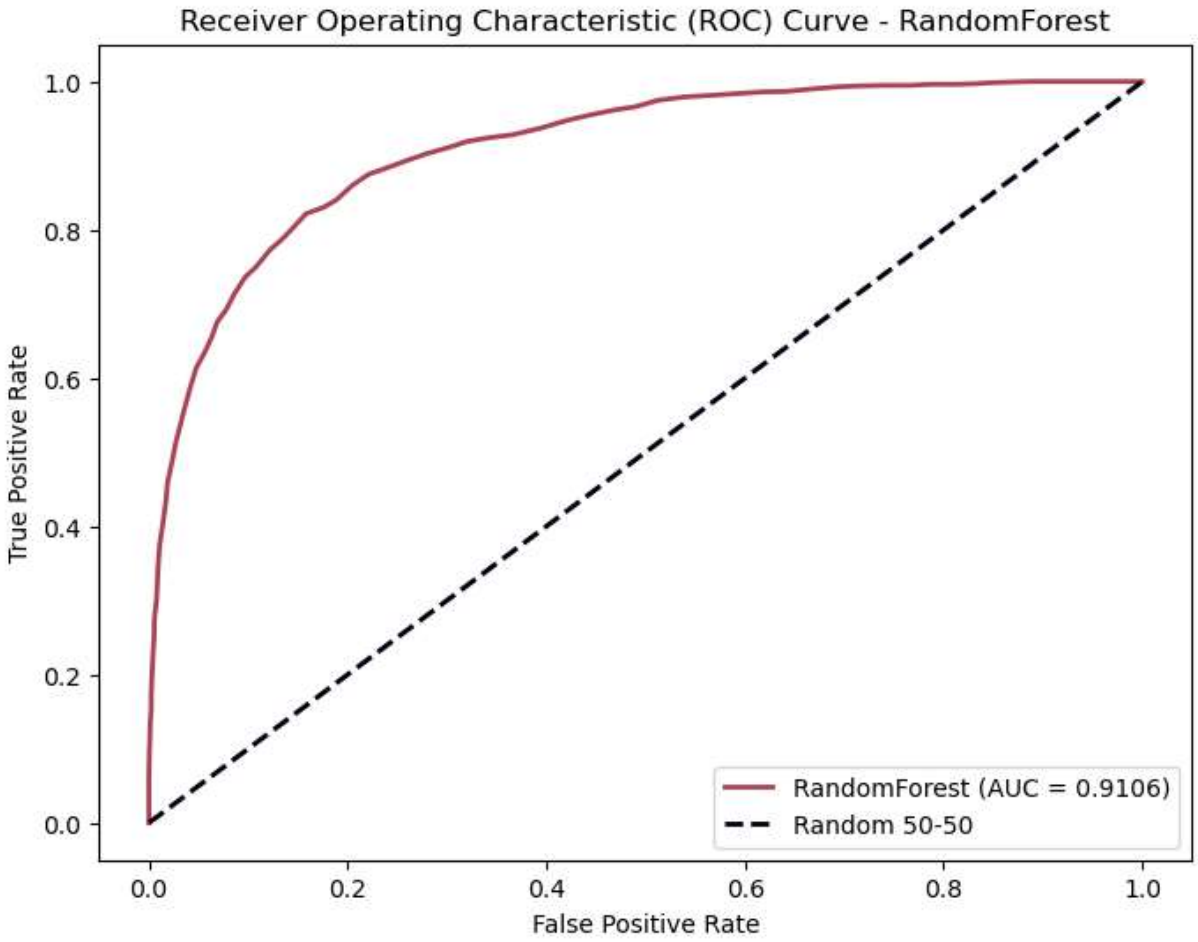
In [285]:
```
# Recording and displaying the confusion matrix
rf_confusion_matrix = confusion_matrix_display(rf_pipeline, y_test, rf_y_pred)
```



The results are now starting to be comparable with the ones from Naive Bayes, but recall for detractors remain lower than the best recorded results.

5) ROC Curve and AUC

In [286]:
```
# Plotting the ROC curve for the baseline model
rf_y_pred_proba = rf_pipeline.predict_proba(X_test['lemmatized_review'])[:, 1]
auc_rf = plot_roc_curve(y_test, rf_y_pred_proba, model_name=rf_model_name)
```



In [287]:
```
# Printing the interpretation
auc_interpretation(auc_rf)
```

```
AUC = 0.9106. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [288]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_dt, auc_rf)
print()


# # Interpreting AUC:
# print('Compared to the best model so far: ')
# metrics_comparison('AUC', auc_resamp1, auc_prep)
```

```
Compared to the last run model:
AUC improved from 0.7215 to 0.9106.
```

In [289]:
```
# Appending each metric to the lists
modelnames.append(rf_model_name)
accuracies.append(accuracy_rf)
f1s.append(f1_rf)
recalls.append(recall_rf)
recalls_detractors.append(recall_detractors_rf)
cv_accs.append(cv_rf)
roc_aucs.append(auc_rf)
```

5. e) TfidfVectorizer and Gradient Boosting

13th iteration: Gradient Boosting Tfidf Vectorizer

Since RandomForest, an ensemble of Decision Trees, performed better than Decision Trees, I will try to take these models to the next levels with Gradient Boosting.

Gradient boosing is a class of ensemble algorithns constructed from decision tree models. Trees are added consecutively and are modified to rectify the errors in prediction made by the models which ran prior. This is why they are called `boosting` .

1) Fitting and training on train data

In [290]:
```
# Importing the relevant packages
from sklearn.ensemble import GradientBoostingClassifier


# Defining the pipeline with the fixed tfidf parameters and GradientBoostingClass
gb_pipeline = imblearn.pipeline.Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=new_stopwords)),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', GradientBoostingClassifier())
])

# Fitting the pipeline on training data
gb_pipeline.fit(X_train['lemmatized_review'], y_train)

# Making predictions on test data
gb_y_pred = gb_pipeline.predict(X_test['lemmatized_review'])
```

2) Evaluation Metrics

In [291]:
```
# Naming the model and calling the function to evaluate it
gb_model_name = 'GradientBoosting'


# Calling the function and recording into the defined values
accuracy_gb, f1_gb, recall_gb, cv_gb = evaluation_metrics(
    y_test,
    gb_y_pred,
    gb_pipeline,
    X_train['lemmatized_review'],
    y_train
)
```

```
Accuracy: 0.8251
F1-Score: 0.8313
Recall: 0.8251
Mean Cross-Validated Accuracy: 0.8373
```

In [292]:
```
# Comparing the evaluation metrics
metrics_comparison('Accuracy', accuracy_rf, accuracy_gb)
metrics_comparison('F1', f1_rf, f1_gb)
metrics_comparison('Recall', recall_rf, recall_gb)
metrics_comparison('Cross-validated Accuracy', cv_rf, cv_gb)
```

```
Accuracy decreased from 0.8366 to 0.8251.
F1 decreased from 0.8421 to 0.8313.
Recall decreased from 0.8366 to 0.8251.
Cross-validated Accuracy decreased from 0.8507 to 0.8373.
```

In [293]:
```
# Printing the overall summary
print('Random Forest actually produced better results than Gradient Boosting.')
```

```
Random Forest actually produced better results than Gradient Boosting.
```

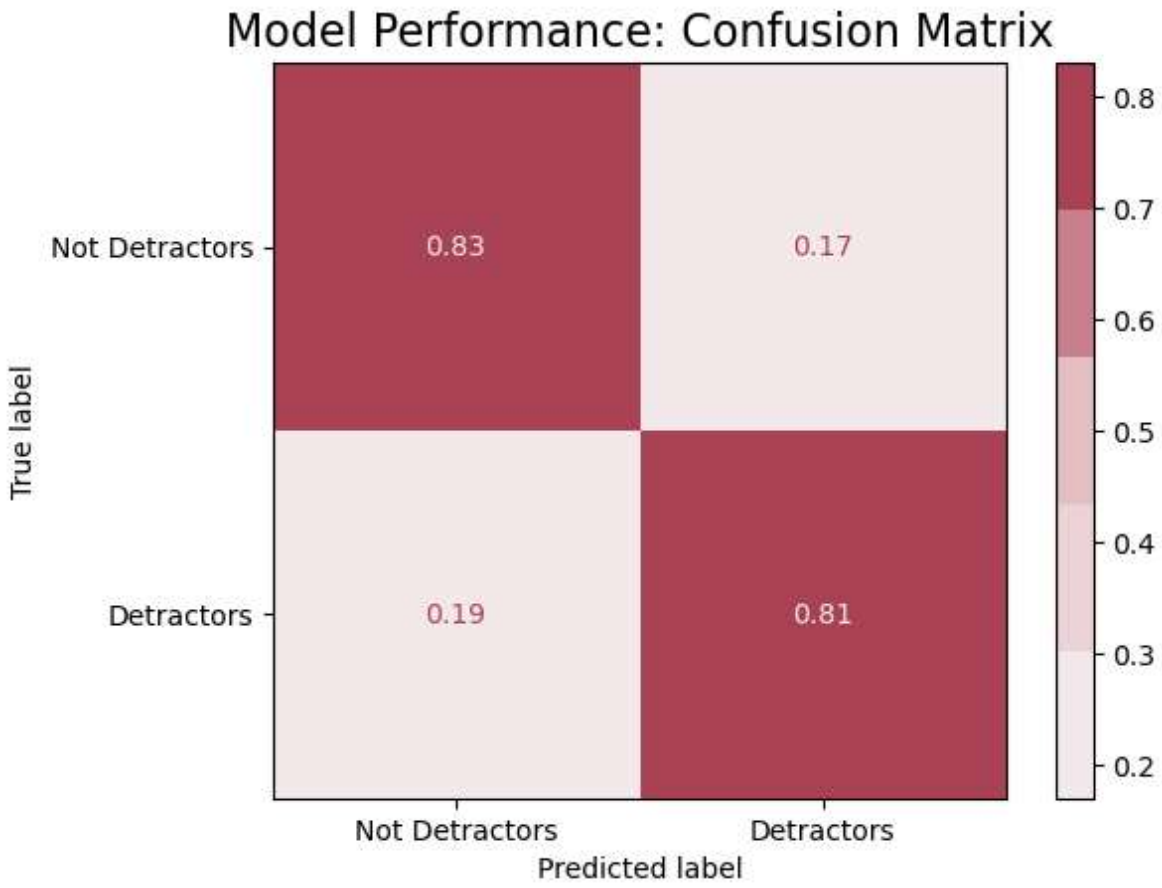
3) Classification Report

In [294]:
```
# Calling the function
gb_class_report, recall_detractors_gb = class_calculation(y_test, gb_y_pred)
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.92471   0.83011   0.87486      3773
    Detractors    0.63076   0.81111   0.70966      1350

      accuracy                        0.82510      5123
     macro avg    0.77774   0.82061   0.79226      5123
  weighted avg    0.84725   0.82510   0.83133      5123
```

4) Confusion Matrix

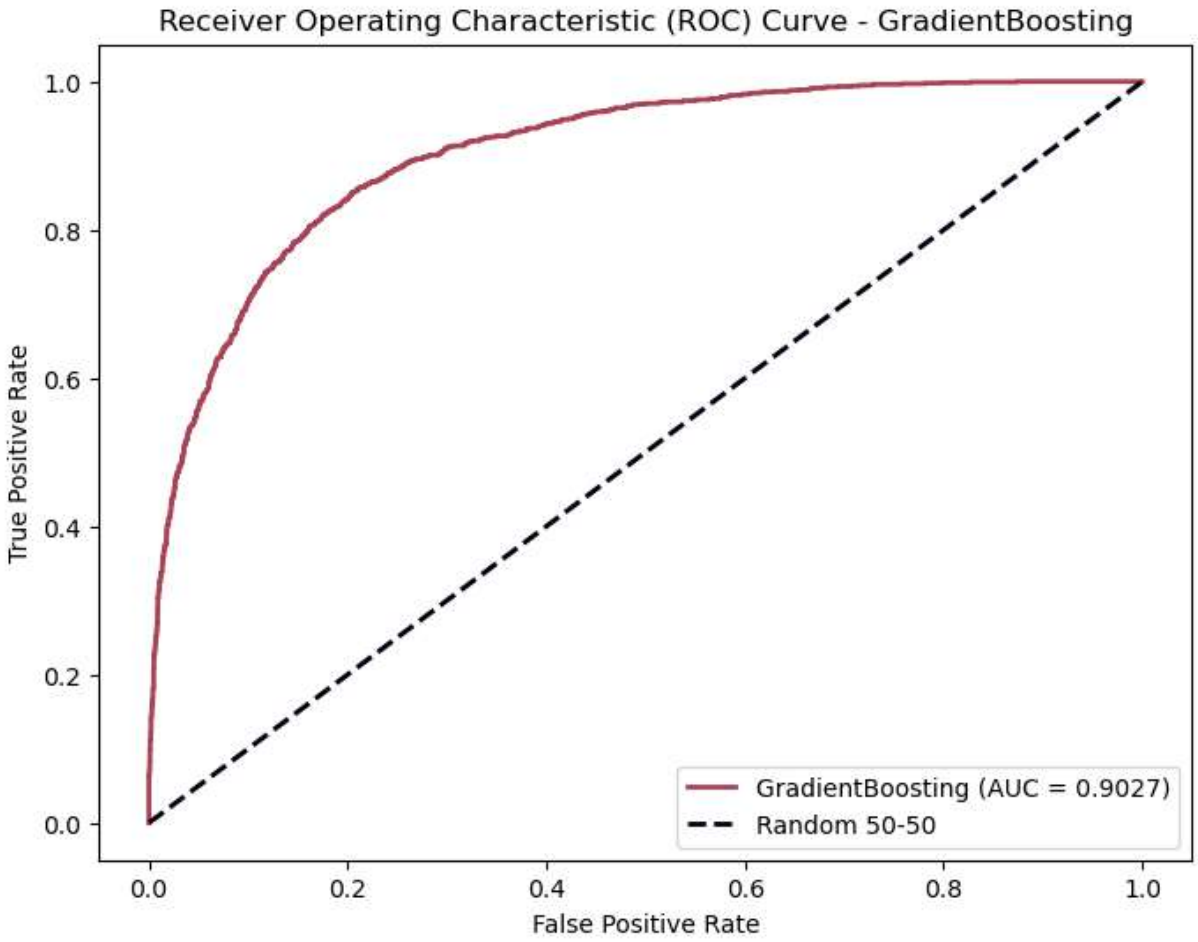
In [295]:
```
# Recording and displaying the confusion matrix
gb_confusion_matrix = confusion_matrix_display(gb_pipeline, y_test, gb_y_pred)
```



Even the detractors class got a lower true positive rate with Gradient Boosting than Random Forest.

5) ROC Curve and AUC

In [296]:
```
# Plotting the ROC curve for the baseline model
gb_y_pred_proba = gb_pipeline.predict_proba(X_test['lemmatized_review'])[:, 1]
auc_gb = plot_roc_curve(y_test, gb_y_pred_proba, model_name=gb_model_name)
```



In [297]:
```
# Printing the interpretation
auc_interpretation(auc_gb)
```

```
AUC = 0.9027. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [298]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_rf, auc_gb)
print()
```

```
Compared to the last run model:
AUC decreased from 0.9106 to 0.9027.
```

In [299]:
```
# Appending each metric to the lists
modelnames.append(gb_model_name)
accuracies.append(accuracy_gb)
f1s.append(f1_gb)
recalls.append(recall_gb)
recalls_detractors.append(recall_detractors_gb)
cv_accs.append(cv_gb)
roc_aucs.append(auc_gb)
```

Since Gradient Boosting did not provide better results than Random Forest, I will try another ensemble classifier.

5. f) TfidfVectorizer and AdaBoost

14th iteration: AdaBoost Tfidf Vectorizer

While Gradient Boosting is often considered more flexible and powerful, these iterations have proven to provide surprising results. AdaBoost focuses on improving the classification of the instances that are more challenging for the current ensemble.

1) Fitting and training on train data

In [300]:
```
# Importing the relevant packages
from sklearn.ensemble import AdaBoostClassifier


# Defining the pipeline with the fixed tfidf parameters and GradientBoostingClass
ab_pipeline = imblearn.pipeline.Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=new_stopwords)),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', AdaBoostClassifier())
])

# Fitting the pipeline on training data
ab_pipeline.fit(X_train['lemmatized_review'], y_train)

# Making predictions on test data
ab_y_pred = ab_pipeline.predict(X_test['lemmatized_review'])
```

2) Evaluation Metrics

In [301]:
```
# Naming the model and calling the function to evaluate it
ab_model_name = 'AdaBoost'


# Calling the function and recording into the defined values
accuracy_ab, f1_ab, recall_ab, cv_ab = evaluation_metrics(
    y_test,
    ab_y_pred,
    ab_pipeline,
    X_train['lemmatized_review'],
    y_train
)
```

```
Accuracy: 0.8150
F1-Score: 0.8221
Recall: 0.8150
Mean Cross-Validated Accuracy: 0.8276
```

In [302]:
```
# Comparing the evaluation metrics between undersampled model (best model) and nc
metrics_comparison('Accuracy', accuracy_gb, accuracy_ab)
metrics_comparison('F1', f1_gb, f1_ab)
metrics_comparison('Recall', recall_gb, recall_ab)
metrics_comparison('Cross-validated Accuracy', cv_gb, cv_ab)
```

```
Accuracy decreased from 0.8251 to 0.8150.
F1 decreased from 0.8313 to 0.8221.
Recall decreased from 0.8251 to 0.8150.
Cross-validated Accuracy decreased from 0.8373 to 0.8276.
```

In [303]:
```
# Printing the overall summary
print('AdaBoost did not provide better results than Gradient Boosting.')
```

```
AdaBoost did not provide better results than Gradient Boosting.
```

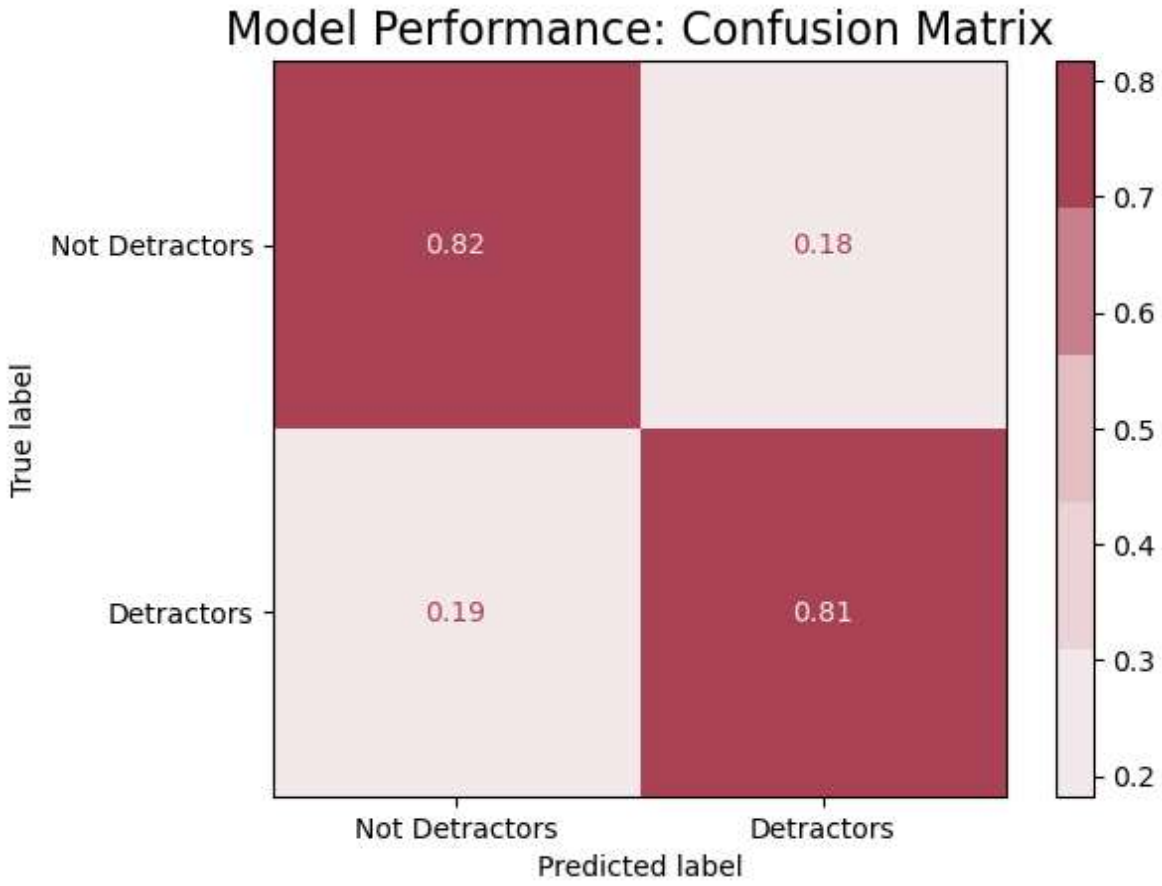
3) Classification Report

In [304]:
```
# Calling the function
ab_class_report, recall_detractors_ab = class_calculation(y_test, ab_y_pred)
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.92278   0.81712   0.86674      3773
    Detractors    0.61279   0.80889   0.69732      1350

      accuracy                        0.81495      5123
     macro avg    0.76779   0.81301   0.78203      5123
  weighted avg    0.84109   0.81495   0.82210      5123
```

4) Confusion Matrix

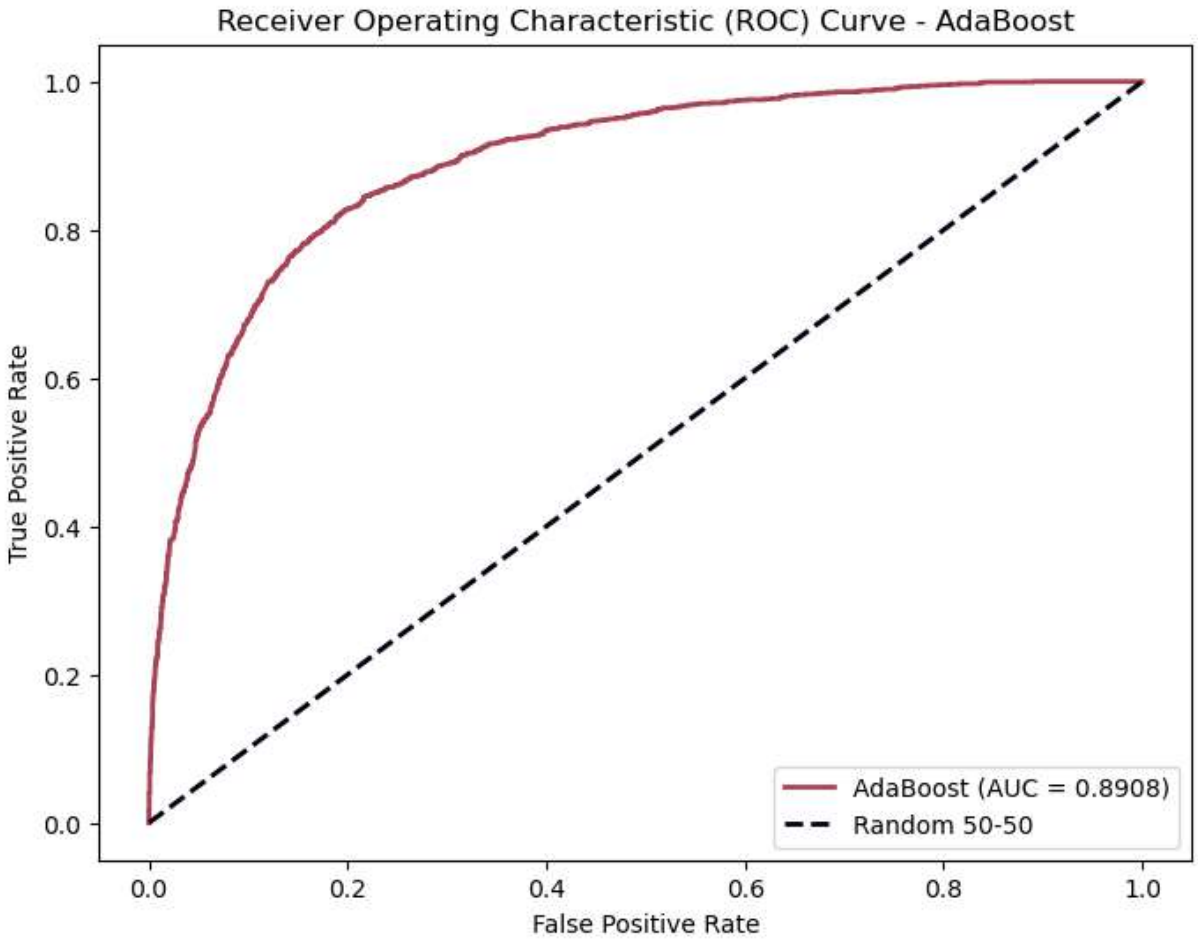
In [305]:
```
# Recording and displaying the confusion matrix
ab_confusion_matrix = confusion_matrix_display(ab_pipeline, y_test, ab_y_pred)
```



The lower results are also reflected on recall for detractors.

5) ROC Curve and AUC

In [306]:
```
# Plotting the ROC curve for the baseline model
ab_y_pred_proba = ab_pipeline.predict_proba(X_test['lemmatized_review'])[:, 1]
auc_ab = plot_roc_curve(y_test, ab_y_pred_proba, model_name=ab_model_name)
```



In [307]:
```
# Printing the interpretation
auc_interpretation(auc_ab)
```

```
AUC = 0.8908. The model has a good predictive power.
```

In [308]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_gb, auc_ab)
print()


# # Interpreting AUC:
# print('Compared to the best model so far: ')
# metrics_comparison('AUC', auc_resamp1, auc_prep)
```

```
Compared to the last run model:
AUC decreased from 0.9027 to 0.8908.
```

In [309]:
```
# Appending each metric to the lists
modelnames.append(ab_model_name)
accuracies.append(accuracy_ab)
f1s.append(f1_ab)
recalls.append(recall_ab)
recalls_detractors.append(recall_detractors_ab)
cv_accs.append(cv_ab)
roc_aucs.append(auc_ab)
```

5. g) TfidfVectorizer and XGBoost

15th iteration: XGBoost Tfidf Vectorizer

Since I am not reaching stronger results than with NaiveBayes with the previous classifiers, I will try XGBoost, for its popularity among classifiers algorithms.

XGBoost, or Extreme Gradient Boosting, is a powerful and efficient machine learning algorithm that belongs to the gradient boosting family, known for its speed, scalability, and ability to handle diverse data types.

1) Fitting and training on train data

In [310]:
```
# Importing the relevant packages
from xgboost import XGBClassifier


# Defining the pipeline with the fixed tfidf parameters and GradientBoostingClass
xgb_pipeline = imblearn.pipeline.Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=new_stopwords)),
    ('us', RandomUnderSampler(sampling_strategy=1, random_state=42)),
    ('classifier', XGBClassifier())  # Change AdaBoostClassifier to XGBClassifier
])

# Fitting the pipeline on training data
xgb_pipeline.fit(X_train['lemmatized_review'], y_train)

# Making predictions on test data
xgb_y_pred = xgb_pipeline.predict(X_test['lemmatized_review'])
```

2) Evaluation Metrics

In [311]:
```
# Naming the model and calling the function to evaluate it
xgb_model_name = 'XGBoost'


# Calling the function and recording into the defined values
accuracy_xgb, f1_xgb, recall_xgb, cv_xgb = evaluation_metrics(
    y_test,
    xgb_y_pred,
    xgb_pipeline,
    X_train['lemmatized_review'],
    y_train
)
```

```
Accuracy: 0.8468
F1-Score: 0.8520
Recall: 0.8468
Mean Cross-Validated Accuracy: 0.8574
```

In [312]:
```
# Comparing the evaluation metrics between the last model the current one
metrics_comparison('Accuracy', accuracy_ab, accuracy_xgb)
metrics_comparison('F1', f1_ab, f1_xgb)
metrics_comparison('Recall', recall_ab, recall_xgb)
metrics_comparison('Cross-validated Accuracy', cv_ab, cv_xgb)
```

```
Accuracy improved from 0.8150 to 0.8468.
F1 improved from 0.8221 to 0.8520.
Recall improved from 0.8150 to 0.8468.
Cross-validated Accuracy improved from 0.8276 to 0.8574.
```

In [313]:
```
# Printing the overall summary
print('XGBoost popularity can be confirmed! Results highly improved.')
```

```
XGBoost popularity can be confirmed! Results highly improved.
```

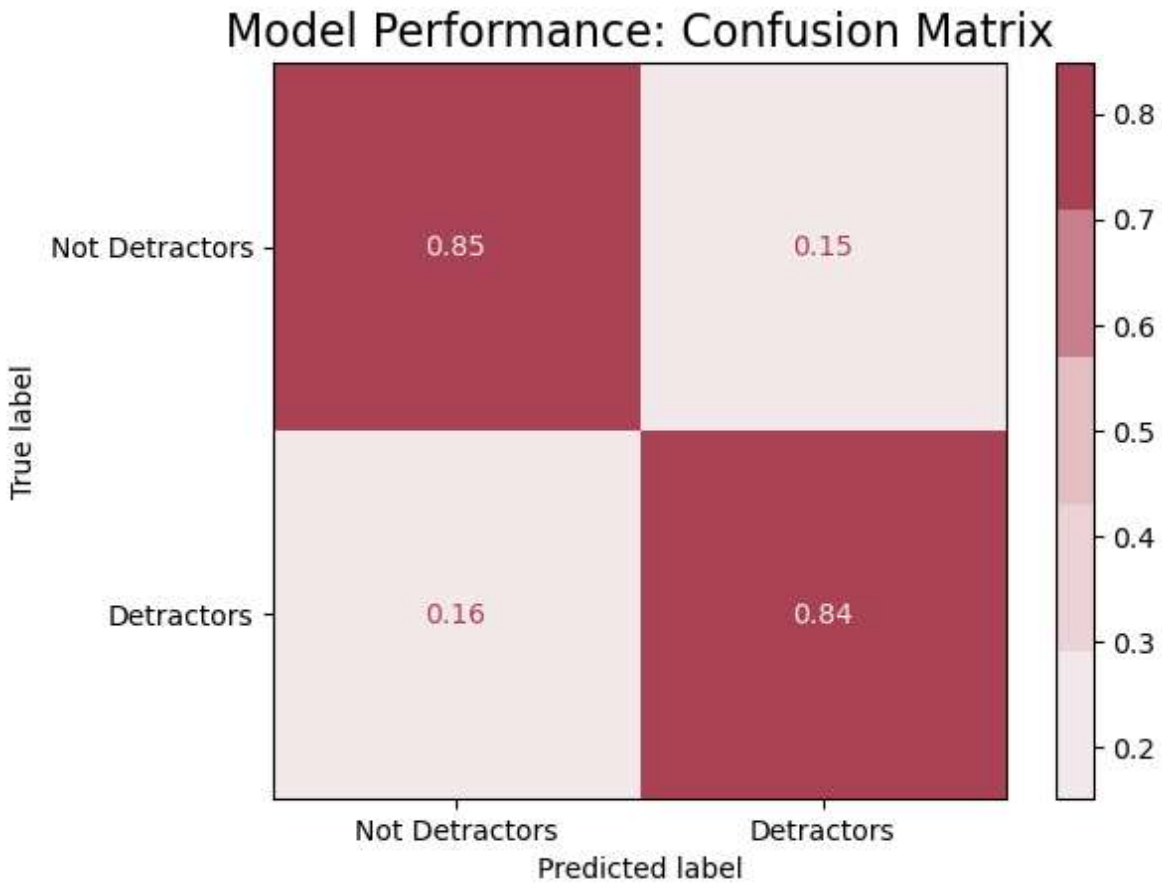
3) Classification Report

In [314]:
```
# Calling the function
xgb_class_report, recall_detractors_xgb = class_calculation(y_test, xgb_y_pred)
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.93812   0.84787   0.89071      3773
    Detractors    0.66492   0.84370   0.74372      1350

      accuracy                        0.84677      5123
     macro avg    0.80152   0.84579   0.81721      5123
  weighted avg    0.86613   0.84677   0.85198      5123
```

4) Confusion Matrix

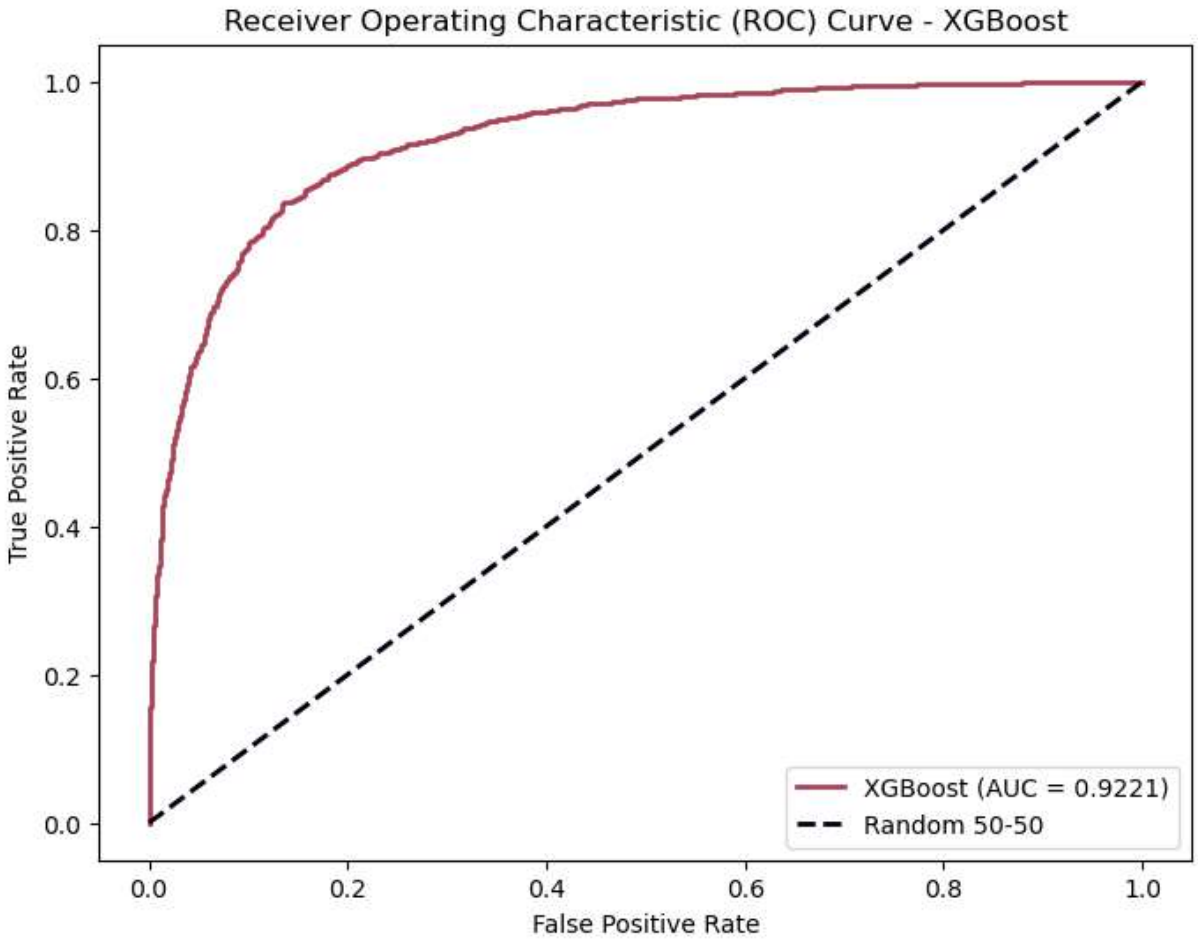
In [315]:
```
# Recording and displaying the confusion matrix
xgb_confusion_matrix = confusion_matrix_display(xgb_pipeline, y_test, xgb_y_pred)
```



Nevertheless, recall for detractors is still not reaching higher results than with the Naive Bayes model.

5) ROC Curve and AUC

In [316]:
```
# Plotting the ROC curve for the baseline model
xgb_y_pred_proba = xgb_pipeline.predict_proba(X_test['lemmatized_review'])[:, 1]
auc_xgb = plot_roc_curve(y_test, xgb_y_pred_proba, model_name=xgb_model_name)
```



In [317]:
```
# Printing the interpretation
auc_interpretation(auc_xgb)
```

```
AUC = 0.9221. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [318]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_ab, auc_xgb)
print()
```

```
Compared to the last run model:
AUC improved from 0.8908 to 0.9221.
```

In [319]:
```
# Appending each metric to the lists
modelnames.append(xgb_model_name)
accuracies.append(accuracy_xgb)
f1s.append(f1_xgb)
recalls.append(recall_xgb)
recalls_detractors.append(recall_detractors_xgb)
cv_accs.append(cv_xgb)
roc_aucs.append(auc_xgb)
```

I finally got another classifier competing with the results provided from Multinomial Naive Bayes. I will try to tune it and find the best parameters to see if these results can be beaten.

16th iteration: Tuned XGBoost

1) Fitting and training on train data

I will try to apply combinatoric grid searching for XGBoost to get the best parameters and try to improve further the model.

In [320]:
```
# Defining the parameter grid
param_grid = {
    'classifier__learning_rate': [0.01, 0.1, 0.2],
    'classifier__n_estimators': [50, 100, 150],
    'classifier__max_depth': [3, 5, 7],
#     'classifier__subsample': [0.8, 0.9, 1.0],
}
```

In [321]:
```
# Creating the GridSearchCV object
grid_search = GridSearchCV(xgb_pipeline, param_grid, cv=5, scoring='recall', n_jo

# Fitting the grid search to the data
grid_search.fit(X_train['lemmatized_review'], y_train)
```

Out[321]:
```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('tfidf',
                                        TfidfVectorizer(stop_words=['hotel',
                                                                    'room',
                                                                    'night',
                                                                    'day',
                                                                    'stay',
                                                                    'resort',
                                                                    'place'])),
                                       ('us',
                                        RandomUnderSampler(random_state=42,
                                                           sampling_strategy=
1)),
                                       ('classifier',
                                        XGBClassifier(base_score=None,
                                                      booster=None,
                                                      callbacks=None,
                                                      colsample_bylevel=None,
                                                      colsample_bynode=None,
                                                      colsample_bytree=None,
                                                      device=None...
                                                      max_delta_step=None,
                                                      max_depth=None,
                                                      max_leaves=None,
                                                      min_child_weight=None,
                                                      missing=nan,
                                                      monotone_constraints=Non
e,
                                                      multi_strategy=None,
                                                      n_estimators=None,
                                                      n_jobs=None,
                                                      num_parallel_tree=None,
                                                      random_state=None,
...))]),
             n_jobs=-1,
             param_grid={'classifier__learning_rate': [0.01, 0.1, 0.2],
                         'classifier__max_depth': [3, 5, 7],
                         'classifier__n_estimators': [50, 100, 150]},
             scoring='recall')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [322]:
```
# Storing the best params
best_xgb_params = grid_search.best_params_

# Storing the best model from the grid search
best_xgb_pipeline = grid_search.best_estimator_

# Printing the best parameters and the corresponding score
print("Best Parameters: ", grid_search.best_params_)
```

```
Best Parameters:  {'classifier__learning_rate': 0.2, 'classifier__max_depth':
7, 'classifier__n_estimators': 150}
```

In [323]:
```
# Fitting the best pipeline on training data
best_xgb_pipeline.fit(X_train['lemmatized_review'], y_train)

# Making predictions on the test set
best_xgb_y_pred = best_xgb_pipeline.predict(X_test['lemmatized_review'])
```

2) Evaluation Metrics

In [324]:
```
# Naming the model and calling the function to evaluate it
xgbtuned_model_name = 'Tuned XGBoost'


# Calling the function and recording into the defined values
accuracy_xgbtuned, f1_xgbtuned, recall_xgbtuned, cv_xgbtuned = evaluation_metrics
    y_test,
    best_xgb_y_pred,
    best_xgb_pipeline,
    X_train['lemmatized_review'],
    y_train
)
```

```
Accuracy: 0.8518
F1-Score: 0.8565
Recall: 0.8518
Mean Cross-Validated Accuracy: 0.8567
```

In [325]:
```
# Comparing the evaluation metrics between the last model the current one
metrics_comparison('Accuracy', accuracy_xgb, accuracy_xgbtuned)
metrics_comparison('F1', f1_xgb, f1_xgbtuned)
metrics_comparison('Recall', recall_xgb, recall_xgbtuned)
metrics_comparison('Cross-validated Accuracy', cv_xgb, cv_xgbtuned)
```

```
Accuracy improved from 0.8468 to 0.8518.
F1 improved from 0.8520 to 0.8565.
Recall improved from 0.8468 to 0.8518.
Cross-validated Accuracy decreased from 0.8574 to 0.8567.
```

In [326]:
```
# Printing the overall summary
print('Despite the expensive computing, overall scores are the best ones recorded
decreased compared to the untuned XBGBoost model but remains over 85%.')
```

```
Despite the expensive computing, overall scores are the best ones recorded. Cro
ss-validated accuracy slightly decreased compared to the untuned XBGBoost model
but remains over 85%.
```

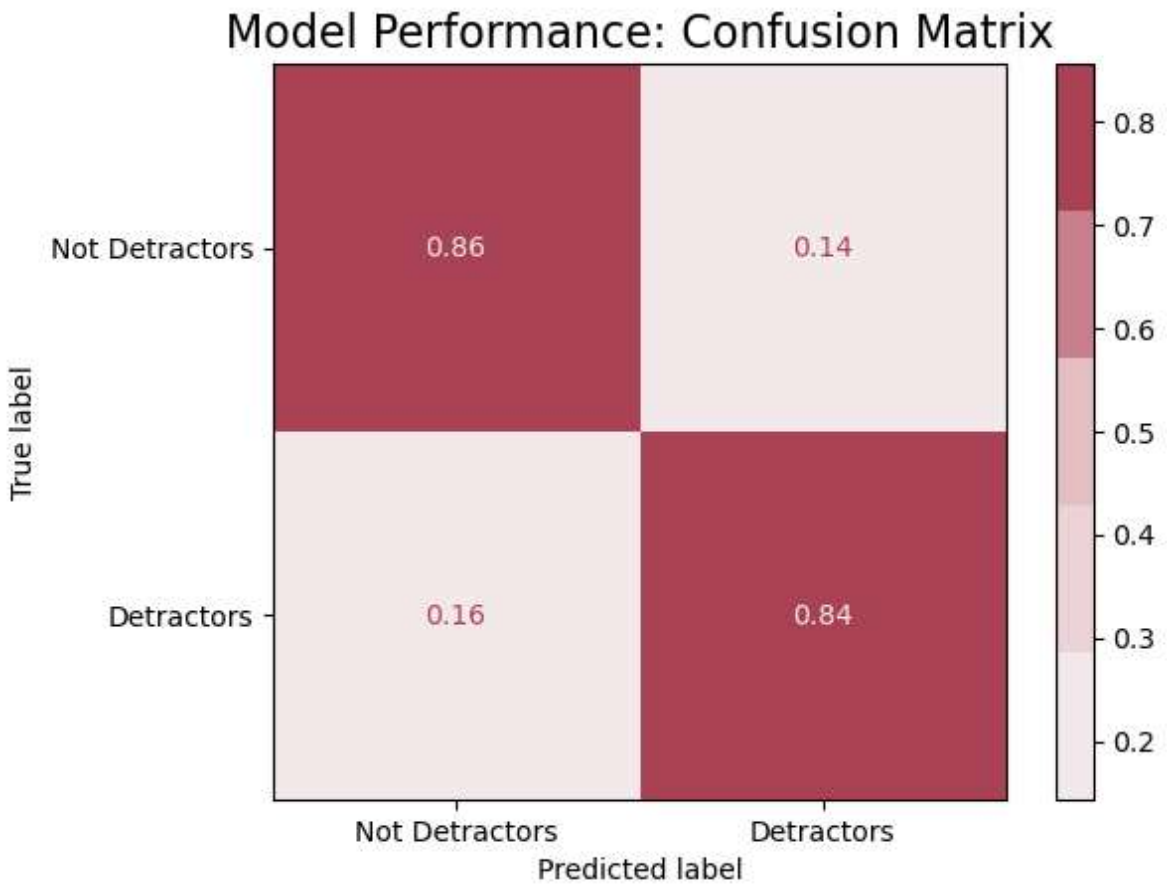
3) Classification Report

In [327]:
```
# Calling the function
tunedxgb_class_report, recall_detractors_tunedxgb = class_calculation(y_test, bes
```

```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.93706   0.85635   0.89489      3773
    Detractors    0.67642   0.83926   0.74909      1350

      accuracy                        0.85184      5123
     macro avg    0.80674   0.84780   0.82199      5123
  weighted avg    0.86838   0.85184   0.85647      5123
```

4) Confusion Matrix

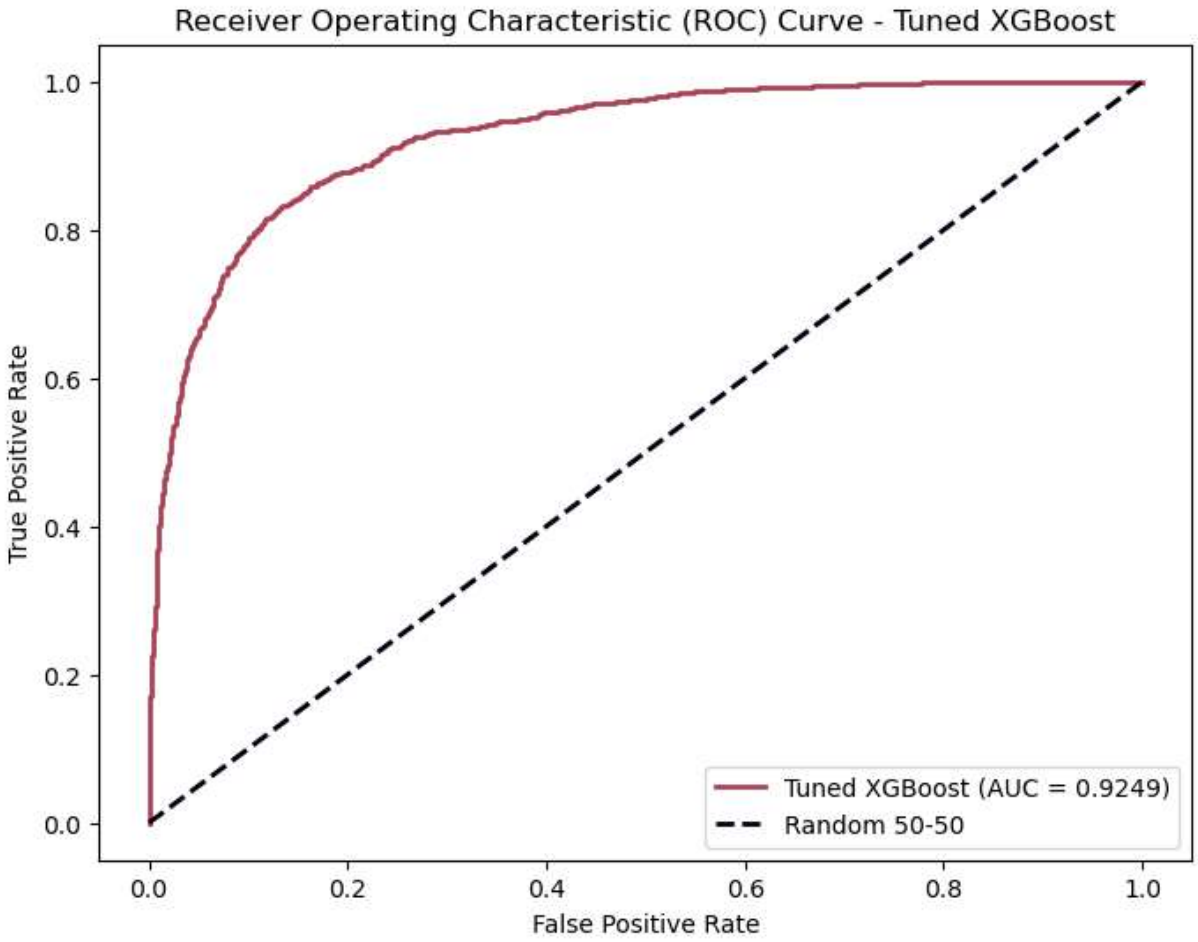
In [328]:
```
# Recording and displaying the confusion matrix
tunedxgb_confusion_matrix = confusion_matrix_display(best_xgb_pipeline, y_test, t
```



Against all odds, tuned XGBoost did not provide best results for recall for detractors.

5) ROC Curve and AUC

In [329]:
```
# Plotting the ROC curve for the baseline model
tunedxgb_y_pred_proba = best_xgb_pipeline.predict_proba(X_test['lemmatized_review
auc_xgbtuned = plot_roc_curve(y_test, tunedxgb_y_pred_proba, model_name=xgbtuned_
```



In [330]:
```
# Printing the interpretation
auc_interpretation(auc_xgbtuned)
```

```
AUC = 0.9249. The model has a great predictive power in distinguishing between
the positive and negative classes.
```

In [331]:
```
# Interpreting AUC:
print('Compared to the last run model: ')
metrics_comparison('AUC', auc_xgb, auc_xgbtuned)
print()
```

```
Compared to the last run model:
AUC improved from 0.9221 to 0.9249.
```

In [332]:
```
# Appending each metric to the lists
modelnames.append(xgbtuned_model_name)
accuracies.append(accuracy_xgbtuned)
f1s.append(f1_xgbtuned)
recalls.append(recall_xgbtuned)
recalls_detractors.append(recall_detractors_tunedxgb)
cv_accs.append(cv_xgbtuned)
roc_aucs.append(auc_xgbtuned)
```

6. Evaluation

6. a) Final Model and Classification Metrics

In [333]:
```
# Verifying that all lists have the same size
# print(len(modelnames))
# print()
# print(len(accuracies))
# print()
# print(len(f1s))
# print()
# print(len(recalls))
# print()
# print(len(recalls_detractors))
# print()
# print(len(cv_accs))
# print()
# print(len(roc_aucs))
# print()
```

In [334]:
```
# Creating a DataFrame with stored best scores

models_results = {
    'models': modelnames,
    'accuracy': accuracies,
    'f1': f1s,
    'recall': recalls,
    'recall_detractors': recalls_detractors,
    'cross-val_accuracies': cv_accs,
    'ROC-AUCs': roc_aucs
}

overall = pd.DataFrame(models_results)
```

In [335]:
```
# Inspecting all iterations' results
overall
```

Out[335]:

	models	accuracy	f1	recall	recall_detractors	cross-val_accuracies	ROC-AUCs
0	Baseline	0.754636	0.665579	0.754636	0.06963	0.750716	0.911003
1	Resampled	0.884248	0.878426	0.884248	0.65481	0.883589	0.927415
2	Sampling 1	0.831154	0.837767	0.831154	0.84296	0.844221	0.919755
3	No Stopwords	0.831934	0.838196	0.831934	0.83333	0.846628	0.918632
4	Industry Stopwords	0.833106	0.839580	0.833106	0.84444	0.845977	0.920165
5	Full Stopwords	0.834472	0.840474	0.834472	0.83259	0.848060	0.919093
6	Lemmatized	0.827054	0.834410	0.827054	0.85630	0.838169	0.919214
7	Preprocessed	0.827054	0.834132	0.827054	0.84593	0.840381	0.917364
8	NaiveBayes GS	0.827445	0.834899	0.827445	0.86148	0.836998	0.914550
9	kNN	0.745071	0.756900	0.745071	0.72815	0.756442	0.803643
10	DecisionTree	0.729260	0.742067	0.729260	0.70519	0.724426	0.721530
11	RandomForest	0.836619	0.842079	0.836619	0.82148	0.850663	0.910567
12	GradientBoosting	0.825102	0.831326	0.825102	0.81111	0.837323	0.902704
13	AdaBoost	0.814952	0.822096	0.814952	0.80889	0.827563	0.890777
14	XGBoost	0.846769	0.851977	0.846769	0.84370	0.857366	0.922079
15	Tuned XGBoost	0.851845	0.856469	0.851845	0.83926	0.856715	0.924855

In [336]:
```
# Storing the maximum recall detractors value and its index
max_recall_detrac = max(recalls_detractors)
max_recall_index_detrac = recalls_detractors.index(max_recall_detrac)
```

In [337]:
```
# Printing metrics for max recall index
overall.iloc[max_recall_index_detrac]
```

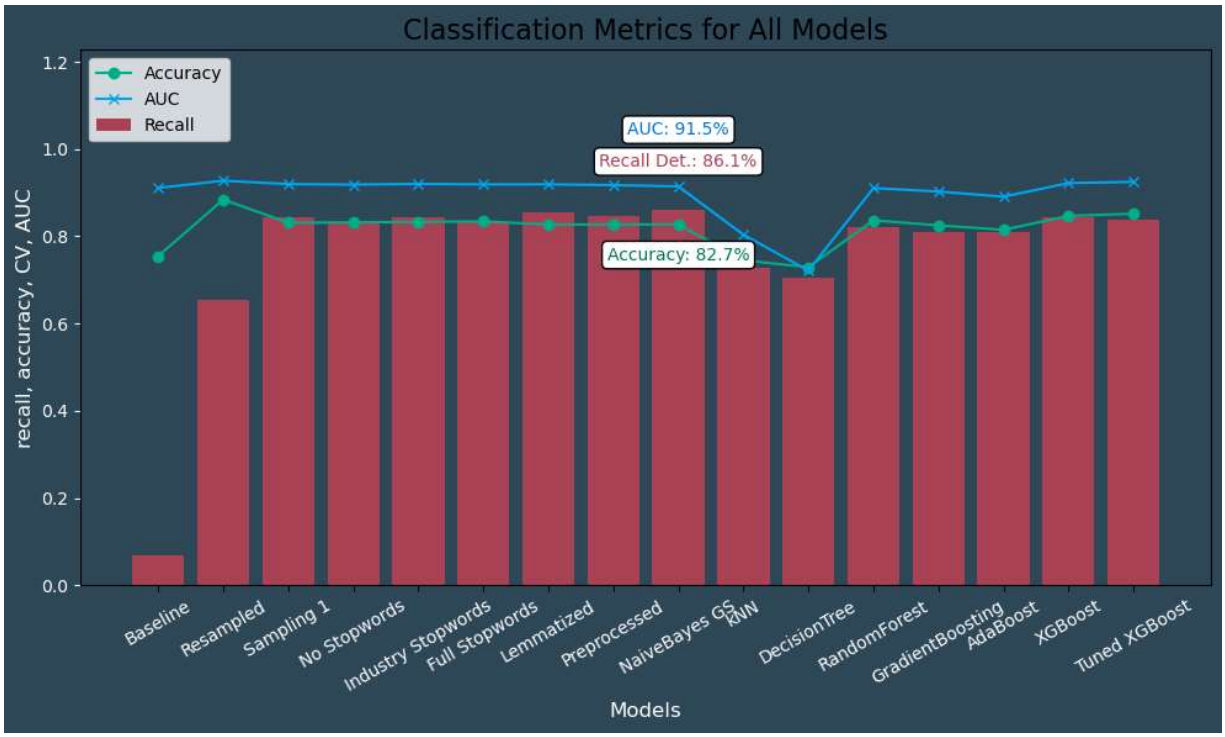
Out[337]:
```
models                  NaiveBayes GS
accuracy                     0.827445
f1                           0.834899
recall                       0.827445
recall_detractors             0.86148
cross-val_accuracies         0.836998
ROC-AUCs                      0.91455
Name: 8, dtype: object
```

In [339]:
```
# Storing both recalls the columns I wanted to average
both_recalls = ['recall', 'recall_detractors']

# # Calculate the average for each row
overall['average_both_recalls'] = overall[both_recalls].mean(axis=1)
```

In [340]:
```
# overall
```

In [341]:
```
# Storing the maximum recall for both recalls and its index
max_both_recalls = max(both_recalls)
max_both_recalls_index = both_recalls.index(max_both_recalls)
```

In [342]:
```
# Creating a bar chart to review all

#  Creating a bar chart for recall
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(10,6))

# Setting facecolor
fig.set_facecolor('#2F4858')
ax.set_facecolor('#2F4858')

# plt.figure(figsize=(10, 6))
plt.bar(overall['models'], overall['recall_detractors'], color='#AA4255', label='

# Creating line plots for Log Loss and accuracy
plt.plot(overall['models'], overall['accuracy'], marker='o', color='#00AF87', lab
# plt.plot(overall['models'], overall['f1'], marker='x', color='#00A4EF', label=
plt.plot(overall['models'], overall['ROC-AUCs'], marker='x', color='#00A4EF', lab


# Setting labels and title
plt.xlabel('Models', fontsize=12, color='white')
plt.xticks(rotation=30, color='white')
ax.tick_params(axis='y', colors='white')

plt.ylabel('recall, accuracy, CV, AUC', fontsize=12, color='white')



# Annotating the last index of each category
plt.annotate(f'Recall Det.: {recalls_detractors[max_recall_index_detrac] * 100:.1
plt.annotate(f'Accuracy: {accuracies[max_recall_index_detrac] * 100:.1f}%', (mode
plt.annotate(f'AUC: {roc_aucs[max_recall_index_detrac] * 100:.1f}%', (modelnames[

# Defining the max value of y
max_y = max(overall[['ROC-AUCs', 'recall_detractors']].max())
plt.ylim(0, max_y  + 0.3)
plt.title('Classification Metrics for All Models', fontsize=16)

# Displaying the legend
plt.legend(loc='upper left')

# Saving the plot as a PNG with a transparent background
plt.savefig('images/final_model_names.png', transparent=True)

# Showing the plot
plt.tight_layout()
plt.show()
```



Summarizing the best classification model types

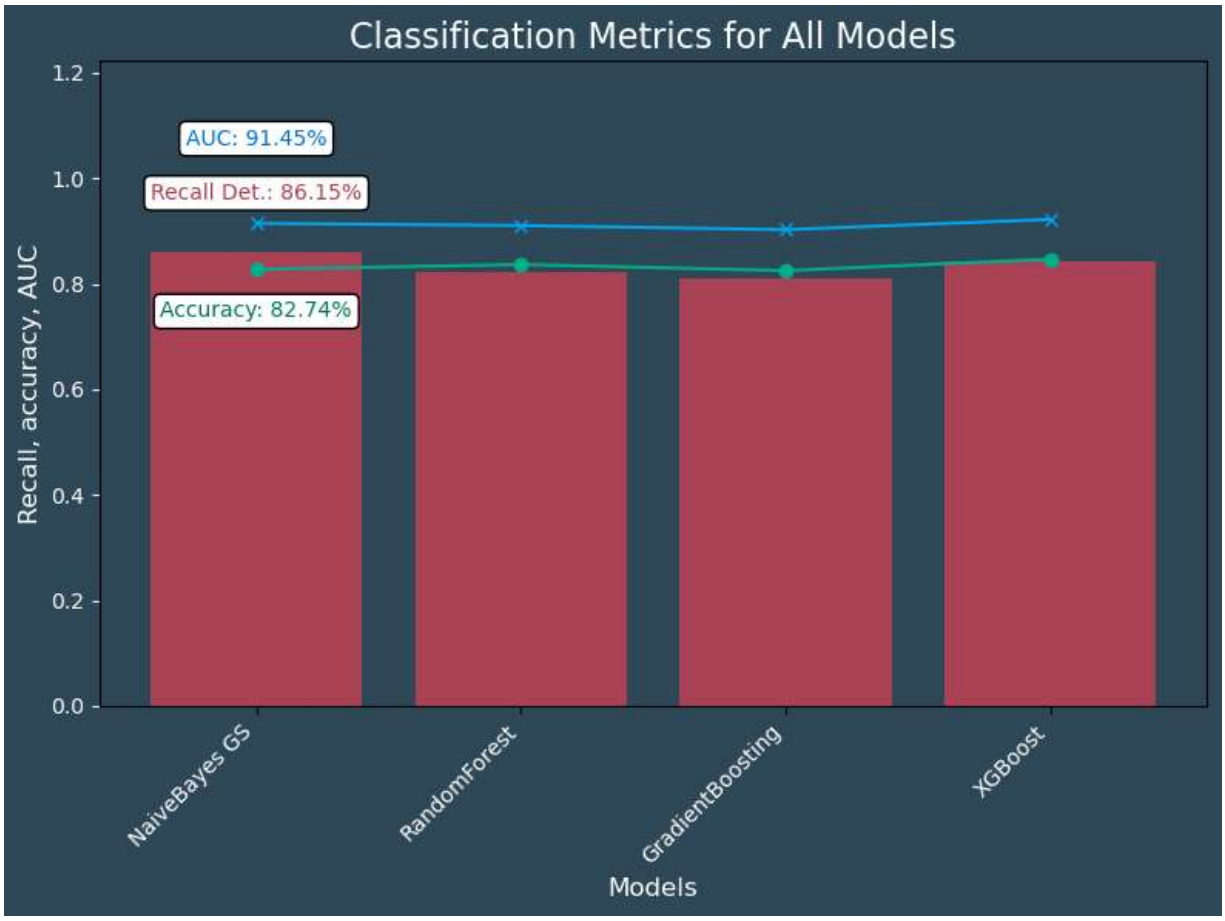
In [343]:
```
# Defining indices to filter on
models_indices = [8, 11, 12, 14]
```

In [344]:
```
# Creating the dataframe with all models
main_model_types = overall.iloc[models_indices]

# Viewing them
main_model_types
```

Out[344]:

	models	accuracy	f1	recall	recall_detractors	cross-val_accuracies	ROC-AUCs	avera
8	NaiveBayes GS	0.827445	0.834899	0.827445	0.86148	0.836998	0.914550	
11	RandomForest	0.836619	0.842079	0.836619	0.82148	0.850663	0.910567	
12	GradientBoosting	0.825102	0.831326	0.825102	0.81111	0.837323	0.902704	
14	XGBoost	0.846769	0.851977	0.846769	0.84370	0.857366	0.922079	

In [345]:
```
# Creating a bar chart to review all

#  Creating a bar chart for recall
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(8,6))

# Setting facecolor
fig.set_facecolor('#2F4858')
ax.set_facecolor('#2F4858')

# plt.figure(figsize=(10, 6))
plt.bar(main_model_types['models'], main_model_types['recall_detractors'], color=

# Creating line plots for Log Loss and accuracy
plt.plot(main_model_types['models'], main_model_types['accuracy'], marker='o', co
# plt.plot(main_model_types['models'], main_model_types['f1'], marker='x', color=
plt.plot(main_model_types['models'], main_model_types['ROC-AUCs'], marker='x', co


# Setting labels and title
plt.xlabel('Models', fontsize=12, color='white')
plt.xticks(rotation=45, color='white', ha='right')
ax.tick_params(axis='y', colors='white')

plt.ylabel('Recall, accuracy, AUC', fontsize=12, color='white')



# Annotating the last index of each category
plt.annotate(f'Recall Det.: {recalls_detractors[max_recall_index_detrac] * 100:.2
plt.annotate(f'Accuracy: {accuracies[max_recall_index_detrac] * 100:.2f}%', (mode
plt.annotate(f'AUC: {roc_aucs[max_recall_index_detrac] * 100:.2f}%', (modelnames[

# Defining the max value of y
max_y = max(main_model_types[['ROC-AUCs', 'recall_detractors']].max())
plt.ylim(0, max_y  + 0.3)
plt.title('Classification Metrics for All Models', fontsize=16, color='white')

# Displaying the legend
# plt.legend(loc='upper left')

# Showing the plot
plt.tight_layout()

# Saving the plot as a PNG
plt.savefig('images/final_models.png')

plt.show()
```



In [346]:
```
print(f'All top 5 models resulted in very close results. The {modelnames[max_reca
results for recall specifically for reviews from detractors, which is the goal fo
```

```
All top 5 models resulted in very close results. The NaiveBayes GS model record
ed the higest results for recall specifically for reviews from detractors, whic
h is the goal for predictions.
```

6. b) Model Performance

2) Evaluation Metrics

- Train data

In [347]:
```
# Naming the model and calling the function to evaluate it
best_model_name = modelnames[max_recall_index_detrac]
best_model = best_pipeline

# Reminding the evaluation metrics for the model
print(f'Evaluation Metrics on Train Data for {best_model_name} model.')

print(f'Accuracy: {overall.iloc[max_recall_index_detrac]["accuracy"]:.4f}')
print(f'F1-Score: {overall.iloc[max_recall_index_detrac]["f1"]:.4f}')
print(f'Recall: {overall.iloc[max_recall_index_detrac]["recall"]:.4f}')
print(f'Recall Detractors: {overall.iloc[max_recall_index_detrac]["recall_detract
print(f'Mean Cross-Validated Accuracy: {overall.iloc[max_recall_index_detrac]["cr
```

```
Evaluation Metrics on Train Data for NaiveBayes GS model.
Accuracy: 0.8274
F1-Score: 0.8349
Recall: 0.8274
Recall Detractors: 0.8615
Mean Cross-Validated Accuracy: 0.8370
```

- Test data

As I have evaluated the models, I will run our function again. The same results are expected for all metrics, only the cross-validated accuracy should change as this one will be evaluated on the test, unseen data.

In [348]:
```
# Running evaluation metrics on test data
print(f'Evaluation Metrics on Unseen Data for {best_model_name} model.')
print()

# Calculating predictions using this model
best_y_pred = best_model.predict(X_test['lemmatized_review'])

# Calling the function and recording into the defined values
accuracy_best_test, recall_best_test, f1_best_test, cv_best_test = evaluation_met
    y_test,
    best_y_pred,
    best_model,
    X_test['lemmatized_review'],
    y_test
)
```

```
Evaluation Metrics on Unseen Data for NaiveBayes GS model.

Accuracy: 0.8274
F1-Score: 0.8349
Recall: 0.8274
Mean Cross-Validated Accuracy: 0.8454
```

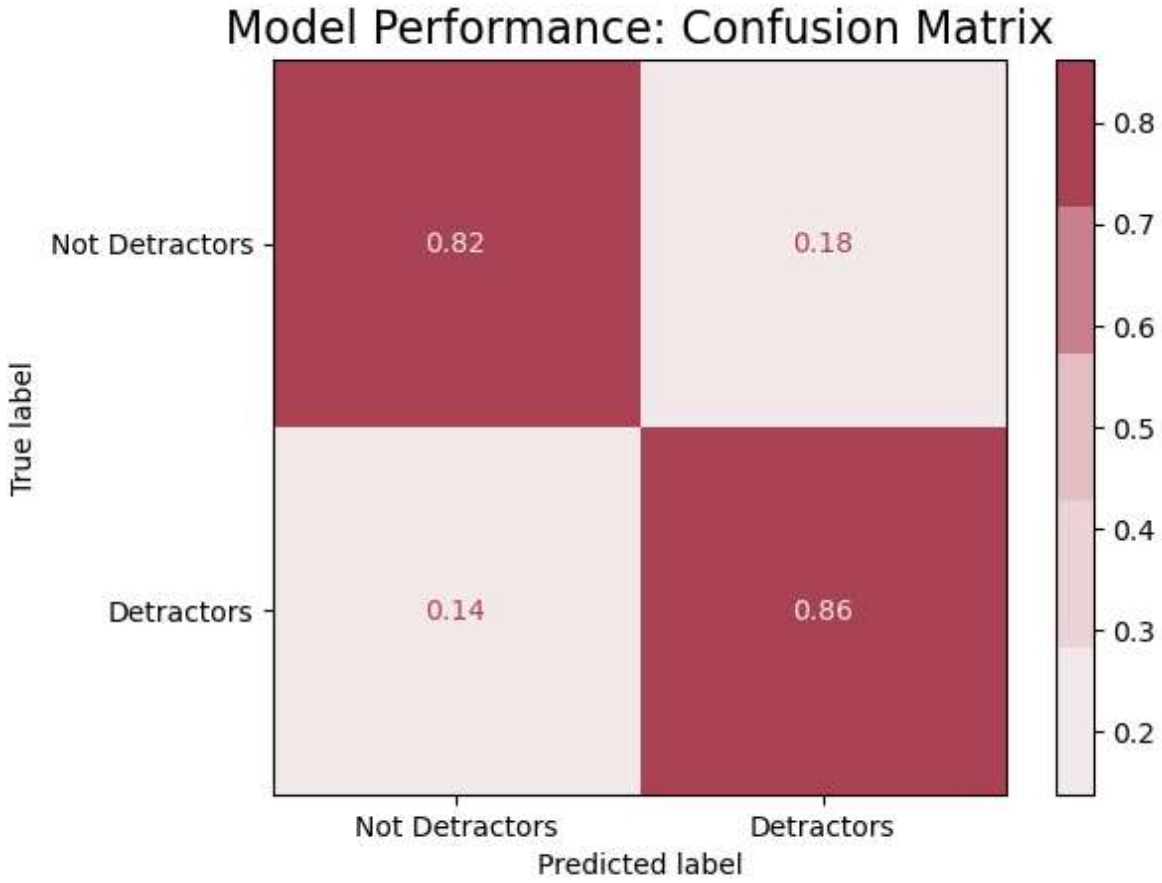
3) Classification Report

In [349]:
```
# Calling the function
best_class_report, recall_detractors_best = class_calculation(y_test, best_y_pred
```

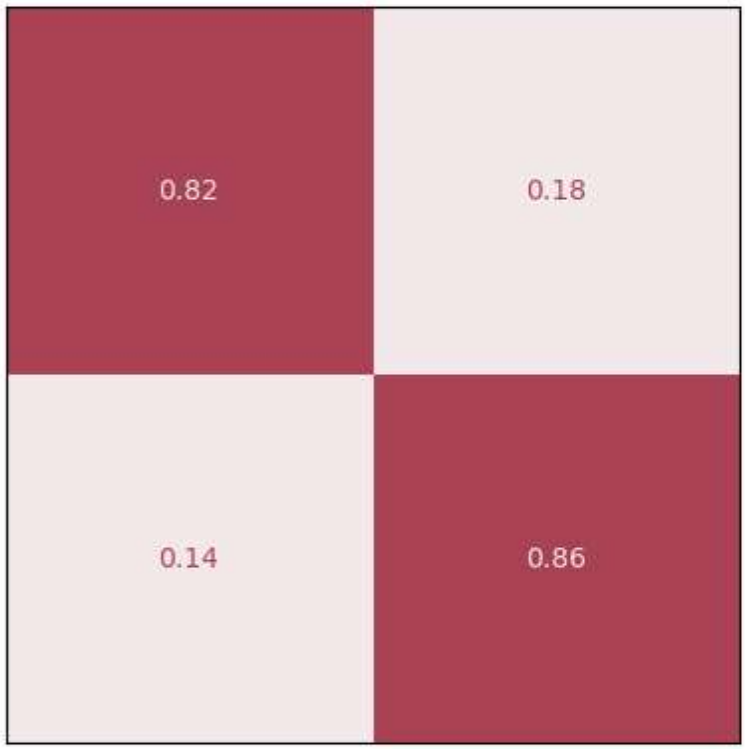
```
Classification Report:
                precision    recall  f1-score   support

Not Detractors    0.94269   0.81527   0.87436      3773
    Detractors    0.62527   0.86148   0.72461      1350

      accuracy                        0.82744      5123
     macro avg    0.78398   0.83837   0.79949      5123
  weighted avg    0.85904   0.82744   0.83490      5123
```

4) Confusion Matrix

In [350]:
```
# Recording and displaying the confusion matrix
best_confusion_matrix = confusion_matrix_display(best_model, y_test, best_y_pred)
```

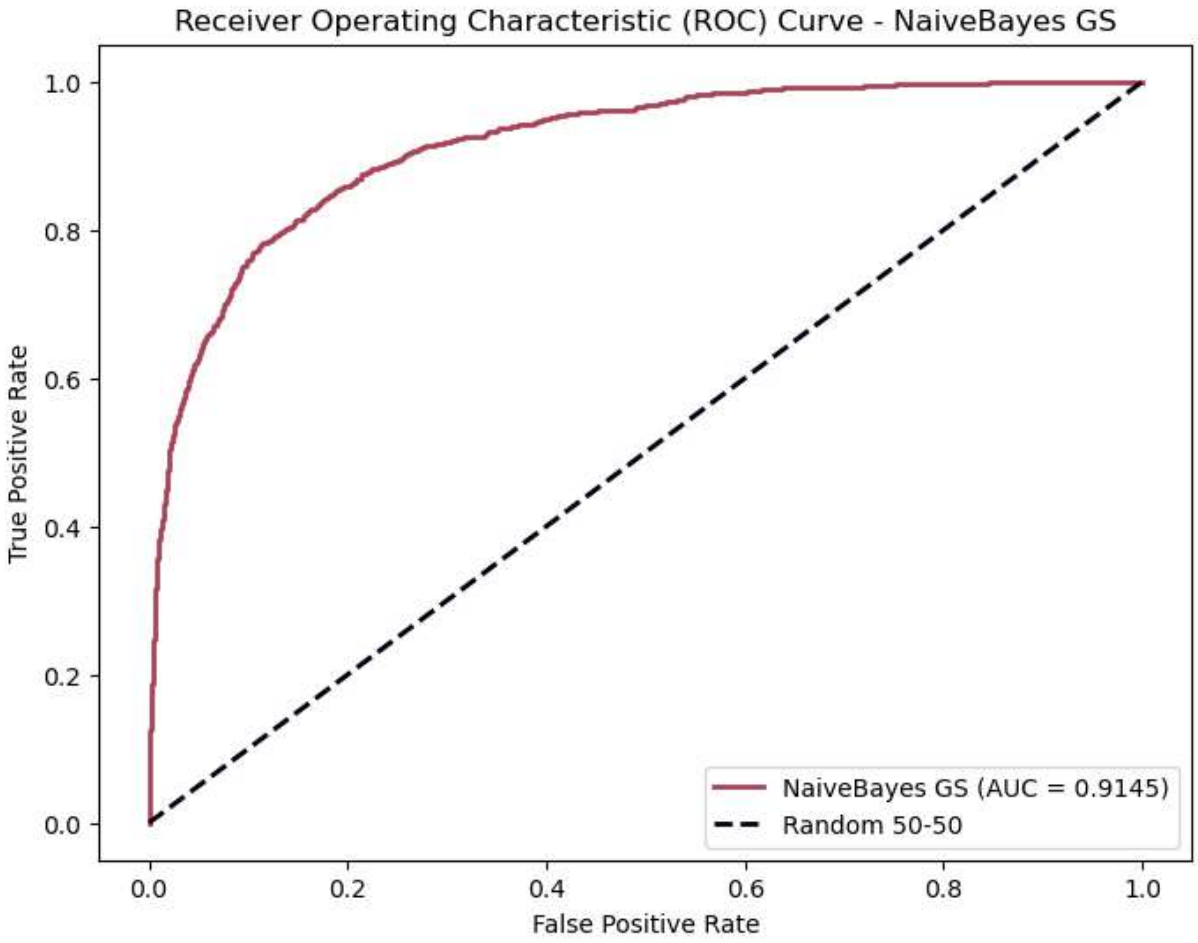


In [351]:
```
# Recording the best confusion matrix for the presentation, with labels in white
prez_cfn_matrix = best_confusion_matrix_only(best_model, y_test, best_y_pred)
```



5) ROC-AUC

In [352]:
```
# Plotting the best model ROC AUC
auc_gs = plot_roc_curve(y_test, gs_y_pred_proba, model_name=gs_model_name)
```



The classification report, confusion matrix and ROC-AUC summarize the evaluation of the model's performance on predicting detractors from hotel TripAdvisor reviews.

The model performs better on predicting *detractors* sentiment. More precisely, it has a true positive rate higher for this class than for the *not_detractors*.

This score was the focus for the evaluation of the models, as the cost of false negative is higher than the cost of false positive. If a review from a detractor was incorrectly identified as "not detractor", the hotel would miss an opportunity to transform a guest's negative experience.

Looking at the details by metric:

Accuracy
The mode correctly identified close to 83% of all reviews.

Recall
86.15% of actual detractors were correctly predicted.

F1 Score
The balance between precision and recall. It is lower (72.5%) for the detractors class, which indicates precision is lower for this class.

AUC

AUC

The Area Under the Receiver Operating Characteristic Curve (AUC) value of 0.9145 indicates a high effectiveness in distinguishing between the two classes: `not_detractors` and `detractors`. The ROC curve, which plots the true positive rate against the false positive rate,

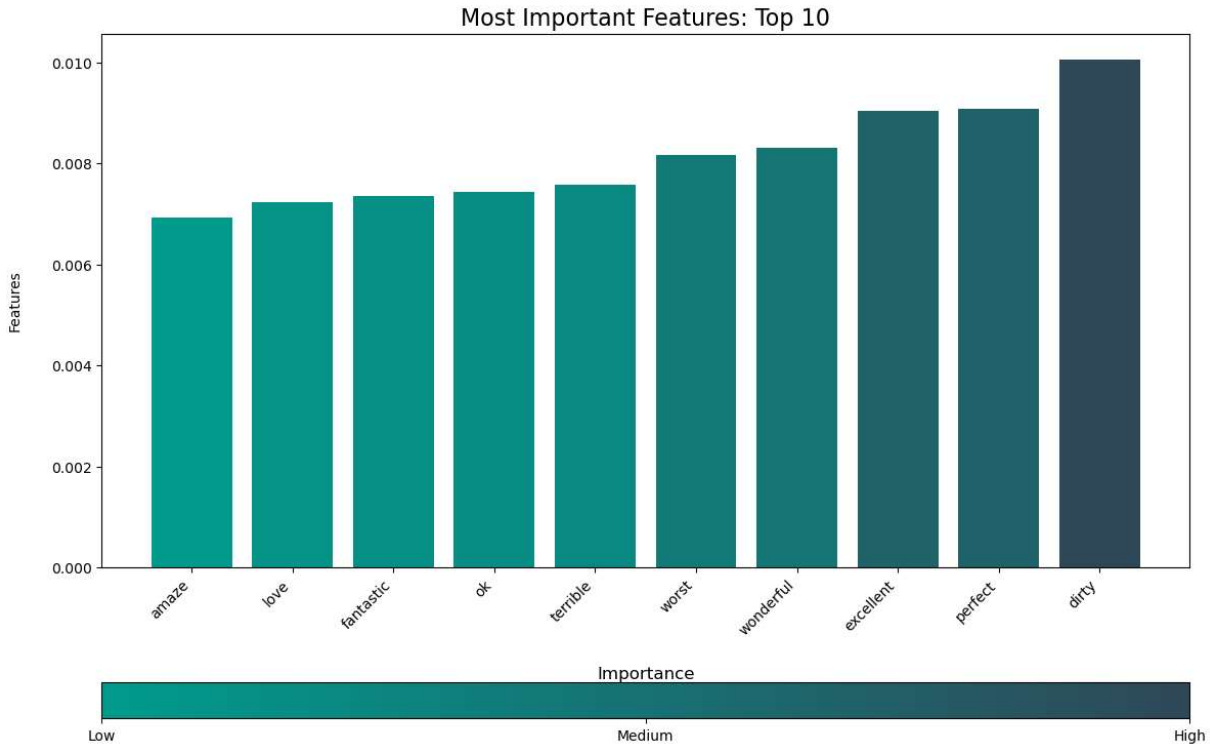
7. Findings & Recommendations

7. a) Most Important Features

In [353]:
```
# Defining a colormap specifically for the top features bar graph
custom_colors = ['#009B8D', '#2F4858']

n_bins = 200

# Creating the custom colormap
features_cmap = LinearSegmentedColormap.from_list("features_cmap", custom_colors,
```

In [354]:
```
from matplotlib.cm import ScalarMappable


def plot_top_feature_importances(pipeline, top_n):
    # Accessing the classifier instance from the pipeline
    classifier = pipeline.named_steps['classifier']

    # Getting the feature names from the vectorizer
    feature_names = pipeline.named_steps['tfidf'].get_feature_names_out()

    # Getting the feature importances
    feature_importances = classifier.feature_importances_

    # Sorting feature importances in descending order and get the corresponding i
    sorted_indices = np.argsort(feature_importances)[::-1]

    # Selecting the top 'top_n' features
    top_indices = sorted_indices[:top_n]

    # Sorting the feature names based on the selected top indices
    top_feature_names = feature_names[top_indices]

    # Sorting the feature importances for the selected top features
    top_feature_importances = feature_importances[top_indices]

    # Reverse the order to make it descending
    top_feature_names = top_feature_names[::-1]
    top_feature_importances = top_feature_importances[::-1]

    # Creating a colormap
    cmap = plt.get_cmap(features_cmap)

    # Manually normalizing importance values
    normalized_importances = (top_feature_importances - np.min(top_feature_import
            np.max(top_feature_importances) - np.min(top_feature_importances))

    # Creating the figure and axis
    fig, ax = plt.subplots(figsize=(12, 8))

    # Using normalized_importances directly for color mapping
    bars = ax.bar(range(top_n), top_feature_importances, align='center', color=cr
    plt.xticks(range(top_n), top_feature_names, rotation=45, ha='right')  # Rotat
    plt.ylabel('Features', labelpad=20)
    plt.xlabel('Importance', fontsize=12, labelpad=20)
    plt.title('Most Important Features: Top {}'.format(top_n), fontsize=16)

    # Adding a colorbar below the plot
    sm = ScalarMappable(cmap=cmap)
    sm.set_array([])
    colorbar = plt.colorbar(sm, ax=ax, orientation='horizontal', aspect=30)
    # Hide colorbar ticks
    # Defining custom tick positions
    custom_ticks = [0, 0.5, 1]
    custom_ticklabels = ['Low', 'Medium', 'High']
    colorbar.set_ticks(custom_ticks)
    colorbar.set_ticklabels(custom_ticklabels)
    plt.tight_layout()
    # Saving the plot as PNG with a transparent background
```

```
    # Saving the plot as PNG with a transparent background
    plt.savefig('images/most_important_features.png', transparent=True)

    # Showing the plot
    plt.show()
    return top_feature_names[::-1]


# Calling the function with the best model
top_10_features = plot_top_feature_importances(best_xgb_pipeline, top_n=10)
```



The most important features for the model from the 2nd best model can help hotels knowing what words to pay attention to, also when they communicate with guests. These features refer to both classes.

In [355]:
```
# Inspecting the returned top 10 features
top_10_features
```

Out[355]:
```
array(['dirty', 'perfect', 'excellent', 'wonderful', 'worst', 'terrible',
       'ok', 'fantastic', 'love', 'amaze'], dtype=object)
```

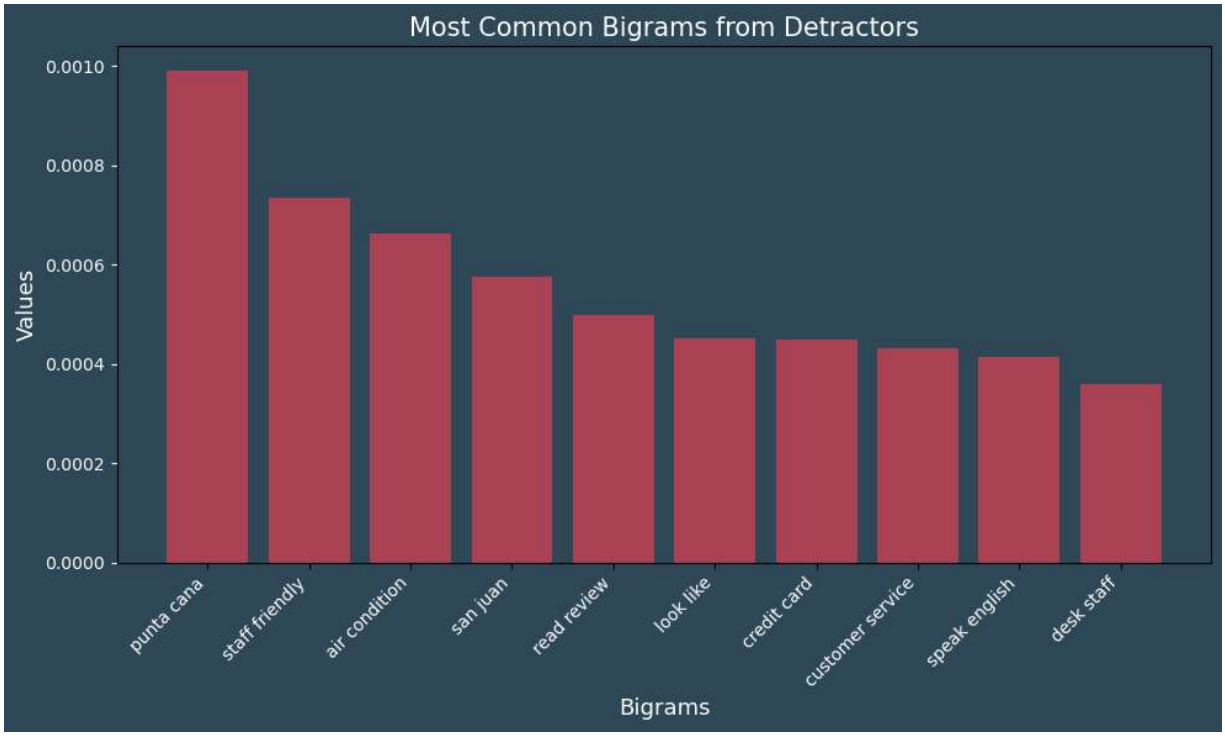
7. b) Recommendations

The advantage of our new service is the deployment can be immediate. Based on historical guest complaints, below are the areas we recommend to focus on to immediately start improving guest satisfactions.

1. Focus on resort hotels
   → These hotels have the most unsatisfied guests

2. Develop a maintenance program with engineering teams
   → Appearance & dysfunctionality cause frustrations
3. Train staff to enhance friendliness
   → Also ensure more languages are spoken
4. Respond to reviews
   → Detractors advise to read reviews

In [356]:
```
# Plotting the detractor bigrams as a bar chart

# Extracting bigrams and values
bigrams, values = zip(*detractors_bigrams_10)

# Sort bigrams and values in descending order based on values
sorted_indices = sorted(range(len(values)), key=lambda k: values[k], reverse=True
sorted_bigrams = [bigrams[i] for i in sorted_indices]
sorted_values = [values[i] for i in sorted_indices]

# Create a bar chart
fig, ax = plt.subplots(figsize=(10, 6))
fig.set_facecolor('#2F4858')
ax.set_facecolor('#2F4858')
ax.bar(range(len(sorted_values)), sorted_values, align='center', color='#AA4255')
ax.set_xticks(range(len(sorted_values)))
ax.set_xticklabels([f'{bigram[0]} {bigram[1]}' for bigram in sorted_bigrams], rot
ax.set_xlabel('Bigrams', color='white', fontsize=13)
ax.set_ylabel('Values', color='white', fontsize=13)
ax.tick_params(axis='y', colors='white')
ax.set_title('Most Common Bigrams from Detractors', color='white', fontsize=15)
plt.tight_layout()

plt.savefig('images/top_10_bigrams')
plt.show()
```



8. Next Steps & Limits

- **Deployment**
  Once the immediates areas of focus are implemented, the next step is only to
  - entice guests to leave real-time reviews. Strategically display the QR code to facilitate access to the tool.
  - only provide us with 2 contacts details. These persons will receive the alerts when guests provide negative feedback
  - communicate about this new tool to your teams

Very quickly, you will see the share of negative reviews decrease in your hotels, and your NPS score increase!

Extract the reviews specifically for your hotel chains, to ensure the model is based on your specific reviews. We, at TripAdvisor can do this for you for a supplement.

- **Limits**
  The model provided very high scores for recommendations. If basing the model on your own hotel chains' reviews is not an option, ensure the most recent data for reviews is taken into account.

- **Next Steps**
  Deploy the model for a ternary classification, following the actual sentiments defined by NPS score: `promoters` , `neutral` , `detractors` .