# Analysing customer reviews using innovative data science techniques - a Loqbox project

## Analytics Problem

In this project, we have been working alongside client Loqbox, a local financial technology company that works with selected lenders to help boost their credit score. To help do this, Loqbox offers a tiered membership service to their customers. Firstly, they offer a Lite membership, which is free and provides customers with access to just the Loqbox 'Save' membership feature. Alternatively, they offer a Full membership service which customers must pay for, either weekly or monthly, and offers access to 3 extra membership features, those being Loqbox 'Spend', 'Rent' and 'Coach'. Note that customers only pay monthly when they choose to activate the Loqbox 'Spend' membership feature.

The business problem Loqbox are facing is that thousands of customers are downgrading from the paid membership platform to the free alternative. Therefore, the problem Loqbox are experiencing surrounds their customer satisfaction and retention rates. They wish to resolve this issue so that they can increase their corporate revenues, boost their corporate reputation, and become more competitive in the market.

When downgrading, customers leave feedback providing their reason, or reasons, for leaving the paid membership service, information provided to us by the client. Alongside this, the data set provided also contains information regarding the upgrade and downgrade dates and times, and the membership features each customer chose to activate during their membership period. From the upgrade and downgrade dates and times we were able to extract the duration of each customer's relationship with Loqbox's membership scheme. Further, information regarding the activation of membership features was presented in categorical Boolean columns. On top of this, there was also a column which provided data in a dictionary, array structure which we were required to reformat to extract information on the reasons customers gave for choosing to leave the membership service.

Our analytics problem has been to identify relationships between the duration of a customer's membership, the features they were utilising, and the reasons they gave for leaving the membership service. Furthermore, where customers provide feedback regarding their dissatisfaction of the product, we have been required to analyse vast amounts of text data to provide meaningful insights on why customers are choosing to leave the membership platform.

The aim of our analysis is to utilise quantitative and qualitative analytic techniques to produce descriptive and diagnostic data visualisations. We have done this with the aim of establishing relationships between key variables, so that we can propose impactful recommendations to the reformation of the membership scheme. A reformation of the membership scheme aims to improve customer satisfaction and retention rates and extend the mean average duration that a customer holds their membership for. An increased membership duration means customers are happier with the service and more likely to recommend it to others. Furthermore, it will also lead to customers paying membership fees for a longer period, thus benefitting our client by increasing their corporate revenues and their competitiveness within the market.

**Analytics Solution**

Based on existing studies on customer review analysis (Vecchio et al., 2018; Cheng and Jin, 2019; Lee et al., 2019), the methodology in this report consists of three stages: data cleaning, data pre-processing, and data analysis (Figure 1). After cleaning and pre-processing the initial data set, we conducted descriptive analysis to evaluate the performance of each feature with membership duration as a proxy, as well as diagnostic analysis to identify the most popular reasons for membership downgrades.

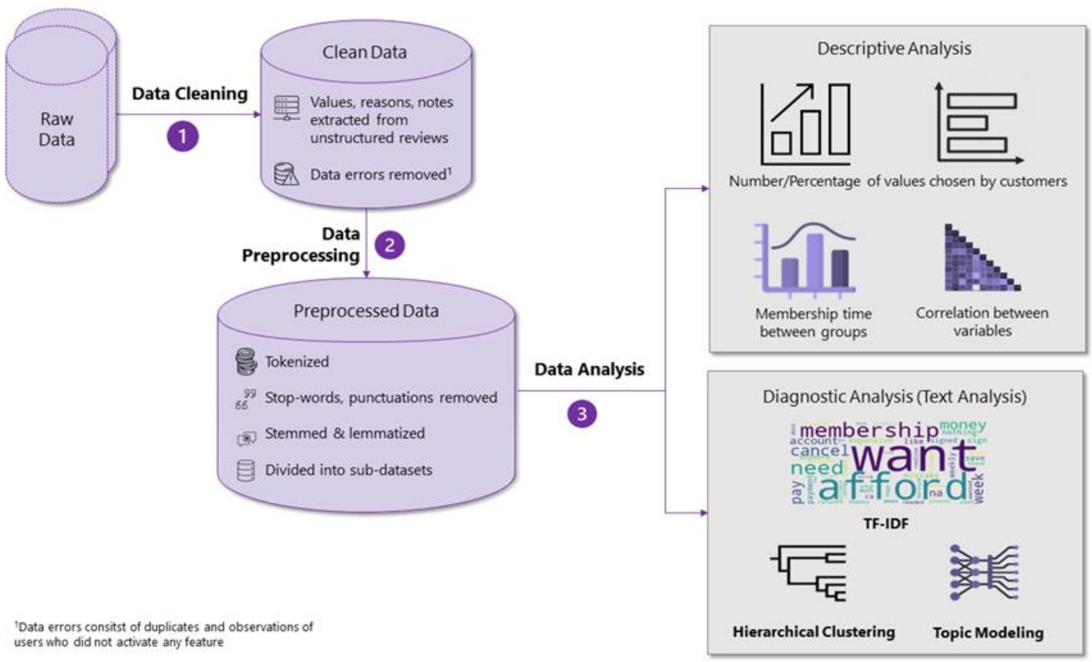



*Figure 1 – Summary of methodology*

The first notable observation we were able to make was that most customers downgrade their Loqbox membership within the first month (Figure 2). Moreover, 50% of customers downgrade in the first 14 days since this is a refundable trial period.

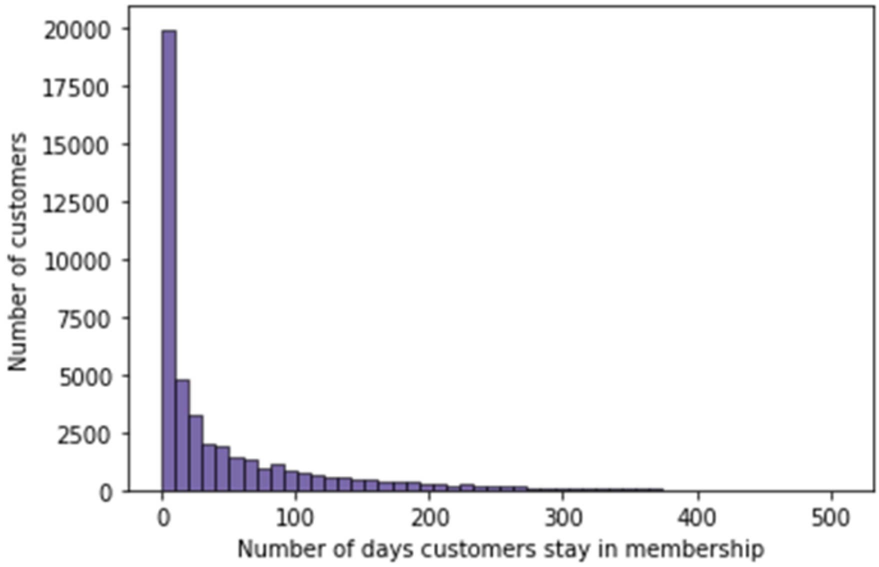



*Figure 2 – Membership length distribution*

The descriptive results, presented in Figure 3, suggest that the greater the number of features a customer activates, the longer their expected membership duration. Furthermore, among all combinations of the 'Save' option with another premium feature, 'Save' & 'Spend' presents itself as the most outstanding bundle with the highest difference in membership duration between those who have them activated and those who did not. This finding shows that out of all features, 'Spend' is most optimal for boosting customer retention, interestingly 'Spend' is the only feature which enforces customers to pay monthly rather than weekly.

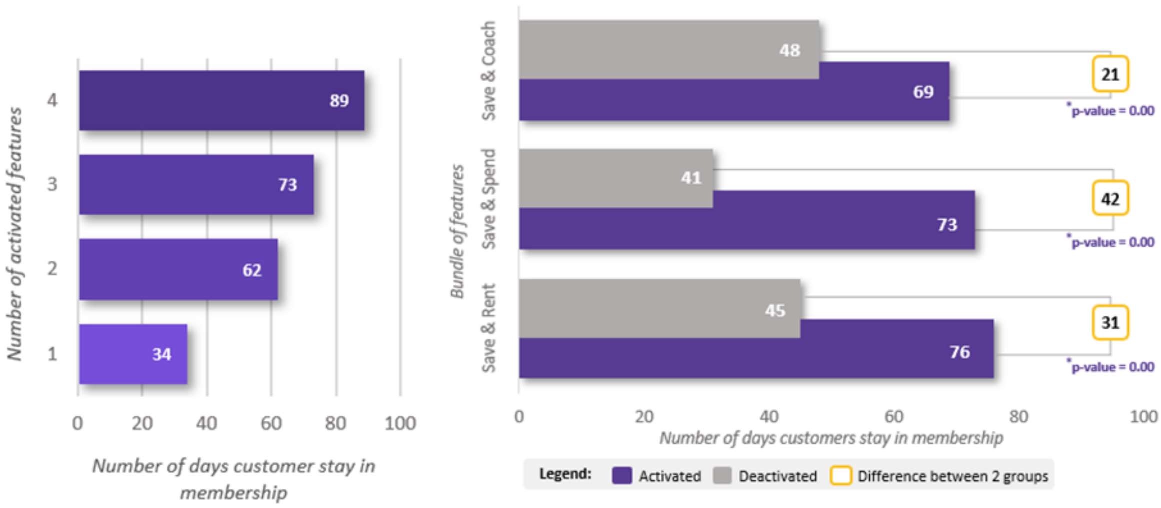



*Figure 3 – Membership length by number of activated features*

Our diagnostic analytic results provide insights into what motivations most often lie behind customer downgrades. Firstly, Figure 4 presents a word cloud where the size of each word is determined by the frequency used by customers, which provides our client with an informative overview of customer concerns.


*Figure 4 – Word cloud of most frequent words*

To develop more detailed insights, words must be clustered into groups for analysis (Lee et al., 2019). To do this, we chose second-order clustering (Lee et al., 2019) for its capability to group synonyms or different forms of a word into the same cluster. From this we formed 4 clusters, which Figure 5 displays the ensuing average-linkage hierarchical clustering results.

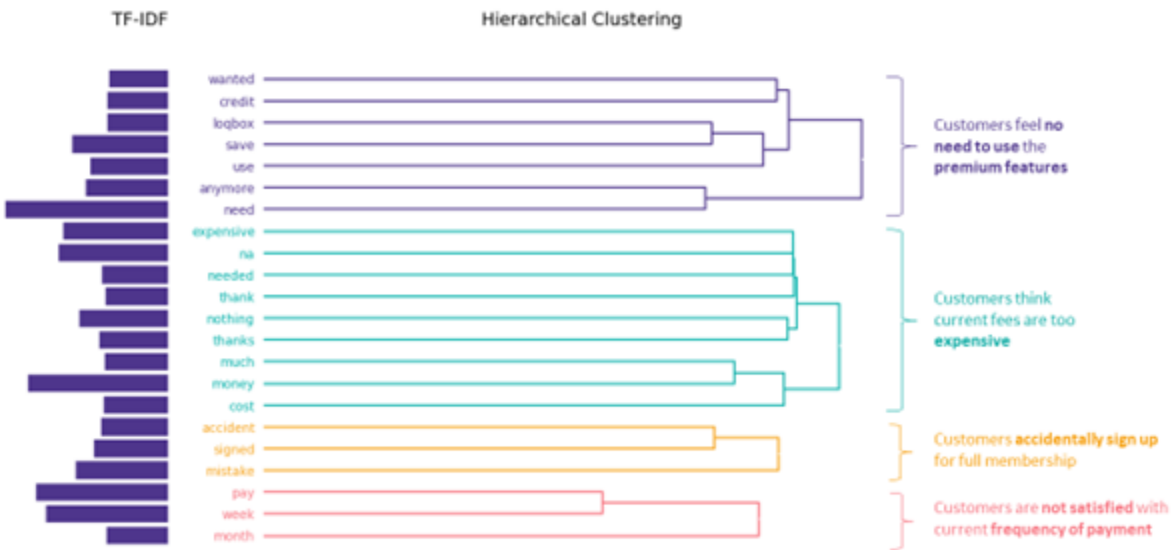



*Figure 5 – TF-IDF and Hierarchical Clustering results*

From our hierarchical clustering analysis, we computed the proportion of customers choosing each reason as their motivation for downgrading. Figure 6 presents our results, and from these we deduce that dissatisfaction with the membership fee and no longer needing the membership features are the most popular reasons for membership downgrades.

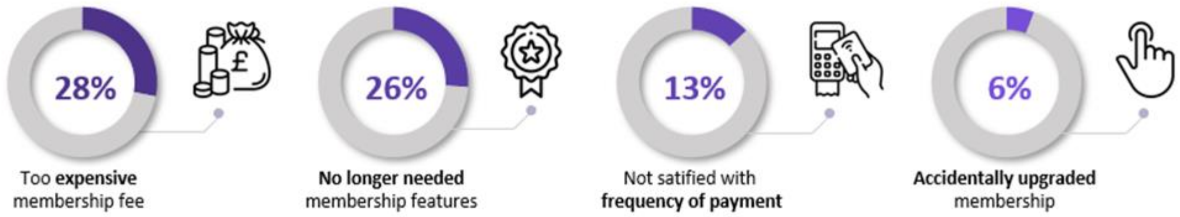



*Figure 6 – Most common downgrade reasons*

Based on above findings, we propose analytic solutions to solve our business problem. Firstly, Loqbox should offer rewards or discounts to customers who activate bundles of features to combat the customers who feel the membership is too expensive. Secondly, price packages should be tailored to the active number of features, as this can connect customers emotionally and intellectually to the organisation (Gruen et al., 2000). Additionally, to resolve customer concerns surrounding the need for extra features, Loqbox should evaluate their effectiveness through a comprehensive survey. Furthermore, more frequent payment cycles appear to increase a customer's likelihood to churn, therefore we suggest Loqbox transpose their cycles from weekly payments to allowing customers to adopt monthly or yearly payment plans. Finally, we suggest improving the user interface to allow customers to familiarise themselves with their membership options and prevent accidental sign-ups.

Additionally, the current questionnaire is limited to revealing the most common downgrade reasons. Therefore, for further customer retention enhancement, Loqbox should design a more quantitative and comprehensive downgrade questionnaire so that they can conduct regression analyses to evaluate customer concerns more effectively. For this, we recommend a 5-Likert scale, as presented in Figure 7.

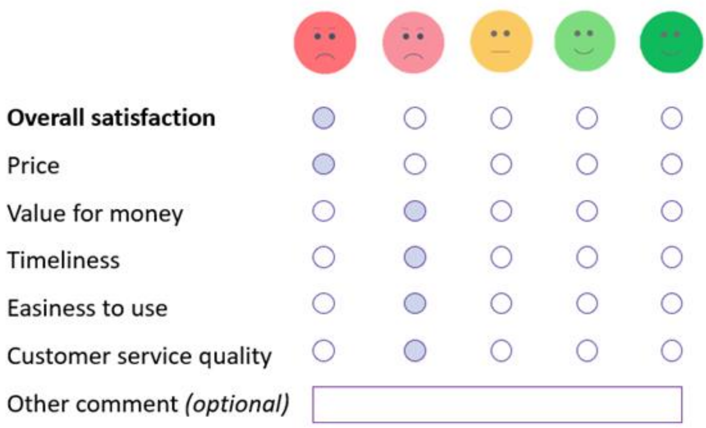



*Figure 7 – An example of 5-Likert quantitative survey*

**Technical Development**

Throughout this project, we have developed our technical skills and knowledge in multiple ways. We continue by highlighting three integral technical developments. Firstly, we have learnt how to effectively utilise Python classes to ensure replicability of work. Secondly, we have researched and implemented new analytical methodologies, which led to the application of Python libraries and functions, with which we were previously unfamiliar. Finally, we have developed our client interaction skills, by implementing new presentation styles.

In the preliminary stages of this project, we met with Loqbox to discuss their vision of what shape our product would take. One of the primary requests put forward by the client was to ensure our data work could become replicable for future customer feedback datasets. This request required us to ensure all data cleaning, processing, and visualisations could be generalised into replicable functions. With our data cleaning phase, we conducted research into the usage and application of the '*DateTime*' library for obtaining the duration of each customer's membership and the '*JSON*' library for disassembling the dictionary, array format of the customer downgrade feedback. After completing this, we decided the best plan of action would be to present all data cleaning and future processing functions in a Python class. To do this we taught ourselves how to implement a class, this entailed converting all code into functions and then into methods. From this we were able to form our data cleaning and processing classes which would be fully automated and replicable for future datasets.

Moving on to data pre-processing phase, our dataset presented us with a large body of unstructured text data which, as a team, we had no experience of working with before. Therefore, we needed to learn and apply new analytic techniques. Firstly, we removed stop words from text to eliminate noise from the data. Then, we performed tokenisation to extract each unique word from the text and undertook lemmatisation and stemming to obtain the root of all words so that they can be clustered more efficiently.

As the project progressed, we engaged in the data analysis phase, which required us to plan what our visualisations would look like to ensure optimal findings could be identified. We completed a vast amount of research to identify useful methodologies for analysis. The methods we discovered to be most optimal for this project were TF-IDF, hierarchical clustering, and topic modelling. Implementing these analytical techniques required us to learn and apply numerous new Python libraries and functions. For example, we used the *Gensim* algorithm to cluster words from the corpus to produce an intertopic distance map with multidimensional scaling to rank the clusters based on the frequency used in the customer downgrade data.

Finally, due to the nature of this project, we developed our technical consultancy skills. Most notably, we developed our communication skills by producing appealing visualisations to help present our findings and project updates to our client in the most concise way possible.

**Effective Teamwork**

From the first team meeting, the importance of effective teamwork was identified as a key contributor to accomplishing success when delivering outstanding deliverables to our client in our given time frame. As a team we immediately identified numerous factors which would constitute towards achieving effective teamwork. The factors are effective decision-making, collaboration, inclusivity, utilisation of individual strengths and experiences, transparent communication, flexibility, and accountability.

At multiple points throughout the project, we were required to collaborate effectively with each other to come to preferable conclusions when making key decisions which would prove to be integral to succeeding in this project. A problem we encountered was when we were required to split into sub teams to complete tasks regarding the data cleaning, processing, and analysis phases. To split teams most optimally we decided to make decisions by consensus, that is that all team members discussed their experiences and strengths and identified their preferred allocation. The ensuing discussion ensured that all team members were heard in equal measure and that we engaged in inclusive practices. Furthermore, this method of allocation ensured that all team members were at least satisfied with the responsibilities they would be taking on and that individual workloads were managed effectively. For the more technical decisions we made we utilised decision by authority, that is that we were able to make optimal decisions by exploiting the knowledge that more experienced members of the team were able to bring to the table.

When working in sub teams we made sure to engage all team members by utilising a transparent information sharing strategy and ensured we were open to flexible project and time management. To ensure consistency with transparent communication we created a Microsoft Teams group and chat to regularly inform team members of progressions made and challenges faced. Furthermore, all data work was carried out via Google Collaboratory to ensure all members had access to the latest progression in the work, and weekly face-to-face meetings were conducted to keep stakeholders updated. Moreover, transparent communication with our client and supervisor was ensured by exchanging regular emails, along with conducting virtual biweekly progression update meetings to ensure the team were working in conjunction with project objectives and our client's vision. Our communication strategy also ensured that when problems occurred, we were able to respond quickly and effectively through collaboration with all team members, and that we were able to respond flexibly and alter timeline objectives to ensure all work could be completed precisely and thoroughly.

Finally, our team decision-making strategy regarding the sub-team division ensured all team members felt clarity with individual roles and responsibilities. This clarity ensured that all constituents felt comfortable taking responsibility for their workloads, that individual strengths were utilised most effectively, and that all timeline objectives were met as we were all aware of our individual goals and time constraints. All work was performed on schedule and to an extraordinary standard, demonstrating effective project management. The opportunity to take responsibility for our work guaranteed that all team members knew the project's ongoings.

## References

Cheng, M. and Jin, X. (2019) 'What do Airbnb users care about? An analysis of online review comments', *International Journal of Hospitality Management*, 76, pp. 58–70. Available at: https://doi.org/10.1016/j.ijhm.2018.04.004.

Gruen, T.W., Summers, J.O. and Acito, F. (2000) 'Relationship Marketing Activities, Commitment, and Membership Behaviors in Professional Associations', *Journal of Marketing*, 64(3), pp. 34–49. Available at: https://doi.org/10.1509/jmkg.64.3.34.18030.

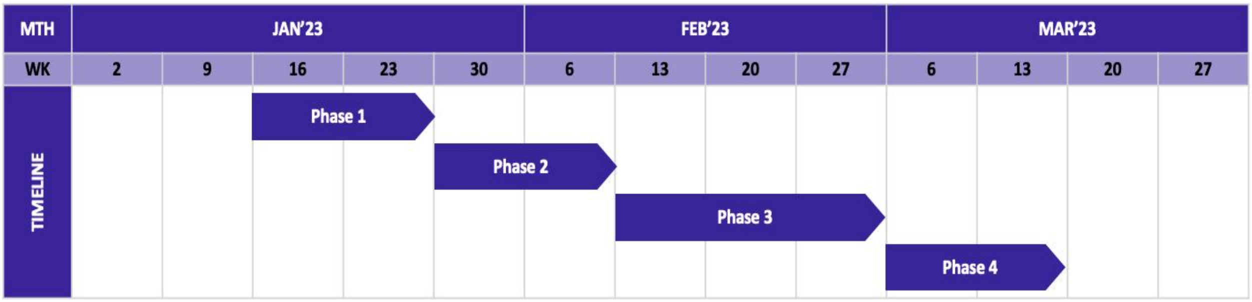
Lee, C.K.H. *et al.* (2019) 'Analysing online reviews to investigate customer behaviour in the sharing economy: The case of Airbnb', *Information Technology & People*, 33(3), pp. 945–961. Available at: https://doi.org/10.1108/ITP-10-2018-0475.

Vecchio, P.D. *et al.* (2018) 'Creating value from Social Big Data: Implications for Smart Tourism Destinations', *Information Processing & Management*, 54(5), pp. 847–860. Available at: https://doi.org/10.1016/j.ipm.2017.10.006.

## APPENDIX I – ABBREVIATIONS

| Abbreviations | Definition |
|---|---|
| UX/UI | User Experience/ User Interface |
| TF-IDF | Term frequency–inverse document frequency |

## APPENDIX II – PROJECT TIMELINE




| Phase | Activities & Deliverables |
|---|---|
| **Phase 1**<br>Data collection | • Collect the data.<br>• Initial review of data set – identify insights and concerns to discuss with client.<br>• Initiate data cleaning phase by creating a collaborative Python workbook. |
| **Phase 2**<br>Data preprocessing | • Finalise data cleaning phase and initiate data analysis phase.<br>• Identify key methodology for analysis and visualisation. |
| **Phase 3**<br>Data analysis | • Finalise data analysis phase.<br>• Integrate predictive and prescriptive analytic techniques to model an optimal membership scheme.<br>• Finalise modelling analytics phase. |
| **Phase 4**<br>Project report and presentation | • Prepare and finalise project report and presentation.<br>• Offer business insights and recommendations to client. |

## APPENDIX III – VARIABLES IN ORIGINAL DATASET

| Field name | Definition |
|---|---|
| customer_id | Unique reference per member |
| upgraded_at | Timestamp when the member upgraded to Full membership |
| downgraded_at | Timestamp when the member downgraded to Lite membership |
| activated_save | Had the member activated Loqbox Save before they downgraded? 1 = Yes, 0 = No |
| activated_spend | Had the member activated Loqbox Spend before they downgraded? 1 = Yes, 0 = No |
| activated_rent | Had the member activated Loqbox Rent before they downgraded? 1 = Yes, 0 = No |
| activated_coach | Had the member activated Loqbox Coach before they downgraded? 1 = Yes, 0 = No |
| update_reason* | Why did the member downgrade? |

## APPENDIX IV – DEFINITIONS OF SELECTED DOWNGRADING REASONS

| Value | Definition of old version values valid to 7/20/2022 2:30:00 PM |
|---|---|
| 0 | My current plan is too expensive |
| 1 | My current plan isn't good value for money |
| 2 | I don't want to pay for my membership |
| 3 | I'm no longer using Loqbox |
| 4 | I can now access the financial products I need |
| 5 | I've reached the credit score I wanted |
| 6 | Loqbox isn't what I thought I'd signed up for |
| 7 | None of the above |

| Value | Definition of new version valid from 7/20/2022 2:30:00 PM |
|---|---|
| 0 | I'm trying to save as much as I can |
| 1 | I'm not using the extra features that full membership gives me |
| 2 | It's not good value for money |
| 3 | Weekly billing doesn't suit me |
| 4 | I didn't intend to take a full membership |
| 5 | None of the above |

# Import libraries and data

In [ ]:
```
#Install packages
!pip install spacy python -m spacy download en #Language model
!pip install gensim # For topic modeling
!pip install --upgrade gensim==4.0.0
!pip install pyLDAvis # For visualizing topic models
!pip install fastcluster
!pip install stop_words
```

```
Usage:
  pip3 install [options] <requirement specifier> [package-index-options] ...
  pip3 install [options] -r <requirements file> [package-index-options] ...
  pip3 install [options] [-e] <vcs project url> ...
  pip3 install [options] [-e] <local project path> ...
  pip3 install [options] <archive url/path> ...

no such option: -m
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gensim in /usr/local/lib/python3.9/dist-packages (3.6.0)
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.9/dist-packages (from gensim) (6.3.0)
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.9/dist-packages (from gensim) (1.10.1)
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.9/dist-packages (from gensim) (1.22.4)
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.9/dist-packages (from gensim) (1.16.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting gensim==4.0.0
  Downloading gensim-4.0.0.tar.gz (23.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.1/23.1 MB 53.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.9/dist-packages (from gensim==4.0.0) (1.22.4)
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.9/dist-packages (from gensim==4.0.0) (1.10.1)
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.9/dist-packages (from gensim==4.0.0) (6.
3.0)
Building wheels for collected packages: gensim
  Building wheel for gensim (setup.py) ... done
  Created wheel for gensim: filename=gensim-4.0.0-cp39-cp39-linux_x86_64.whl size=26057836 sha256=742e4424b92777277f
3244adc73da643dcd147f4b30fc65e8442cf7b02af549e
  Stored in directory: /root/.cache/pip/wheels/58/8b/5f/53deafbdad45cf0d3d3c0189d1f29b309bfd6950abe2f58a70
Successfully built gensim
Installing collected packages: gensim
  Attempting uninstall: gensim
    Found existing installation: gensim 3.6.0
    Uninstalling gensim-3.6.0:
      Successfully uninstalled gensim-3.6.0
Successfully installed gensim-4.0.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyLDAvis
  Downloading pyLDAvis-3.4.0-py3-none-any.whl (2.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.6/2.6 MB 23.8 MB/s eta 0:00:00
Collecting funcy
  Downloading funcy-1.18-py2.py3-none-any.whl (33 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (1.10.1)
```

```
Requirement already satisfied: numpy>=1.22.0 in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (1.22.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (63.4.3)
Requirement already satisfied: gensim in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (4.0.0)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (1.2.2)
Requirement already satisfied: pandas>=1.3.4 in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (1.4.4)
Requirement already satisfied: numexpr in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (2.8.4)
Collecting joblib>=1.2.0
  Downloading joblib-1.2.0-py3-none-any.whl (297 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 298.0/298.0 KB 18.9 MB/s eta 0:00:00
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from pyLDAvis) (3.1.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=1.3.4->pyLDAvis)
(2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas>=1.3.4-
>pyLDAvis) (2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=1.
0.0->pyLDAvis) (3.1.0)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.9/dist-packages (from gensim->pyLDAvis)
(6.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from jinja2->pyLDAvis) (2.
1.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pand
as>=1.3.4->pyLDAvis) (1.16.0)
Installing collected packages: funcy, joblib, pyLDAvis
  Attempting uninstall: joblib
    Found existing installation: joblib 1.1.1
    Uninstalling joblib-1.1.1:
      Successfully uninstalled joblib-1.1.1
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This beha
viour is the source of the following dependency conflicts.
pandas-profiling 3.2.0 requires joblib~=1.1.0, but you have joblib 1.2.0 which is incompatible.
Successfully installed funcy-1.18 joblib-1.2.0 pyLDAvis-3.4.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting fastcluster
  Downloading fastcluster-1.2.6-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014
_x86_64.whl (193 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 193.7/193.7 KB 4.8 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.9 in /usr/local/lib/python3.9/dist-packages (from fastcluster) (1.22.4)
Installing collected packages: fastcluster
Successfully installed fastcluster-1.2.6
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting stop_words
  Downloading stop-words-2018.7.23.tar.gz (31 kB)
```

```
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: stop_words
  Building wheel for stop_words (setup.py) ... done
  Created wheel for stop_words: filename=stop_words-2018.7.23-py3-none-any.whl size=32910 sha256=344b0b096b48454f176
296222a7af072fb3026b47838edec9f45cb4a3ca20f84
  Stored in directory: /root/.cache/pip/wheels/da/d8/66/395317506a23a9d1d7de433ad6a7d9e6e16aab48cf028a0f60
Successfully built stop_words
Installing collected packages: stop_words
Successfully installed stop_words-2018.7.23
```

In [ ]:

```
#Install libraries
import pandas as pd
import datetime
import json
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import re
import stop_words
import scipy
import scipy.cluster.hierarchy as sch
import nltk
import gensim
import gensim.corpora as corpora
import spacy# Plotting tools
import pyLDAvis
import pyLDAvis.gensim
import pyLDAvis.sklearn
import panel as pn

from sklearn.feature_extraction.text import TfidfVectorizer
from scipy.cluster.hierarchy import ward, dendrogram, fcluster, leaves_list, set_link_color_palette
from wordcloud import WordCloud, STOPWORDS
from fastcluster import linkage
from nltk import word_tokenize,sent_tokenize
from nltk.stem import WordNetLemmatizer, PorterStemmer
from stop_words import get_stop_words
from nltk.corpus import stopwords
from sklearn.metrics.pairwise import cosine_similarity
from pprint import pprint# Gensim
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel# spaCy for preprocessing
```

```
from sklearn.decomposition import NMF
from sklearn.feature_extraction.text import CountVectorizer

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
%matplotlib inline
```

```
/usr/local/lib/python3.9/dist-packages/gensim/similarities/__init__.py:15: UserWarning: The gensim.similarities.leve
nshtein submodule is disabled, because the optional Levenshtein package <https://pypi.org/project/python-Levenshtei
n/> is unavailable. Install Levenhstein (e.g. `pip install python-Levenshtein`) to suppress this warning.
  warnings.warn(msg)
/usr/local/lib/python3.9/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

In [ ]:
```
print('Which downgrade data file would you like to access?')
value = input()
```

```
Which downgrade data file would you like to access?
https://raw.githubusercontent.com/MatticusBa/consultinggroup/main/DATA-462%20Loqbox%20Analytics%20Project%20-%20Memb
ership%20Downgrades%20Data%20(1).csv?token=GHSAT0AAAAAAB6DKJ2W5A53SX7P64ZONFEMY6X7Q2Q
```

## Our file name

https://raw.githubusercontent.com/MatticusBa/consultinggroup/main/DATA-462%20Loqbox%20Analytics%20Project%20-%20Membership%20Downgrades%20Data%20(1).csv?token=GHSAT0AAAAAAB6DKJ2W5A53SX7P64ZONFEMY6X7Q2Q

In [ ]:
```
df = pd.read_csv(value)
```

# 1. Data Cleaning

In [ ]:
```
class data_cleaning:

    def __init__(self, df):
        self.df = df

    def view_df(self, value=10):
        self.value = value
        return self.df.head(value)

    def invalid_rows(self):
        self.df = self.df[(self.df['activated_save']!=0)|(self.df['activated_spend']!=0)|(self.df['activated_rent']!=
        return self.df

    def split_reasons(self):
        self.df['update_reason_dict'] = self.df['update_reason'].str[2:].str[:-2]
        self.df['update_reason_dict'] = self.df['update_reason_dict'].str.split("}, {")

        self.df = self.df.explode('update_reason_dict')
        self.df['update_reason_dict'] = "{" + self.df['update_reason_dict'] + "}"


        a = self.df[self.df.update_reason_dict == "{}"]
        b = self.df[self.df.update_reason_dict != "{}"].dropna()
        c = self.df[self.df.update_reason_dict.isna()]

        b2 = pd.concat([b, b.update_reason_dict.apply(json.loads).apply(pd.Series)], axis=1)

        self.df = pd.concat([a, b2, c])[["customer_id", "upgraded_at", "downgraded_at", "activated_save", "activated_

        return self.df

    def reason_sort(self):
        x = self.df.copy()

        self.df['value'] = self.df['value'].astype(str)
        self.df = self.df.groupby(["customer_id","upgraded_at", "downgraded_at", "activated_save", "activated_spend",
        self.df = pd.DataFrame(self.df)
        self.df.columns=['customer_id','upgraded_at', "downgraded_at", "activated_save", "activated_spend", "activate

        for i in range(8):
            self.df.loc[self.df['values'].str.contains(str(i)), ['value_'+str(i)]]="1"
```

```
            self.df['value_'+str(i)] = self.df['value_'+str(i)].fillna("0")

        x = x.pivot(index=['customer_id','upgraded_at','note'], columns='value', values=['reason'])
        x.columns = ['_'.join([str(i)for i in col]).strip() for col in x.columns.values]
        x = x.reset_index(level=None, drop=False, inplace=False, col_level=0, col_fill='')

        self.df = pd.merge(self.df, x)

        self.df = self.df.drop(columns= ['reason_3', 'reason_5.0', 'values'])

        return self.df

    def reorder_columns(self):
        value_cols = [col for col in self.df.columns if 'value_' in col]
        reason_cols = [col for col in self.df.columns if 'reason_' in col]
        dropcols = value_cols + reason_cols
        cols = [col for col in self.df.columns if col not in dropcols]
        new_order = []
        for i in range(len(value_cols)):
            new_order.append('value_'+str(i)+'')
            new_order.append('reason_'+str(i))
        new_order.append('reason_nan')
        self.df = self.df.reindex(columns= cols + new_order)
        return self.df

    def to_dataframe(self):
        df = pd.DataFrame(self.df)
        return df
```

In [ ]:

```
loq = data_cleaning(df)

loq.invalid_rows()
loq.split_reasons()
loq.reason_sort()
loq.reorder_columns()
loq = loq.to_dataframe()
loq.head(5)
```

```
<ipython-input-5-ad99dd6f3c33>:15: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy
  self.df['update_reason_dict'] = self.df['update_reason'].str[2:].str[:-2]
<ipython-input-5-ad99dd6f3c33>:16: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy
  self.df['update_reason_dict'] = self.df['update_reason_dict'].str.split("}, {")
```

Out[ ]:

| | customer_id | upgraded_at | downgraded_at | activated_save | activated_spend | activated_rent | activated_coach | note | value_0 | reason_( |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 18 | 2022-06-06 11:05:27.000 | 2022-11-16 20:34:41.000 | 1 | 0 | 1 | 0 | NaN | 0 | NaN |
| **1** | 252 | 2022-02-12 09:23:22.000 | 2022-03-14 16:17:01.000 | 1 | 0 | 0 | 0 | `Member wants to proceed with Save only | 0 | NaN |
| **2** | 267 | 2022-03-01 08:49:00.000 | 2022-05-08 12:37:56.000 | 1 | 0 | 0 | 0 | NaN | 1 | My curren plan i: too expensive |
| **3** | 332 | 2022-08-05 23:25:15.000 | 2022-08-25 15:52:59.000 | 1 | 1 | 0 | 0 | NaN | 0 | NaN |
| **4** | 445 | 2022-05-27 07:41:27.000 | 2022-05-27 07:43:36.000 | 1 | 0 | 0 | 1 | NaN | 1 | My curren plan i: too expensive |

5 rows × 25 columns

# 2. Data pre-processing

## 2.1 For descriptive analysis

In [ ]:
```
class data_processing_num:

    def __init__(self, df):
        self.df = df
```

```
    def find_interval(self):
        self.df['downgraded_at'] = self.df.downgraded_at.astype('datetime64')
        self.df['upgraded_at'] = self.df.upgraded_at.astype('datetime64')
        self.df['interval'] = (self.df['downgraded_at'] - self.df['upgraded_at']).dt.days.astype(int)
        return self.df

    def find_new_vals(self):
        self.df['new_vals'] = self.df['downgraded_at'] >=datetime.datetime(2022,7,20,14,30)
        return self.df

    def find_notes(self):
        self.df['is_note'] = self.df['note'].notna()
        return self.df

    def col_names(self):

        name_change = {'activated_save':'save',
                       'activated_spend':'spend',
                       'activated_rent':'rent',
                       'activated_coach':'coach'}

        self.df = self.df.rename(columns=name_change)
        return self.df

    def combine_notes(self):
        self.df['notes'] = self.df['note'].fillna(self.df['reason_nan'])
        self.df = self.df.drop(columns=['note', 'reason_nan'])
        return self.df

    def df_numerical(self):
        want = ['customer_id', 'interval', 'new_vals', 'is_note', 'save', 'spend', 'rent', 'coach']

        for i in range(8):
            want.append(f'value_{i}')

        self.df = self.df[want].astype(int)

        return self.df

    def interval_groups(self):
        groups = {range(0,15): 'money_back',
                  range(15,31): 'short',
```

```
                 range(31,91): 'medium',
                 range(91,10000): 'long'}

        self.df['length_member'] = self.df['interval'].replace(groups)
        return self.df

    def to_dataframe(self):
        df = pd.DataFrame(self.df)
        return df
```

In [ ]:
```
loqnum = data_processing_num(loq)

loqnum.find_interval()
loqnum.find_new_vals()
loqnum.find_notes()
loqnum.col_names()
loqnum.combine_notes()
loqnum.df_numerical()
loqnum.interval_groups()
loqnum = loqnum.to_dataframe()
```

In [ ]:
```
loqnum.head()
```

Out[ ]:

| | customer_id | interval | new_vals | is_note | save | spend | rent | coach | value_0 | value_1 | value_2 | value_3 | value_4 | value_5 | value_6 | valu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 18 | 163 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| **1** | 252 | 30 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| **2** | 267 | 68 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | |
| **3** | 332 | 19 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | |
| **4** | 445 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |

## 2.2 For diagnostic analysis

In [ ]:
```
loqtext = data_cleaning(df)
```

```
loqtext.invalid_rows()
loqtext.split_reasons()
loqtext = loqtext.to_dataframe()

loqtext = data_processing_num(loqtext)
loqtext.find_interval()
loqtext.find_new_vals()
loqtext = loqtext.to_dataframe()
loqtext.head(5)
```

```
<ipython-input-5-ad99dd6f3c33>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy
  self.df['update_reason_dict'] = self.df['update_reason'].str[2:].str[:-2]
<ipython-input-5-ad99dd6f3c33>:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy
  self.df['update_reason_dict'] = self.df['update_reason_dict'].str.split("}, {")
```

Out[ ]:

| | customer_id | upgraded_at | downgraded_at | activated_save | activated_spend | activated_rent | activated_coach | value | reason | note |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 18 | 2022-06-06 11:05:27 | 2022-11-16 20:34:41 | 1 | 0 | 1 | 0 | 1 | None | NaN |
| **1** | 252 | 2022-02-12 09:23:22 | 2022-03-14 16:17:01 | 1 | 0 | 0 | 0 | 2 | NaN | `Member wants to proceed with Save only |
| **2** | 267 | 2022-03-01 08:49:00 | 2022-05-08 12:37:56 | 1 | 0 | 0 | 0 | 2 | I’m not taking advantage of all the benefits o... | NaN |
| **2** | 267 | 2022-03-01 08:49:00 | 2022-05-08 12:37:56 | 1 | 0 | 0 | 0 | 1 | My current plan isn’t good value for money | NaN |
| **2** | 267 | 2022-03-01 08:49:00 | 2022-05-08 12:37:56 | 1 | 0 | 0 | 0 | 0 | My current plan is too expensive | NaN |

In [ ]:

```
# Get stop word list
stop_words = list(get_stop_words('en'))
nltk_words = list(stopwords.words('english'))
stop_words.extend(nltk_words)
```

In [ ]:

```
def preprocessing(df):
    df_old = df[df.new_vals == False]
    #Remove rows with NaN reason or non-NaN notes
    df_old_1 = df_old[(df_old["reason"].notna()) & (df_old["note"].isnull())]

    #Find written reasons by customers (not suggested by LOQBOX)
```

```
    distinct_reason = df_old_1.groupby(['value', 'reason']).size().reset_index(name='counts')
    written_reason = distinct_reason[distinct_reason.counts < 20]
    df_old_2 = df_old[(df_old["reason"].notna()) & (df_old["note"].isnull())].drop(columns="value").drop_duplicates(
    df_old_2 = df_old_2[df_old_2.reason.isin(written_reason.reason)]

    df_new = df[df.new_vals == True]
    df_new_2 = df_new[(df_new["reason"].notna()) & (df_new["note"].isnull())].drop(columns="value").drop_duplicates(

    df_preprocessed = pd.concat([df_new_2, df_old_2])

    #Tokenize - Lowercase
    df_preprocessed["reason_v2"] = df_preprocessed["reason"].str.lower().apply(word_tokenize)

    #Remove stop words
    df_preprocessed["reason_v3"] = df_preprocessed["reason_v2"].apply(lambda x: [word for word in x if word.isalpha(

    #Lemmatize
    lmtzr = WordNetLemmatizer()
    df_preprocessed["reason_lmt"] = df_preprocessed["reason_v3"].apply(lambda x: [lmtzr.lemmatize(word) for word in

    #Stemming
    ps = PorterStemmer()
    df_preprocessed["reason_stem"] = df_preprocessed["reason_v3"].apply(lambda x: [ps.stem(word) for word in x])

    df_preprocessed["len_reason"] = df_preprocessed["reason_lmt"].str.len()
    loqtext_preprocessed = df_preprocessed.drop(columns = ["reason_v2", "reason_v3"])


    return loqtext_preprocessed
```

In [ ]:
```
loqtext_preprocessed = preprocessing(loqtext)
```

# 3. Data analysis

## 3.1 Descriptive analysis

In [ ]:
```
def correlation_heatmap(df, var):
```

```
    if var == 'new':
        df = df[(df['new_vals']==1)&(df['is_note']==0)]
        df = df.drop(columns=['value_6', 'value_7'])

    elif var == 'old':
        df = df[(df['new_vals']==0)&(df['is_note']==0)]

    elif var == 'option':
        for i in range(8):
            df = df.drop(columns=[f'value_{i}'])

    plt.figure(figsize=(20,8))
    heatmap = sns.heatmap(df.drop(columns=['customer_id', 'new_vals', 'is_note']).corr(), vmin=-1, vmax=1, annot=True
    heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':20}, pad=15)
    return heatmap;
```

In [ ]:
```
def option_combinations(df):
    df = df.groupby(['save', 'spend', 'rent', 'coach'], as_index=False).mean()
    df = df[['save', 'spend', 'rent', 'coach', 'interval']]
    df = df.sort_values(by=['interval'], axis=0, ascending=False)
    df = df.rename(columns={'interval':'avg_interval'})
    return df
```

In [ ]:
```
def interval_averages(opt1, opt2, opt3, opt4, df):

    data = pd.concat([df.groupby([opt], as_index=False).mean()[[opt, 'interval']].rename(columns={'interval':f'{opt}_
    data = data.drop([opt1, opt2, opt3, opt4], axis=0).rename(columns={0:'deactivated', 1:'activated'}).reset_index(
    d2 = data.copy()
    d2['difference'] = d2['activated'] - d2['deactivated']
    bar = data.plot(x='index', kind='bar', stacked=False, color=['#4e358c', '#00b1aa'])
    d2 = d2.set_index('index')

    return d2
```

In [ ]:
```
def reasons_options(opt1, opt2, opt3, opt4, df, age):

    df = df[df['is_note']==0]

    if age=='old':
        df = df[df['new_vals']==0]
        df = df[[f'{opt1}', f'{opt2}', f'{opt3}', f'{opt4}', 'value_0', 'value_1', 'value_2', 'value_3', 'value_4',
```

```
    else:
        df = df[df['new_vals']==1]
        df = df[[f'{opt1}', f'{opt2}', f'{opt3}', f'{opt4}', 'value_0', 'value_1', 'value_2', 'value_3', 'value_4',

    options = [opt1, opt2, opt3, opt4]
    data_frames = []

    for option in options:
        df_option = df.groupby([f'{option}'], as_index=False).mean()
        df_option = df_option[df_option[f'{option}']==1]
        df_option = df_option.drop(columns=options)
        df_option = df_option.T.reset_index().rename(columns={1:f'{option}'}).set_index('index')
        data_frames.append(df_option)

    data = pd.concat(data_frames, axis=1).reset_index()

    bar=data.plot(x='index',
                  kind='bar',
                  stacked=False,
                  color=['#4e358c', '#00b1aa', '#f7a51e', '#cb8fde']
                  )

    df = data.set_index('index')

    return df
```

In [ ]:

```
def option_bar(df):

    df = df.groupby(['length_member'], as_index=False).mean()
    df = df[['length_member', 'save', 'spend', 'rent', 'coach']]
    df = df.set_index('length_member')
    df = df.T
    df = df.reset_index()

    bar=df.plot(x='index',
                kind='bar',
                stacked=False,
                color=['#4e358c', '#00b1aa', '#f7a51e', '#cb8fde']
                )

    df = df.set_index('index')
```

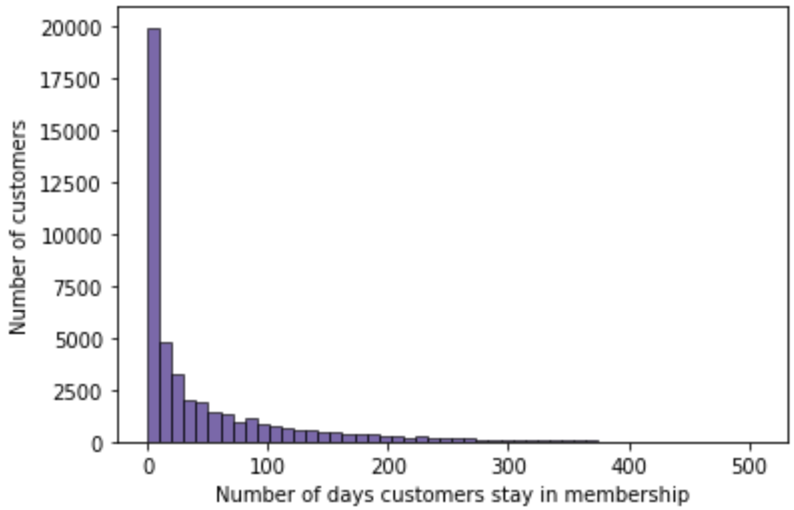
```
    return df
```

In [ ]:
```
def value_bar(df, age):
    if age=='old':
        df = df[(df['is_note']==0)&(df['new_vals']==0)]
        df = df[['length_member', 'value_0', 'value_1', 'value_2', 'value_3', 'value_4', 'value_5', 'value_6', 'value

    else:
        df = df[(df['is_note']==0)&(df['new_vals']==1)]
        df = df[['length_member', 'value_0', 'value_1', 'value_2', 'value_3', 'value_4', 'value_5']]

    df = df.groupby(['length_member'], as_index=False).mean()
    df = df.set_index('length_member')
    df = df.T
    df = df.reset_index()

    bar = df.plot(x='index',
                  kind='bar',
                  stacked=False,
                   color=['#4e358c', '#00b1aa', '#f7a51e', '#cb8fde'])

    df = df.set_index('index')

    return df
```

In [ ]:
```
def value_bar_count(df, age):
    if age=='old':
        df = df[(df['is_note']==0)&(df['new_vals']==0)]
        df = df[['length_member', 'value_0', 'value_1', 'value_2', 'value_3', 'value_4', 'value_5', 'value_6', 'value

    else:
        df = df[(df['is_note']==0)&(df['new_vals']==1)]
        df = df[['length_member', 'value_0', 'value_1', 'value_2', 'value_3', 'value_4', 'value_5']]

    df = df.groupby(['length_member'], as_index=True).sum()
    df = df.set_index('length_member')
    df = df.T
    df = df.reset_index()

    bar = df.plot(x='index',
```

```
                    kind='bar',
                    stacked=False,
                     color=['#4e358c', '#00b1aa', '#f7a51e', '#cb8fde'])

    df = df.set_index('index')

    return df
```

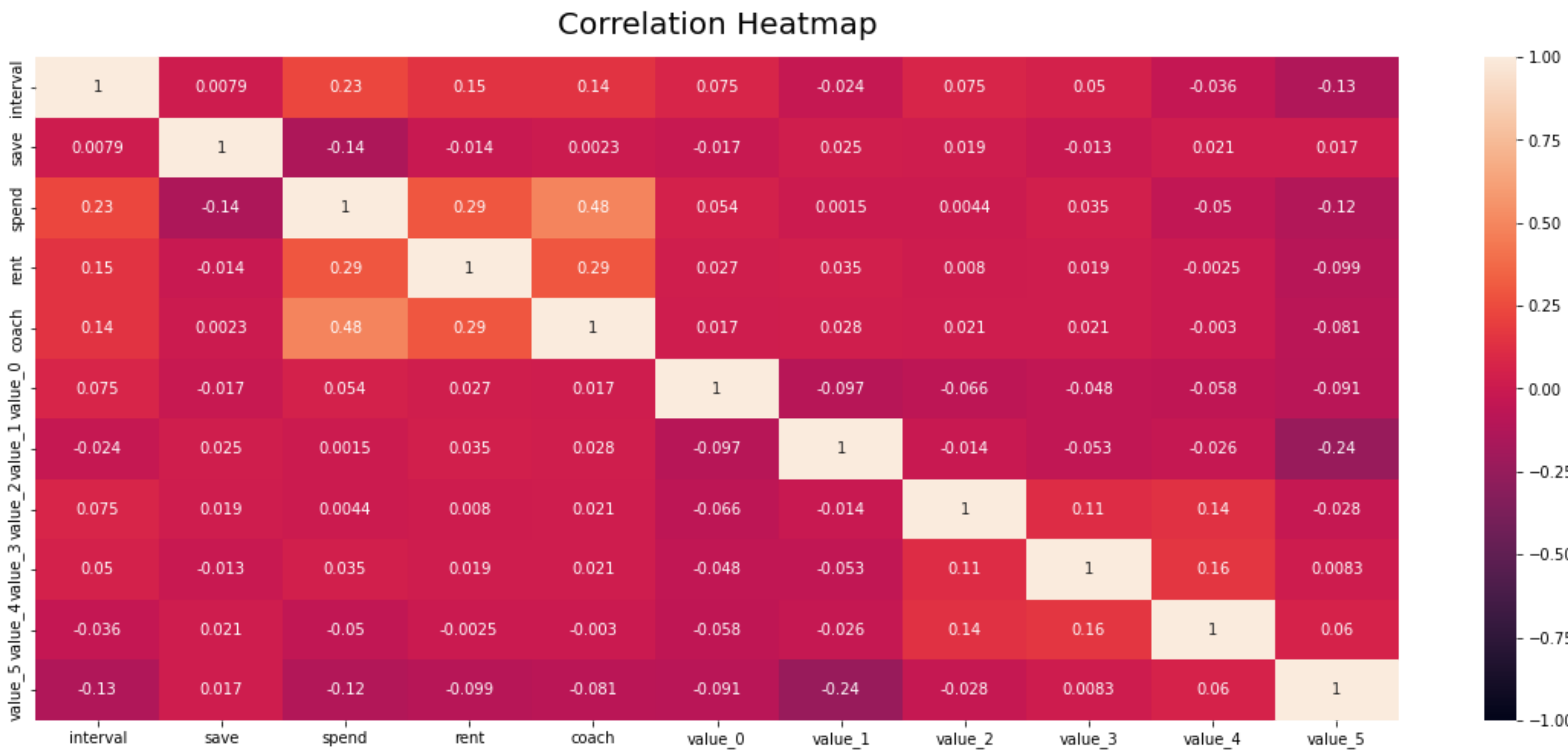
In [ ]:
```
#histogram of interval distribution
sns.histplot(loqnum.interval, color='#4e358c', bins=50)

plt.xlabel("Number of days customers stay in membership")
plt.ylabel("Number of customers")

plt.show()
```



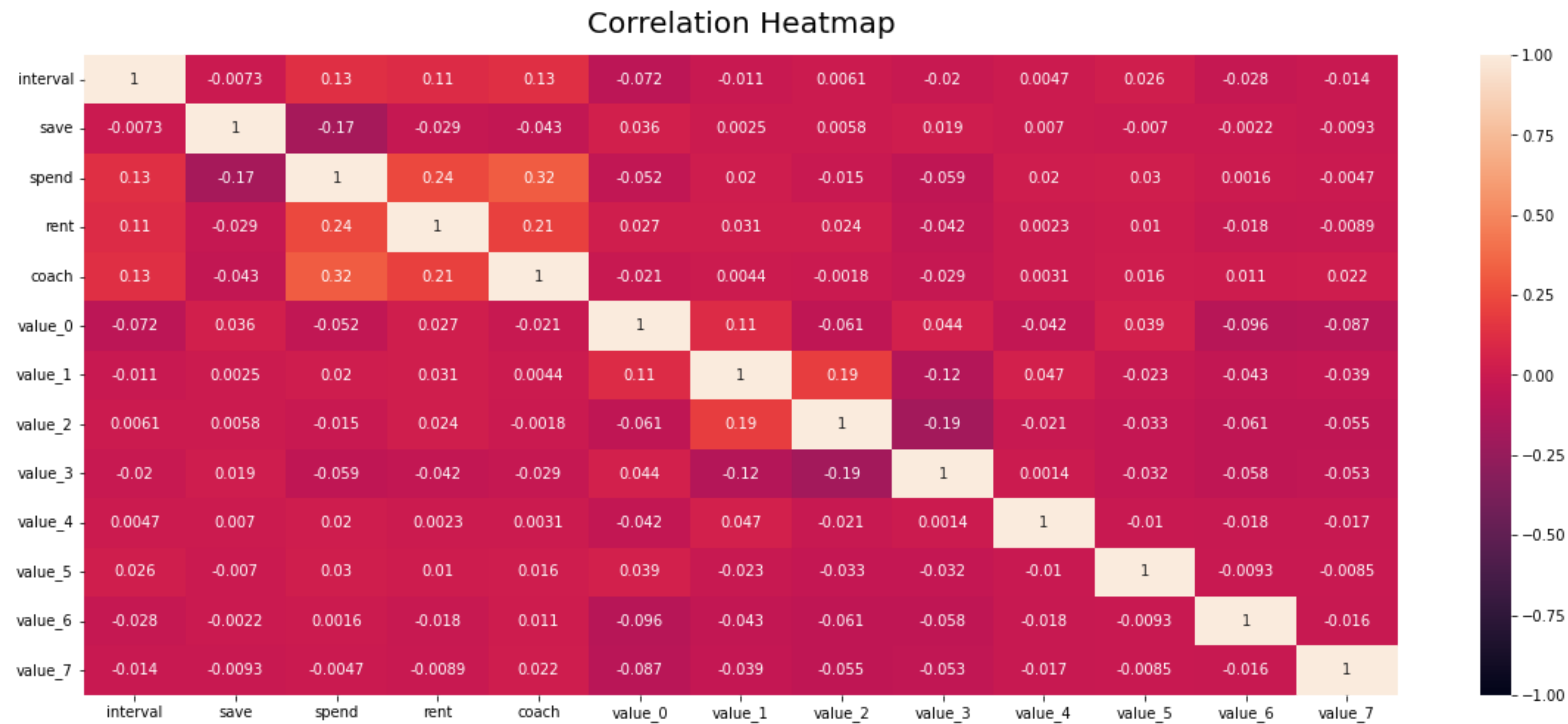

In [ ]:
```
# % of customers deactivate within 14 days
loqnum[loqnum["interval"] <= 14]["customer_id"].count()/loqnum["customer_id"].count()
```

Out[ ]: 0.5024947978430627

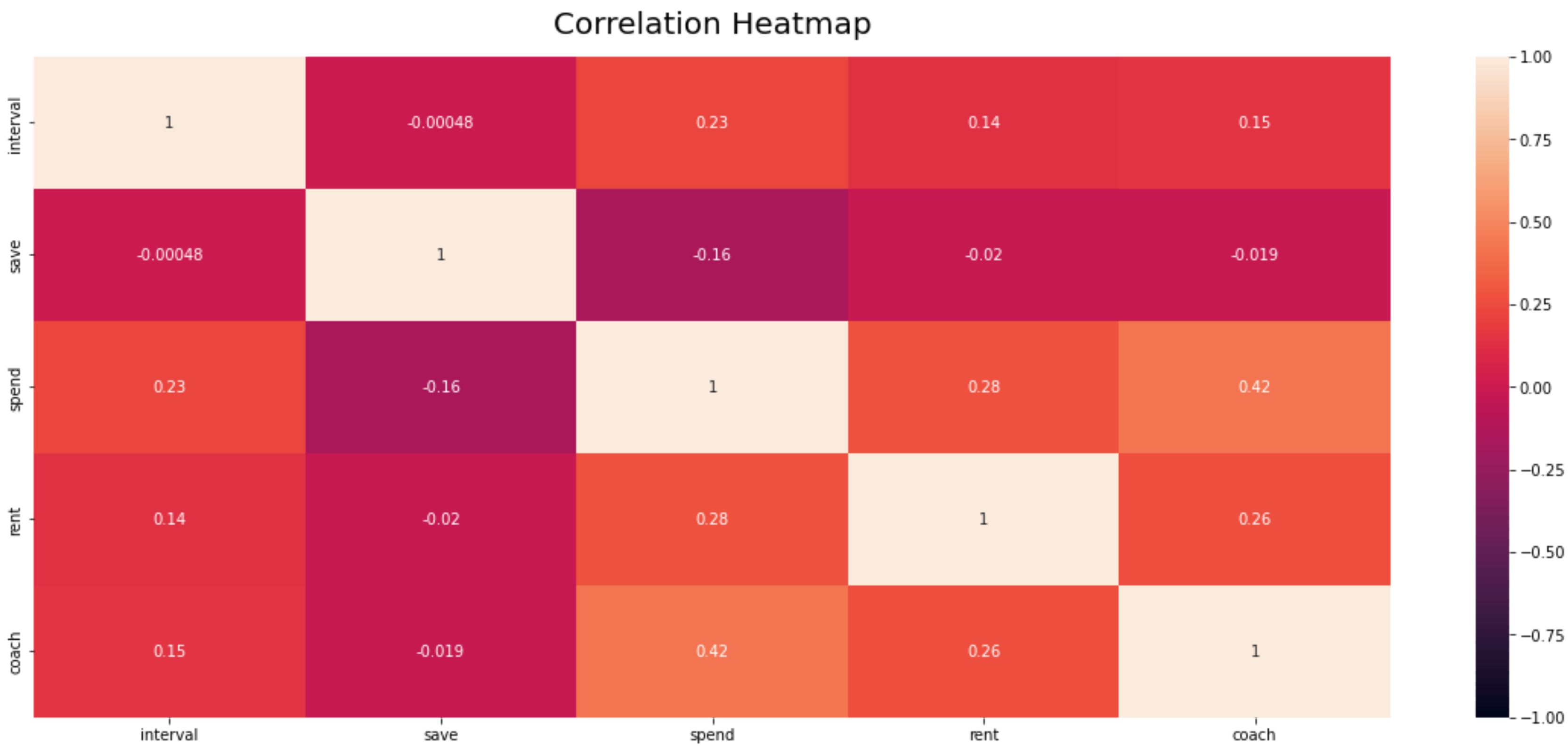
In [ ]:
```
correlation_heatmap(loqnum, 'new');
```




In [ ]:
```
correlation_heatmap(loqnum, 'old');
```




In [ ]:
```
correlation_heatmap(loqnum, 'option');
```




In [ ]:
```
option_combinations(loqnum)
```

Out[ ]:

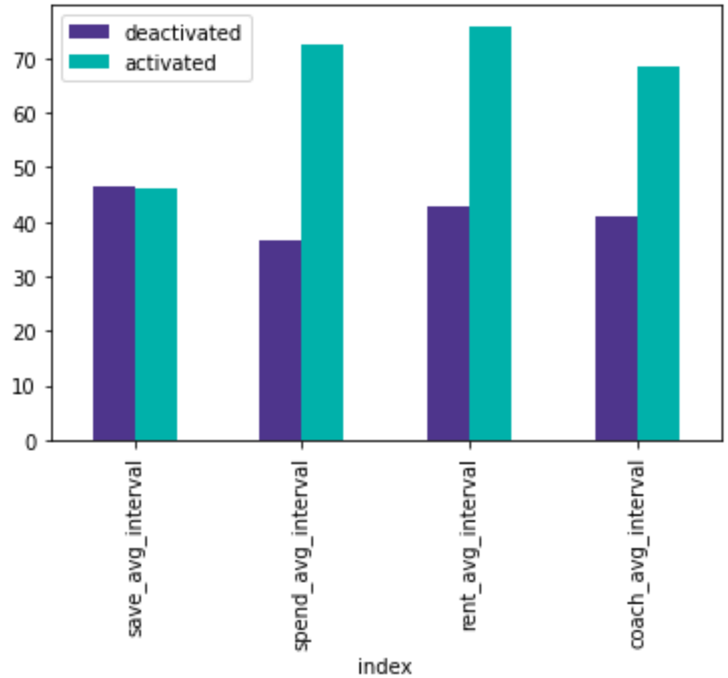
| | save | spend | rent | coach | avg_interval |
|---|---|---|---|---|---|
| **13** | 1 | 1 | 1 | 0 | 90.194196 |
| **14** | 1 | 1 | 1 | 1 | 88.679360 |
| **12** | 1 | 1 | 0 | 1 | 69.279933 |
| **11** | 1 | 1 | 0 | 0 | 68.075583 |
| **10** | 1 | 0 | 1 | 1 | 67.616352 |
| **5** | 0 | 1 | 1 | 0 | 62.939394 |
| **6** | 0 | 1 | 1 | 1 | 57.157895 |
| **8** | 1 | 0 | 0 | 1 | 54.342271 |
| **2** | 0 | 0 | 1 | 1 | 50.444444 |
| **9** | 1 | 0 | 1 | 0 | 49.875937 |
| **3** | 0 | 1 | 0 | 0 | 47.981383 |
| **4** | 0 | 1 | 0 | 1 | 44.644737 |
| **1** | 0 | 0 | 1 | 0 | 36.760000 |
| **7** | 1 | 0 | 0 | 0 | 34.090617 |
| **0** | 0 | 0 | 0 | 1 | 22.953488 |

In [ ]:

```
interval_averages('save', 'spend', 'rent', 'coach', loqnum)
```

Out[ ]:

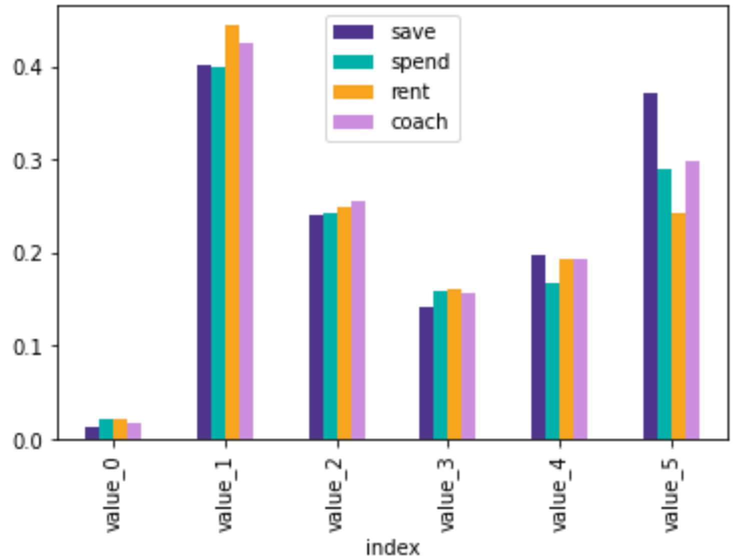
| | deactivated | activated | difference |
|---|---|---|---|
| **index** | | | |
| **save_avg_interval** | 46.397590 | 46.101084 | -0.296507 |
| **spend_avg_interval** | 36.552042 | 72.423962 | 35.871921 |
| **rent_avg_interval** | 42.920456 | 75.919314 | 32.998858 |
| **coach_avg_interval** | 40.951511 | 68.383873 | 27.432362 |




In [ ]:
```
reasons_options('save', 'spend', 'rent', 'coach', loqnum, 'new')
```

Out[ ]:

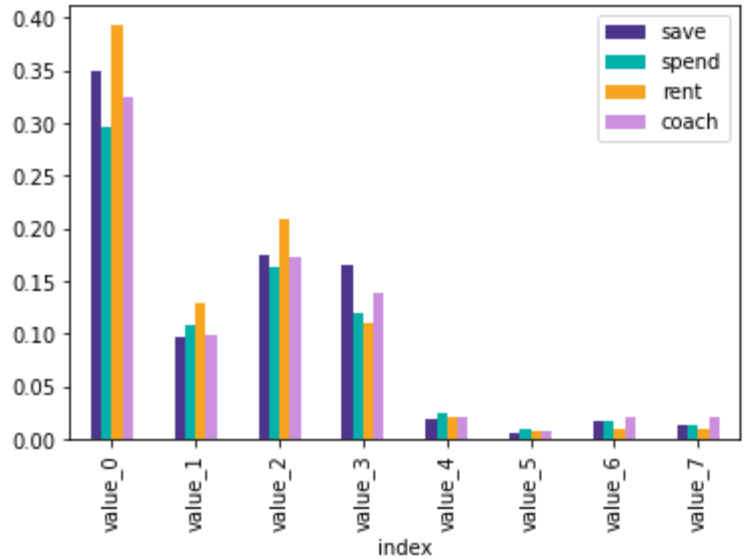
| index | save | spend | rent | coach |
|---|---|---|---|---|
| **value_0** | 0.013606 | 0.022495 | 0.022267 | 0.017513 |
| **value_1** | 0.400801 | 0.400234 | 0.444534 | 0.424518 |
| **value_2** | 0.240193 | 0.241747 | 0.248178 | 0.255819 |
| **value_3** | 0.142586 | 0.159947 | 0.161134 | 0.156728 |
| **value_4** | 0.196755 | 0.168273 | 0.193117 | 0.193527 |
| **value_5** | 0.371483 | 0.290827 | 0.243320 | 0.298825 |




In [ ]:
```
reasons_options('save', 'spend', 'rent', 'coach', loqnum, 'old')
```

Out[ ]:

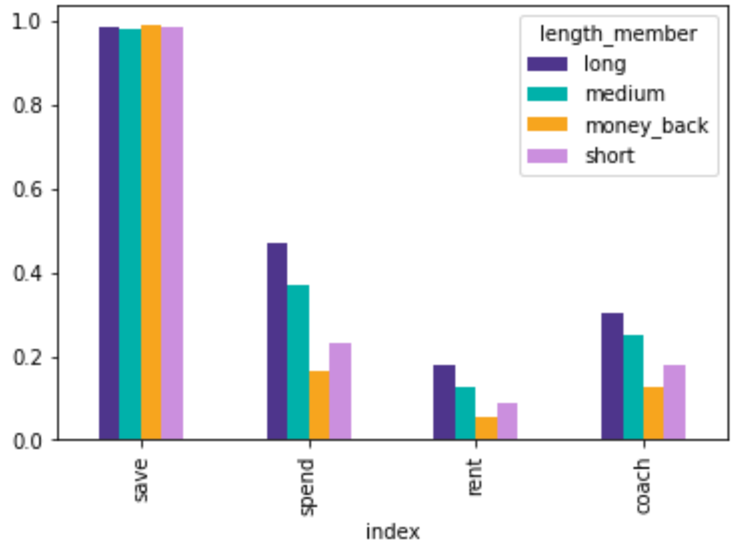
| index | save | spend | rent | coach |
|---|---|---|---|---|
| **value_0** | 0.349754 | 0.295591 | 0.392882 | 0.324306 |
| **value_1** | 0.096596 | 0.109145 | 0.128679 | 0.099632 |
| **value_2** | 0.175334 | 0.163310 | 0.208077 | 0.173521 |
| **value_3** | 0.165491 | 0.118672 | 0.109514 | 0.139084 |
| **value_4** | 0.019482 | 0.025313 | 0.020534 | 0.020395 |
| **value_5** | 0.004972 | 0.009526 | 0.007529 | 0.007690 |
| **value_6** | 0.016945 | 0.017420 | 0.008898 | 0.020395 |
| **value_7** | 0.013850 | 0.012793 | 0.010267 | 0.020060 |




In [ ]:
```
option_bar(loqnum)
```

Out[ ]:

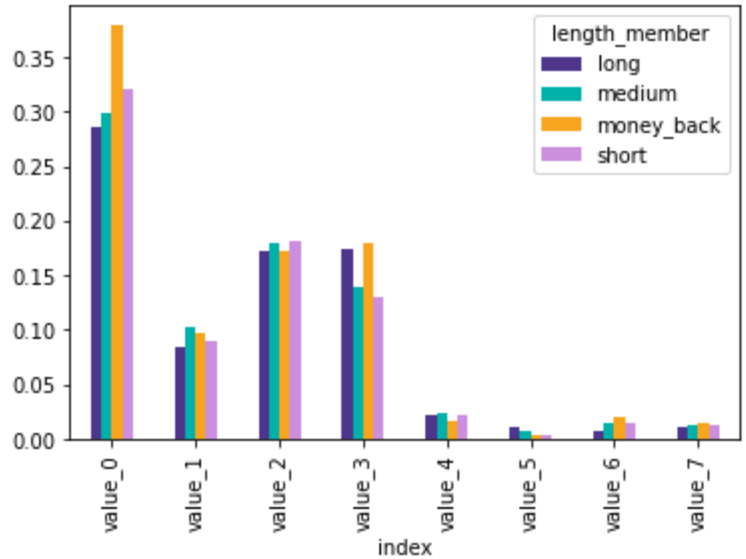
| length_member | long | medium | money_back | short |
|---|---|---|---|---|
| **index** | | | | |
| **save** | 0.986692 | 0.982923 | 0.988690 | 0.986934 |
| **spend** | 0.468159 | 0.370005 | 0.162971 | 0.231430 |
| **rent** | 0.179229 | 0.126859 | 0.056817 | 0.090209 |
| **coach** | 0.302612 | 0.252207 | 0.124054 | 0.180061 |




In [ ]:
```
value_bar(loqnum, 'old')
```

Out[ ]:

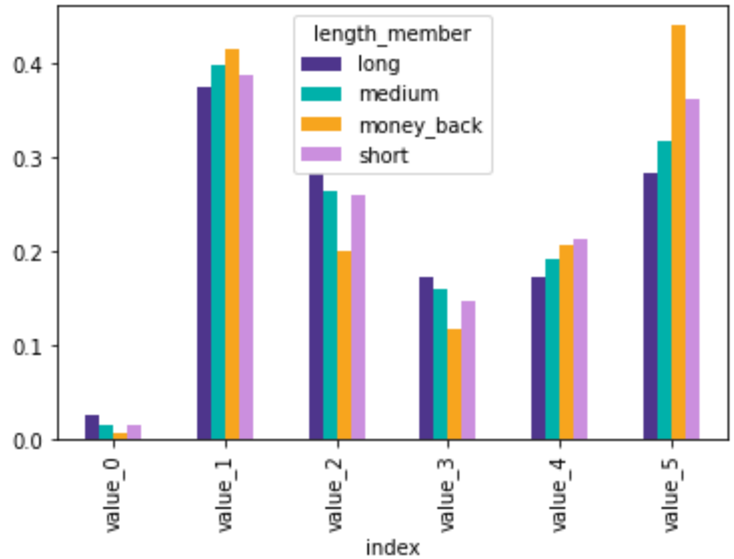
| length_member | long | medium | money_back | short |
|---|---|---|---|---|
| **index** | | | | |
| **value_0** | 0.286255 | 0.299451 | 0.379242 | 0.321364 |
| **value_1** | 0.084416 | 0.103297 | 0.097789 | 0.090316 |
| **value_2** | 0.172619 | 0.179121 | 0.172952 | 0.180631 |
| **value_3** | 0.174784 | 0.139011 | 0.179208 | 0.130939 |
| **value_4** | 0.021645 | 0.023077 | 0.017312 | 0.021763 |
| **value_5** | 0.010281 | 0.007143 | 0.003771 | 0.003990 |
| **value_6** | 0.008117 | 0.014286 | 0.019541 | 0.015597 |
| **value_7** | 0.011364 | 0.013462 | 0.014827 | 0.012695 |




In [ ]:
```
value_bar(loqnum, 'new')
```

Out[ ]:

| length_member | long | medium | money_back | short |
|---|---|---|---|---|
| **index** | | | | |
| **value_0** | 0.025787 | 0.014694 | 0.006465 | 0.015888 |
| **value_1** | 0.375228 | 0.398344 | 0.415515 | 0.387850 |
| **value_2** | 0.283655 | 0.263959 | 0.199510 | 0.259813 |
| **value_3** | 0.172589 | 0.161101 | 0.118368 | 0.148131 |
| **value_4** | 0.171980 | 0.191558 | 0.206420 | 0.212617 |
| **value_5** | 0.284467 | 0.318194 | 0.441596 | 0.361682 |




In [ ]:
```
#value_bar_count(loqnum, 'old')
```

## 3.2 Diagnostic analysis

### 3.2.1 TF-IDF & Hierarchical Clustering

In [ ]:
```
data = loqtext_preprocessed[loqtext_preprocessed["len_reason"] > 0]
```

In [ ]:
```
def tfidf(input_data_words):
    # Generate count vectors
    word_vec = input_data_words.apply(pd.value_counts).fillna(0)

    # Compute term frequencies
    tf = word_vec.divide(np.sum(word_vec, axis=1), axis=0)

    # Compute inverse document frequencies
    idf = np.log10(len(tf) / word_vec[word_vec > 0].count())

    # Compute TF-IDF vectors
    tfidf = np.multiply(tf, idf.to_frame().T)
```

```
    tfidf_sum = tfidf.T.sum(axis=1).sort_values(ascending = False).reset_index().rename(columns={"index": "word", 0:

    tfidf_transposed = tfidf.T

    return tfidf, tfidf_sum, tfidf_transposed
```

In [ ]:
```
def wordcloud(df, max_words):
    wordcloud = WordCloud(background_color="white", max_words=max_words, random_state=1).generate_from_frequencies(df
    plt.figure(figsize = (10, 10), facecolor = 'white', edgecolor='blue')
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)

    plt.show()
```

In [ ]:
```
def clustering(df, number_words, number_clusters):

    tfidf_sum = tfidf.T.sum(axis=1).sort_values(ascending = False).reset_index().rename(columns={"index": "word", 0:

    tfidf_transposed = tfidf.T


    # Get top important words (higest TF-IDF)
    important_words = tfidf_transposed[tfidf_transposed.index.isin(tfidf_sum.head(number_words).word)].index
    important_words_tfidf = tfidf_sum.head(number_words)
    n = len(important_words)

    # Calculate distance between words
    dist = 1 - cosine_similarity(tfidf_transposed[tfidf_transposed.index.isin(tfidf_sum.head(number_words).word)])

    Z = ward(dist) #define the linkage_matrix using ward clustering pre-computed distances


    # Number of desired clusters
    T = sch.fcluster(Z, number_clusters, 'maxclust')

    # calculate labels
    labels=list('' for i in range(n))
    for i in range(n):
      labels[i]=str(i)+ ',' + str(T[i])
```

```
    # calculate color threshold
    ct=Z[-(number_clusters-1),2]

    fig = plt.figure(figsize=(15, 12))
    gs = fig.add_gridspec(nrows=5, ncols=2, width_ratios=[1, 2], height_ratios=[1,1,100,1,1])
    ax0 = fig.add_subplot(gs[0, 1])

    # Plot dendrogram - Hierarchical Clustering
    ax2 = fig.add_subplot(gs[1:4, 1])
    set_link_color_palette(["gray", "gray", "gray", "gray", "gray", "#f7727e", "#f7a51e", "#00b1aa", "#4e358c"])
    ax = dendrogram(Z=Z,
                orientation="right",
                labels=important_words,
                color_threshold=ct,
                above_threshold_color="white",
                ax=ax2,
                leaf_font_size=10
                );
    ax2.set_xlim(0.5, 1.6)

    for leaf, leaf_color in zip(plt.gca().get_yticklabels(), ax["leaves_color_list"]):
      leaf.set_color(leaf_color)

    important_words_tfidf['word'] = pd.Categorical(important_words_tfidf['word'], categories=ax["ivl"], ordered=True)
    important_words_tfidf = important_words_tfidf.sort_values('word')

    # Plot bar chart - TF-IDF
    ax1 = fig.add_subplot(gs[:, 0])
    x = important_words_tfidf["word"]
    y = important_words_tfidf["tfidf_sum"]
    ax1.barh(x, y, align='center', color="#4e358c")
    ax1.set_yticks(x)
    ax1.set_yticklabels(x)
    ax1.set_xlim(175,0)

    # Edit the plots
    ax0.spines['right'].set_visible(False)
    ax0.spines['top'].set_visible(False)
    ax0.set_axis_off()
    ax1.spines['top'].set_visible(False)
    ax1.set_axis_off()
    ax2.spines['left'].set_visible(False)
```

```
    ax2.spines['right'].set_visible(False)
    ax2.spines['top'].set_visible(False)
    ax2.spines['bottom'].set_visible(False)
    ax2.get_xaxis().set_visible(False)

    ax1.set_title("TF-IDF",loc="right", fontsize=14)
    ax0.set_title("Hierarchical Clustering", fontsize=14)

    #plt.tick_params(axis="y", labelsize=12)
    plt.show()
```

In [ ]:

```
tfidf, tfidf_sum, tfidf_transposed = tfidf(data["reason_lmt"])
```

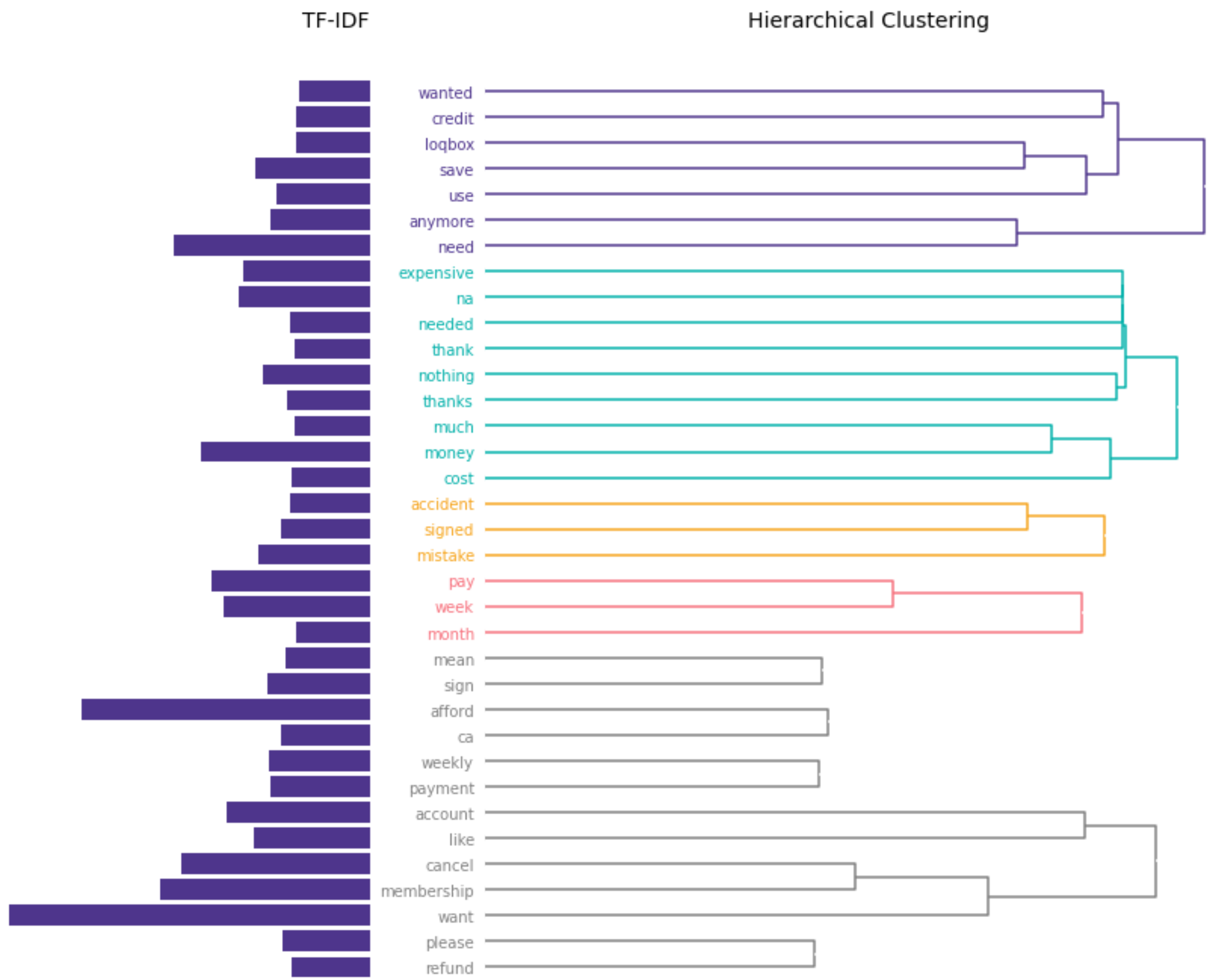
```
<ipython-input-77-fb889ccaaed7>:12: FutureWarning: Calling a ufunc on non-aligned DataFrames (or DataFrame/Series co
mbination). Currently, the indices are ignored and the result takes the index/columns of the first DataFrame. In the
future , the DataFrames/Series will be aligned before applying the ufunc.
Convert one of the arguments to a NumPy array (eg 'ufunc(df1, np.asarray(df2)') to keep the current behaviour, or al
ign manually (eg 'df1, df2 = df1.align(df2)') before passing to the ufunc to obtain the future behaviour and silence
this warning.
  tfidf = np.multiply(tf, idf.to_frame().T)
```

In [ ]:

```
wordcloud(tfidf, 50)
```


In [ ]:
```
clustering(tfidf, 35, 9)
```

```
<ipython-input-87-d3ea1f50d440>:50: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy
  important_words_tfidf['word'] = pd.Categorical(important_words_tfidf['word'], categories=ax["ivl"], ordered=True)
```




## 3.2.2 TF-IDF & Topic Modeling

In [ ]:
```
#Create dictionaries and corpus needed for topic modeling
#filter out the words that appear in more than 95% of the documents, less than 5 documents, specific words = loqbox

def createCorpus(input_data_words):

  # Create Dictionary
  id2word = corpora.Dictionary(input_data_words)

  # Create stop words list
  stop_words = ['loqbox']

  # Fitler out extremes
  id2word.filter_extremes(no_above=0.95, no_below=5, keep_tokens=[tokenid for tokenid, freq in id2word.cfs.items() if

  # Create Corpus
  texts = input_data_words

  # Term Document Frequency
  corpus = [id2word.doc2bow(text) for text in texts]

  return id2word,texts, corpus
```

In [ ]:
```
#Build topic modeling
def buildGensimModel(corpus, id2word, optimal_num_topics):
    lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                               id2word=id2word,
                                               num_topics=optimal_num_topics,
                                               random_state=100,
                                               update_every=1,
                                               chunksize=100,
                                               passes=10,
                                               alpha='auto',
                                               per_word_topics=True)
    for idx, topic in lda_model.show_topics(num_topics=optimal_num_topics, formatted=False):
        print('Topic: {} \nWords: {}'.format(idx, [w[0] for w in topic]))

    return lda_model

def buildNMFModel(input_list):
  # Create a TfidfVectorizer object to transform text into a matrix of TF-IDF features
```

```
  vectorizer = TfidfVectorizer(max_df=0.95, min_df=5)


  # Fit and transform the text data
  tfidf = vectorizer.fit_transform(input_list)

  # Create an NMF object and fit the TF-IDF data to it (number of topic = 5)
  nmf_model = NMF(n_components=5, init='random', random_state=42)
  nmf_model.fit(tfidf)

  # Print the top 10 words for each of the 10 topics
  for i, topic in enumerate(nmf_model.components_):
    print(f"Topic {i}:")
    print([vectorizer.get_feature_names()[index] for index in topic.argsort()[-10:]])

  return vectorizer, nmf_model
```

In [ ]:

```
#Visualise the result
def visualiseDashboard(model,corpus,id2word):
  pyLDAvis.enable_notebook()

  return pyLDAvis.gensim.prepare(model,corpus,id2word)
```

In [ ]:

```
#All model Lemmatisation
id2word_all_lmt, texts_all_lmt, corpus_all_lmt = createCorpus(data['reason_lmt'])
all_model = buildGensimModel(corpus_all_lmt,id2word_all_lmt, 3)
```

```
Topic: 0
Words: ['want', 'pay', 'week', 'afford', 'save', 'money', 'need', 'account', 'sign', 'signed']
Topic: 1
Words: ['membership', 'cancel', 'credit', 'free', 'score', 'please', 'first', 'full', 'expensive', 'signing']
Topic: 2
Words: ['mean', 'mistake', 'like', 'refund', 'paid', 'clicked', 'accident', 'get', 'next', 'spend']
```

In [ ]:

```
visualiseDashboard(all_model, corpus_all_lmt, id2word_all_lmt)
```

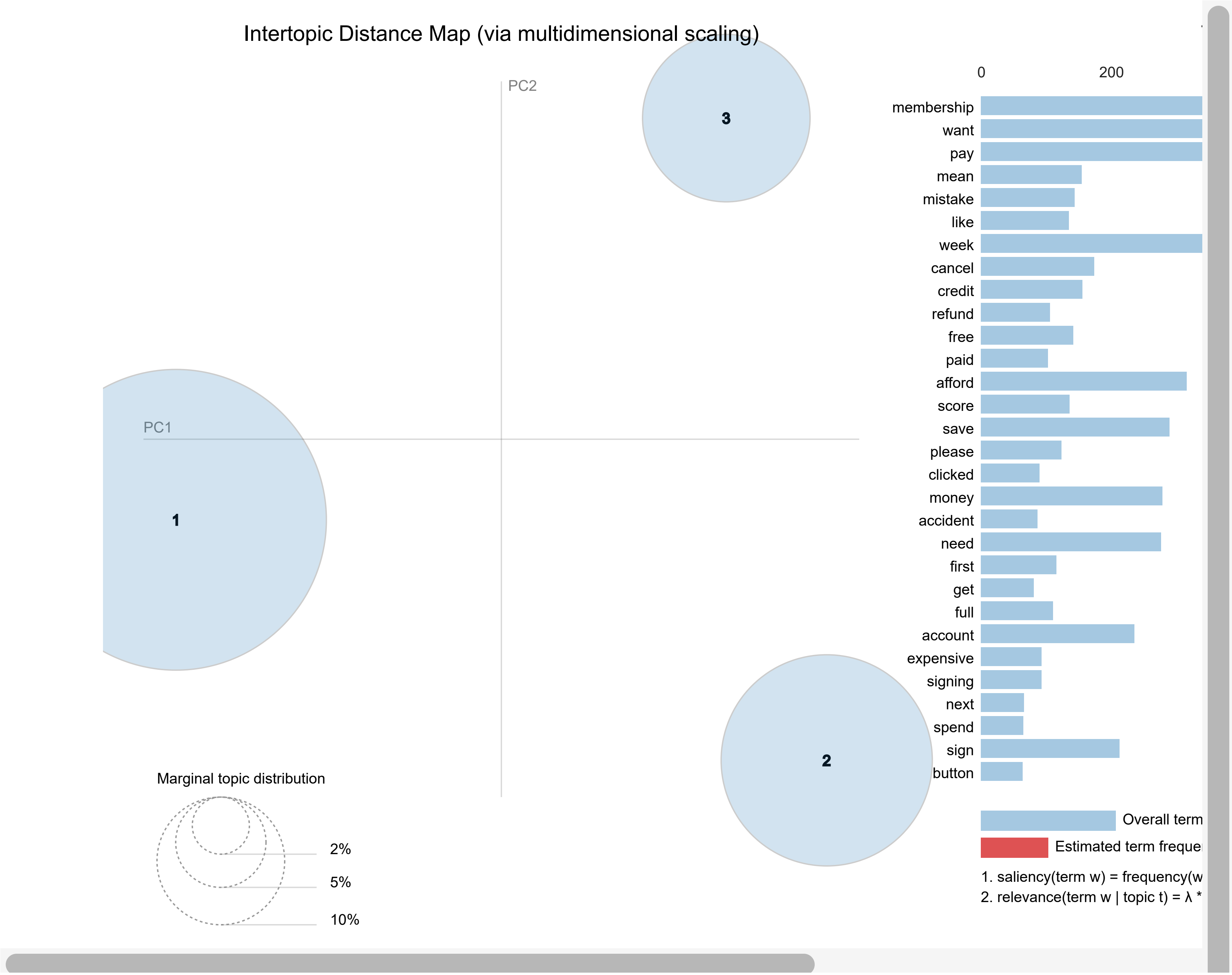
```
/usr/local/lib/python3.9/dist-packages/pyLDAvis/_prepare.py:243: FutureWarning: In a future version of pandas all
guments of DataFrame.drop except for the argument 'labels' will be keyword-only
  default_term_info = default_term_info.sort_values(
```

Out[ ]: Selected Topic: 0 Previous Topic Next Topic Clear Topic

Slide to adjust relevance m

λ = 1

(2)



Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
Marginal topic distribution
2%
5%
10%
0
200
membership
want
pay
mean
mistake
like
week
cancel
credit
refund
free
paid
afford
score
save
please
clicked
money
accident
need
first
get
full
account
expensive
signing
next
spend
sign
button
Overall term
Estimated term freque
1. saliency(term w) = frequency(w
2. relevance(term w | topic t) = λ *

In [ ]:
```
#All model stemming
id2word_all_stm, texts_all_stm, corpus_all_stm = createCorpus(data['reason_stem'])
all_model_stm = buildGensimModel(corpus_all_stm,id2word_all_stm, 3)
```

```
Topic: 0
Words: ['want', 'membership', 'need', 'afford', 'money', 'account', 'weekli', 'cancel', 'payment', 'fee']
Topic: 1
Words: ['pay', 'save', 'week', 'month', 'per', 'use', 'extra', 'free', 'cost', 'much']
Topic: 2
Words: ['sign', 'charg', 'servic', 'upgrad', 'mean', 'credit', 'score', 'first', 'get', 'paid']
```

In [ ]:
```
visualiseDashboard(all_model_stm, corpus_all_stm, id2word_all_stm)
```

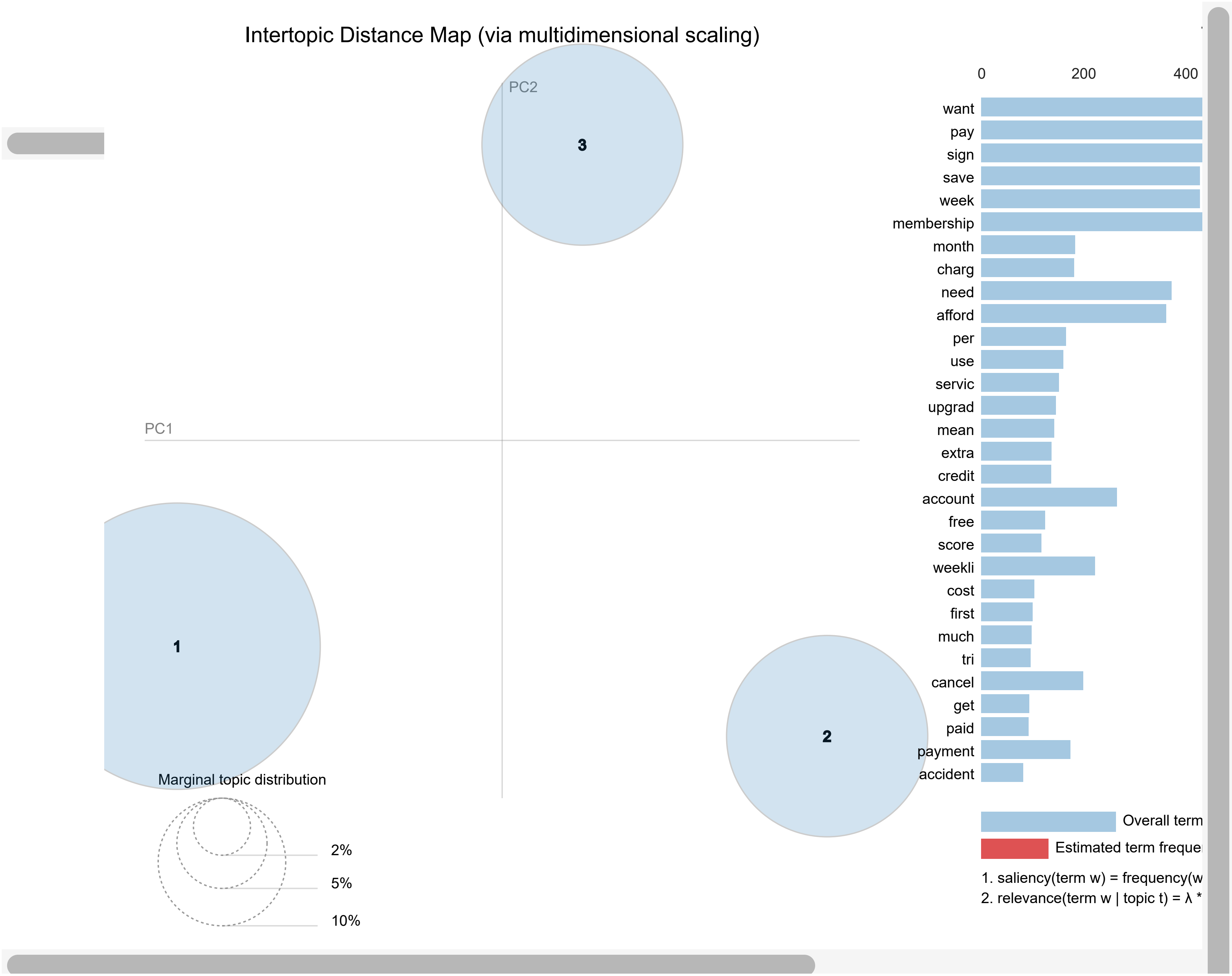
```
/usr/local/lib/python3.9/dist-packages/pyLDAvis/_prepare.py:243: FutureWarning: In a future version of pandas all
guments of DataFrame.drop except for the argument 'labels' will be keyword-only
  default_term_info = default_term_info.sort_values(
```

Out[ ]: Selected Topic: 0 Previous Topic Next Topic Clear Topic

Slide to adjust relevance m

λ = 1

(2)


Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
3
1
2
Marginal topic distribution
2%
5%
10%
0
200
400
want
pay
sign
save
week
membership
month
charg
need
afford
per
use
servic
upgrad
mean
extra
credit
account
free
score
weekli
cost
first
much
tri
cancel
get
paid
payment
accident
Overall term
Estimated term freque
1. saliency(term w) = frequency(w
2. relevance(term w | topic t) = λ *

In [ ]:
```
#Calculate percentage

#Define words in each topic based on the result from TF-IDF
topics = [
    {'name':'no longer needed','keywords':['use', 'loqbox', 'credit', 'dont', 'like','anymor','need']},
    {'name':'expensive','keywords':['mistake','expens', 'na', 'noth', 'thank','plan','realis','charg','much','cost',
    {'name':'frequency of payment','keywords':['month', 'week', 'pay', 'afford', 'ca','weekli','payment']},
    {'name':'accidtally upgraded','keywords':['accid','click','mean','sign','accident','upgrad']}
]


#Create data_words
comments = data['reason_stem']

# assign comments to topics
topic_counts = [0] * (len(topics) + 1) # +1 for "other" category
for comment in comments:
    matched_topic = False
    for i, topic in enumerate(topics):
        for keyword in topic['keywords']:
            if keyword in comment:
                topic_counts[i] += 1
                matched_topic = True
                break
        if matched_topic:
            break
    if not matched_topic:
        topic_counts[-1] += 1

# calculate percentage of comments for each topic
total_comments = len(comments)
for i, topic in enumerate(topics):
    percentage = topic_counts[i] / total_comments * 100
    print(f"Topic {i}: {percentage}% ({topic_counts[i]} comments)")
print(f"Other: {topic_counts[-1] / total_comments * 100}% ({topic_counts[-1]} comments)")
```

```
Topic 0: 24.694636218799786% (930 comments)
Topic 1: 26.287838555496545% (990 comments)
Topic 2: 13.887413701540098% (523 comments)
Topic 3: 7.169410515135423% (270 comments)
Other: 27.960701009028142% (1053 comments)
```