ETL Project

Apple App Store vs Google Play Store

Document Revision History			
Revision	**Date**	**Rev. by:**	**Remarks**
1.0	05/06/2019	Ellise Carpenter, Rohith Bhattaram, Jonas Haskins, Kavya Shabnavees	

Table of Contents

1 Project Description

When it comes to mobile ecosystems, there are two giants locked in a battle, not only for revenue, but also for the hearts and minds of developers and consumers alike. The purpose of our project is to extract data sets of Google's Play Store and Apple's App Store for year 2017 from Kaggle data source, transform by normalizing both the datasets and load the datasets to a database. Ultimately, the transformed data set will allow for a better understanding of the relative variances between vendors.

2 Extract

We used 2 datasets from Kaggle. Both data sets were the most recent available and were originally obtained from via web scraping. The specific datasets used for the project are as follows:

Kaggle - Google Play Store Stats
Kaggle - Apple App Store Stats

Conversion of CSVs to Pandas Dataframe:

```
csv_Apple = "AppleStore.csv"
csv_GPlay = "googleplaystore.csv"
appleDF = pd.read_csv(csv_Apple)
googleDF = pd.read_csv(csv_GPlay)
```

```
# Check files imported correctly
# googleDF.head()
# appleDF.head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0

3 Transform

File Clean-up with selected columns:

Drop columns for Apple store

```
# Drop columns for apple store
cleaned_apple = appleDF[["track_name","price","user_rating","rating_count_tot","prime_genre"]]
cleaned_apple_df = cleaned_apple.rename(columns={"track_name":"app_name","price":"price","user_rating":"us
er_rating","rating_count_tot":"number_of_reviews","prime_genre":"genre"})
# cleaned_apple_df.head()
```

Drop columns for Google Store

```
# Drop columns for google store
cleaned_google = googleDF[["App","Category","Rating","Reviews","Price"]]
cleaned_google_df = cleaned_google.rename(columns = {"App":"app_name","Price":"price","Rating":"user_ratin
g","Reviews":"num_of_reviews","Category":"genre"})
# cleaned_google_df.head()
```

Removing special characters

```
def clean_data(strs):
   listOfStrings=[' ','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','
1,2,3,4,5,6,7,8,9,0]
   name=""
   string_=""
   word = strs
   #print(word)
   flag=bool(re.match('^[a-zA-Z0-9]+$', word))
   #print(flag)
   if flag :
       name = word
       #print(f"printing{name}")
       return name
   else :

       #print("control reached major else block")

       for letter in word :

           #print(f"letter is :{letter}")
           if letter in listOfStrings :
               #print("control enterd to if list of strings")
               string_=string_+letter
               #print(f"updated name value at this point:{string_}")
           else :
               #print("control enterd to else list of strings")
               letter=""
               string_=string_+letter
               #print(f"updated name value at this point:{string_}")
   name=string_
   return name
```

```
# Remove special characters from DB
for index, record in cleaned_google_df.iterrows():
    y = record['app_name']
    if len(y) > 3:
        #print(y)
        x = re.sub(r'[~|!|@|#|$|%|^|&|*|(|)|"|;|<|>|/|?]',r'',y)
        cleaned_google_df.loc[index, 'app_name'] = clean_data(x)

    else:
        cleaned_google_df.loc[index, 'app_name'] = "NA"
```

```
# Remove special characters from DB
for index, record in cleaned_apple_df.iterrows():
    y = record['app_name']
    if len(y) > 3:
        #print(y)
        x = re.sub(r'[~|!|@|#|$|%|^|&|*|(|)|"|;|<|>|/|?]',r'',y)
        cleaned_apple_df.loc[index, 'app_name'] = clean_data(x)

    else:
        cleaned_apple_df.loc[index, 'app_name'] = "NA"
```

```
for index, record in cleaned_google_df.iterrows():
       y = record['price']
       if len(y) > 2:
           #print(y)
           x = re.sub(r'[~|!|@|#|$|%|^|&|*|(|)|"|;|<|>|/|?]',r'',y)
           cleaned_google_df.loc[index, 'price'] = x

       else :
            cleaned_google_df.loc[index, 'price'] = 0
```

```
for index, record in cleaned_apple_df.iterrows():
       y = str(record['price'])
       if len(y) > 2:
           #print(y)
           x = re.sub(r'[~|!|@|#|$|%|^|&|*|(|)|"|;|<|>|/|?]',r'',y)
           cleaned_apple_df.loc[index, 'price'] = x

       else :
            cleaned_apple_df.loc[index, 'price'] = 0
```

Normalizing of Google Categories:

```
counter = 0
for index, row in cleaned_google_df.iterrows():
    try:
        location1 = row["genre"].upper()
           #print(row["Category"])
           #print(f"counter value is : {counter}")
        if  location1 == "GAME" :
                    location1 = "GAMES"
        elif location1 == "AUTO_AND_VEHICLES":
                    location1 = "REFERENCE"
        elif location1 == "FAMILY" :
                    location1 = "REFERENCE"
        elif location1 == "PARENTING" :
                location1 = "REFERENCE"
        elif location1 == "TOOLS" :
               location1 = "UTILITIES"
        elif location1 == "PERSONALIZATION" :
                location1 = "UTILITIES"


        elif location1 == "SOCIAL":
                location1 = "SOCIAL_NETWORKING"
        elif location1 == "DATING" :
                location1 = "SOCIAL_NETWORKING"
        elif location1 == "EVENTS" :
                location1 = "SOCIAL_NETWORKING"
        elif location1 == "PHOTOGRAPHY" :
                location1 = "PHOTOS_AND_VIDEOS"
        elif location1 == "VIDEO_PLAYERS" :
                location1 = "PHOTOS_AND_VIDEOS"
        elif location1 == "MAPS_AND_NAVIGATION" :
                location1 = "NAVIGATION"

        elif location1 == "HOUSE_AND_HOME":
                location1 = "CATALOG"
        elif location1 == "1.9" :
                location1 = "CATALOG"
        elif location1 == "BEAUTY" :
                location1 = "CATALOG"
        elif location1 == "ART_AND_DESIGN" :
                location1 = "CATALOG"
        elif location1 == "COMICS" :
                location1 = "CATALOG"
        else :
              location1 = location1

               #print(f"value to be inserted : {location1}")
        cleaned_google_df.loc[index, 'genre'] = location1
        counter = counter+1

        if counter > 100000 :
            break
```

Normalizing Apple Categories:

```
counter = 0
for index, row in cleaned_apple_df.iterrows():

      try:

          location1 = row["genre"].upper()
          #print(row["prime_genre"])
          #print(f"counter value is : {counter}")

          if  location1 == "HEALTH & FITNESS" :
              location1 = "HEALTH_AND_FITNESS"
          elif location1 == "PHOTO & VIDEO":
              location1 = "PHOTOS_AND_VIDEOS"
          elif location1 == "FOOD & DRINK" :
              location1 = "FOOD_AND_DRINK"
          elif location1 == "BOOK" :
              location1 = "BOOKS"
          else :
              location1 = location1

          #print(f"value to be inserted : {location1}")
          cleaned_apple_df.loc[index, 'genre'] = location1
          counter = counter+1

          if counter > 100000 :
               break


      except (KeyError, IndexError):
          print("Missing field/some exception - so skipping those")
print("Task Completed....")
```

4 Load

The last step was loading our final data frames into Database. We created a MYSQL database and respective tables to match the columns from the final Panda's Data Frames and then connected to the database via SQL Alchemy.

Use pandas to load dataframe to MySQL

```
# def load():
#     try:
#         cleaned_google_df.to_sql(name='google_stats', con=engine, if_exists='append', index=False)
#         print("Data loading completed")
#     except:
#         return ("oops...missed one field")

# print(load())

cleaned_google_df.to_sql(name='google_stats', con=engine, if_exists='append', index=False)
```

```
cleaned_apple_df.to_sql(name='apple_stats', con=engine, if_exists='append', index=False)
```

```
# Confirm data was added
pd.read_sql_query('select * from apple_stats', con=engine).head()
```

	app_name	price	user_rating	number_of_reviews	genre
0	PACMAN Premium	3.99	4.0	21292	GAMES
1	Evernote stay organized	0.00	4.0	161065	PRODUCTIVITY
2	WeatherBug Local Weather Radar Maps Alerts	0.00	3.5	188583	WEATHER
3	eBay Best App to Buy Sell Save Online Shopping	0.00	4.0	262241	SHOPPING
4	Bible	0.00	4.5	985920	REFERENCE

5 Preliminary Analysis

Genre Rating (Avg)

Apple

```
1 • use etl_project;
2 • select avg(user_rating)
3   ,genre
4   from apple_stats
5   group by genre
6   order by avg(user_rating) DESC;
```

Result Grid | Filter Rows: | Export:

avg(user_rating)	genre
4.00562	PRODUCTIVITY
3.97826	MUSIC
3.80086	PHOTOS_AND_VIDEOS
3.74561	BUSINESS
3.70000	HEALTH_AND_FITNESS
3.68501	GAMES
3.59722	WEATHER
3.54098	SHOPPING
3.45313	REFERENCE
3.37654	TRAVEL
3.37638	EDUCATION
3.36957	MEDICAL
3.27823	UTILITIES
3.24673	ENTERTAINMENT
3.18254	FOOD_AND_DRINK
2.98503	SOCIAL NETWORKING
2.98246	SPORTS
2.98000	NEWS
2.80556	LIFESTYLE

esult 7 X

Google

```
1 • use etl_project;
2 • select avg(user_rating)
3   ,genre
4   from google_stats
5   group by genre
6   order by avg(user_rating) DESC;
```

esult Grid | Filter Rows: | Export:

avg(user_rating)	genre
4.32773	EDUCATION
4.13529	ENTERTAINMENT
4.11700	FINANCE
4.03971	GAMES
3.86709	WEATHER
3.81131	PHOTOS_AND_VIDEOS
3.76931	SHOPPING
3.75951	CATALOG
3.66430	REFERENCE
3.63588	NAVIGATION
3.59547	HEALTH_AND_FITNESS
3.53333	FOOD_AND_DRINK
3.48590	UTILITIES
3.47489	TRAVEL_AND_LOCAL
3.37292	SPORTS
3.36684	PRODUCTIVITY
3.36047	SOCIAL_NETWORKING
3.34952	COMMUNICATION
3.33902	LIFESTYLE

NAVIGATION

sult 6 X

Genre Price (Avg)

Apple

```
1 • use etl_project;
2 • select avg(price)
3   ,genre
4   from apple_stats
5   group by genre
6   order by avg(price);
```

Result Grid | Filter Rows: | Expor

avg(price)	genre
0.016311	SHOPPING
0.339880	SOCIAL NETWORKING
0.421154	FINANCE
0.517733	NEWS
0.799000	CATALOGS
0.885417	LIFESTYLE
0.889701	ENTERTAINMENT
0.953070	SPORTS
1.120370	TRAVEL
1.432923	GAMES
1.473295	PHOTOS_AND_VIDEOS
1.552381	FOOD_AND_DRINK
1.605417	WEATHER
1.647621	UTILITIES
1.790536	BOOKS
1.916444	HEALTH_AND_FITNESS
4.028234	EDUCATION
4.124783	NAVIGATION
4.330562	PRODUCTIVITY
4.835435	MUSIC

Google

```
2 • select avg(price)
3   ,genre
4   from google_stats
5   group by genre
6   order by avg(price);
```

esult Grid | Filter Rows: | Ex

avg(price)	genre
0.011786	LIBRARIES_AND_DEMO
0.015669	NEWS_AND_MAGAZINES
0.024170	CATALOG
0.027129	SHOPPING
0.076396	FOOD_AND_DRINK
0.078235	ENTERTAINMENT
0.150924	EDUCATION
0.205725	NAVIGATION
0.224216	HEALTH_AND_FITNESS
0.228082	TRAVEL_AND_LOCAL
0.263937	COMMUNICATION
0.291267	PHOTOS_AND_VIDEOS
0.297085	GAMES
0.307692	SPORTS
0.324313	SOCIAL_NETWORKING
0.348399	UTILITIES
0.410380	WEATHER
0.418353	BUSINESS
0.539505	BOOKS_AND_REFERENCE
0.670936	PRODUCTIVITY

Highest Rated Apps:

Apple

```
use etl_project;
select app_name
,number_of_reviews
,user_rating
,genre
from apple_stats
where number_of_reviews > 10000
order by user_rating DESC;
```

app_name	number_of_reviews	user_rating	genre
Head Soccer	481564	5.0	GAMES
Plants vs Zombies	426463	5.0	GAMES
Sniper D Assassin Sh...	386521	5.0	GAMES
Geometry Dash Lite	370370	5.0	GAMES
Infinity Blade	326482	5.0	GAMES
Geometry Dash	266440	5.0	GAMES
Dominos Pizza USA	258624	5.0	FOOD_AND_DRINK
CSR Racing	257100	5.0	GAMES
Pictoword Fun Pics G...	186089	5.0	GAMES
Plants vs Zombies HD	163598	5.0	GAMES
The Room	143908	5.0	GAMES
Iron Force	141634	5.0	GAMES
Sniper Shooter Gun S...	134080	5.0	GAMES
Flashlight	130450	5.0	UTILITIES
Pic Collage Picture E...	123433	5.0	PHOTOS_AND_VIDEOS
Zappos shop shoes ...	103655	5.0	SHOPPING
Credit Karma Free Cr...	101679	5.0	FINANCE
PewDiePies Tuber Si...	90851	5.0	GAMES

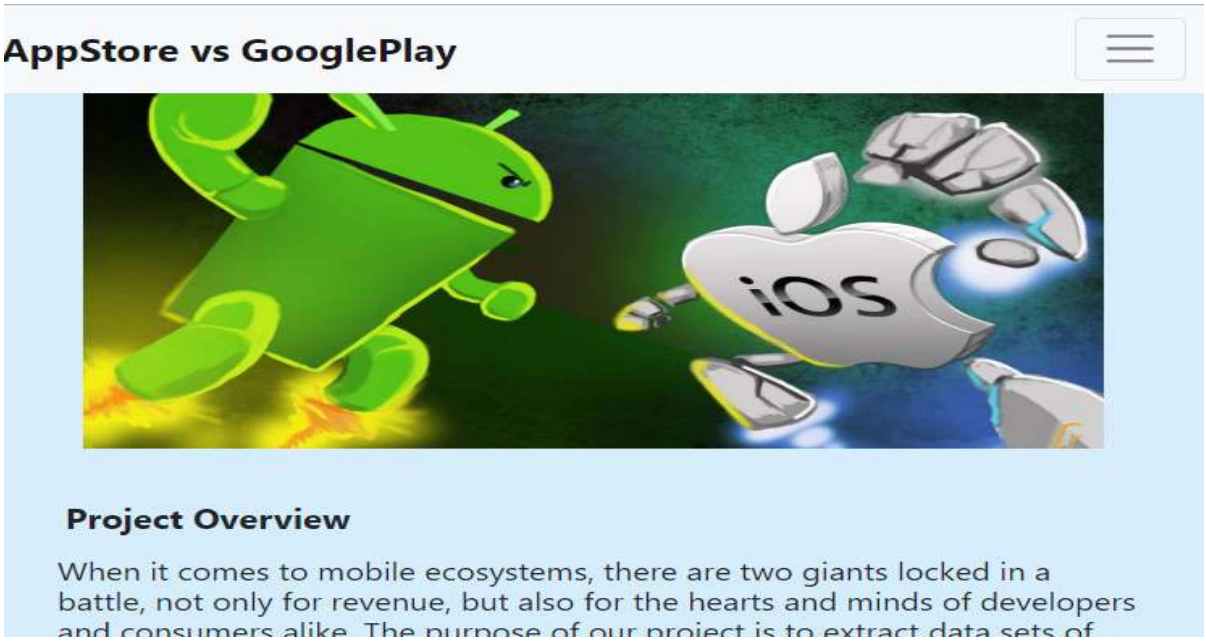
Google

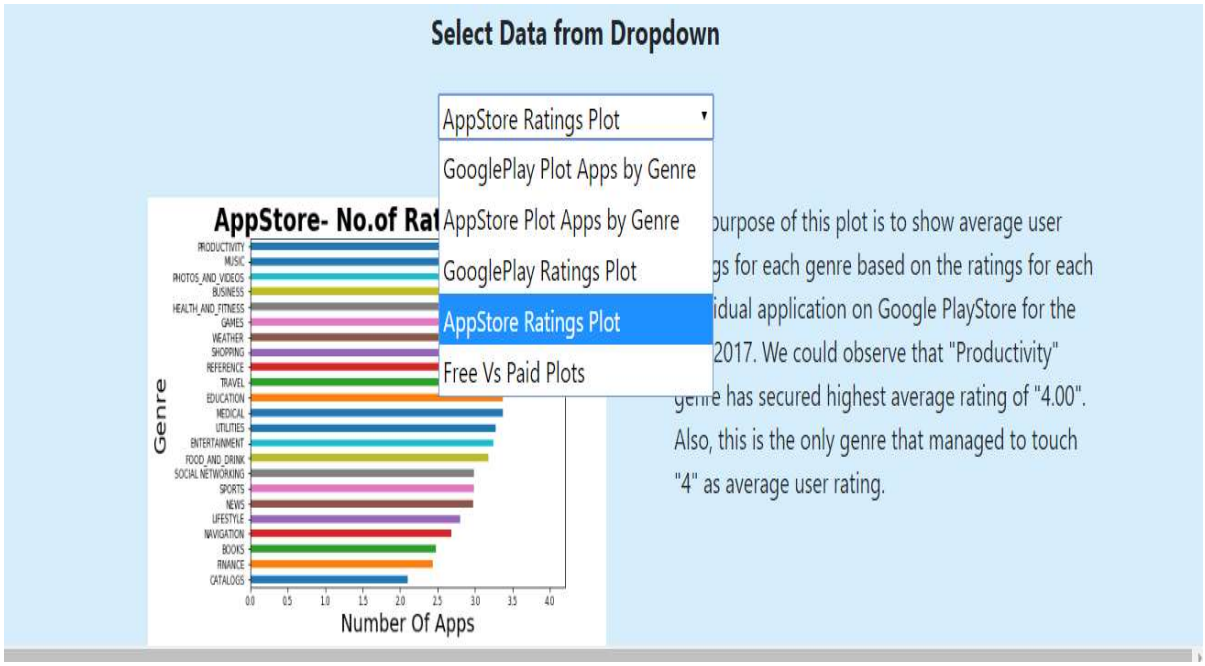
```
use etl_project;
select app_name
,num_of_reviews
,user_rating
,genre
from google_stats
where num_of_reviews > 10000
order by user_rating DESC;
```

app_name	num_of_reviews	user_rating	genre
Tickets SDA and Exam from the State Traffic S...	10479	4.9	REFERENCE
Tickets PDA Exam	197136	4.9	REFERENCE
ipsy Makeup Beauty and Tips	49790	4.9	CATALOG
Learn Japanese Korean Chinese Offline Free	133136	4.9	EDUCATION
Lose Belly Fat in Days Flat Stomach	38098	4.9	HEALTH_AND_FITNESS
Six Pack in Days Abs Workout	272337	4.9	HEALTH_AND_FITNESS
StrongLifts x Workout Gym Log Personal Trainer	66791	4.9	HEALTH_AND_FITNESS
Down Dog Great Yoga Anywhere	28945	4.9	HEALTH_AND_FITNESS
The Room Old Sins	21119	4.9	GAMES
Hungry Hearts Diner A Tale of StarCrossed Souls	46253	4.9	REFERENCE
Solitaire Decked Out Ad Free	37302	4.9	GAMES
PinPanda Color by Number Pixel Art Coloring Book	55723	4.9	REFERENCE
JW Library	922752	4.9	BOOKS_AND_REFERENCE
FC Porto	15883	4.9	SPORTS
Fines of the State Traffic Safety Inspectorate a...	116986	4.8	REFERENCE
Free Books Spirit Fanfiction and Stories	116307	4.8	BOOKS_AND_REFERENCE
ReadEra free ebook reader	47303	4.8	BOOKS_AND_REFERENCE

6 HTML Screenshots/Visualizations

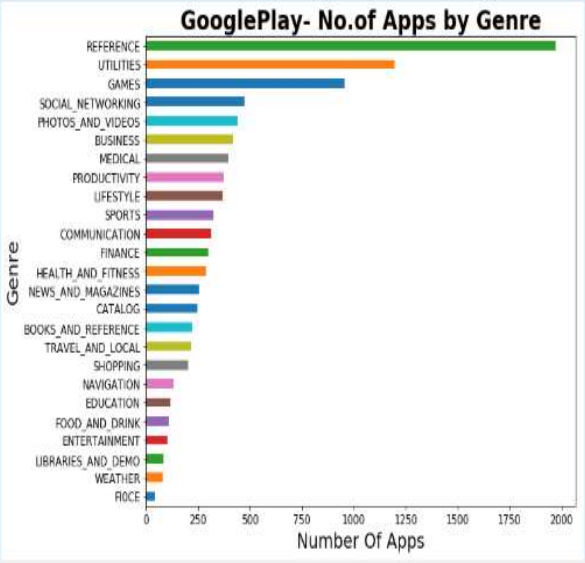
- **Page View:**





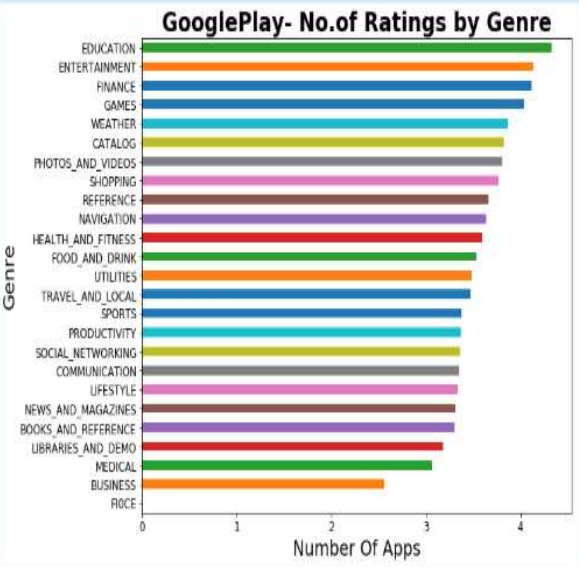
- **Google PlayStore:**

1. Number of Applications per Genre



The purpose of this bar chart is to showcase the distribution of 9000+ applications based on their genres from a dataset of GooglePlayStore for the year 2017.We could visualize that highest number of applications(~2000) falls under "Reference" genre and next highest applications count(~1250) is bagged by "Utilities" genre.

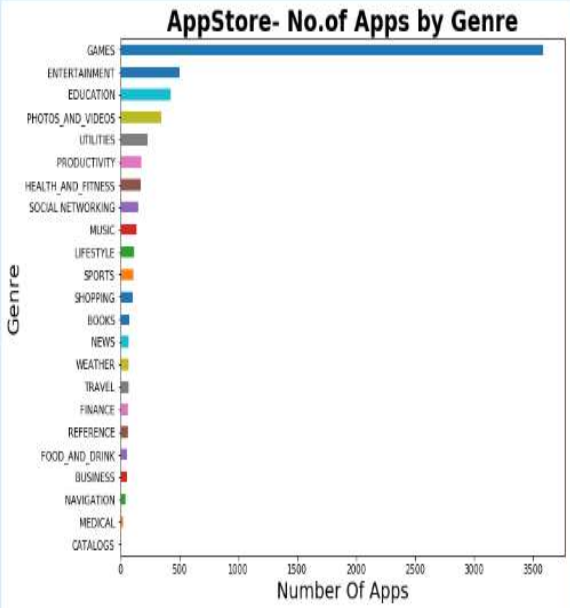
2. Average User Ratings per Genre



The purpose of this plot is to show average user ratings for each genre based on the ratings for each individual application on Google PlayStore for the year 2017. We could observe that "Education" genre has secured highest average rating of "4.32". All together, only four genres managed to maintain an average of "4+" user rating.

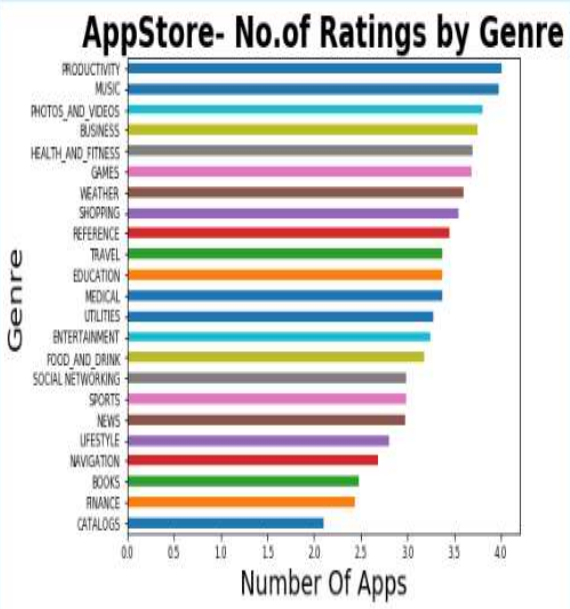
➢ Apple AppStore:

1. Number of Applications per Genre



The purpose of this bar chart is to showcase the distribution of 7500+ applications based on their genres from a dataset of Apple AppStore for the year 2017.We could visualize that highest number of applications(3500+) falls under "Games" genre and for the remaining genres, none of them recorded more than 500 applications each.

2. Average User Ratings per Genre



The purpose of this plot is to show average user ratings for each genre based on the ratings for each individual application on Google PlayStore for the year 2017. We could observe that "Productivity" genre has secured highest average rating of "4.00". Also, this is the only genre that managed to touch "4" as average user rating.