How to Name Long-Lasting Businesses! - A Case Study for Hairdressers

February 3, 2023

0.1 Importing Common Libraries & Settings

[1]:
```
!pip3 install requests
!pip install wordcloud
import requests
import json
import time
import pandas as pd
import math
from datetime import timedelta, date, datetime as dt
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator
import matplotlib.pyplot as plt
print('Libraries Imported')
```

```
Requirement already satisfied: requests in /opt/conda/lib/python3.9/site-
packages (2.27.1)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.9/site-
packages (from requests) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/opt/conda/lib/python3.9/site-packages (from requests) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.9/site-packages (from requests) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.9/site-packages (from requests) (2022.9.24)
Requirement already satisfied: wordcloud in /opt/conda/lib/python3.9/site-
packages (1.8.1)
Requirement already satisfied: numpy>=1.6.1 in /opt/conda/lib/python3.9/site-
packages (from wordcloud) (1.21.6)
Requirement already satisfied: pillow in /opt/conda/lib/python3.9/site-packages
(from wordcloud) (9.1.1)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.9/site-
packages (from wordcloud) (3.5.2)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.9/site-
```

```
packages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.9/site-
packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->wordcloud) (3.0.8)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->wordcloud) (4.38.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.9/site-
packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)

/opt/conda/lib/python3.9/site-packages/requests/__init__.py:102:
RequestsDependencyWarning: urllib3 (1.26.9) or chardet
(5.0.0)/charset_normalizer (2.0.12) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({})/charset_normalizer ({}) doesn't
match a supported "

Libraries Imported
```

[2]:
```
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
print('Dataframe view set up')
```

```
Dataframe view set up
```

[3]:
```
def call_api_with(url_extension):
    your_company_house_api_key ="4fad447a-8aae-4fda-849c-9e64c9fbb86c"

    login_headers = {"Authorization":your_company_house_api_key}
    url = f"https://api.company-information.service.gov.uk/{url_extension}"
    print(f'requesting: {url}')
    res = requests.get(url, headers=login_headers)
    return res.json()
print('Created Function to authenticate and call API')
```

```
Created Function to authenticate and call API
```

0.2 How to Name Long-Lasting Businesses! - A Case Study for Hairdressers

0.3 Setting the Context:

Studies indicate that the name of a brand could make or break it. Our clients are planning on opening a chain of salons and have approached us for recommendations based on historical data. They want a brand name which facilitates a long-lasting run for the chain.

The report begins by fetching, organising, transforming & checking the data in the **'Data Import-**

ing & Cleaning' section. This is followed by exploratory analysis to define what 'long-lasting' means for a hairdressing business in the **'Defining Long-Lasting'** section. The **'Analysing Names'** section focuses on cleaning of names & keyword analysis. The report ends with visualisation of trends under **'Visualising Results'** followed by an outline of the problem, approach and findings in the **'Mini-Report'**.

0.3.1 Business Question 2: Code:

0.4 Data Importing & Cleaning

[4]:
```
def fetch_data_for_hairdressers(strt_creat_year, strt_creat_month,␣
 ↪strt_creat_day,end_creat_year,
                                end_creat_month, end_creat_day, days,␣
 ↪number_of_companies, sic):
    size=5000
    index_number = math.ceil(number_of_companies/size)
    data_in_list = []
    creat_dt_strt = date(strt_creat_year,strt_creat_month,strt_creat_day)
    creat_dt_end = date(end_creat_year,end_creat_month,end_creat_day)
    creat_dt_end_interim = creat_dt_strt + timedelta(days=days)
    while creat_dt_strt<=creat_dt_end:
        for index in range(0,index_number):
            data_in_list += call_api_with(f"""advanced-search/companies?
 ↪incorporated_from={creat_dt_strt}&incorporated_to={creat_dt_end_interim}&sic_codes={sic}&si
 ↪get('items', [])
            time.sleep(0.6)
        creat_dt_strt = creat_dt_strt + timedelta(days=days+1)
        creat_dt_end_interim = creat_dt_strt + timedelta(days=days)
    return data_in_list
print('Created Function to fetch data for hairdressers created between 2010 and␣
 ↪2020')
```

```
Created Function to fetch data for hairdressers created between 2010 and 2020
```

[5]:
```
hairdressers = pd.
 ↪json_normalize(fetch_data_for_hairdressers(2010,1,1,2021,1,1,300,10000,96020))
hairdressers = hairdressers.drop(columns='sic_codes').drop_duplicates()
print("""Fetched Data for hairdressers created between 2010 and 2020, Dropped␣
 ↪SIC Code Column and removed duplicates""")
```

```
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2010-01-01&incorporated_to=2010-10-28&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2010-01-01&incorporated_to=2010-10-28&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2010-10-29&incorporated_to=2011-08-25&sic_codes=96020&siz
```

```
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2010-10-29&incorporated_to=2011-08-25&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2011-08-26&incorporated_to=2012-06-21&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2011-08-26&incorporated_to=2012-06-21&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2012-06-22&incorporated_to=2013-04-18&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2012-06-22&incorporated_to=2013-04-18&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2013-04-19&incorporated_to=2014-02-13&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2013-04-19&incorporated_to=2014-02-13&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2014-02-14&incorporated_to=2014-12-11&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2014-02-14&incorporated_to=2014-12-11&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2014-12-12&incorporated_to=2015-10-08&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2014-12-12&incorporated_to=2015-10-08&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2015-10-09&incorporated_to=2016-08-04&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2015-10-09&incorporated_to=2016-08-04&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2016-08-05&incorporated_to=2017-06-01&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2016-08-05&incorporated_to=2017-06-01&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2017-06-02&incorporated_to=2018-03-29&sic_codes=96020&siz
```

```
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2017-06-02&incorporated_to=2018-03-29&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2018-03-30&incorporated_to=2019-01-24&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2018-03-30&incorporated_to=2019-01-24&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2019-01-25&incorporated_to=2019-11-21&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2019-01-25&incorporated_to=2019-11-21&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2019-11-22&incorporated_to=2020-09-17&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2019-11-22&incorporated_to=2020-09-17&sic_codes=96020&siz
e=5000&start_index=5000
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2020-09-18&incorporated_to=2021-07-15&sic_codes=96020&siz
e=5000&start_index=0
requesting: https://api.company-information.service.gov.uk/advanced-search/compa
nies?incorporated_from=2020-09-18&incorporated_to=2021-07-15&sic_codes=96020&siz
e=5000&start_index=5000
Fetched Data for hairdressers created between 2010 and 2020, Dropped SIC Code
Column and removed duplicates
```

[6]:
```
if hairdressers.shape[0] == hairdressers['company_number'].nunique():
    print('There are no duplicates in the data and the number of companies are␣
 ↪'+str(hairdressers.shape[0]))
else:
    print('The data has duplicates')
```

```
There are no duplicates in the data and the number of companies are 69532
```

[7]:
```
def replace_digit_with_space(string):
    if string == string:
        for char in string:
            if char.isdigit():
                string = string.replace(char, ' ')

    return string
```

```
def identify_postal_region(string):
    if string==string:
        return string.split(' ')[0]
    else:
        return np.nan

print('Functions to Postal Region from registered_office_address.postal_code')
```

```
Functions to Postal Region from registered_office_address.postal_code
```

[8]:
```
hairdressers['postal_region'] = hairdressers['registered_office_address.
 ↪postal_code'].map(replace_digit_with_space)
hairdressers['postal_region'] = hairdressers['postal_region'].
 ↪map(identify_postal_region)
print('Fetched Postal Region from registered address')
```

```
Fetched Postal Region from registered address
```

[9]:
```
postal_lookup = pd.read_csv('postal_lookup.txt', sep='\t',␣
 ↪encoding='ISO-8859-1')
print('Imported Postal Code Lookup')

hairdressers = hairdressers.merge(postal_lookup,how = 'left',left_on =␣
 ↪['postal_region'],
                                                  right_on = ['post_area'])
print('Merged area name with original dataset')
```

```
Imported Postal Code Lookup
Merged area name with original dataset
```

[10]:
```
def cessation_year_calc(date):
    if date != date:
        return 2022
    else:
        return int(date[0:4])
```

[11]:
```
hairdressers['creation_year'] = pd.
 ↪DatetimeIndex(hairdressers['date_of_creation']).year
hairdressers['cessation_year'] = hairdressers['date_of_cessation'].
 ↪map(cessation_year_calc)
hairdressers['tenure'] = hairdressers['cessation_year'] -␣
 ↪hairdressers['creation_year']
print('Calculated Creation & Cessation Year and Tenure')
```

```
Calculated Creation & Cessation Year and Tenure
```

[12]:
```
hairdressers_top_locs = hairdressers.groupby(['area_name'])['company_number'].
 ↪count().to_frame().reset_index()
hairdressers_top_locs['rank'] =  hairdressers_top_locs['company_number'].
 ↪rank(ascending = False)

list_of_top_10_localities =␣
 ↪list(hairdressers_top_locs[hairdressers_top_locs['rank']<=10]['area_name'])
print('Created list of top 10 localities by number of hairdressers')
```

```
Created list of top 10 localities by number of hairdressers
```

0.5 Defining Long-Lasting

[13]:
```
fig = px.box(hairdressers[hairdressers['area_name'].
 ↪isin(list_of_top_10_localities)], x='area_name', y='tenure',
              color='area_name', title ='Tenure across Top Localities')
fig.show()
```

[14]:
```
fig = px.box(hairdressers, x='company_status', y='tenure',␣
 ↪color='company_status',
              title ='Tenure across Company Status')
fig.show()
```

The median tenure across localities seems to be 2-4 years, while the 3rd quartile ranges between 4-5 years. Hence, we define long-lasting as a tenure of >5 years, and low tenure as dissolved, <2 year for comparison

[15]:
```
def creating_tenure_bucket(tenure, status):
    if tenure>=5:
        return 'High Tenure'
    elif tenure<=2 and status=='dissolved':
        return 'Low Tenure'
    else:
        return 'Others'
```

[16]:
```
hairdressers['tenure_type'] = hairdressers.apply(lambda x:␣
 ↪creating_tenure_bucket(x['tenure'],x['company_status']), axis=1)
print('Created Buckets for High/Low Tenure')
print('Identified '+str(hairdressers[hairdressers['tenure_type'] == 'High␣
 ↪Tenure'].shape[0])+' High Tenure and␣
 ↪'+str(hairdressers[hairdressers['tenure_type'] == 'Low Tenure'].shape[0])+'␣
 ↪Low Tenure Brands')
```

```
Created Buckets for High/Low Tenure
Identified 19399 High Tenure and 21104 Low Tenure Brands
```

0.6 Analysing Names

[17]:
```
words_to_remove = [' LIMITED', ' LTD', ' ENTERPRISES', ' ENTERPRISE']
def cleaning_names(name):
    #Removing Regular Company Suffixes
    for word in words_to_remove:
        name = name.replace(word,'')
    list_of_name = list()
    #Removing words with 1 character
    for word in name.split():
        if len(word) > 1:
            list_of_name.append(word)
    name = ' '.join(list_of_name)
    return name
print('Created Function to Clean Names')
```

```
Created Function to Clean Names
```

[18]:
```
hairdressers['company_name_cleaned'] = hairdressers['company_name'].
 ↪map(cleaning_names)
hairdressers['words_in_name'] = hairdressers['company_name'].str.split().str.
 ↪len()
print('Cleaned Names and Counted # of Words')
```

```
Cleaned Names and Counted # of Words
```

[19]:
```
def most_frequent_words(dataframe, tenure_type):
    dataframe = dataframe[dataframe['tenure_type'] == tenure_type].reset_index()
    combining_all_names = ''
    for i in range(dataframe.shape[0]):
        combining_all_names = combining_all_names+' '+dataframe.
 ↪loc[i]['company_name_cleaned']
    print('Combining all names')

    split_words = combining_all_names.split()
    print('Splitting into List')
    from collections import Counter
    counter = Counter(split_words)
    most_occur_100 = counter.most_common(100)
    most_occur_100 = dict(most_occur_100)
    print('Getting Most Frequest Words for '+tenure_type)
    return most_occur_100
print('Created Function to get Most Frequent Words')
```

```
Created Function to get Most Frequent Words
```

[20]:
```
most_occur_high = most_frequent_words(hairdressers,'High Tenure')
most_occur_low = most_frequent_words(hairdressers,'Low Tenure')
```

```
Combining all names
Splitting into List
Getting Most Frequest Words for High Tenure
Combining all names
Splitting into List
Getting Most Frequest Words for Low Tenure
```

[21]:
```
def conv_dict_to_df(dictionary, tenure_type):
    dataframe = pd.DataFrame(dictionary.items(), columns=['Word', 'Frequency'])
    dataframe = dataframe.sort_values('Frequency')
    dataframe['tenure_type'] = tenure_type
    return dataframe
print('Created function to convert dictionary to dataframe')
```

```
Created function to convert dictionary to dataframe
```

[22]:
```
most_occur_high_df = conv_dict_to_df(most_occur_high, 'High Tenure')
most_occur_low_df = conv_dict_to_df(most_occur_low, 'Low Tenure')
high_not_low_words = [k for k in most_occur_high if k not in most_occur_low]
low_not_high_words = [k for k in most_occur_low if k not in most_occur_high]
```

[23]:
```
most_occur_df = pd.concat([most_occur_high_df,most_occur_low_df])
print('Collating Most Frequent Words for High & Low Tenure Businesses')
```

```
Collating Most Frequent Words for High & Low Tenure Businesses
```

[24]:
```
most_occur_high_df['only_in_high'] = most_occur_high_df['Word'].map(lambda x: x␣
 ↪in high_not_low_words)
print('Adding Flag to denote if word occurs among low tenure names')
most_occur_low_df['only_in_low'] = most_occur_low_df['Word'].map(lambda x: x in␣
 ↪low_not_high_words)
print('Adding Flag to denote if word occurs among high tenure names')
```

```
Adding Flag to denote if word occurs among low tenure names
Adding Flag to denote if word occurs among high tenure names
```

0.7 Visualising Results

[25]:
```
graph1 = px.box(hairdressers, x='tenure_type', y='words_in_name',
             color='tenure_type', title =' Graph 1: # of Words across Tenure␣
 ↪Type')
graph1.show('notebook')
```

[26]:
```
graph2 = px.histogram(most_occur_df, x="Word", y="Frequency",␣
 ↪color="tenure_type", opacity=0.8,
                   title='Graph 2: Top Words - High vs Low Tenure', height=800,␣
 ↪width=1000)
```

```
graph2.show('notebook')
```

[27]:
```
def plot_word_cloud(dataframe, demarcator, color):
    dict_of_freq = {}
    only_specific_tenure_words = dataframe[dataframe[demarcator]==True].
↪reset_index()

    for index in range(only_specific_tenure_words.shape[0]):
        dict_of_freq[only_specific_tenure_words.loc[index]['Word']] =␣
↪only_specific_tenure_words.loc[index]['Frequency']
    wordcloud = WordCloud(background_color='white',width=1600, height=800,␣
↪colormap=color)
    wordcloud.generate_from_frequencies(dict_of_freq)
    plt.figure()
    plt.figure(figsize=(10,5), facecolor='k')
    plt.imshow(wordcloud)
    plt.axis("off")
    print(demarcator.upper())
```

0.7.1 Business Question 2: Mini-report and visualisation (252 Words):

0.7.2 Problem Statement:

A **name can make or break a business**. A potential entrant to the **hairdressing market has approached us for recommendations on how to name the brand to facilitate a long-run**. This report identifies hairdressers from history that have had a **high and low tenure**. This is followed by a **comparative analysis on the length and keywords among these names to come up with recommendations for our clients**.

0.7.3 Insights Drawn:

The median tenure across localities seems to be 3 years, while the 3rd quartile ranges between 4-5 years. Hence, we define long-lasting as a tenure of >5 years, and low tenure as dissolved in <2 years for comparison

Three to Four Words is the Norm As seen in Graph 1, while there is no significant difference between the # of words in names of high and low tenure hairdressers, 3-4 words seems to be the usual length

DOs Words such as 'Hairdressers', 'Centre, 'Cutting', 'Serenity, 'Retreat', 'Pure, 'Gents, 'Creative and 'Rooms are frequent among the high-tenure brand names, but don't appear among the low-tenure brands, as seen in Graph 2

DON'Ts Words such as 'Lashes', 'Glow', 'Pretty', 'Mobile', 'Aesthetic', 'Natural', 'Touch', 'Massage', 'Cutz', 'Pretty' and 'Golden' are frequent among the low-tenure brand names, but don't appear among the high-tenure ones, as seen in Graph 2

Hairdressers, Studios & Centre seem to be more prevalent among high tenure brands as opposed to Club and Saloon among low tenure ones. Adjectives such as Pure, Creative, Unique, Professional, Permanent as recommended as well.

[28]:
```
graph1.show('notebook')
```

[29]:
```
print('\033[1mGraph 2 - High vs Low Tenure exclusive Words')
plot_word_cloud(most_occur_high_df,'only_in_high', 'Greens')
plot_word_cloud(most_occur_low_df,'only_in_low', 'Reds')
```

Graph 2 - High vs Low Tenure exclusive Words

ONLY_IN_HIGH

ONLY_IN_LOW

<Figure size 432x288 with 0 Axes>

<Figure size 432x288 with 0 Axes>


AESTHETIC
COSMETIC
GLOW
ELITE
CLUB
MASSAGE
MOBILE
BELLA
CUTZ
SKINCARE
GLAMOUR
LUXE
LASHES
LOCKS
LOVE
ALL
BOX
TATTOO
INTERNATIONAL
SCISSORS
SALOON
ON
PRETTY
TOUCH
NATURAL
GOLDEN