Podcast Market Analysis: Selecting The Most Promising Podcast Category

```
import sqlite3
import warnings
import pandas as pd
import plotly.express as px
from scipy.stats import chi2_contingency, chi2, kruskal, spearmanr
from statsmodels.stats.proportion import proportion_confint
from utils.functions import *
```

Dataset Cleaning

The main objectives for this part of the notebook are:

- Do univariate analysis to get familiar with the dataset.
- Discover and address data quality issues.

At the end of this section, "Key Takeaways" provides a condensed overview of the main findings and observations. The goal to distil the most important conclusions, setting the stage for bivariate data analysis.

The database consists of three tables: "podcasts", "reviews" and "ratings". They are connected by the "podcast_id", which is used to identify individual podcasts. In this section, we will look at the tables individually.

Podcasts Table

```
con = sqlite3.connect("database.sqlite")
query = """
SELECT *
FROM podcasts
"""
pd.read_sql_query(query, con).head()
```

```
                         podcast_id   itunes_id  \
0  a00018b54eb342567c94dacfb2a3e504  1313466221
1  a00043d34e734b09246d17dc5d56f63c   158973461
2  a0004b1ef445af9dc84dad1e7821b1e3   139076942
3  a00071f9aaae9ac725c3a586701abf4d  1332508972
4  a000a500f06555f81220c3eb641aded7  1544900779

                                    slug  \
0                         scaling-global
1  cornerstone-baptist-church-of-orlando
```

```
2            mystery-dancing-in-the-dark
3                      kts-money-matters
4       word-on-the-street-w-dreak-swift

                                          itunes_url  \
0  https://podcasts.apple.com/us/podcast/scaling-...
1  https://podcasts.apple.com/us/podcast/cornerst...
2  https://podcasts.apple.com/us/podcast/mystery-...
3  https://podcasts.apple.com/us/podcast/kts-mone...
4  https://podcasts.apple.com/us/podcast/word-on-...

                                   title
0                         Scaling Global
1  Cornerstone Baptist Church of Orlando
2           Mystery: Dancing in the Dark
3                      KTs Money Matters
4      Word on the Street w/ Dreak Swift
```

- This table conatins identification data rather than information that can be used to the understand data content.
- For a broader trend analysis, this information will have no use.

How many podcasts are in the table?

```
query = """
SELECT count(*)
FROM podcasts
"""
con.execute(query).fetchall()[0][0]

110024
```

- There are 110024 podcasts in the table.

Missing Values

```
query = """
SELECT
    COUNT(CASE WHEN podcast_id IS NULL THEN 1 END) AS
podcast_id_null_count,
    COUNT(CASE WHEN itunes_id IS NULL THEN 1 END) AS
itunes_id_null_count,
    COUNT(CASE WHEN slug IS NULL THEN 1 END) AS slug_null_count,
    COUNT(CASE WHEN itunes_url IS NULL THEN 1 END) AS
itunes_url_null_count,
    COUNT(CASE WHEN title IS NULL THEN 1 END) AS title_null_count
FROM podcasts;
"""
pd.read_sql_query(query, con)
```

```
   podcast_id_null_count  itunes_id_null_count  slug_null_count  \
0                      0                     0                0

   itunes_url_null_count  title_null_count
0                      0                 0
```

- The dataset seems to be clean with no missing values.

Duplicates

```
query = """
SELECT DISTINCT podcast_id
FROM podcasts
GROUP BY podcast_id
HAVING COUNT(*) > 1;
"""
con.execute(query).fetchall()

[]
```

- By the ID number (primary key), all podcats are unique.

Maybe some podcasts have the same titles?

```
query = """
SELECT COUNT(*)
FROM (
    SELECT title
    FROM podcasts
    GROUP BY title
    HAVING COUNT(*) > 1
)
"""
con.execute(query).fetchall()[0][0]

675
```

- 675 podcasts have a shared title with another podcast.

Reviews Table

Next, I will explore the variables that will be used to analyze podcast market.

```
query = """
SELECT *
FROM reviews
"""
all_reviews = pd.read_sql_query(query, con)
all_reviews.head()
```

```
                         podcast_id  \
0  c61aa81c9b929a66f0c1db6cbe5d8548
1  c61aa81c9b929a66f0c1db6cbe5d8548
2  ad4f2bf69c72b8db75978423c25f379e
3  ad4f2bf69c72b8db75978423c25f379e
4  ad4f2bf69c72b8db75978423c25f379e

                                              title  \
0                               really interesting!
1  Must listen for anyone interested in the arts!!!
2                                 nauseatingly left
3                                   Diverse stories
4                                              □□□□

                                             content  rating
author_id  \
0  Thanks for providing these insights.  Really e...       5
F7E5A318989779D
1  Super excited to see this podcast grow. So man...       5
F6BF5472689BD12
2  I'm a liberal myself, but its pretty obvious a...       1
1AB95B8E6E1309E
3  I find Tedx talks very inspirational but I oft...       5
11BB760AA5DEBD1
4                I love this podcast, it is so good.       5
D86032C8E57D15A

                  created_at
0  2018-04-24T12:05:16-07:00
1  2018-05-09T18:14:32-07:00
2  2019-06-11T14:53:39-07:00
3  2018-05-31T13:08:09-07:00
4  2019-06-19T13:56:05-07:00
```

- This table has multiple useful features related to the reception of the podcast: review rating, time of the review, and reviewer's ID.
- For this project, I will analyze how these three particular features relate to the podcast category, which will be discussed in the "categories" table.

Lets see the number of unique podcasts in this table:

```
query = """
SELECT COUNT(DISTINCT podcast_id)
FROM reviews
"""
con.execute(query).fetchall()[0][0]

111544
```

- There are more reviewed podcasts than podcasts in the "podcasts" table. The "reviews" table has 111,544 unique podcasts, while the Podcasts table has 110,024.

How many podcasts (podcast_id) between the tables are matching?

```
query = """
SELECT COUNT(DISTINCT r.podcast_id)
FROM reviews r
JOIN podcasts p
ON r.podcast_id = p.podcast_id
"""
con.execute(query).fetchall()[0][0]

110024
```

- Reviews table includes all podcasts from the podcasts table and has a few additional ones.

Misssing Values

```
query = """
SELECT
    COUNT(CASE WHEN podcast_id IS NULL THEN 1 END) AS
podcast_id_null_count,
    COUNT(CASE WHEN title IS NULL THEN 1 END) AS title_null_count,
    COUNT(CASE WHEN content IS NULL THEN 1 END) AS content_null_count,
    COUNT(CASE WHEN rating IS NULL THEN 1 END) AS rating_null_count,
    COUNT(CASE WHEN author_id IS NULL THEN 1 END) AS
author_id_null_count,
    COUNT(CASE WHEN created_at IS NULL THEN 1 END) AS
created_at_null_count
FROM reviews;
"""
pd.read_sql_query(query, con)

   podcast_id_null_count  title_null_count  content_null_count  \
0                      0                 0                   0

   rating_null_count  author_id_null_count  created_at_null_count
0                  0                     0                      0
```

- No missing values in the table.

Duplicates

Duplicates by the ID number (review count):

```
query = """
SELECT podcast_id, COUNT(*) as number_of_reviews
FROM reviews
```

```
GROUP BY podcast_id
ORDER BY COUNT(*) DESC
"""
pd.read_sql_query(query, con).head()

                         podcast_id  number_of_reviews
0  bf5bf76d5b6ffbf9a31bba4480383b7f              33104
1  bc5ddad3898e0973eb541577d1df8004              10675
2  bad6c91efdbee814db985c7a65199604               9698
3  f5fce0325ac6a4bf5e191d6608b95797               8248
4  f2377a9b0d9a2e0fb05c3dad55759328               7389
```

- As expected, many podcasts appear multiple times in the table due to having multiple reviews.
- The number of reviews can be considered a metric for listener engagement and podcast popularity. Podcasts with the most reviews are the most popular, while those with the fewest reviews are the least popular.
- A single podcast is doing exceptionally well with almost three times the reviews as the second most popular podcast.

What is this podcast?

```
query = """
SELECT DISTINCT p.title, c.category, p.itunes_url
FROM podcasts p
JOIN reviews r
ON p.podcast_id = r.podcast_id
JOIN categories c
ON p.podcast_id = c.podcast_id
WHERE r.podcast_id = 'bf5bf76d5b6ffbf9a31bba4480383b7f'
"""
pd.read_sql_query(query, con)

          title    category
itunes_url
0  Crime Junkie  true-crime
https://podcasts.apple.com/us/podcast/crime-ju...
```

- The performance of this podcast may distort the overall trend for this genre. I will remove it from the analysis.

Anomalies In Podcast Review Counts

Some people may try to artificially inflate their review count to appear more popular or manipulate the podcast recommendation system. Setting a standard for when a review is considered suspicious is difficult, as it's natural for some reviews to contain the same information. For example, short reviews like "good job" or "great podcast" are common. I will try a few different approaches based on author IDs, content, and the title of the review.

First, let's examine the number of reviews that might be falsified, by looking at instances where a single user leaves more than three reviews with the same title. For this, I've chosen the threshold of 3 because I believe that people are unlikely to write the exact same text multiple times. I did not set this limit to 2 because I think than frequent listeners might repeat a compliment multiple times.

```
query = """
SELECT SUM(counts)
FROM (
    SELECT COUNT(*) counts
    FROM reviews
    GROUP BY author_id, content
    HAVING COUNT(*) > 3
)
"""
con.execute(query).fetchall()[0][0]
```

```
2621
```

- If a user has more than three reviews that look suspicious, then 2,621 reviews would likely be fake.
- That is a small number considering there are over 2 million reviews in total.

Let's run the same test, but this time focusing on duplicate titles instead of review content.

```
query = """
SELECT SUM(counts)
FROM (
    SELECT COUNT(*) counts
    FROM reviews
    GROUP BY author_id, title
    HAVING COUNT(*) > 3
)
"""
con.execute(query).fetchall()[0][0]
```

```
21300
```

- The number of suspicious reviews identified by duplicated titles for each author has increased almost tenfold to 21,300.

How may authors have left these reviews?

```
query = """
SELECT COUNT(author_id)
FROM (
    SELECT DISTINCT author_id
    FROM reviews
    GROUP BY author_id, title
    HAVING COUNT(*) > 3
)
"""
con.execute(query).fetchall()[0][0]
```

```
2141
```

- There are 2,141 users that left the previously identified 21,300 reviews.

People are probably more sophisticated and would use multiple accounts to leave fake reviews. Let's try a different approach based on identical review content for individual podcasts. Since we are now examining the entire lifespan of the podcast, there's a higher chance of encountering the same reviews. I will look for reviews that appear more than 10 times.

```
query = """
SELECT SUM(counts)
FROM (
    SELECT COUNT(*) as counts
    FROM reviews
    GROUP BY content, podcast_id
    HAVING COUNT(*) > 10
)
"""
con.execute(query).fetchall()[0][0]
```

```
524
```

- Only 524 reviews have identical content.

Previously, using the review title helped identify more suspicious reviews. Let's run the same test focusing on titles this time.

```
query = """
SELECT SUM(counts)
FROM (
    SELECT COUNT(*) as counts
    FROM reviews
    GROUP BY title, podcast_id
    HAVING COUNT(*) > 10
)
"""
con.execute(query).fetchall()[0][0]
```

```
40789
```

- The number of duplicated reviews is now much higher, totaling 40,789.

Now lets try to identify individual podcasts that have a lot of possible fraudulent reviews. I will use the title, as the content has very few duplications, and continue with the threshold of 15 identical reviews for a podcast.

```
query = """
SELECT r.podcast_id, rc.number_of_podcasts, COUNT(*) AS
num_of_suspicious_podcasts, SUM(r.counts) AS num_of_duplicate_reviews
FROM (
    SELECT podcast_id, title, COUNT(*) AS counts
    FROM reviews
    GROUP BY title, podcast_id
    HAVING COUNT(*) > 15
) AS r
JOIN (
    SELECT podcast_id, COUNT(*) AS number_of_podcasts
    FROM reviews
    GROUP BY podcast_id
) AS rc
ON r.podcast_id = rc.podcast_id
GROUP BY r.podcast_id, rc.number_of_podcasts
ORDER BY num_of_suspicious_podcasts DESC
"""
pd.read_sql_query(query, con).head()
```

```
                         podcast_id  number_of_podcasts  \
0  bf5bf76d5b6ffbf9a31bba4480383b7f               33104
1  bad6c91efdbee814db985c7a65199604                9698
2  bc5ddad3898e0973eb541577d1df8004               10675
3  c8bde52bde033bac86f0892998f7c062                5099
4  f5fce0325ac6a4bf5e191d6608b95797                8248

   num_of_suspicious_podcasts  num_of_duplicate_reviews
0                         180                      9215
1                          46                      2013
2                          31                      1052
3                          27                       969
4                          26                       666
```

- Two podcasts stand out for potentially having a lot of anomalous activity. One is the previously identified outlier, "Crime Junkie," which has by far the most reviews. The other podcast in question has nearly twice the number of possible duplicates (2013) compared to the next podcast on the list (1052).

What is this podcast?

```
query = """
SELECT DISTINCT p.title, c.category, p.itunes_url
FROM podcasts p
JOIN reviews r
ON p.podcast_id = r.podcast_id
JOIN categories c
ON p.podcast_id = c.podcast_id
WHERE r.podcast_id = 'bad6c91efdbee814db985c7a65199604'
```

```
"""
pd.read_sql_query(query, con)

              title                         category  \
0  Wow in the World                        education
1  Wow in the World                      kids-family
2  Wow in the World  kids-family-stories-for-kids

                                          itunes_url
0  https://podcasts.apple.com/us/podcast/wow-in-t...
1  https://podcasts.apple.com/us/podcast/wow-in-t...
2  https://podcasts.apple.com/us/podcast/wow-in-t...
```

- This is a podcast for kids. I will exclude this from the analysis.

Anomaly Detection Summary

The difficulty mainly comes from deciding when a review seems suspicious. In my quick look at this, I used certain standards, but they were only guesses that could be set higher or lower. Creating a better system to spot questionable reviews or podcasts would need much more work.

The key points are: 1) This approach may not catch all fraudulent reviews, but I will go with the straightforward and less contentious method of removing users who have posted the same review more than three times. 2) Exclude two particular podcasts that show a notably higher number of duplicated reviews compared to others.

Podcasts With Very Low Number Of Reviews

By excluding podcasts with minimal review count, we should get a clearer and more representative analysis. Low-review podcasts might suggest creators with limited commitment or only a few uploads. This could distort insights and misrepresent the genuine efforts. I'll remove all podcasts with fewer than three reviews, but this threshold could be set higher.

```
cursor = con.cursor()
cursor.execute('''CREATE VIEW IF NOT EXISTS FilteredReviews AS
                    SELECT podcast_id
                    FROM reviews
                    WHERE podcast_id !=
'bf5bf76d5b6ffbf9a31bba4480383b7f'
                      AND podcast_id !=
'bad6c91efdbee814db985c7a65199604'
                      AND author_id NOT IN (
                        SELECT DISTINCT author_id
                        FROM reviews
                        GROUP BY author_id, title
                        HAVING COUNT(*) > 3
                      )
                    GROUP BY podcast_id
                    HAVING COUNT(*) > 3''')
```

```
con.commit()
query = """
SELECT COUNT(*)
FROM FilteredReviewss
"""
con.execute(query).fetchall()[0][0]
```

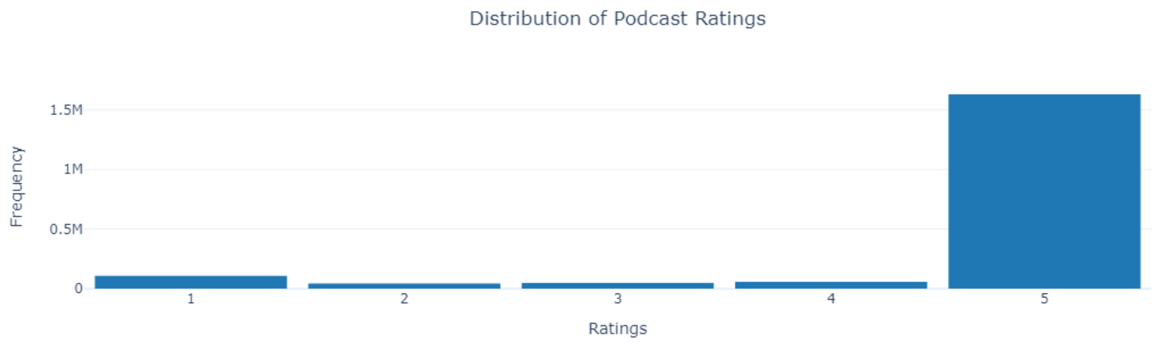
```
46911
```

- Dropping anomalous reviews and podcasts with fewer than three reviews leaves 46,911 out of 111,544 podcasts, which is a 58% reduction.
- Majority of podcasts in Apple Podcasts are empty, abandoned or very successful.

I will continue data cleaning without these podcasts and reviews.

Ratings Distribution

```
query = """
SELECT rating
FROM reviews
WHERE podcast_id IN (
    SELECT podcast_id
    FROM reviews
    WHERE podcast_id != 'bf5bf76d5b6ffbf9a31bba4480383b7f'
    AND podcast_id != 'bad6c91efdbee814db985c7a65199604'
    GROUP BY podcast_id
    HAVING COUNT(*) > 3
)
AND author_id NOT IN (
    SELECT DISTINCT author_id
    FROM reviews
    GROUP BY author_id, title
    HAVING COUNT(*) > 3
)
"""
df = pd.read_sql_query(query, con)
plot_hist(df)
```

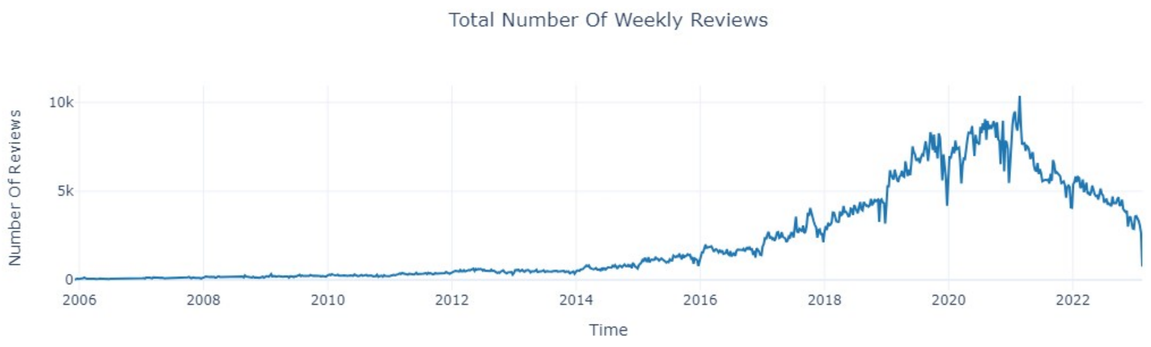


- 87% of all ratings are "5". This could mean that this data has limited usefulness for drawing insights about the podcasts.

Reviews Over Time

To visualize the overall trend, I will include podcasts with fewer than three reviews.

```
query = """
SELECT strftime('%Y-%m-%d', created_at, 'weekday 0', '-6 days') as
review_week, COUNT(*) as num_reviews
FROM reviews
WHERE podcast_id != 'bf5bf76d5b6ffbf9a31bba4480383b7f'
AND podcast_id != 'bad6c91efdbee814db985c7a65199604'
AND author_id NOT IN (
    SELECT DISTINCT author_id
    FROM reviews
    GROUP BY author_id, title
    HAVING COUNT(*) > 3
)
GROUP BY review_week
ORDER BY review_week
"""
df = pd.read_sql_query(query, con)
plot_line(df)
```



- Apple Podcasts (iTunes) is a waning platform with a decreasing user base, reaching its peak at the beginning of 2021 and diminishing ever since.
- A shrinking user base means fewer potential listeners. A declining platform can hinder the visibility, reach, and monetization opportunities.
- Since the platform is losing many listeners, it does not make sense to choose the best category for a podcast based on the size of the market. Instead, we'll focus on areas where podcasts are retaining their audience or are declining the least.
- In the future, to improve the visualization's clarity, I'll exclude dates before 2017 and after 2023. A platform typically wouldn't lose most of its users in one month, causing such a significant drop in reviews.
- Every December, the number of reviews drops.

Categories Table

This table contains the main variable of interest (target feature) for this analysis.

```
query = """
SELECT *
FROM categories
"""
pd.read_sql_query(query, con).head()
```

```
                         podcast_id               category
0  c61aa81c9b929a66f0c1db6cbe5d8548                   arts
1  c61aa81c9b929a66f0c1db6cbe5d8548  arts-performing-arts
2  c61aa81c9b929a66f0c1db6cbe5d8548                  music
3  ad4f2bf69c72b8db75978423c25f379e                   arts
4  ad4f2bf69c72b8db75978423c25f379e            arts-design
```

- The podcast category provides insights into the niche and genre of the content. It identifies the target audience and the specific interests that the podcast focuses on. This is the target feature for this analysis.
- From the first few rows of the dataframe there is a clearly visible issue: some podcasts have multiple categories. As mentioned in the Readme, this table includes all historical data for every category a podcast has ever had, without any indication which category could be considered the correct one or providing any ability to track changes.
- This issue introduces a lot of uncertainty into the analysis because it is unclear which category to associate with the reviews. I'll need to decide how to address this.

Number of podcasts in "categories" table:

```
query = """
SELECT COUNT(DISTINCT podcast_id)
FROM categories
"""
con.execute(query).fetchall()[0][0]
```

```
110024
```

- The same number as matching podcasts from "podcast" and "reviews" tables. Since this is really unlikely to happen by accident, I will assume these are the same podcasts (same podcast_id).
- We will join the reviews with the categories. Additional information from the "reviews" table will not be used.

Number Of Podcasts For Analysis

```
query = """
SELECT COUNT(DISTINCT c.podcast_id)
FROM categories c
WHERE c.podcast_id IN (SELECT podcast_id FROM FilteredReviews)
```

```
"""
con.execute(query).fetchall()[0][0]
```

```
46240
```

- After joining the "category" and "review" tables and removing anomalous reviews, there are 46,240 podcasts availible for analysis.
- This number is slightly smaller than 46,911, which was from the "reviews" table. This makes sense, since we know that the "categories" table has fewer unique podcasts than the "reviews" table.

Number Of Categories

```
query = """
SELECT COUNT(DISTINCT category)
FROM categories
"""
con.execute(query).fetchall()[0][0]
```

```
110
```

- Categories feature has 110 individual categories.

Does removing very small podcasts and anomalies (reviews), remove any categories?

```
query = """
SELECT COUNT(DISTINCT category)
FROM categories c
WHERE c.podcast_id IN (
    SELECT podcast_id FROM FilteredReviews
)
"""
con.execute(query).fetchall()[0][0]
```

```
110
```

- Number of categories remains the same.

Duplicates

How many podcasts have changed their category?

```
query = """
SELECT COUNT(*)
FROM (
    SELECT c.podcast_id
    FROM categories c
    WHERE c.podcast_id IN (
        SELECT podcast_id FROM FilteredReviews
    )
    GROUP BY c.podcast_id
    HAVING COUNT(*) > 1
)
"""
con.execute(query).fetchall()[0][0]
```

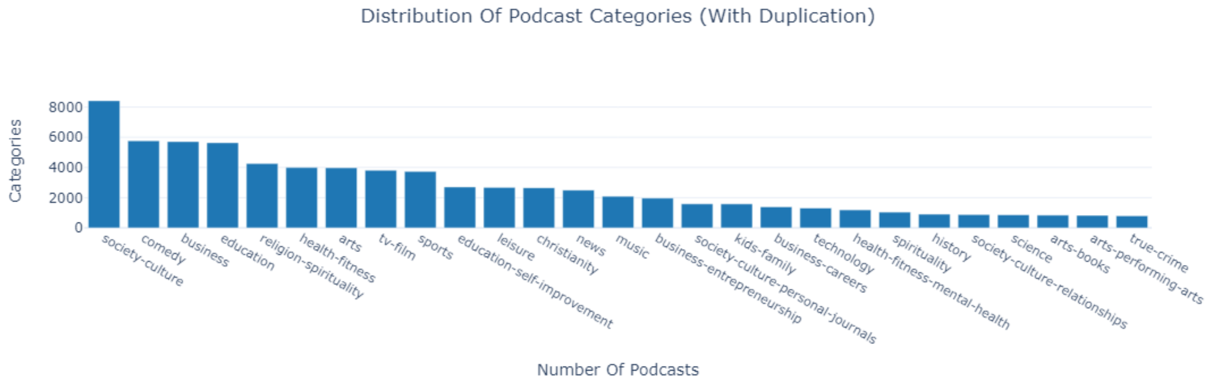
```
29592
```

- Out of 46,240 available podcasts, 29,592 (63%) have changed their category at least once.

Category Distribution

Because of the duplication, there is no clear way to understand the exact podcast categories, but it is still important to know what this variable contains. Since there are so many categories, they would not fit in a single plot, I will limit the plot to include the catogories that have more than 800 podcasts.

```
query = """
SELECT category, COUNT(*) AS counts
FROM categories c
WHERE c.podcast_id IN (
    SELECT podcast_id FROM FilteredReviews
)
GROUP BY category
HAVING counts > 800
ORDER BY COUNT(*) DESC
"""
df = pd.read_sql_query(query, con)
plot_counts(df, "Distribution Of Podcast Categories (With Duplication)")
```

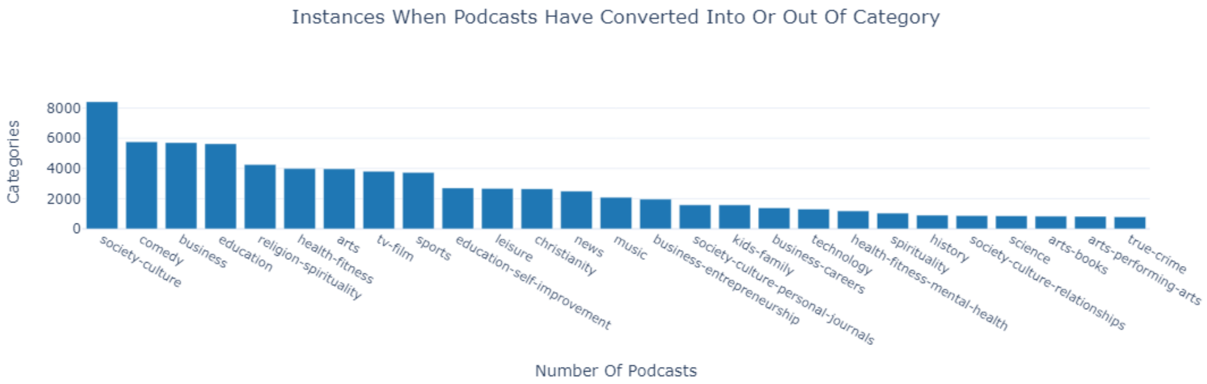


- Most podcasts have chosen broad categories.
- Various categories with fewer podcasts are more specific and also technically belong to a larger group. This is more visible, if the plot includes all podcasts.
- Society-culture is the biggest category by a clear margin.

Changes In Categories

Perhaps there is a pattern that could lead to a straightforward solution to the duplication issue. Let's see which categories have the most changes.

```
query = """
SELECT cc.category, COUNT(*) AS counts
FROM categories cc
WHERE cc.podcast_id IN (
    SELECT podcast_id FROM FilteredReviews
)
GROUP BY cc.category
HAVING COUNT(*) > 800
ORDER BY counts DESC
"""
df = pd.read_sql_query(query, con)
plot_counts(df, "Instances When Podcasts Have Converted Into Or Out Of
Category")
```



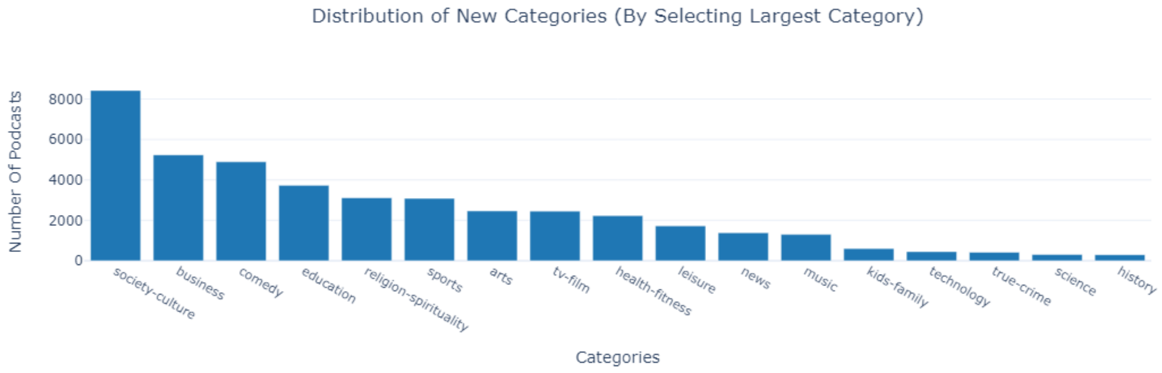
- The duplications are spread out.
- The big categories (that are characterised by fewer words) have the most changes. More precise categories have fewer conversions.
- There is no simple way to address this problem.

Handling Duplicated Categories

To tackle this duplication issue, I'm using a "Majority Category Determination" strategy. For each podcast that has multiple category entries, I'll pick the category that appears most often in the whole dataset. This should allow to analyse the broad trends. There is also a "Minority Category Determination" which would identify the least frequent categories associated with each podcast. This method can provide insights into niche or specialized content areas within that may be overlooked in a majority-focused analysis. Ultimately, both methods should be used, and their results compared to provide a more balanced view of the podcast market. For this project, I'll focus only on a single method (Majority Category Determination).

For clarity, the plot will contain only categories with more than 300 podcasts.

```
query = """
WITH RankedCategories AS (
    SELECT
        category,
        ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC) AS ordering_number
    FROM categories c
    WHERE c.podcast_id IN (
        SELECT podcast_id FROM FilteredReviews
    )
    GROUP BY category
)
SELECT c.podcast_id, c.category, rc.ordering_number
FROM categories c
JOIN RankedCategories rc
ON rc.category = c.category
WHERE c.podcast_id IN (
    SELECT DISTINCT r.podcast_id FROM FilteredReviews r
)
"""
df = pd.read_sql_query(query, con)
df_fixed_categories = df.loc[df.groupby("podcast_id")
["ordering_number"].idxmin()]
df_to_plot = df_fixed_categories["category"].value_counts()
df_to_plot_filtered = df_to_plot[df_to_plot >= 300]
plot_counts_series(df_to_plot_filtered)
```



- As expected, all niche categories are now much smaller.

Merging Reworked Categories With Reviews

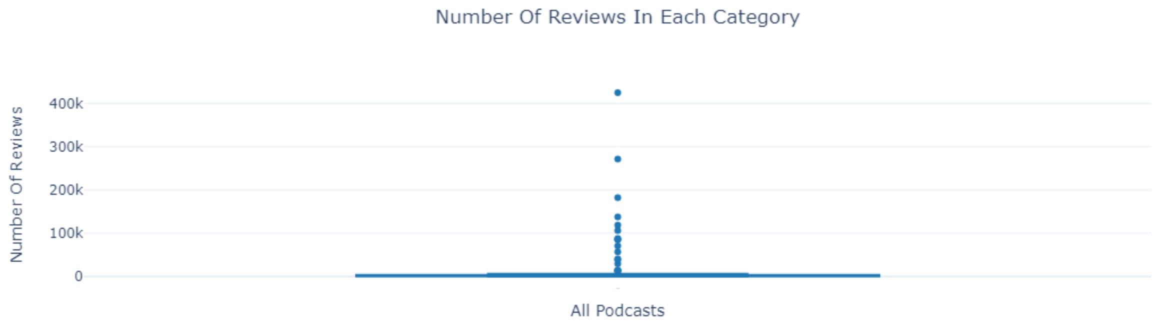
```
keep_from_fixed_categories = ["podcast_id", "category"]
keep_from_all_reviews = ["podcast_id", "rating", "author_id",
"created_at"]
df = pd.merge(
    df_fixed_categories[keep_from_fixed_categories],
    all_reviews[keep_from_all_reviews],
    on="podcast_id",
)
df.head()

                         podcast_id category  rating        author_id
\
0  a000aa69852b276565c4f5eb9cdd999b     arts       5  F43E015ADAF7828

1  a000aa69852b276565c4f5eb9cdd999b     arts       5  D3988F2DCD317FB

2  a000aa69852b276565c4f5eb9cdd999b     arts       5  923E47C12ACD3D5

3  a000aa69852b276565c4f5eb9cdd999b     arts       5  5A71004E5146D3C

4  a000aa69852b276565c4f5eb9cdd999b     arts       5  F0B119E5348C41C


                  created_at
0  2018-05-12T09:31:51-07:00
1  2018-04-06T04:14:20-07:00
2  2018-05-03T19:47:17-07:00
3  2018-02-06T15:39:23-07:00
4  2018-01-31T21:09:40-07:00
```

Outliers

We have previously looked into the number of reviews per podcast, but now there is a new metric - the number of reviews per category. If this number for a category is low, the insights could be unreliable and possibly wrong.

Let's look into a traditional method for detecting these outliers using IQR:

```
plot_box(df)
```



- The box plot detects outliers only for the upper range of the IQR, and individual points with a low number of reviews are not considered outliers. From the box plot we can see that the lowest count of reviews for a category is 6. There are multiple categories that have a really low number of reviews.

- This means that I have to set a standard for droping categories based on my own opinion when there a too few reviews.
- An option would be setting a standard for a review count of 1,200. This would mean an approximately one review every other day from when the iTunes platform started gaining traction in 2017 up to the present day, spanning six years.

- While this is a low frequency and identifying categories with close to this number of reviews would require further validation regarding when these reviews were acquired, it serves as a decent starting point.

```
df_below_1200 = df["category"].value_counts()[lambda x: x < 1200]
filtered_categories = df_below_1200.index
df = df[~df["category"].isin(filtered_categories)]
len(df_below_1200)
```

```
54
```

- Filtering categories to have more than 1,200 total reviews drops 54 categories, leaving 56 categories.

Takeaways

The following issues with the data were identified, and steps taken to address them:

- There are more reviewed podcasts than there are podcasts associated with a category. This additional information will not be used.
- To study more serious efforts, all podcasts with fewer than three reviews have been dropped.
- To account for review manipulation, accounts that have posted more than three reviews with identical titles have been removed.
- Some podcasts have multiple category entries due to category changes over time. A single category was determined by selecting the overall largest category from the data. This approach provides a broader overview, while selecting by the smaller category would allow to consider more niche segments.
- Multivariate analysis in the next section should be done on both largest and smallest selected categories. However, for this project, I'll work only on the largest categories.

After the cleaning process, the analysis will be based on 47,467 podcasts.

Insights

- Apple Podcasts are on decline and shouldn't be used as the primary platform for distributing podcasts.

Data Analysis

The main objectives for this part of the notebook are:

- Investigate the relationships between features and the target by performing multivariate analysis.
- Extract insights to identify which podcast category could have the highest chance of success.

To understand market situation, information from different feature relations will have to be combined and interpreted. My initial plans are:

1) Look at ratings for each category. Maybe there is something useful. 2) Find categories with the least decrease in the number of reviews - most important factor, since the platform is declining. 3) Discover if there are categories where reviewers engage with content from various different podcasts within the same genre to see if they would be open to new content. 4) Compare the number of podcasts to number of reviews for different categories to see if there is a higher demand for comparatively smaller supply.

Relationship Between Categories And Ratings

In this section, we explore how podcast categories relate to ordinal ratings. Ordinal data can be treated as numeric or categorical, so I'll apply both chi-square and Kruskal-Wallis tests. By using multiple tests, my aim is to validate the findings. Using statistical tests helps understand whether there are differences in ratings across categories and if they are statistically meaningful.

After conducting these tests, I'll will visualize the relationship as categorical-categorical using proportional bar plots, providing a way to interpret the results.

Chi-square Test

The chi-square test is a statistical test used to determine if there is a significant association between two categorical variables. It compares the observed frequencies with the frequencies that would be expected, if there was no association. If the calculated chi-square statistic exceeds a critical value from the table based on degrees of freedom and a chosen significance level, which for this project will be 0.05, it suggests a relationship between variables.

Additionally, we calculate a p-value to determine how significant this relationship is. If the p-value is greater than a chosen standard (in this case, 0.05), we reject the null hypothesis in favor of an alternative hypothesis, indicating a significant association between the variables. These two hypotheses are the same for all statistical tests.

```
alpha = 0.05
contingency_table = pd.crosstab(df["category"], df["rating"])
chi, p_value, degrees_of_freedom, expected_freq =
chi2_contingency(contingency_table)
critical_value = chi2.ppf(1 - alpha, degrees_of_freedom)
print(f"Chi-square statistic: {round(chi, 2)}")
print(f"p-value: {p_value}")
print(f"Critical Value: {round(critical_value, 2)}")
```

```
Chi-square statistic: 63779.67
p-value: 0.0
Critical Value: 246.97
```

- A computed chi-square value is much larger than the critical value, indicating an association between two variables.
- The p-value is very close to zero, which is below the common significance level of 0.05. This low p-value suggests strong evidence against the null hypothesis, meaning that this association is statistically significant.

Kruskal-Wallis Test

The Kruskal-Wallis test is a statistical test used to compare the medians of three or more independent groups. The H-statistic in the Kruskal-Wallis test measures the variability in the medians across all the different groups or categories. A larger H-statistic indicates greater variability or differences in the medians of the groups.

I chose this test because, as we've seen from the ratings distribution, our data is heavily skewed, and using the median instead of the mean is likely a better option.

```
grouped_data = [group["rating"] for name, group in
df.groupby("category")]
h_stat, p_value = kruskal(*grouped_data)
print("H-statistic:", round(h_stat, 2))
print("p-value:", p_value)
```

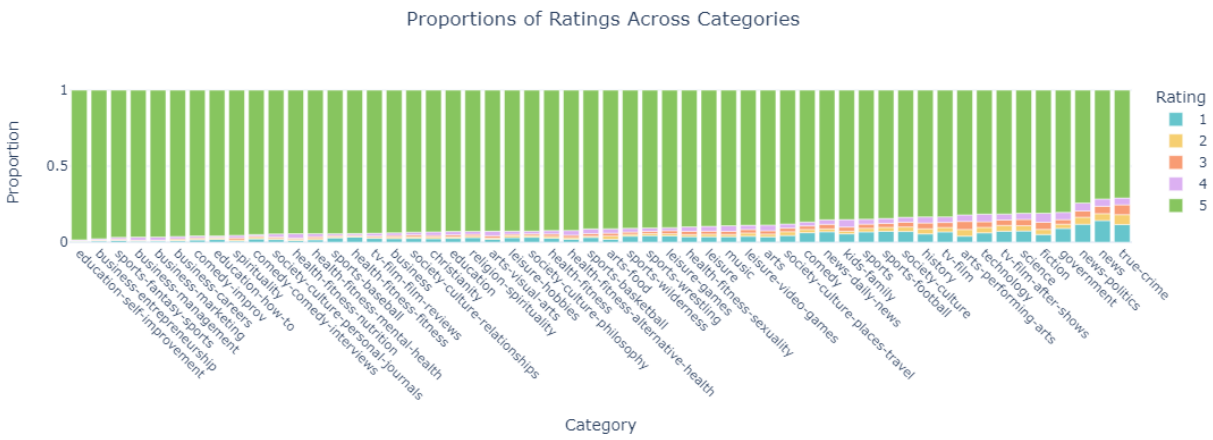
```
H-statistic: 58241.25
p-value: 0.0
```

- The H-statistic value is large, indicating considerable variability in ratings across the categories. The p-value is extremely small (0.0). This low p-value provides strong evidence against the null hypothesis, suggesting that there are significant differences in ratings among the different categories.

Visualization

Since the vast majority of reviews are "5", I consider anything other than "5" to be a negative review.

```
rating_proportions = pd.crosstab(df["category"], df["rating"],
normalize="index")
sorted_proportions = rating_proportions.sort_values(by=5,
ascending=False).reset_index()
plot_ratings_categories(sorted_proportions)
```



- Genres with a large number of podcasts have a more negative reviews then genres with fewer podcasts. This trend is likely accidental due to a small sample size.
- Podcasts covering controversial topics about politics have a lot of negative reviews.
- Another category with many negative reviews is true-crime, though the reason for this is unclear to me.
- From the perspective of creating a successful new podcast, it may be good to avoid these subjects, because they are more risky.

Let's check if my idea about niche podcasts having higher ratings solely due to smaller sample size is correct. I will calculate 95% confidence intervals for the proportions of ratings with a value of 5 for the five most highly rated podcasts. For comparison, I'll do the same for the five podcasts with the worst ratings as well.

```
first_5 = sorted_proportions["category"][:5].tolist()
last_5 = sorted_proportions["category"][-5:].tolist()
names = first_5 + last_5
for name in names:
    category_df = df[df["category"] == name]
    total_count = len(category_df)
    num_ratings_5 = len(category_df[category_df["rating"] == 5])
    proportion_5 = num_ratings_5 / total_count
    lower_bound, upper_bound = proportion_confint(
        num_ratings_5, total_count, alpha=0.05
    )
    if name == 'fiction':
        print('-------------')
    print(f"{name}: ({round(lower_bound, 3)}, {round(upper_bound,
3)})")
```

```
education-self-improvement: (0.979, 0.99)
business-entrepreneurship: (0.974, 0.981)
sports-fantasy-sports: (0.959, 0.977)
business-management: (0.957, 0.977)
business-marketing: (0.96, 0.972)
-------------
```

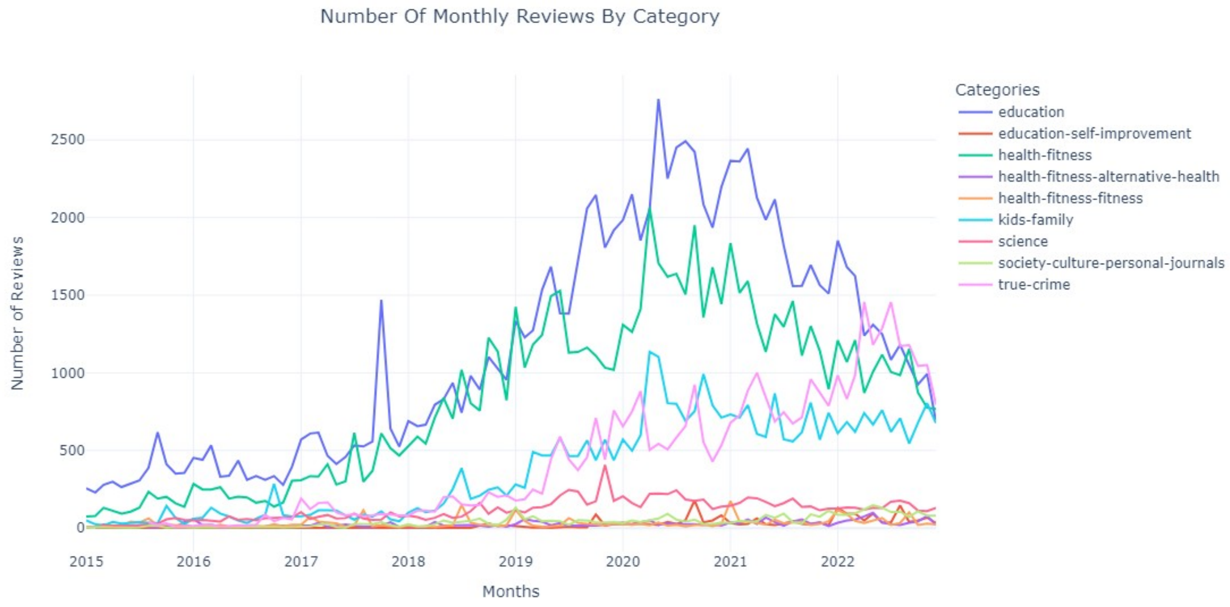
```
fiction: (0.803, 0.816)
government: (0.791, 0.817)
news-politics: (0.722, 0.762)
news: (0.713, 0.718)
true-crime: (0.705, 0.714)
```

- My hypothesis was completely incorrect. The narrow confidence intervals for all podcasts indicate a high level of precision, giving confidence in the calculated proportions.
- This means that smaller niche categories are, in fact, slightly more appreciated (rated) than broader categories.

Relationship Between Category And Number Of Reviews Over Time

I have already looked at the plot showing the change in the number of reviews from 2017 and selected the most interesting cases. These include all cases (of which there are few) where the number of reviews is not falling, and two additional lines for "education" and "health-fitness" to comment on the findings. These two lines also represent the decline that iTunes is experiencing.

```
warnings.filterwarnings("ignore", category=UserWarning, module=".*")
df_plot = df[["category", "created_at"]].copy()
categories_to_keep = [
    "education",
    "kids-family",
    "true-crime",
    "education-self-improvement",
    "health-fitness",
    "health-fitness-alternative-health",
    "health-fitness-fitness",
    "science",
    "society-culture-personal-journals",
]
df_plot = df_plot[df_plot["category"].isin(categories_to_keep)]
df_plot["created_at"] = pd.to_datetime(df_plot["created_at"],
format="mixed")
df_plot = df_plot[df_plot["created_at"].dt.year >= 2015]
df_plot = df_plot[df_plot["created_at"].dt.year < 2023]
df_plot["year_month"] =
df_plot["created_at"].dt.to_period("M").astype(str)
monthly_reviews = (
    df_plot.groupby(["category",
"year_month"]).size().reset_index(name="num_reviews")
)
plot_reviews_month(monthly_reviews)
```



Bad options:

- Some podcasts, like "education-self-improvement", "health-fitness", "health-fitness-alternative-health", "health-fitness-fitness", and "society-culture-personal-journals" are unlikely to be good choices for new podcasts. The trend for the larger genres that encompass them is strongly trending downwards, and their comparable stability is likely accidental due to a small sample size. For "society-culture", the trend is not plotted because there are too many reviews, making the plot uncomfortable to view.

A possibility:

- An option could be science. This category has a slightly higher count of reviews than the ones I rejected previously, and there's no obvious reason why this isn't a good category. However, the overall count of reviews may be too low, indicating that people may not be particularly interested in this topic.

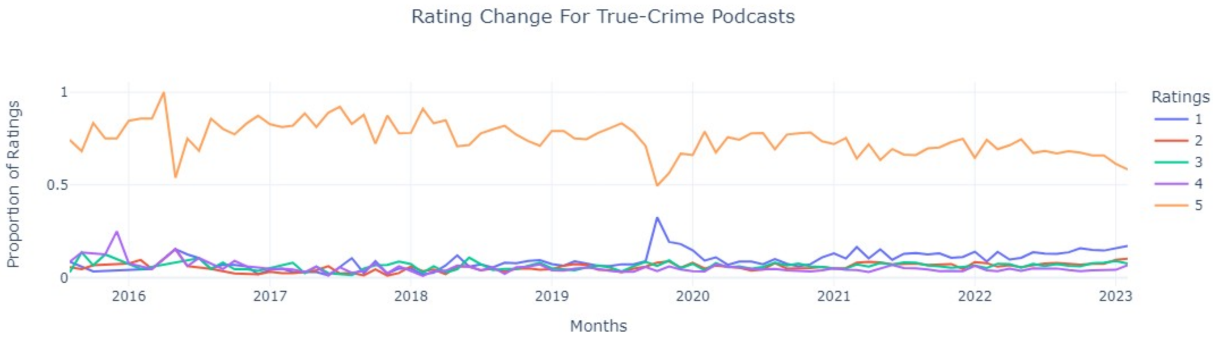
Good options:

- There are only two niches that have a relatively high number of reviews, and their number is seemingly stable: "kids-family" and "true-crime". These are the types of podcasts where listenership is not losing interest, and I would recommend them for a new podcast.

What Is Happening With True-Crime?

True crime has come up three times during this analysis: once when we removed the outlier, next when we looked at the lowest-rated types, and when we found it again among the podcasts that are not yet losing listeners. Let's look to see if the negative reviews are a recent event.

```
df_true_crime = df[df["category"] == "true-crime"]
df_true_crime = df_true_crime[["rating", "created_at"]].copy()
df_true_crime["created_at"] =
pd.to_datetime(df_true_crime["created_at"])
df_true_crime["year_month"] =
df_true_crime["created_at"].dt.to_period("M").astype(str)
df_rating_counts = (
    df_true_crime.groupby(["year_month",
"rating"]).size().reset_index(name="count")
)
df_total_counts = (

df_true_crime.groupby("year_month").size().reset_index(name="total_cou
nt")
)
df_merged = pd.merge(df_rating_counts, df_total_counts,
on="year_month")
df_merged["proportion"] = df_merged["count"] /
df_merged["total_count"]
plot_true_crime_month(df_merged)
```



- Review ratings are steadily declining. The reasons for this are unclear. To find out why this is happening, I would have to read the content of the reviews.
- Review entries for this genre are fairly recent, with the most recent appearing in 2015, while most of the podcasts have much older data.

Is This Podcast A Better Opportunity For A New Podcast Than Kids-Family?

Answer this would require more research, but I think it is not. On one hand, a growing discontent among fans could mean a good opportunity to fix issues (maybe replicate what was done right before). On the other hand, fixing content quality is more ambitious and risky.

In a subsection later on, I will try to estimate competition and saturation for all genres, including this on.

Relationship Between Podcast Category And User Engagement Across A Category

In this chapter, I will investigate the relationship between podcast categories and user engagement by analyzing the frequency of repeated user IDs within each category. I explore whether users tend to engage with multiple podcasts within the same category or they are loyal to their favourite podcast.

Kruskal-Wallis Test

Let's use a Kruskal-Wallis test to see if there are significant differences. The interpretation of this test is provided in a sub-section above.

Since this essentially creates a new feature, I will display this new data for clarity.

```
df_users = df[["podcast_id", "category", "author_id"]]
grouped = (
    df_users.groupby(["category", "author_id"])
["podcast_id"].nunique().reset_index()
)
grouped = grouped.rename(columns={'podcast_id': 'number_of_reviews'})
grouped.head()
```

```
  category        author_id  number_of_reviews
0     arts  000260281A737A3                  1
1     arts  0003FE5763C5790                  1
2     arts  00043DCB0A64EA3                  1
3     arts  00044218B20DAED                  1
4     arts  00047F7D109332F                  1
```

For example: We can see that the user with the ID '000260281A737A3' left a review on a single podcast in the arts category. The relationship we are interested in is between the "category" and "number_of_reviews".

```
grouped_data = [group["number_of_reviews"] for name, group in
grouped.groupby("category")]
h_stat, p_value = kruskal(*grouped_data)
print(f"H-statistic: {round(h_stat, 2)}")
print(f"p-value: {p_value}")
```

```
H-statistic: 15628.15
p-value: 0.0
```

- H-statistic and a p-value indicate that there are statistically significant differences in the medians of the groups.

Descriptive Statistics

```
descriptive_stat = (
    grouped.groupby("category")["number_of_reviews"]
    .describe()
    .sort_values(by="mean", ascending=False)
)
min_mean = round(descriptive_stat["mean"].min(), 2)
max_mean = round(descriptive_stat["mean"].max(), 2)
print(f"The range of means is from: {min_mean} to {max_mean}")
print("All unique medians are:", descriptive_stat["50%"].unique()[0])
```

```
The range of means is from: 1.0 to 1.19
All unique medians are: 1.0
```

- Between all categories medians are all the same, and the mean differs at most by 0.19.
- People generally don't leave comments on more than one podcast within a single category, and this holds true for all types of podcasts.
- This result contradicts with the statistical test, which suggests differences in medians across different podcast categories. This is probably because the test detects small changes that are not practically meaningful.

Relationship Between Podcasts And Reviews

Let's take a direct approach to examining supply, represented by the number of podcasts, and demand, indicated by the number of reviews. We will be looking for outliers—points that deviate significantly from the general trend. Specifically, we're interested in identifying points with high review counts but low podcast numbers. This analysis allows to point out categories with exceptional listener engagement relative to other categories.

Correlation Test

There is probably a strong connection between these two variables, but I'll test it with a statistical test. A correlation test assesses the strength and direction of the relationship between two continuous variables. It measures how changes in one variable correspond to changes in another. A correlation coefficient close to 1 signifies a strong positive correlation, close to -1 indicates a strong negative correlation, and around 0 suggests no correlation. Since I'm keeping the outliers by high review counts, I'll use Spearman's correlation, since it handles them better.

```
df_summary = (
    df.groupby("category")
    .agg(num_podcasts=("podcast_id", "nunique"),
total_reviews=("rating", "count"))
    .reset_index()
)
spearman_corr, spearman_p_value = spearmanr(
    df_summary["num_podcasts"], df_summary["total_reviews"]
)
print(
    f"Spearman correlation coefficient: {round(spearman_corr, 2)}, p-value: {round(spearman_p_value, 3)}"
)
```

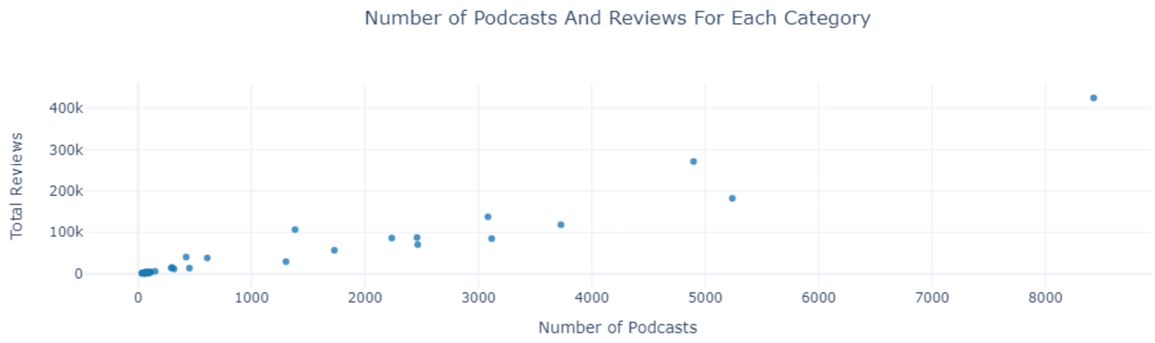
```
Spearman correlation coefficient: 0.88, p-value: 0.0
```

- The Spearman correlation coefficient indicates a strong positive relationship. The p-value of 0.0 (below 0.05) suggests we can reject null hypothesis and that this correlation is statistically significant.

Visualization

To declutter the plot, I won't display the names of the categories, but they'll be visible when hovering over the points.

```
plot_podcasts_reviews(df_summary)
```



- Comedy podcasts have slightly more reviews than could be expected from the overall trend, but no category really stands out too much for me.
- Both "kids-family" and "true-crime" are smaller categories, with about 40 thousand reviews and 300-400 podcasts each.

All Insights

- Reviews over time indicate that Apple Podcasts is losing a significant number of listeners and should not be used as the primary platform for distributing podcasts.
- Review changes over time reveal that the "kids-family" and "true-crime" categories are the most promising for new podcasts, while other categories are declining in popularity.
- Ratings indicate that podcasts in politics and true-crime are polarizing and, therefore, riskier.
- If it was possible to fix the issue of declining ratings for "true-crime", this may be the best category for a new podcast.
- Ratings also indicate that smaller niche podcasts receive slightly more positive appreciation, although the difference is not large.
- Individual reviews: Within each genre, listeners have a favorite podcast to which they are loyal.

Recommendation: Create a podcast for kids.

Conclusions

In addition to analyzing the largest categories, I should do a separate analysis on the data after resolving the duplicated categories by selecting the smaller option. There may be insights hidden at a higher level. It's possible that there are no differences between the smaller and larger encompassing categories, but this would need verification.

Further avenues to explore:

- Investigate why there are so many negative revies for "true-crime" podcasts by identifying common themes or issues mentioned by listeners. Determine whether these reviews come from new listeners or long-time followers. If the issues are fixable, this category could potentially be the best choice for a new podcast.
- Identify exceptional performers within the "kids-family" and "true-crime" categories. This information could help understand the attributes contributing to the success of these podcasts.

Validate assumptions for each test:

- For the chi-square test, the expected frequency count for each cell should be at least 5. I did not check this because I know that each group has more than 1200 counts, and most likely this assumption is met.
- Check if the assumption of "homogeneity of variances" is met for the Kruskal-Wallis test by comparing the spread of ranks within each group using box plots or statistical test. If the boxes (interquartile ranges) are similar in size across all groups, it would suggest that the variability within groups is roughly similar and this condition is met.

Additional ideas to improve the project:

- For the relationship between podcast category and the number of reviews left by individuals, I focused solely on central tendency, leaving the variability unclear. Calculating the percentage of individuals who left only a single review within a specific category would be a good information.
- People are loyal to their podcast in one category. Would this appy to cross-category listenership? I would guess that people listen to podcasts from various categories.
- Some variables can be renamed to follow PEP8 recommendations (df, df_plot).
- I could have reaserched domain knowledge thresholds when to drop small categories.
- It would be interesting to see if there is any association between having multiple categories in the past and the number of reviews (popularity).
- Maybe drop all older data.