Machine Learning with Church Music

By Jake Bodea

Introduction

As a worship leader at Agape Church of Orange County, I found difficulty in picking a worship set at times. I dreamed of a program that would make that process easier. Upon finding that our church's organizational software, Planning Center, held a data set of our church music data available for download, it dawned on me that the program I envisioned could actually become a reality.

Part One of this project will focus on cleaning up that poorly-managed data set and expanding on that data, whereas Part Two will demonstrate how machine learning algorithms can help group such data into clusters of similar songs to help streamline the song selection process.

Part One: Cleaning and Improving the data

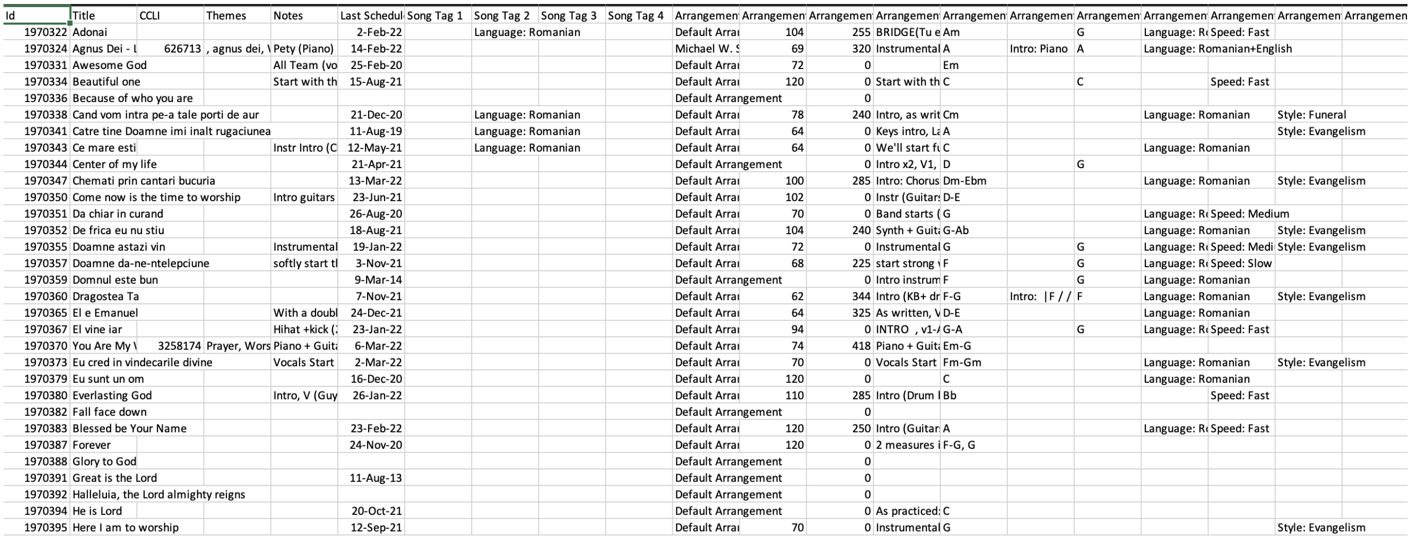
The first thing that needs to be done before any analysis is collecting the data. Below is some code that downloads my church's Planning Center data. The Python library Pandas converts the csv data from Planning Center into a Data Frame, which is a 2-dimensional table of rows and columns, similar to a spreadsheet on Excel.

In [1]:

```python
import pandas as pd
import numpy as np

temp = pd.read_csv("pco_songs.csv")
temp = temp.drop_duplicates()
print("There are currently {} rows of data.".format(temp.shape[0]))
```

```
There are currently 565 rows of data.
```



Id	Title	CCLI	Themes	Notes	Last Schedul	Song Tag 1	Song Tag 2	Song Tag 3	Song Tag 4	Arrangemen	Arrangemen	Arrangemen	Arrangemen	Arrangemen	Arrangemen	Arrangemen	Arrangemen	Arrangemen	Arrangemen	Arrangemen
1970322	Adonai				2-Feb-22		Language: Romanian			Default Arrai	104	255	BRIDGE(Tu e	Am		G	Language: Ro	Speed: Fast		
1970324	Agnus Dei - L	626713	, agnus dei, \	Pety (Piano)	14-Feb-22					Michael W. S	69	320	Instrumental	A	Intro: Piano	A	Language: Romanian+English			
1970331	Awesome God			All Team (vo	25-Feb-20					Default Arrai	72	0		Em						
1970334	Beautiful one			Start with th	15-Aug-21					Default Arrai	120	0	Start with th	C		C		Speed: Fast		
1970336	Because of who you are									Default Arrangement		0								
1970338	Cand vom intra pe-a tale porti de aur				21-Dec-20		Language: Romanian			Default Arrai	78	240	Intro, as writ	Cm			Language: Romanian		Style: Funeral	
1970341	Catre tine Doamne imi inalt rugaciunea				11-Aug-19		Language: Romanian			Default Arrai	64	0	Keys intro, La	A					Style: Evangelism	
1970343	Ce mare esti			Instr Intro (C	12-May-21		Language: Romanian			Default Arrai	64	0	We'll start fu	C			Language: Romanian			
1970344	Center of my life				21-Apr-21					Default Arrangement		0	Intro x2, V1,	D		G				
1970347	Chemati prin cantari bucuria				13-Mar-22					Default Arrai	100	285	Intro: Chorus	Dm-Ebm			Language: Romanian		Style: Evangelism	
1970350	Come now is the time to worship			Intro guitars	23-Jun-21					Default Arrai	102	0	Instr (Guitars	D-E						
1970351	Da chiar in curand				26-Aug-20					Default Arrai	70	0	Band starts (	G			Language: Ro	Speed: Medium		
1970352	De frica eu nu stiu				18-Aug-21					Default Arrai	104	240	Synth + Guita	G-Ab			Language: Romanian		Style: Evangelism	
1970355	Doamne astazi vin			Instrumental	19-Jan-22					Default Arrai	72	0	Instrumental	G		G	Language: Ro	Speed: Medi	Style: Evangelism	
1970357	Doamne da-ne-ntelepciune			softly start tl	3-Nov-21					Default Arrai	68	225	start strong \	F		G	Language: Ro	Speed: Slow		
1970359	Domnul este bun				9-Mar-14					Default Arrangement		0	Intro instrum	F		G	Language: Romanian			
1970360	Dragostea Ta				7-Nov-21					Default Arrai	62	344	Intro (KB+ dr	F-G	Intro: |F / /	F	Language: Romanian		Style: Evangelism	
1970365	El e Emanuel			With a doubl	24-Dec-21					Default Arrai	64	325	As written, V	D-E			Language: Romanian			
1970367	El vine iar			Hihat +kick (2	23-Jan-22					Default Arrai	94	0	INTRO , v1-A	G-A		G	Language: Ro	Speed: Fast		
1970370	You Are My \	3258174	Prayer, Wors	Piano + Guita	6-Mar-22					Default Arrai	74	418	Piano + Guita	Em-G						
1970373	Eu cred in vindecarile divine			Vocals Start	2-Mar-22					Default Arrai	70	0	Vocals Start	Fm-Gm			Language: Romanian		Style: Evangelism	
1970379	Eu sunt un om				16-Dec-20					Default Arrai	120	0		C			Language: Romanian			
1970380	Everlasting God			Intro, V (Guy	26-Jan-22					Default Arrai	110	285	Intro (Drum l	Bb				Speed: Fast		
1970382	Fall face down									Default Arrangement		0								
1970383	Blessed be Your Name				23-Feb-22					Default Arrai	120	250	Intro (Guitar	A			Language: Ro	Speed: Fast		
1970387	Forever				24-Nov-20					Default Arrai	120	0	2 measures i	F-G, G						
1970388	Glory to God									Default Arrangement		0								
1970391	Great is the Lord				11-Aug-13					Default Arrangement		0								
1970392	Halleluia, the Lord almighty reigns									Default Arrangement		0								
1970394	He is Lord				20-Oct-21					Default Arrangement		0	As practiced:	C						
1970395	Here I am to worship				12-Sep-21					Default Arrai	70	0	Instrumental	G					Style: Evangelism	

Looking at the above data, it can be seen that there are a lot of variables that have not been inputted consistently, such as "Themes" and "Song Tags". When trying to analyze large sets of data, it is crucial to have a "clean" data set, meaning that there are no empty spaces where data is expected. Unfortunately, it seems that this data set has too many columns with little to no information. Let's take only the Title and BPM data and see what else can be derived.

Note: BPM stands for "Beats per Minute" and is a measurement of a song's speed

In [2]:

```
df = temp[["Title", "Arrangement 1 BPM"]]
df = df.rename(columns={"Arrangement 1 BPM": "BPM"})
df.head()
```

Out[2]:

	Title	BPM
0	Adonai	104.0
1	Agnus Dei - Lord of All	69.0
2	Awesome God	72.0
3	Beautiful one	120.0
4	Because of who you are	NaN

In [ ]:

```

```

Let us now limit the data to entries that have a valid BPM ("NaN" refers to "Not a number", and is shown when my church did not input any data for the BPM).

In [3]:

```
df = df.dropna()
df = df.reset_index(drop=True)
df.head()
```

Out[3]:

	Title	BPM
0	Adonai	104.0
1	Agnus Dei - Lord of All	69.0
2	Awesome God	72.0
3	Beautiful one	120.0
4	Cand vom intra pe-a tale porti de aur	78.0

In [ ]:

```

```

We now have two variables: an identifying variable, `Title` , and a numeric variable, `BPM` . Given that my church had a multilingual congregation and thus multilingual songs, let us try to explore this data more by trying to figure out the language of each song.

`LangDetect` is an open source library claiming to use Google's translation features. Notice the implementation below:

In [4]:

```
import langdetect as lang

df["Language"] = ""
df["Lang Probs"] = df["Title"].apply(lang.detect_langs)
```

In [5]:

```
df.head()
```

Out[5]:

	Title	BPM	Language	Lang Probs
0	Adonai	104.0		[tl:0.8571386056006786, id:0.14286069370907353]
1	Agnus Dei - Lord of All	69.0		[en:0.8571393258573531, nl:0.1428571257547347]
2	Awesome God	72.0		[af:0.9999939556167616]
3	Beautiful one	120.0		[ro:0.8571402589640329, fr:0.14285972316680953]
4	Cand vom intra pe-a tale porti de aur	78.0		[it:0.5713876077958037, pt:0.4285697122261661]

In [ ]:

```
# since every run of detect or detect_langs is different,
# this ensures that df["Language"] will just be the most likely language
from a single result in string format
i = 0
for lan in df["Lang Probs"]:
    language = str(lan)[1:3]
    if language != "en":
        df["Language"].iloc[i] = "not en"
    else:
        df["Language"].iloc[i] = "en"
    i += 1
```

In [7]:

```
df.head()
```

Out[7]:

	Title	BPM	Language	Lang Probs
0	Adonai	104.0	not en	[tl:0.8571386056006786, id:0.14286069370907353]
1	Agnus Dei - Lord of All	69.0	en	[en:0.8571393258573531, nl:0.1428571257547347]
2	Awesome God	72.0	not en	[af:0.9999939556167616]
3	Beautiful one	120.0	not en	[ro:0.8571402589640329, fr:0.14285972316680953]
4	Cand vom intra pe-a tale porti de aur	78.0	not en	[it:0.5713876077958037, pt:0.4285697122261661]

In [ ]:

```

```

Notice that there are lots of errors with the `LangDetect` library. Most explicitly is at index 3: "Beautiful One" is considered as Romanian! Also, every time the code is rerun, a different result is outputted (called a nondeterministic algorithm). This makes it really difficult to parse the language without more text. Let's try to automate some lyric data retrieval by web scraping Google search results.

In [8]:

```
'''
Below code is commented due to length, see next codeblock for a download
of csv file.

import requests
from bs4 import BeautifulSoup

lyrics = df["Title"] # lyrics is a series, not df. needs to be converted
lyrics = lyrics.to_frame()
lyrics["Text"] = ""
i=0
for i in range(len(lyrics)):
    search = "{} lyrics".format(lyrics["Title"].iloc[i])
    url = f"https://www.google.com/search?&q={search}"
    req = requests.get(url).text
    sor = BeautifulSoup(req, "html.parser")
    check = type(sor.find("div", class_='hwc'))
    if str(check) != "<class 'NoneType'>":
        words = sor.find("div", class_='hwc').text
        lyrics["Text"].iloc[i] = words
    i += 1

# since "newline" keys are processed as "/n", lets remove that from each
instance in lyrics["Text"]
lyrics["Text"] = lyrics["Text"].replace("/n", " ")

lyrics.head()
'''
;
```

Out[8]:

```
''
```

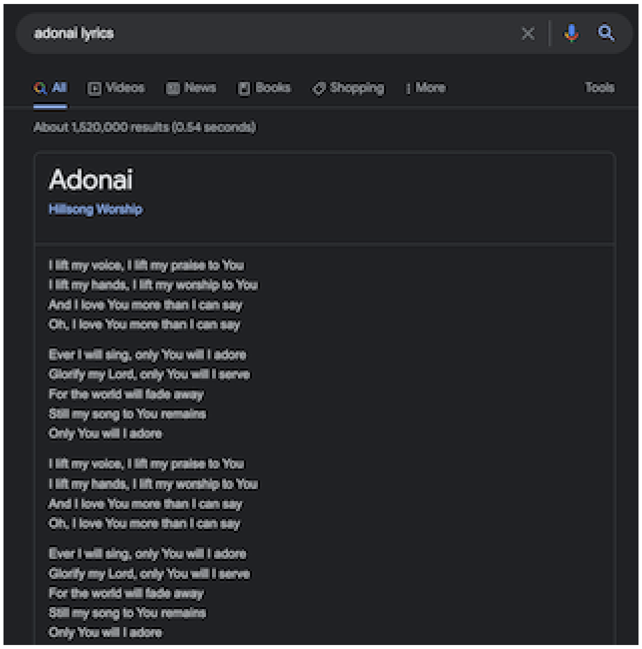
In [ ]:

Note: *The above cell takes a long time to run, so instead the output was saved into a separate .csv file that is simply downloaded below*

The code above does the following:

- Google search of '{title} lyrics'
- Looks to see if Google provides lyrics based on the search
- Saves those lyrics to a new variable

Here is a screenshot of what it looks like when Google provides lyrics:



About 1,520,000 results (0.54 seconds)

Adonai

Hillsong Worship

I lift my voice, I lift my praise to You
I lift my hands, I lift my worship to You
And I love You more than I can say
Oh, I love You more than I can say

Ever I will sing, only You will I adore
Glorify my Lord, only You will I serve
For the world will fade away
Still my song to You remains
Only You will I adore

I lift my voice, I lift my praise to You
I lift my hands, I lift my worship to You
And I love You more than I can say
Oh, I love You more than I can say

Ever I will sing, only You will I adore
Glorify my Lord, only You will I serve
For the world will fade away
Still my song to You remains
Only You will I adore

In [9]:
```
# saved excel of above code so we don't have to wait a long time to do i
t again, just import csv file
# lyrics.to_excel('lyrics.xlsx', index=False)
```

In [10]:

```python
lyrics = pd.read_excel('lyrics.xlsx')
lyrics = lyrics.drop_duplicates()
lyrics.head()
```

Out[10]:

	Title	Text
0	Adonai	I lift my voice, I lift my praise to You\nI li...
1	Agnus Dei - Lord of All	Alleluia, Alleluia\nFor the Lord God Almighty ...
2	Awesome God	Our God is an awesome God\nHe reigns from heav...
3	Beautiful one	Wonderful, so wonderful\nIs Your unfailing lov...
4	Because of who you are	I worship you Lord\nLord, I love you, Lord\n\n...

In [ ]:

Now we can rerun the language detection algorithm based on the lyrics instead of the title. If the lyrics are blank (as in, Google was unable to find lyrics), then the algorithm would look at the title as before.

In [11]:

```python
lyrics["Language Probs"] = ""

i = 0
for i in range(len(lyrics)):
    if str(lyrics["Text"].iloc[i]) != "nan": # if no lyrics availalbe, use title
        lyrics["Language Probs"].iloc[i] = lang.detect_langs(lyrics["Text"].iloc[i])
    else:
        lyrics["Language Probs"].iloc[i] = lang.detect_langs(lyrics["Title"].iloc[i])
    i+=1

lyrics.head()
```

Out[11]:

	Title	Text	Language Probs
0	Adonai	I lift my voice, I lift my praise to You\nI li...	[en:0.9999962671595386]
1	Agnus Dei - Lord of All	Alleluia, Alleluia\nFor the Lord God Almighty ...	[en:0.9999970413285042]
2	Awesome God	Our God is an awesome God\nHe reigns from heav...	[en:0.9999957372229569]
3	Beautiful one	Wonderful, so wonderful\nIs Your unfailing lov...	[en:0.9999957797556582]
4	Because of who you are	I worship you Lord\nLord, I love you, Lord\n\n...	[en:0.9999978096878703]

In [12]:

```
lyrics["Language"] = ""

lyrics = lyrics[["Title", "Language", "Language Probs", "Text"]]
# rearranges df


i = 0
for lan in lyrics["Language Probs"]:
    language = str(lan)[1:3]
    if language != "en":
        lyrics["Language"].iloc[i] = "not en"
    else:
        lyrics["Language"].iloc[i] = "en"
    i += 1

lyrics.head()
```

Out[12]:

	Title	Language	Language Probs	Text
0	Adonai	en	[en:0.9999962671595386]	I lift my voice, I lift my praise to You\nI li...
1	Agnus Dei - Lord of All	en	[en:0.9999970413285042]	Alleluia, Alleluia\nFor the Lord God Almighty ...
2	Awesome God	en	[en:0.9999957372229569]	Our God is an awesome God\nHe reigns from heav...
3	Beautiful one	en	[en:0.9999957797556582]	Wonderful, so wonderful\nIs Your unfailing lov...
4	Because of who you are	en	[en:0.9999978096878703]	I worship you Lord\nLord, I love you, Lord\n\n...

In [ ]:

```
```

So, now that we have the best possible language data Google's lyrics provide, we can go back to our old data frame and update the language probabilities with the ones from our lyrics data by performing a left join.

In [13]:

```
# creating an inner join on df and lyrics data

df = df.drop(columns=["Lang Probs", "Language"])

df = pd.merge(df, lyrics, on="Title", how='left')
df.head()
```

Out[13]:

	Title	BPM	Language	Language Probs	Text
0	Adonai	104.0	en	[en:0.9999962671595386]	I lift my voice, I lift my praise to You\nI li...
1	Agnus Dei - Lord of All	69.0	en	[en:0.9999970413285042]	Alleluia, Alleluia\nFor the Lord God Almighty ...
2	Awesome God	72.0	en	[en:0.9999957372229569]	Our God is an awesome God\nHe reigns from heav...
3	Beautiful one	120.0	en	[en:0.9999957797556582]	Wonderful, so wonderful\nIs Your unfailing lov...
4	Cand vom intra pe-a tale porti de aur	78.0	not en	[it:0.7142817428145498, ro:0.14286050323373645...	NaN

In [ ]:

When scanning through the data, I noticed "I will celebrate" is not considered primarily in English. When looking through the data, it appears that the frequency of the word "Hallelujah" in the lyrics causes other languages to be prioritized. I will ignore this and use the algorithm as it stands, but it is important to note that the data for lyrics and language not 100% accurate. A potential fix would be to say that if there is any chance of English being the language then to consider the song English, but in the instances where lyrics are not available and only the title is considered, there is a high chance of resulting in incorrect data (i.e. "Oceanele (Still)" could be incorrectly interpreted as English)

Also, not every song produced the correct lyrics, due to the lack of additional information. The lyrics for "Forever" returned lyrics including curse words -- which is definitely not the one we sing in church. However, the lyrics serve the purpose of determining the language of a song. It should just be noted that the lyrics data should not be counted on in circumstances directly involving a song's lyrics.

Let us now create a new data frame containing only rows where the songs are in English.

In [14]:
```
english = df[df["Language"] == "en"]
english = english[["Title", "BPM"]]
english = english.reset_index(drop=True)
english.head()
```

Out[14]:

	Title	BPM
0	Adonai	104.0
1	Agnus Dei - Lord of All	69.0
2	Awesome God	72.0
3	Beautiful one	120.0
4	Come now is the time to worship	102.0

In [15]:
```
print("There are now " + str(df.shape[0]) + " entries, out of which "
      + str(english.shape[0]) + " are in English.")
```

```
There are now 364 entries, out of which 259 are in English.
```

In [ ]:

So after that process, we have a dataframe with 259 entries, where all the entries are songs in English that have `BPM` data. Let us continue to explore what this looks like where we have `CCLI` entries.

CCLI is a music license number attached to all worship songs. The licensing serves as a way of maintaining copyright laws, but also allowing for the music to be accessible to churches looking to play them weekly. With a CCLI number, searching for data on the song such as themes and scripture references is quite easy. The problem with the current data set, though, is that my church has not been inputting them for all of our songs. Below is some code bringing back the initial data from the Planning Center csv download and creating a left join with the current English song data.

In [16]:
```
ccli_original = english
temp_ccli = temp[["Title", "CCLI"]]
ccli_original = pd.merge(ccli_original, temp_ccli, on='Title', how='lef
t' )
ccli_original.head()
```

Out[16]:

	Title	BPM	CCLI
0	Adonai	104.0	NaN
1	Agnus Dei - Lord of All	69.0	626713.0
2	Awesome God	72.0	NaN
3	Beautiful one	120.0	NaN
4	Come now is the time to worship	102.0	NaN

In [ ]:

Notice that majority of the songs do not have CCLI data. Unfortunately, there is not a great way of obtaining CCLI numbers for music given just the title. So, this project will have to make due with the data available. `ccli_original` will house our English songs and their CCLI numbers for those that have them.

In [17]:

```
ccli_original = ccli_original.dropna()
ccli_original = ccli_original.reset_index(drop=True)

# convert bpm and ccli to integers
ccli_original = ccli_original.astype({'CCLI': 'int32', 'BPM' : 'int32'})
ccli_original.head()
```

Out[17]:

	Title	BPM	CCLI
0	Agnus Dei - Lord of All	69	626713
1	You Are My World	74	3258174
2	Forever	120	7001228
3	In Christ Alone	68	3350395
4	How Great Is Our God	76	4348399

In [18]:

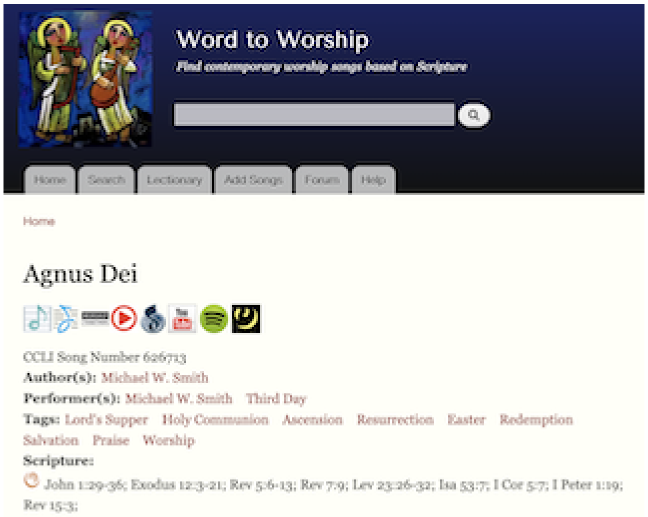
```
print("There are now " + str(ccli_original.shape[0]) + " entries to work with.")
```

```
There are now 160 entries to work with.
```

In [ ]:

Notice that the original data set of 565 songs has been limited to 160 English songs and their CCLI numbers. While this is not ideal, this project is limited to the data available.

However, now that we have a cleaner data set, we can do some more web scraping to retrieve data given the CCLI number that was previously unavailable. Using the website Word to Worship (https://wordtoworship.com/song/502), the program below was able to scrape themes ("Tags" on the website) for each song that appeared in a search result.



Agnus Dei

CCLI Song Number 626713
Author(s): Michael W. Smith
Performer(s): Michael W. Smith Third Day
Tags: Lord's Supper Holy Communion Ascension Resurrection Easter Redemption Salvation Praise Worship
Scripture:
John 1:29-36; Exodus 12:3-21; Rev 5:6-13; Rev 7:9; Lev 23:26-32; Isa 53:7; I Cor 5:7; I Peter 1:19; Rev 15:3;

```
In [19]: import requests
         from bs4 import BeautifulSoup
```

```
In [ ]: ccli_original["Themes"] = ""

        for i in range(ccli_original.shape[0]):
            search = ccli_original["CCLI"].iloc[i]
            url = f"https://wordtoworship.com/search/node/{search}"
            req = requests.get(url, headers={"User-Agent": "XY"}).text
            sor = BeautifulSoup(req, "html.parser")
            soup = sor.find("h3", class_="title")
            try:
                link = [a['href'] for a in soup.find_all('a', href=True)][0]
                song_req = requests.get(link, headers={"User-Agent": "XY"}).text
                song_sor = BeautifulSoup(song_req, "html.parser")
                song_soup = song_sor.find("div", class_="field field-name-field-tags field-type-taxonomy-term-reference field-label-inline clearfix")
                themes = [a.text for a in song_soup.find_all('a')]
                ccli_original["Themes"].iloc[i] = themes
            except:
                pass
```

In [22]:

```python
ccli_df = ccli_original
ccli_df.head()
```

Out[22]:

	Title	BPM	CCLI	Themes
0	Agnus Dei - Lord of All	69	626713	[Lord's Supper, Holy Communion, Ascension, Res...
1	You Are My World	74	3258174	[Prayer, Surrender, Worship]
2	Forever	120	7001228	[Adoration & Praise, Grace & Mercy, Resurrecti...
3	In Christ Alone	68	3350395	[Hope, Security, Resurrection, Life of Jesus, ...
4	How Great Is Our God	76	4348399	[Hymn, Kingship, Majesty, Praise, Worthiness]

In [ ]:

```python

```

`ccli_df` will house all the data where theme information was found.

In [23]:

```python
ccli_df = ccli_df[ccli_df["Themes"] != ""]
ccli_df = ccli_df.reset_index(drop=True)
ccli_df
```

Out[23]:

	Title	BPM	CCLI	Themes
0	Agnus Dei - Lord of All	69	626713	[Lord's Supper, Holy Communion, Ascension, Res...
1	You Are My World	74	3258174	[Prayer, Surrender, Worship]
2	Forever	120	7001228	[Adoration & Praise, Grace & Mercy, Resurrecti...
3	In Christ Alone	68	3350395	[Hope, Security, Resurrection, Life of Jesus, ...
4	How Great Is Our God	76	4348399	[Hymn, Kingship, Majesty, Praise, Worthiness]
...	...	...	...	...
76	Promises	86	7149439	[promises of God, God's faithfulness, Blessing...
77	Graves Into Gardens	70	7138219	[Fulfillment, God's Love, Grace, Mercy, Praise...
78	Hymn Of Heaven	71	7168994	[Worship, Heaven, Hymn, Eternity, Revelation, ...
79	Here As In Heaven	69	7051506	[Kingdom of God, Kingdom of Heaven, Lord's Pra...
80	House Of The Lord	86	7168995	[Celebration, Revival, Church, Acceptance, Gra...

81 rows × 4 columns

In [ ]:

```python

```

Unfortunately, this further limits the data to solely 81 rows. This data however is very clean and can be analyzed much better than the original data.

The code blocks below expand our data into a long format. While less visually appealing, this process makes machine learning algorithms much easier to implement. The lists of themes attached to each song title will be expanded into their own columns, and a `0` will refer to that title not having that theme, while `1` will mean that the song has that theme.

In [29]:
```
theme_set = set()
for theme in ccli_df["Themes"]:
    for i in range(len(theme)):
        text = theme[i].lower()
        if "." in text: # some typos in the data
            texts = text.split(".")
            theme_set.update(texts)
        elif " & " in text:
            texts = text.split(" & ")
            theme_set.update(texts)
        else:
            theme_set.add(text)

theme_df = pd.DataFrame(columns=list(theme_set))
theme_df["Title"] = ccli_df["Title"]


#sets title as first column
titles = theme_df.pop('Title')
theme_df.insert(0, "Title", titles)
theme_df.head()

# initializes 0 for each theme
for i in range(theme_df.shape[0]):
    for theme in theme_set:
        theme_df.at[i, theme.lower()] = 0
```

In [30]:
```
ccli_themes = pd.merge(ccli_df, theme_df, how='left', on='Title')
```

In [31]:
```
for i in range(ccli_themes.shape[0]):
    list_themes = ccli_themes["Themes"].iloc[i]
    for theme in list_themes:
        if " & " in theme:
            themes = theme.split(" & ")
            for tag in themes:
                ccli_themes.at[i, tag.lower()] = 1
        else:
            ccli_themes.at[i, theme.lower()] = 1
```

```
In [50]: ccli_themes = ccli_themes.drop(columns=["CCLI", "Themes", "npursue", "he
         aling. god's presence"])
         #for bug fixing purposes

         ccli_themes = ccli_themes.drop_duplicates()
         ccli_themes = ccli_themes.reset_index(drop=True)
         ccli_themes
```

Out[50]:

	Title	BPM	revival	palm sunday	cross	grace	help	christ	rejoice	victor	...	holy is the lamb	jesus
0	Agnus Dei - Lord of All	69	0	0	0	0	0	0	0	0	...	0	0
1	You Are My World	74	0	0	0	0	0	0	0	0	...	0	0
2	Forever	120	0	0	0	1	0	0	0	0	...	0	0
3	In Christ Alone	68	0	0	0	0	0	0	0	0	...	0	0
4	How Great Is Our God	76	0	0	0	0	0	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
76	Promises	86	0	0	0	0	0	0	0	0	...	0	0
77	Graves Into Gardens	70	0	0	0	1	0	0	0	0	...	0	0
78	Hymn Of Heaven	71	0	0	0	0	0	0	0	0	...	1	0
79	Here As In Heaven	69	0	0	0	0	0	0	0	0	...	0	0
80	House Of The Lord	86	1	0	0	1	0	0	0	0	...	0	0

81 rows × 187 columns

In [33]:

```
df = ccli_themes #since df is shorter to write out than ccli_themes
df.to_excel("clean_data.xlsx", index=False)
df.head()
```

Out[33]:

	Title	BPM	revival	palm sunday	npursue	cross	grace	help	christ	rejoice	...	holy is the lamb	jesus
0	Agnus Dei - Lord of All	69	0	0	0	0	0	0	0	0	...	0	0
1	You Are My World	74	0	0	0	0	0	0	0	0	...	0	0
2	Forever	120	0	0	0	0	1	0	0	0	...	0	0
3	In Christ Alone	68	0	0	0	0	0	0	0	0	...	0	0
4	How Great Is Our God	76	0	0	0	0	0	0	0	0	...	0	0

5 rows × 188 columns

In [ ]:

```

```

Thus, we have cleaned and retrieved as much data as possible from the original messy data into a new data set called `df`, containing song `Title`, `BPM`, and themes. We can now begin setting up some machine learning models.

Part Two: Machine Learning Analysis

The field of data science is in practically unanimous agreement that the most difficult/tedious part of data analysis is the cleaning and preparation of the data sets. From this point on, it is a matter of picking a machine learning model and implementing that into our data set.

Given the type of data we have, a clustering machine learning algorithm would be best to use. Clustering algorithms group variables of data into a certain number of groups, or clusters, where all items in the clusters have similar characteristics. Below, we will use the KMeans algorithm to group the data based on similar themes.

We will first remove "rare" themes from our data set, defining rare to mean that fewer than 4 songs share that theme.

In [34]:

```
rare_cols = []
for col in df.columns:
    if col != "Title" and col != "BPM":
        if sum(df[col]) < 4:
            rare_cols.append(col)
```

In [35]:

```
df = df.drop(columns=rare_cols)
df.head()
```

Out[35]:

	Title	BPM	cross	grace	salvation	worship	praise	declaration	resurrection	freedom	...
0	Agnus Dei - Lord of All	69	0	0	1	1	1	0	1	0	...
1	You Are My World	74	0	0	0	1	0	0	0	0	...
2	Forever	120	0	1	0	0	1	0	1	0	...
3	In Christ Alone	68	0	0	0	0	0	0	1	0	...
4	How Great Is Our God	76	0	0	0	0	1	0	0	0	...

5 rows × 37 columns

In [36]:

```
df.shape[1] # number of columns
```

Out[36]:

```
37
```

In [ ]:

Note that this limits us to 37 columns, so 35 themes (one column is "Title" and one is "BPM"). Now we can start to define our model with 8 clusters. The number 8 was chosen because I want groups of about 10 related songs.

In [37]:

```
from sklearn.cluster import KMeans

# 1. Defining the model: KMeans with 8 clusters
km = KMeans(n_clusters=8)
```

In [38]:

```
# 2. Fitting the model to the data

# gets theme columns to create a df of just themes
themes = []
for col in df.columns:
    if col != "Title" and col != "BPM":
        themes.append(col)

theme_df = pd.DataFrame()
for theme in themes:
    theme_df[theme] = df[theme]


# fitting the model to our theme data
km.fit(theme_df)
centroids = km.cluster_centers_ # will be demonstrated later

theme_df['Cluster'] = km.predict(theme_df)
```

In [39]:

```
theme_df['Cluster']
```

Out[39]:

```
0      2
1      3
2      5
3      2
4      7
      ..
76     0
77     3
78     2
79     7
80     0
Name: Cluster, Length: 81, dtype: int32
```

In [ ]:

```

```

A couple things to note thus far:

- We have temporarily abandoned our BPM data. This model will group similar songs solely by their themes. If BPM data were to be included, the BPM and theme data would have to be normalized so that the BPM data (ranges from approx. 60 to 150) does not count as more important than theme data (0 or 1). Also, the output would not just group songs by their themes, but by BPM as well. This result does not interest me, as I would care firstly about the theme of a song when creating a worship set, but it is worth mentioning that using BPM data is a possible variation.
- The "output" of this model is `theme_df` , which is a df of all the themes plus a column listing the cluster of each song. Now it is just a matter of extracting those cluster groups.

```
In [40]: for i in range(8):
             print("Cluster {}:".format(i))
             print(centroids[i])
             print("----------")
             print()
```

```
Cluster 0:
[-1.38777878e-17  1.00000000e-01  0.00000000e+00  7.00000000e-01
  1.00000000e+00  1.00000000e-01 -5.55111512e-17  0.00000000e+00
 -1.38777878e-17  1.00000000e-01  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  1.00000000e-01  1.00000000e-01 -4.16333634e-17  2.00000000e-01
  0.00000000e+00  2.00000000e-01  6.00000000e-01  0.00000000e+00
  1.00000000e-01  3.00000000e-01  0.00000000e+00  1.00000000e-01
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  2.00000000e-01  0.00000000e+00  2.00000000e-01]
----------

Cluster 1:
[0.         0.         0.         0.33333333 0.         0.33333333
 0.33333333 0.         0.66666667 0.33333333 0.         0.
 0.         0.         0.         1.         0.         0.
 0.33333333 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.        ]
----------

Cluster 2:
[ 1.11111111e-01  1.11111111e-01  1.11111111e-01  2.22222222e-01
  3.33333333e-01  0.00000000e+00  1.00000000e+00  1.11111111e-01
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  2.22222222e-01  0.00000000e+00  0.00000000e+00  1.11111111e-01
  0.00000000e+00  7.77777778e-01 -2.77555756e-17  1.11111111e-01
  0.00000000e+00  0.00000000e+00  2.77555756e-17  1.38777878e-17
  3.33333333e-01  1.11111111e-01  0.00000000e+00  0.00000000e+00
  0.00000000e+00  1.11111111e-01  2.22222222e-01  0.00000000e+00
  0.00000000e+00  0.00000000e+00  1.11111111e-01]
----------

Cluster 3:
[ 1.11111111e-01  2.22222222e-01  1.11111111e-01  2.22222222e-01
  2.22222222e-01  0.00000000e+00 -2.77555756e-17  0.00000000e+00
  1.11111111e-01  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  1.11111111e-01  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  1.11111111e-01  1.11111111e-01
  1.11111111e-01  0.00000000e+00  2.77555756e-17  1.11111111e-01
  0.00000000e+00  0.00000000e+00  1.11111111e-01  1.00000000e+00
  0.00000000e+00  2.22222222e-01  1.11111111e-01  2.22222222e-01
  0.00000000e+00  0.00000000e+00 -2.77555756e-17]
----------

Cluster 4:
[ 2.00000000e-01  6.00000000e-01  8.00000000e-01  0.00000000e+00
  6.00000000e-01  0.00000000e+00 -2.77555756e-17  1.00000000e+00
  2.00000000e-01  0.00000000e+00  0.00000000e+00  8.00000000e-01
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00 -1.38777878e-17  2.00000000e-01
  0.00000000e+00  0.00000000e+00  0.00000000e+00  2.00000000e-01
  2.00000000e-01  4.00000000e-01  2.00000000e-01  0.00000000e+00
  0.00000000e+00  2.00000000e-01  0.00000000e+00  0.00000000e+00
  0.00000000e+00  2.00000000e-01 -1.38777878e-17]
----------
```

```
Cluster 5:
[0.   0.5  0.   0.   1.   0.25 0.75 0.   0.   0.   0.5  0.   0.   0.
 0.   0.   0.   1.   0.25 0.5  0.   0.   1.   0.   0.   0.   0.   0.
 0.   0.5  0.75 0.   0.25 0.   0.  ]
----------

Cluster 6:
[0.   0.   0.   0.   0.25 0.   0.   0.   0.25 0.75 0.   0.   0.25 0.
 0.25 0.   0.   0.   0.   0.   0.   0.   0.   0.   1.   0.25 0.   0.
 0.   0.   0.25 0.   0.25 0.75 0.5 ]
----------

Cluster 7:
[ 5.40540541e-02  1.08108108e-01  2.70270270e-02  5.40540541e-02
  2.16216216e-01  8.10810811e-02  8.32667268e-17  5.55111512e-17
 -1.38777878e-17  2.70270270e-02  5.40540541e-02  0.00000000e+00
  2.70270270e-02  8.10810811e-02  8.10810811e-02  0.00000000e+00
  8.10810811e-02  1.11022302e-16  1.62162162e-01  5.40540541e-02
  8.10810811e-02  5.40540541e-02  2.70270270e-02  1.35135135e-01
  2.70270270e-02  2.70270270e-02  5.40540541e-02 -2.77555756e-17
  1.08108108e-01  2.70270270e-02  4.16333634e-17  5.40540541e-02
  1.08108108e-01  5.40540541e-02  1.08108108e-01]
----------
```

The above code prints out the "centroids". These "points" are what define each cluster. As in, items in Cluster 7 would be as similar as possible to:

- 0 of theme 0
- 0 of theme 1

...

- 0.125 of theme 3, etc.

Now, we can import the cluster data back to the data frame to start presenting the findings.

In [41]:

```
df['Cluster'] = theme_df['Cluster']
```

In [42]:

```
def cluster_getter(df, num):
    '''
    This function takes a df and an integer to output a cluster list
    '''
    cluster_list = []
    for i in range(df.shape[0]):
        if df['Cluster'].iloc[i] == num:
            cluster_list.append(df["Title"].iloc[i])
    return cluster_list
```

In [43]:

```
clusters = dict()
for i in range(8):
    cluster_list = cluster_getter(df, i)
    clusters[i] = cluster_list
```

In [44]:

```
for num in clusters:
    print("---------------------------")
    print("Group {}: ".format(num))
    for i in range(len(clusters[num])):
        print(clusters[num][i])
    print()
```

```
---------------------------
Group 0:
Lord I Need You
All The Poor And Powerless
With Everything
Only King Forever
At Your Name
Way Maker
Raise A Hallelujah
I Give You My Heart
Promises
House Of The Lord

---------------------------
Group 1:
Glorious (Paul Baloche)
Come Lord Jesus (Even So Come)
Open Up The Heavens

---------------------------
Group 2:
Agnus Dei - Lord of All
In Christ Alone
Happy Day
Your Great Name
O Praise The Name (Anástasis)
Glorious Day
Jerusalem
Overcome
Hymn Of Heaven

---------------------------
Group 3:
You Are My World
The Stand
Came To My Rescue
Arms Open Wide
Give Me Faith
Lay Me Down
Goodness Of God / Bunatatea Ta
From The Inside Out
Graves Into Gardens

---------------------------
Group 4:
All Because Of Jesus
O Come To The Altar
Who You Say I Am
Living Hope
No Longer Slaves

---------------------------
Group 5:
Forever
Forever
This I Believe
King Of Kings
```

```
---------------------------
Group 6:
Jesus Paid It All
Break Every Chain
What A Beautiful Name
Yet Not I But Through Christ In Me

---------------------------
Group 7:
How Great Is Our God
Above All
At The Cross
The Same Love
Desert Song
We Fall Down
God Is Able
Your Name
Victor's Crown
Holy Spirit
A Mighty Fortress
This Is Amazing Grace
Enough
Good Good Father
Healer
Beneath The Waters (I Will Rise)
Great I Am
The Greatness Of Our God
None But Jesus
Build My Life
In Jesus' Name
King of My Heart
Oh Our Lord
He Shall Reign Forevermore
My Lighthouse
I Am
Do It Again
See A Victory
Yes And Amen
Cannons
God I Look To You
God So Loved
Alive & Breathing
Whom Shall I Fear
Joy Has Dawned
The Blessing
Here As In Heaven
```

In [45]:

```
df.to_excel("theme_clusters.xlsx", index=False)
```

In [ ]:

There it is! The code above simply formats the results to print out nicely as seen above. Each of the groups have similar themes. Notice how the songs from Group 6 below all have common themes of 'Second Coming'.

In [48]:

```
ccli_df.set_index("Title", inplace=True)

for song in clusters[1]:
    print(song + ":")
    print(ccli_df["Themes"].loc[song])
    print()
```

```
Glorious (Paul Baloche):
['Declaration', 'Faith', 'Glory', 'Resurrection', 'Second Coming']

Come Lord Jesus (Even So Come):
['Advent', 'Second Coming']

Open Up The Heavens:
['Heaven', 'Second Coming', 'Worship', 'Glory', 'Power', 'Singing', 'Ca
ll to Worship', 'Presence', 'Invocation']
```

In [ ]:

Conclusion

In summary, this project began by downloading and processing my church's Planning Center song data. Realizing that lots of work would be needed, all variables except the Title and BPM were abandoned. The data was indexed to only contain songs that were in English, had a BPM, and contained a CCLI number. Given that information, theme variables were web scraped and the result was a data set containing title, BPM, and song variables.

With that data, the KMeans algorithm was used to create 8 groups of songs with closely-linked themes. Thus, all the cleaning and modeling resulted in the following visualization (created using Tableau):

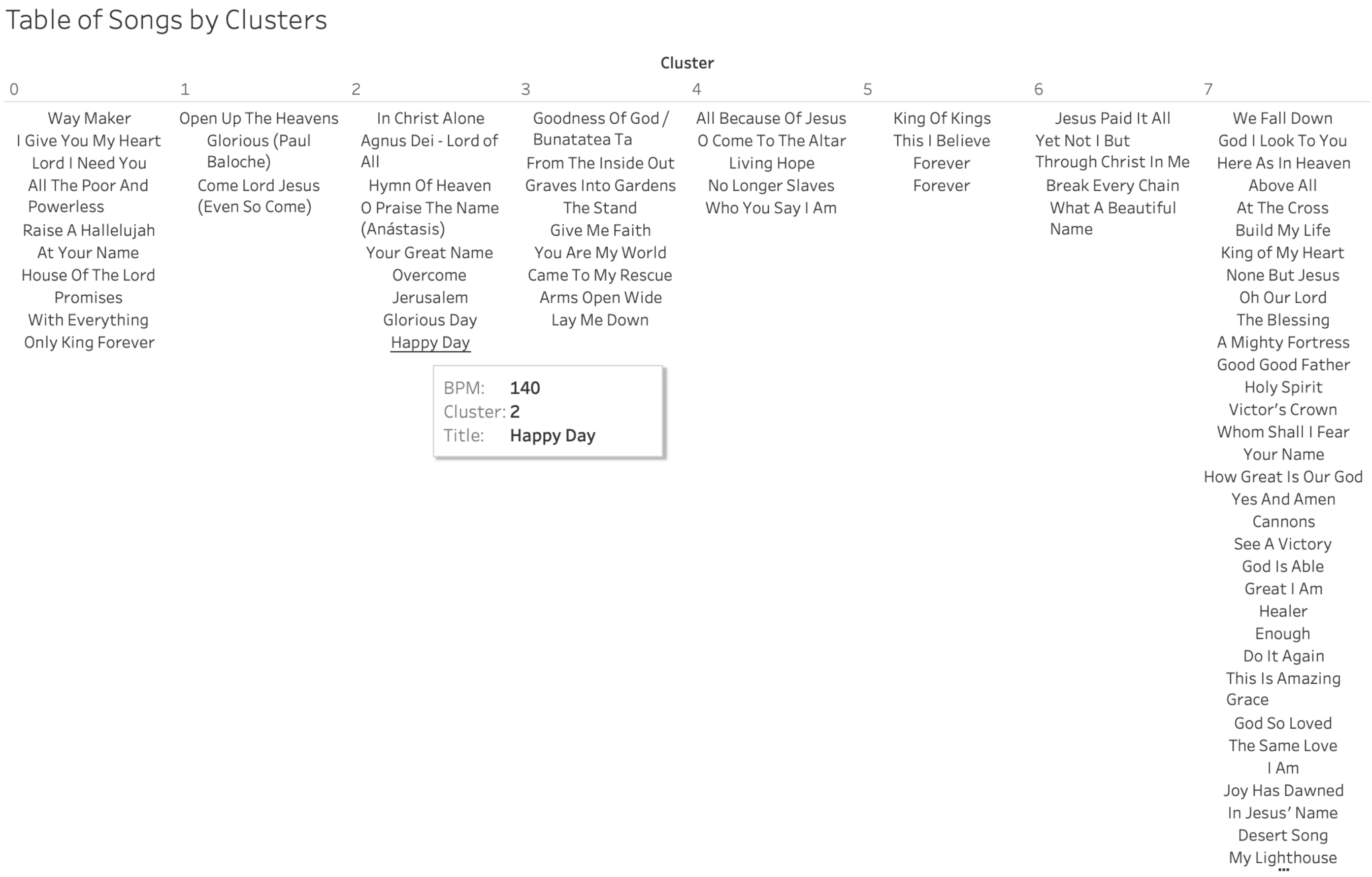


Table of Songs by Clusters

Cluster							
0	1	2	3	4	5	6	7
Way Maker	Open Up The Heavens	In Christ Alone	Goodness Of God / Bunatatea Ta	All Because Of Jesus	King Of Kings	Jesus Paid It All	We Fall Down
I Give You My Heart	Glorious (Paul Baloche)	Agnus Dei - Lord of All	From The Inside Out	O Come To The Altar	This I Believe	Yet Not I But Through Christ In Me	God I Look To You
Lord I Need You	Come Lord Jesus (Even So Come)	Hymn Of Heaven	Graves Into Gardens	Living Hope	Forever	Break Every Chain	Here As In Heaven
All The Poor And Powerless		O Praise The Name (Anástasis)	The Stand	No Longer Slaves	Forever	What A Beautiful Name	Above All
Raise A Hallelujah		Your Great Name	Give Me Faith	Who You Say I Am			At The Cross
At Your Name		Overcome	You Are My World				Build My Life
House Of The Lord		Jerusalem	Came To My Rescue				King of My Heart
Promises		Glorious Day	Arms Open Wide				None But Jesus
With Everything		Happy Day	Lay Me Down				Oh Our Lord
Only King Forever							The Blessing
							A Mighty Fortress
							Good Good Father
							Holy Spirit
							Victor’s Crown
							Whom Shall I Fear
							Your Name
							How Great Is Our God
							Yes And Amen
							Cannons
							See A Victory
							God Is Able
							Great I Am
							Healer
							Enough
							Do It Again
							This Is Amazing Grace
							God So Loved
							The Same Love
							I Am
							Joy Has Dawned
							In Jesus’ Name
							Desert Song
							My Lighthouse
							...

BPM: 140
Cluster: 2
Title: Happy Day

Reflection

While I am very satisfied with the result given the context of the extremely poor data to start with, the overwhelming feeling throughout the entirety of the project was that I really wished I had better data. In the end, I am confident that the implementation of machine learning models used above would have produced even better results with improved data.

Given better data and time, I would have loved to expand on this project by creating a user interface by which users would input the theme of a preacher's message (say, "confession") and the program would output a 4-5 songs that relate to that theme and would be listed in an order matching the flow of the service (i.e. slower to faster tempo).

References

- akashkumarsen4. “Scrape Google Search Results Using Python Beautifulsoup.” GeeksforGeeks, GeeksforGeeks, 29 Dec. 2020, https://www.geeksforgeeks.org/scrape-google-search-results-using-python-beautifulsoup/ (https://www.geeksforgeeks.org/scrape-google-search-results-using-python-beautifulsoup/).
- Khushijain. “K-Means Clustering: Python Implementation from Scratch.” Medium, Towards Data Science, 3 July 2021, https://medium.com/nerd-for-tech/k-means-python-implementation-from-scratch-8400f30b8e5c (https://medium.com/nerd-for-tech/k-means-python-implementation-from-scratch-8400f30b8e5c).
- Pietro, Mauro Di. “Text Analysis & Feature Engineering with NLP.” Medium, Towards Data Science, 15 Mar. 2022, https://towardsdatascience.com/text-analysis-feature-engineering-with-nlp-502d6ea9225d (https://towardsdatascience.com/text-analysis-feature-engineering-with-nlp-502d6ea9225d).