

Examen segundo parcial

# Primer paso

- Primero cargue los datos de testeo para comenzar a ver los datos con los que se van a trabajar. Me di cuenta que había muchos valores guardados en el formato de json y comenze a buscar una manera para cargarlos y trabajar con la librería de sklearn

```
movies = pd.read_csv('train.csv', index_col=None, na_values=['NA'])

#10 valores csv
print(movies.head(10))
#tipos de valores
print(movies.dtypes)
print('')
#datos estadisticos de valores nominales
print(movies.describe())
print('')
#nos da el porcentaje de datos nulos
print(pd.DataFrame({'percent_missing': movies.isnull().sum() * 100 / len(movies)}))
print('')
#Nos dice que tan impotantes son los datos unicos
print(pd.DataFrame({'percent_unique': movies.apply(lambda x: x.unique().size/x.size*100)}))

# print(movies.belongs_to_collection)

# for movie in movies:
#     print(movie)
```

# Segunda parte

Continúe buscando en las clases del csv, para poder ver todos las que tenían un json como dato encontré una manera de que, con esto me dice que

```
print(movies.belongsto_collection)
for movie in movies:
    print(movie)

print(movies.head(1).belongs_to_collection.to_string())
pd.set_option('display.max_columns', len(movies))
print(movies.head(1))
```

belongs\_to\_collection  
genres  
production\_companies  
production\_countries  
spoken\_languages  
Keywords  
cast  
crew

# Tercera parte

- A la hora de buscar como cargar los json me salio el error de que existían valores nulos por lo que procedi a ver el porcentaje de valores nulos y únicos para proceder a quitar las clases
- Procedi a quitar todas las que tuvieran 100 datos únicos, porque no ayudan a la búsqueda de patrones en la clase

	percent_missing
id	0.000000
belongs_to_collection	79.866667
budget	0.000000
genres	0.233333
homepage	68.466667
imdb_id	0.000000
original_language	0.000000
original_title	0.000000
overview	0.266667
popularity	0.000000
poster_path	0.033333
production_companies	5.200000
production_countries	1.833333
release_date	0.000000
runtime	0.066667
spoken_languages	0.666667
status	0.000000
tagline	19.900000
title	0.000000
Keywords	9.200000
cast	0.433333
crew	0.533333
revenue	0.000000
	percent_unique
id	100.000000
belongs_to_collection	14.100000
budget	13.533333
genres	29.100000
homepage	31.400000
imdb_id	100.000000
original_language	1.200000
original_title	99.166667
overview	99.766667
popularity	99.966667
poster_path	100.000000
production_companies	79.466667
production_countries	10.733333
release_date	79.933333
runtime	4.666667
spoken_languages	13.400000
status	0.066667
tagline	80.033333
title	98.966667
Keywords	88.300000
cast	99.200000
crew	99.500000
revenue	95.000000

# Cuarta parte

- Primero con transformamos las columnas que estaban en json para que pudieran ser procesadas convirtiendo los datos que tienen NaN a datos con un json vacío {}

```
''' transform json columns '''

import ast

dict_columns = ['belongs_to_collection', 'genres', 'production_companies',
                 'production_countries', 'spoken_languages', 'Keywords', 'cast', 'crew']

def text_to_dict(df):
    for column in dict_columns:
        df[column] = df[column].apply(lambda x: {} if pd.isna(x) else ast.literal_eval(x))
    return df

movies = text_to_dict(movies)
```

```
0 [{"id": 313576, "name": "Hot Tub Time Machine Collecti
1 [{"id": 107674, "name": "The Princess Diaries Collecti
2 nan
3 nan
4 nan
0 [{"id": 313576, "name": "Hot Tub Time Machine Collecti
1 [{"id": 107674, "name": "The Princess Diaries Collecti
2 {}
3 {}
4 {}}
```

# Quinta parte

- En esta parte comencé a extraer los datos de los json que nos pudieran servir, en el json

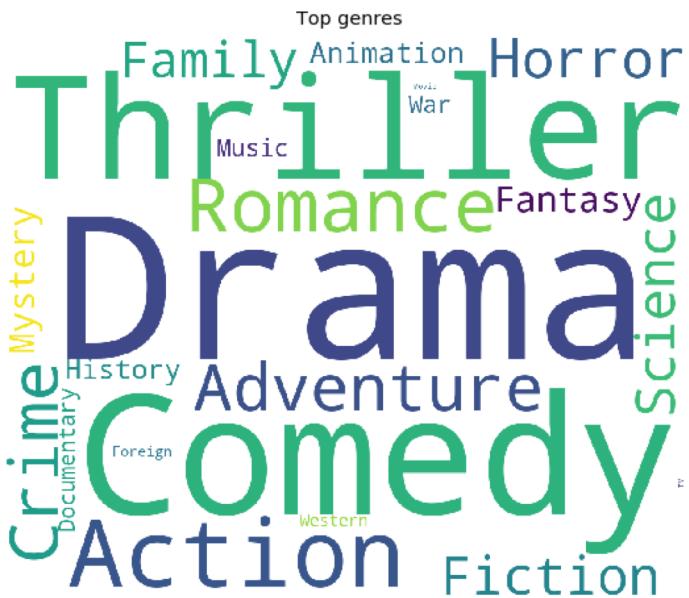
## Belongs to collection

Solo nos quedamos con name, debido a que las demás clases, eran paths hacia imágenes que no nos ayudan a nuestro objetivo

# Generos

- Para los géneros fue un problema grande, porque muchos datos tenían combinaciones de géneros diferentes, lo que hacia las características únicas, entonces con un kernel que me encontré, pase el json a 3 tipos de características, el numero de géneros que tenían, todos los géneros juntos en un string y una columna por cada uno de los 15 géneros mas utilizados, después de esto analizamos los resultados de cada columna agregada para ver si estábamos en lo correcto

Con las tablas obtenidas se puede observar Que fue un poco exagerado el tomar tantos generos y hacerlos valores individuales, por lo que se utilizo la tabla de generos mas utilizados y el porcentaje de valores nulos para determinar que solo 5 eran adecuados



	percent_unique
budget	13.533333
homepage	31.400000
original_language	1.200000
original_title	99.166667
overview	99.766667
popularity	99.966667
release_date	79.933333
runtime	4.666667
status	0.066667
tagline	80.033333
title	98.966667
revenue	95.000000
num_genres	0.266667
all_genres	17.966667
genre_Drama	0.066667
genre_Comedy	0.066667
genre_Thriller	0.066667
genre_Action	0.066667
genre_Romance	0.066667
genre_Crime	0.066667
genre_Adventure	0.066667
genre_Horror	0.066667
genre_Science Fiction	0.066667
genre_Family	0.066667
genre_Fantasy	0.066667
genre_Mystery	0.066667
genre_Animation	0.066667
genre_History	0.066667
genre_Music	0.066667
belongs_to_collection_name	14.100000

	percent_missing
budget	0.000000
homepage	68.466667
original_language	0.000000
original_title	0.000000
overview	0.266667
popularity	0.000000
release_date	0.000000
runtime	0.066667
status	0.000000
tagline	19.900000
title	0.000000
revenue	0.000000
num_genres	0.000000
all_genres	0.000000
genre_Drama	48.966667
genre_Comedy	65.733333
genre_Thriller	73.700000
genre_Action	75.300000
genre_Romance	80.966667
genre_Crime	84.366667
genre_Adventure	85.366667
genre_Horror	89.966667
genre_Science Fiction	90.333333
genre_Family	91.333333
genre_Fantasy	92.266667
genre_Mystery	92.500000
genre_Animation	95.300000
genre_History	95.600000
genre_Music	96.666667
belongs_to_collection_name	79.866667

# Productoras

- Con las productoras, fue el mismo problema que con los géneros y después de hacer el análisis, me di cuenta que no me servía separar las productoras y que el dato mas funcional era el numero de productoras

num_companies	5.200000
all_production_companies	0.000000
production_company_Warner Bros.	92.900000
production_company_Universal Pictures	93.666667
production_company_Paramount Pictures	94.633333
production_company_Twentieth Century Fox Film C...	95.400000
production_company_Columbia Pictures	94.866667
production_company_Metro-Goldwyn-Mayer (MGM)	97.200000
production_company_New Line Cinema	97.500000
production_company_Touchstone Pictures	97.900000
production_company_Walt Disney Pictures	97.933333
production_company_Columbia Pictures Corporation	97.966667
production_company_TriStar Pictures	98.233333
production_company_Relativity Media	98.400000
production_company_Canal+	97.966667
production_company_United Artists	98.533333
production_company_Miramax Films	98.666667
production_company_Village Roadshow Pictures	98.800000
production_company_Regency Enterprises	98.966667
production_company_BBC Films	99.000000
production_company_Dune Entertainment	98.700000
production_company_Working Title Films	99.000000
production_company_Fox Searchlight Pictures	99.033333
production_company_StudioCanal	99.033333
production_company_Lionsgate	99.000000
production_company_DreamWorks SKG	99.100000
production_company_Fox 2000 Pictures	99.166667
production_company_Summit Entertainment	99.200000
production_company_Hollywood Pictures	99.166667
production_company_Orion Pictures	99.166667
production_company_Amblin Entertainment	99.233333
production_company_Dimension Films	99.233333

num\_companies    0.600000  
all\_production\_companies                            79.466667

# Paises

- Se hizo lo mismo y se tomaron 4 mas comunes debido a valores nulos, en valores únicos estaban bastante bien y parece que todos los países que no sean US son outliers

all_production_companies	0.000000
num_countries	1.833333
all_countries	0.000000
production_country_US	23.933333
production_country_GB	87.333333
production_country_FR	92.600000
production_country_DE	94.433333
production_country_CA	96.000000
production_country_IN	97.300000
production_country_IT	97.866667
production_country_JP	97.966667
production_country_AU	97.966667
production_country_RU	98.066667

# Idiomas hablados

Por la cantidad de películas con lenguajes diferentes al inglés, podemos deducir que cualquier película que hable otro idioma es outlier y que la mayoría de las películas hablan solo un idioma, lo que nos permite eliminar esta columna y quedarnos con solo la del idioma original

```
Alfonsos-MacBook-Pro:examenSegundoParcial chatrol51$ python3 movies.py
[('en', 2618), ('fr', 288), ('es', 239), ('de', 169), ('ru', 152), ('it', 124), ('ja', 89), ('zh', 68), ('hi', 56), ('pt', 43)]
Alfonsos-MacBook-Pro:examenSegundoParcial chatrol51$
```

Solo dejamos una opción donde agregara una columna con el numero de idiomas hablados si el numero era 1 o 0 lo pondría en nulo mas mostraría el numero de idiomas hablados

```
"spoken_languages"
movies['num_languages'] = movies['spoken_languages'].apply(lambda x: len(x) if x != {} and len(x) > 1 else None)
# movies[111].spoken_languages
# movies[111].num_languages
```

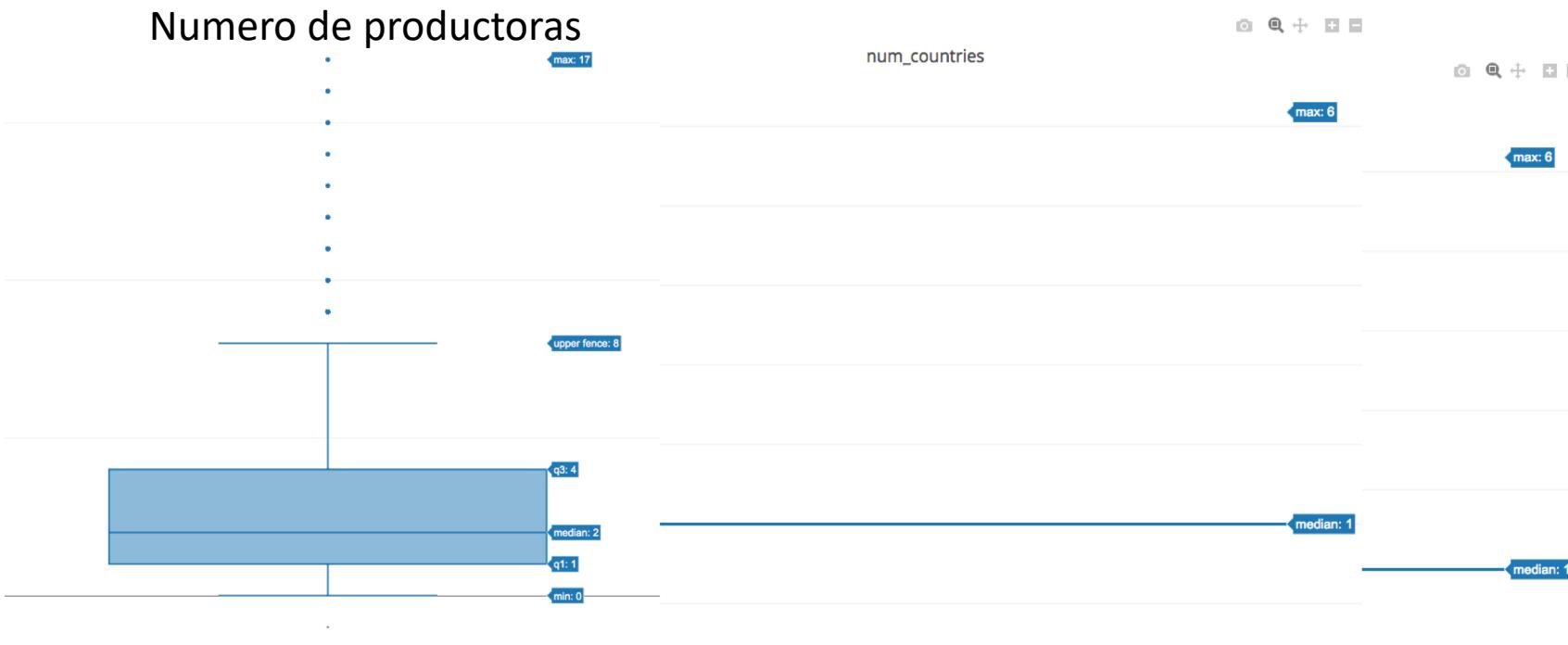
# Elenco

- Este fue otro tipo de problema debido a la información que podíamos extraer del json, el del equipo de producción era muy similar por lo que se le trato de la misma forma

```
0 [{"cast_id": 4, "character": "Lou", "credit_id": "52fe4ee7c3a36847f82afae7", "gender": 2, "id": 52997, "name": "Rob Corddry", "order": 0, "profile_path": "/k2zJL0V1nEZuFT08xUd0d3ucfxz.jpg"}], 1 [{"cast_id": 5, "character": "Dwight Schrute", "credit_id": "52fe4ee7c3a36847f82afae7", "gender": 2, "id": 52998, "name": "Rainn Wilson", "order": 1, "profile_path": "/k2zJL0V1nEZuFT08xUd0d3ucfxz.jpg"}], 2 [{"cast_id": 6, "character": "Pam Beesly", "credit_id": "52fe4ee7c3a36847f82afae7", "gender": 2, "id": 52999, "name": "Jenna Fischer", "order": 2, "profile_path": "/k2zJL0V1nEZuFT08xUd0d3ucfxz.jpg"}], 3 [{"cast_id": 7, "character": "Michael Scott", "credit_id": "52fe4ee7c3a36847f82afae7", "gender": 2, "id": 53000, "name": "Steve Carell", "order": 3, "profile_path": "/k2zJL0V1nEZuFT08xUd0d3ucfxz.jpg"}]
```

# Quitar datos fuera de parametros

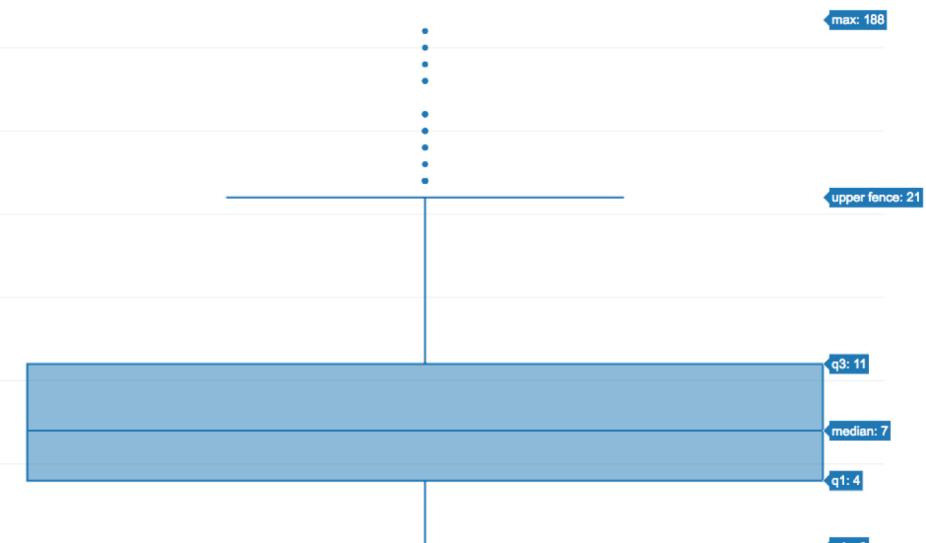
- Primero se busco hacer boxplot de los datos que para revisar si tenían outliers, removimos overview debido a que es un string



# Popularidad

- Esta clase tenía demasiados decimales, lo que hacia parecer que era dato único, lo único que hice fue redondearla a 0 decimales e imprimí los valores

```
movies['popularity'] = movies['popularity'].round(0)
popularity_values = movies['popularity'].value_counts()
print(popularity_values)
```



8.0	188
6.0	184
7.0	174
5.0	147
1.0	145
9.0	143
11.0	127
10.0	127
2.0	126
4.0	118
3.0	114
12.0	100
13.0	84
14.0	63
0.0	55
15.0	43
16.0	26
17.0	17
19.0	12
18.0	12
20.0	9
21.0	5
22.0	5
34.0	3
26.0	3
25.0	3
37.0	2
24.0	2
28.0	2
39.0	2
41.0	2
188.0	1
155.0	1
47.0	1
64.0	1
55.0	1
90.0	1
49.0	1
185.0	1
30.0	1
36.0	1
42.0	1
88.0	1
31.0	1
52.0	1
23.0	1
29.0	1

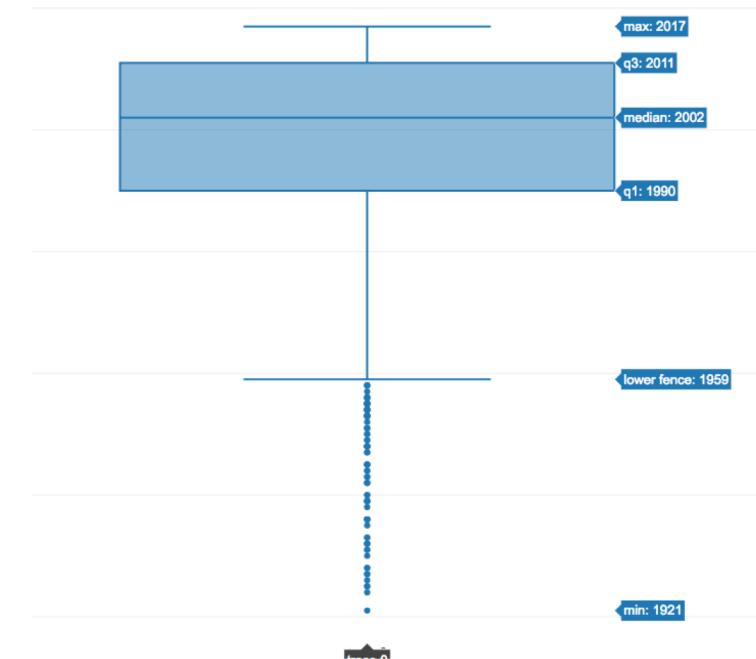
# Día de lanzamiento

- Para el día de lanzamiento, estaba muy alto el nivel de dato único por lo que procedí a redondearlos al año solamente logrando que se redujera muchísimo la originalidad del valor, transformándola a un entero

release\_date 85.332686

release\_date 4.468719

release\_date int64

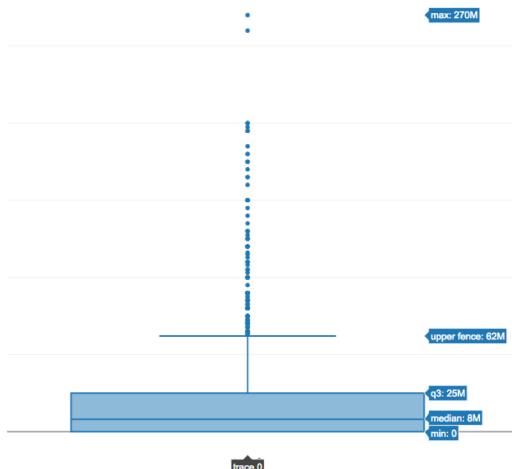


# Transformar objetos con getdummies y quitar outliers

Status: removida

```
Alfonso's-MacBook-Pro:examenSegundoParcial chatrol51$ python3 movies.py
0 Released
1 Released
2 Released
3 Released
4 Released
Alfonso's-MacBook-Pro:examenSegundoParcial chatrol51$ python3 movies.py
0 1
1 1
2 1
3 1
4 1
Alfonso's-MacBook-Pro:examenSegundoParcial chatrol51$ python3 movies.py
1 2996
0 PM 4
Name: status, dtype: int64
Alfonso's-MacBook-Pro:examenSegundoParcial chatrol51$
```

budget



Mas comunes home page: removida

```
[(nan, 689), ('http://skinningmovie.com/synopsis.htm', 1), ('http://www.magnetreleasing.com/vhs/', 1), ('h
833910c-fd4a-4fcf-a734-b3d252473a03', 1), ('http://www.imdb.com/title/tt0095169/', 1), ('http://www.wereth
www.frozen-film.com/', 1), ('http://sonyclassics.com/thebronze/', 1), ('http://www.frankandlolamovie.com/'
16, 1), ('http://www.frankandlolamovie.com/1516', 1)]
```

# Reducción de dimensiones

Con SelectinKbest enconte varios valores en infinito por falta de tiempo  
Solo pude quitarlos

```
test = SelectKBest(score_func=f_classif,k=5)
fit = test.fit(data_features, data_labels)
# print('')
# print(json.dumps(dict(zip(movies.columns, fit.scores_)), indent=2, sort_keys=True))
features = fit.transform(data_features)

scaler = MinMaxScaler(feature_range=(0,1))
features = scaler.fit_transform(data_features)
```

```
{
    "budget": 1.1651587370325274,
    "departments_Art": 3.335896128050371,
    "departments_Camera": 0.15941769818940454,
    "departments_Costume & Make-Up": 3.760412620183901,
    "departments_Crew": 1.620288148850596,
    "departments_Directing": 1.5578311094042296,
    "departments_Editing": 9.305054673474621,
    "departments_Lighting": NaN,
    "departments_Production": 3.923734679692334,
    "departments_Sound": 1.9549663582937444,
    "departments_Visual Effects": 7.302625259378424,
    "departments_Writing": 3.7900018793459918,
    "genders_0.cast": 0.7492816872074479,
    "genders_0.crew": 2.4159964456216455,
    "genders_1.cast": 1.3089591646638028,
    "genders_1.crew": 1.3979496028859681,
    "genders_2.cast": 4.179171446945165,
    "genders_2.crew": 2.0354333436932323,
    "genre_Action": 0.89504038527061194,
    "genre_Comedy": 1.276265383440606,
    "genre_Drama": 1.035328680835737,
    "genre_Romance": 1.1981399307727543,
    "genre_Thriller": 0.770165642015755,
    "has_collection": 1.349813155880203,
    "jobs_Art Direction": 1.196646623466527,
    "jobs_Casting": 1.2297280399880677,
    "jobs_Costume Design": 2.5934550390627713,
    "jobs_Director": 1.8673919032535762,
    "jobs_Director of Photography": 1.467886465098937,
    "jobs_Editor": 2.4261719448532104,
    "jobs_Executive Producer": 8.306122885142964,
    "jobs_Makeup Artist": 8.71223712433844,
    "jobs_Original Music Composer": 1.4988605534960713,
    "jobs_Producer": 5.399489631324722,
    "jobs_Production Design": 1.1870177303013634,
    "jobs_Screenplay": 1.1624525071270146,
    "jobs_Set Decoration": 3.0339931454151223,
    "jobs_Sound Re-Recording Mixer": 3.131746412775437,
    "jobs_Writer": 2.1195999511261636,
    "language_de": 2.298711625999215,
    "language_en": 1.8516495617111246,
    "language_es": 1.6017171434566693,
    "language_fr": 0.92107838440635507,
    "num_Keywords": 0.8624113989717409,
    "num_cast": 0.8575728411956145,
    "num_companies": 1.6440062350743032,
    "num_countries": 0.678378471784596,
    "num_genres": 1.044664101782567,
    "num_languages": 0.6409322436905234,
    "original_language": NaN,
    "popularity": 1.1058835647803794,
    "release_date": 0.6233622475813905,
    "revenue": 3.632875028727336,
    "runtime": 0.31730306717014517
}
```

```
{
    "budget": 1.1651587370325274,
    "cast_character": 1.35042596348884,
    "cast_character_Additional Voices (voice)": 1.1663435885650324,
    "cast_character_Bartender": 1.4160794222564208,
    "cast_character_Dancer": Infinity,
    "cast_character_Debutante": 0.7650765894943008,
    "cast_character_Doctor": 0.551964431969712,
    "cast_character_Frank": 2.8324268856771555,
    "cast_character_Herself": Infinity,
    "cast_character_Himself": 1.2608414352661383,
    "cast_character_Jack": NaN,
    "cast_character_Nurse": 1.8926741624117018,
    "cast_character_Paul": 0.7515927755718395,
    "cast_character_Reportor": 1.951563265020634,
    "cast_character_Security Guard": 1.4607501623013546,
    "cast_character_WAITRESS": 1.287088200321747,
    "crew_name_Avy Kaufman": NaN,
    "crew_name_Bob Weinstein": Infinity,
    "crew_name_Deborah Agra": Infinity,
    "crew_name_Francine Maisler": Infinity,
    "crew_name_Harvey Weinstein": 0.11495418619290758,
    "crew_name_James Horner": 1.5391457415308574,
    "crew_name_James Newton Howard": Infinity,
    "crew_name_Janet Hirshenson": 4.179171446945165,
    "crew_name_Jerry Goldsmith": NaN,
    "crew_name_Kerry Barden": Infinity,
    "crew_name_Luc Besson": 0.5175153295563171,
    "crew_name_Mary Vernieu": Infinity,
    "crew_name_Robert Rodriguez": Infinity,
    "crew_name_Steven Spielberg": Infinity,
    "crew_name_Tricia Wood": 0.4929076029936347,
    "departments_Art": 3.335896128050371,
    "departments_Camera": 0.15941769818940454,
    "departments_Costume & Make-Up": 3.760412620183901,
    "departments_Crew": 1.620288148850596,
    "departments_Directing": 1.5578311094042296,
    "departments_Editing": 9.305054673474621,
    "departments_Lighting": NaN,
    "departments_Production": 3.923734679692334,
    "departments_Sound": 1.9549663502937444,
    "departments_Visual Effects": 7.302625259378424,
    "departments_Writing": 3.7900018793459918,
    "genders_0.cast": 0.7492816872074479,
    "genders_0.crew": 2.4159964456216455,
    "genders_1.cast": 1.3089591646638028,
    "genders_1.crew": 1.3979496028859681,
    "genders_2.cast": NaN,
    "genders_2.crew": 2.0354333436932323,
    "genre_Action": 0.89504038527061194,
    "genre_Comedy": 1.276265383440606,
    "genre_Drama": 1.035328680835737,
    "genre_Romance": 1.1981399307727543,
```

# Aplicar el primer modelo

```
0 # print(pd.DataFrame(['percent_unique': movies.apply(lambda x: x.unique().size/x.size*100)]))  
1  
2 '''Applying models'''  
3  
4 X_train, X_test, Y_train, Y_test = train_test_split(features, data_labels, test_size=0.3)  
5  
6 model = DecisionTreeRegressor(max_depth=7)  
7 model.fit(X_train, Y_train)  
8  
9 # names = ['Decision Tree regressors', 'MLP Classifier',  
#           'Random Forest Classifier', 'AdaBoost',  
#           'Bagging Classifier']  
0 # regressors = [  
#  
# ]  
1 score = mean_squared_error(Y_test, model.predict(X_test))  
2  
3 print(score)
```

```
Data with input dtype object was converted to float64 by MinMaxScaler.  
1280264253830591.2  
Alfonso-MacBook-Pro:examenSegundoParcial chatrol51$ python3 movies.py  
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-pac  
Features [ 1 53] are constant.  
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-pac  
invalid value encountered in true_divide  
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-pac  
Data with input dtype object was converted to float64 by MinMaxScaler.  
1242944343830731.5  
Alfonso-MacBook-Pro:examenSegundoParcial chatrol51$ █
```

- Con nuestro modelo sacamos un MSE muy grande, por lo que jugamos un poco con los valores, pero no tuvo buenos resultados asi que continuamos con la parte de agregar mas modelos y continuamos con el mismo error

```

names = ['Decision Tree regressors', 'MLP regressors',
         'Random Forest regressors', 'AdaBoost',
         'Bagging regressors']

regressors = [
    DecisionTreeRegressor(max_depth=7),
    MLPRegressor(alpha=1),
    RandomForestRegressor(max_depth=5, max_features=10, n_estimators=10),
    AdaBoostRegressor(n_estimators=10),
    BaggingRegressor(max_features=1,n_estimators=10)
]

for name, regressor in zip(names, regressors):
    regressor.fit(X_train,Y_train)
    y_pred = regressor.predict(X_test)
    print_('{}: {}'.format(name, mean_squared_error(Y_test, y_pred)))
```

```

Decision Tree regressors: 1534609408530439.0
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/s

Stochastic Optimizer: Maximum iterations (200) reached and the op

MLP regressors: 1186826301209413.0
Random Forest regressors: 872368149392667.1
AdaBoost: 850541647083990.5
Bagging regressors: 934031777375785.6
```

# Conclusiones

- Tuve muchos problemas lidiando con los datos, desde pasar los json a columnas y que los mismos tuvieran muchísimos, nulos creo que no fue la mejor idea pasarlos todos a columnas y por la gran cantidad de valores iguales, estoy sobre entrenando mis modelos, sigo sin encontrar una manera de pasar adecuadamente las clases extraídas de los json, me gusto mucho hacer este proyecto.
- Creo que pese a que los resultados no fueron los adecuados, aprendí mucho de la manipulación de datos y tratar de acomodar las clases lo mejor posible.