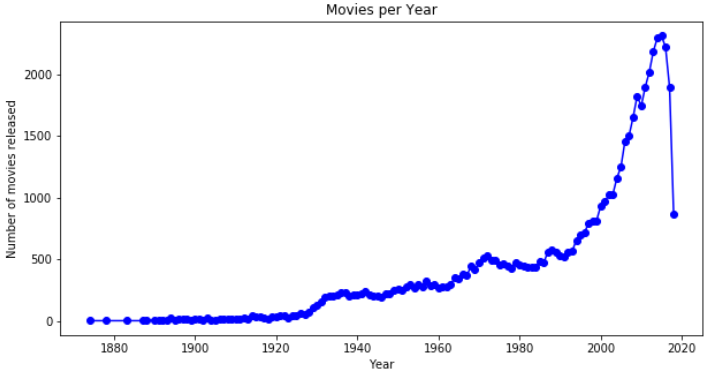
Northwestern

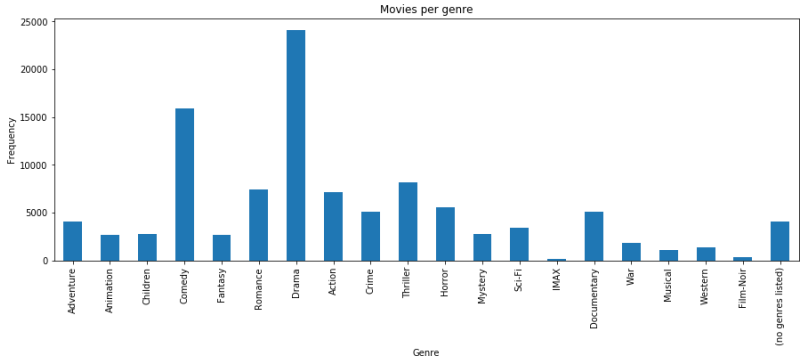
MSiA 423 Mid-project Review: Movie Recommender

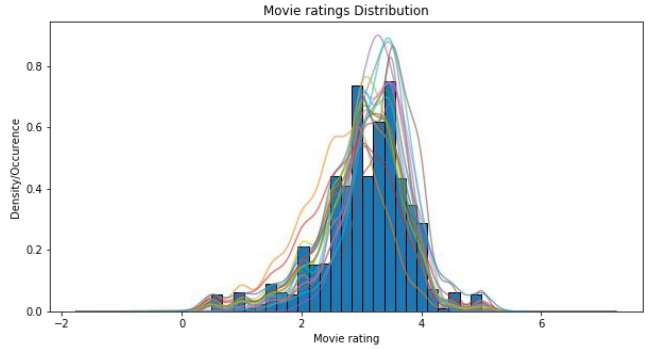
By: Sabarish Chockalingam

Highlights

- Interesting fact: Ratings only started coming in from about 1997 (when movie lens started).
- Below are visualization that helped with EDA and will aid when modelling.







Review progress

- Imported Data and perform EDA. (2 pt)
- Stored data set in S3. (1 pt)

Demo/analysis

- Visualization of notebook showing EDA.

Movie Recommender Development

```
In [1]:
import os
import pandas as pd
from matplotlib import pyplot as plt

In [92]:
ratingspath = os.path.join("..","..","data","ml-latest","ratings.csv")
ratings = pd.read_csv(ratingspath)

moviespath = os.path.join("..","..","data","ml-latest","movies.csv")
movies = pd.read_csv(moviespath)

In [8]:
ratings.head()
```

Out[8]:

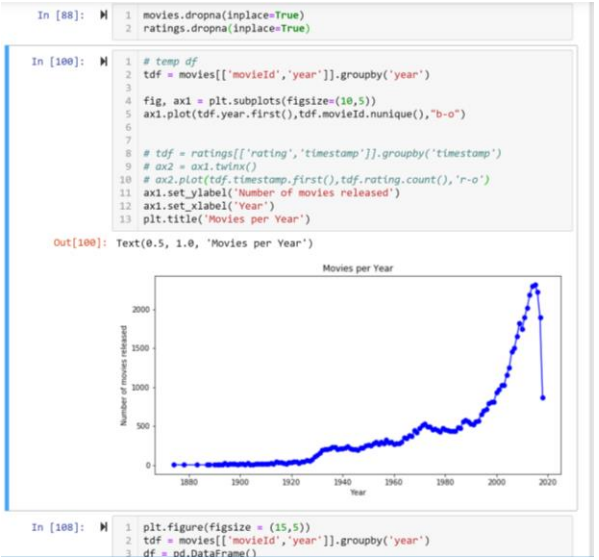
	userId	movieId	rating	timestamp
0	1	307	3.5	1256677221
1	1	481	3.5	1256677456
2	1	1091	1.5	1256677471
3	1	1257	4.5	1256677460
4	1	1449	4.5	1256677264

```
In [53]:
movies.head()
```

Out[53]:

	movieId	title	genres	year
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0
1	2	Jumanji (1995)	Adventure|Children|Fantasy	1995.0

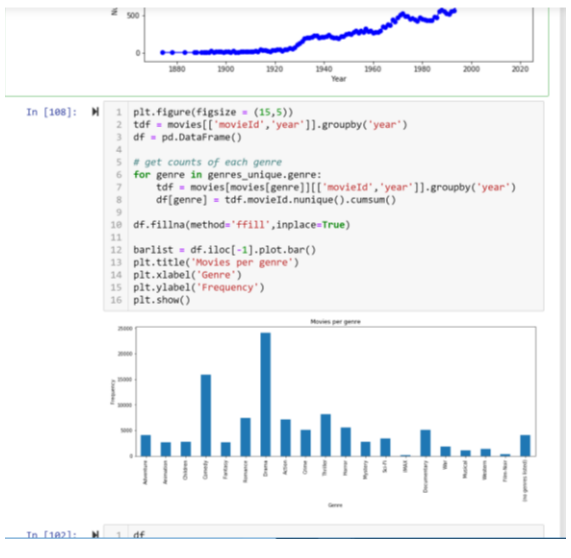
```
In [46]:
# extracting date
movies['year'] = movies.title.str.extract("\(((\d{4})\)", expand=True)

In [48]:
movies.year = pd.to_datetime(movies.year, format = "%Y")
# removing any NAs that may result in decimals
movies.year = movies.year.dt.year
# removing year at end of title
#movies.title = movies.title.str[:-7]

In [60]:
# all different types of genre
genres_unique = pd.DataFrame(movies.genres.str.split('|').tolist()).stac
genres_unique = pd.DataFrame(genres_unique,columns=['genre'])

# creating boolean columns for each genre
movies = movies.join(movies.genres.str.get_dummies().astype(bool))
movies.drop('genres',inplace=True, axis=1)

In [87]:
# converting timestamp to date type
ratings.timestamp = pd.to_datetime(ratings.timestamp,infer_datetime_form
ratings.timestamp = ratings.timestamp.dt.year

In [88]:
movies.dropna(inplace=True)
ratings.dropna(inplace=True)

In [100]:
# temp df
tdf = movies[['movieId','year']].groupby('year')

fig, ax1 = plt.subplots(figsize=(10,5))
ax1.plot(tdf.year.first(),tdf.movieId.nunique(),"b-o")


# tdf = ratings[['rating','timestamp']].groupby('timestamp')
# ax2 = ax1.twinx()
```



```
In [88]:
movies.dropna(inplace=True)
ratings.dropna(inplace=True)

In [100]:
# temp df
tdf = movies[['movieId','year']].groupby('year')

fig, ax1 = plt.subplots(figsize=(10,5))
ax1.plot(tdf.year.first(),tdf.movieId.nunique(),"b-o")


# tdf = ratings[['rating','timestamp']].groupby('timestamp')
# ax2 = ax1.twinx()
# ax2.plot(tdf.timestamp.first(),tdf.rating.count(),'r-o')
ax1.set_ylabel('Number of movies released')
ax1.set_xlabel('Year')
plt.title('Movies per Year')

Out[100]: Text(0.5, 1.0, 'Movies per Year')
```



```
In [108]:
plt.figure(figsize = (15,5))
tdf = movies[['movieId','year']].groupby('year')
df = pd.DataFrame()
```



```
In [108]:
plt.figure(figsize = (15,5))
tdf = movies[['movieId','year']].groupby('year')
df = pd.DataFrame()

# get counts of each genre
for genre in genres_unique.genre:
    tdf = movies[movies[genre]][['movieId','year']].groupby('year')
    df[genre] = tdf.movieId.nunique().cumsum()

df.fillna(method='ffill',inplace=True)

barlist = df.iloc[-1].plot.bar()
plt.title('Movies per genre')
plt.xlabel('Genre')
plt.ylabel('Frequency')
plt.show()
```



```
In [102]:
df
```

Lessons learned

- Recommender may not be as easy to evaluate as previously expected since 5 point rating system is not very accurate. For example, it is hard to distinguish top 5 movies for users who rate most movies 5 stars.
- Planning to do the following to improve speed of recommender:
  - Ask user to pick genre.
  - Limit data to be from 1985 to present.

Recommendations

- The following stories are essential for the core app and should be completed next.
  - Create Train and Test Data set. (0 pt)
  - Build and Train Model. (2 pt)
  - Create User Input Boxes (Movie Name and their Rating). (1 pt)
  - Match Input to Movie in Dataset. (1 pt)
  - Autocomplete Title Input. (1 pt)
  - Returns top 5 recommendations from model. (1 pt)
  - Story 1: Deploy app via Flask (AWS). (4 pts)

- After completion of these stories, most of the other stories involve tuning the model and improving the app.