DATA VISUALIZATION

Project

He NI

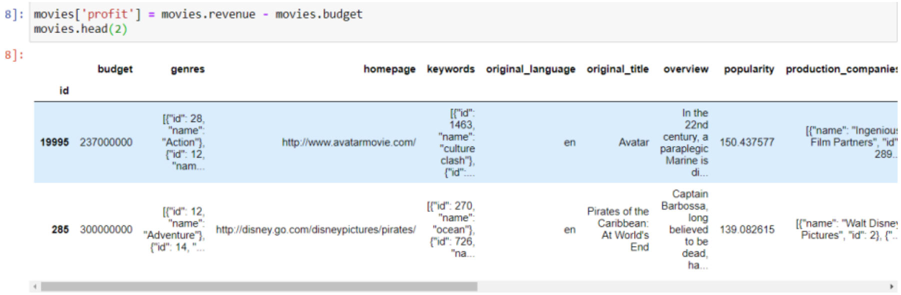
Bader GUETARI

BI2

1. **The dataset link is: https://www.kaggle.com/tmdb/tmdb-movie-metadata**

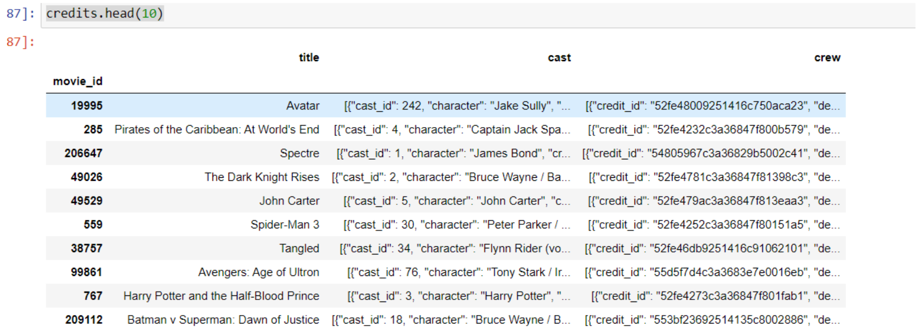
2. **We chose this dataset to find out the trends in the movie market, what types of movies are more popular and rated higher, and our audience is the movie companies, with the aim of providing data analysis to the companies.**

3. **First we get the dataset and find two csv files, movies.csv and credits.csv**



```
8]: movies['profit'] = movies.revenue - movies.budget
    movies.head(2)
8]:
```

id	budget	genres	homepage	keywords	original_language	original_title	overview	popularity	production_companie:
19995	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingeniou: Film Partners", "id" 289..
285	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disne: Pictures", "id": 2}, {"...

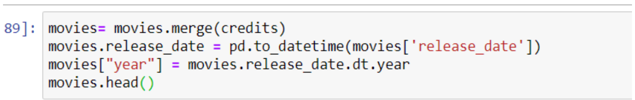


```
87]: credits.head(10)
87]:
```

movie_id	title	cast	crew
19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...
559	Spider-Man 3	[{"cast_id": 30, "character": "Peter Parker / ...	[{"credit_id": "52fe4252c3a36847f80151a5", "de...
38757	Tangled	[{"cast_id": 34, "character": "Flynn Rider (vo...	[{"credit_id": "52fe46db9251416c91062101", "de...
99861	Avengers: Age of Ultron	[{"cast_id": 76, "character": "Tony Stark / Ir...	[{"credit_id": "55d5f7d4c3a3683e7e0016eb", "de...
767	Harry Potter and the Half-Blood Prince	[{"cast_id": 3, "character": "Harry Potter", "...	[{"credit_id": "52fe4273c3a36847f801fab1", "de...
209112	Batman v Superman: Dawn of Justice	[{"cast_id": 18, "character": "Bruce Wayne / B...	[{"credit_id": "553bf23692514135c8002886", "de...

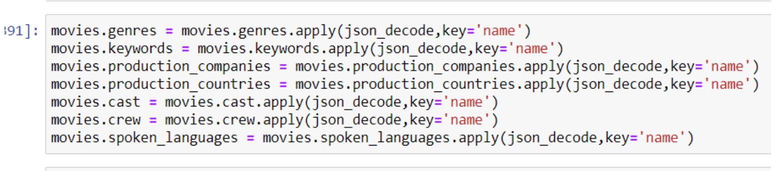
We need to merge and clean up these two data.

--Convert the year to pd.datetime and merge at the end of the table



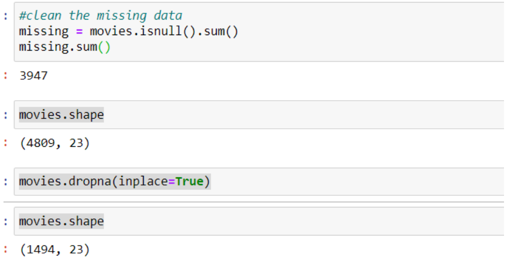
```
89]: movies= movies.merge(credits)
     movies.release_date = pd.to_datetime(movies['release_date'])
     movies["year"] = movies.release_date.dt.year
     movies.head()
```

--Perform data normalisation, i.e. data pre-processing, to split the features: crew, cast, production_countries .... Perform splitting.



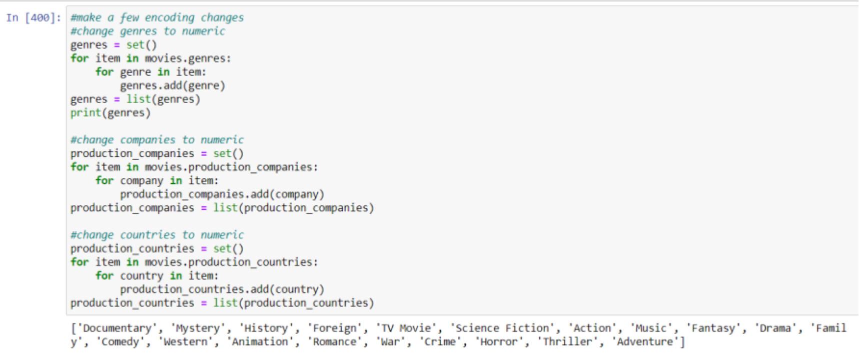
```
91]: movies.genres = movies.genres.apply(json_decode,key='name')
     movies.keywords = movies.keywords.apply(json_decode,key='name')
     movies.production_companies = movies.production_companies.apply(json_decode,key='name')
     movies.production_countries = movies.production_countries.apply(json_decode,key='name')
     movies.cast = movies.cast.apply(json_decode,key='name')
     movies.crew = movies.crew.apply(json_decode,key='name')
     movies.spoken_languages = movies.spoken_languages.apply(json_decode,key='name')
```

--then delete the empty value



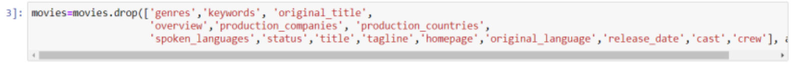
```
: #clean the missing data
  missing = movies.isnull().sum()
  missing.sum()
: 3947

: movies.shape
: (4809, 23)

: movies.dropna(inplace=True)

: movies.shape
: (1494, 23)
```

4. Because there are many non-character types in the data, such as genres, we encode changes to them



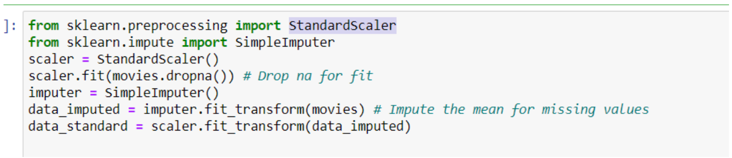
```
In [400]: #make a few encoding changes
          #change genres to numeric
          genres = set()
          for item in movies.genres:
              for genre in item:
                  genres.add(genre)
          genres = list(genres)
          print(genres)

          #change companies to numeric
          production_companies = set()
          for item in movies.production_companies:
              for company in item:
                  production_companies.add(company)
          production_companies = list(production_companies)

          #change countries to numeric
          production_countries = set()
          for item in movies.production_countries:
              for country in item:
                  production_countries.add(country)
          production_countries = list(production_countries)

          ['Documentary', 'Mystery', 'History', 'Foreign', 'TV Movie', 'Science Fiction', 'Action', 'Music', 'Fantasy', 'Drama', 'Famil
          y', 'Comedy', 'Western', 'Animation', 'Romance', 'War', 'Crime', 'Horror', 'Thriller', 'Adventure']
```

5. drop those that cannot be encoded (too many columns)



```
3]: movies=movies.drop(['genres','keywords', 'original_title',
                        'overview','production_companies', 'production_countries',
                        'spoken_languages','status','title','tagline','homepage','original_language','release_date','cast','crew'],
```

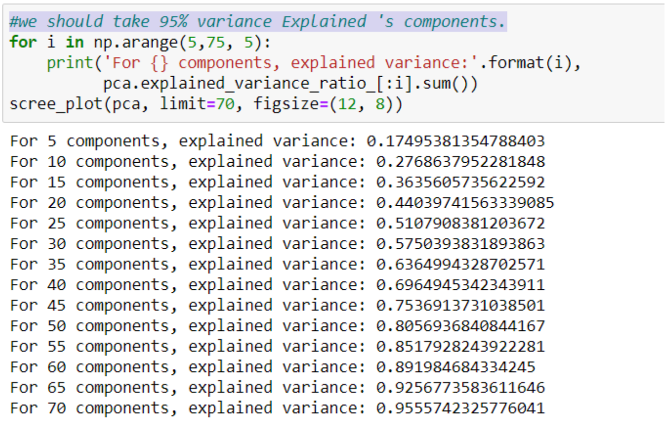
6. we continue with the data pre-processing. Before we apply dimensionality reduction techniques to the data, we need to perform feature scaling so that the principal component vectors are not influenced by the natural differences in scale for features. Here we have chosen the Standard Scaler method.



```
]: from sklearn.preprocessing import StandardScaler
   from sklearn.impute import SimpleImputer
   scaler = StandardScaler()
   scaler.fit(movies.dropna()) # Drop na for fit
   imputer = SimpleImputer()
   data_imputed = imputer.fit_transform(movies) # Impute the mean for missing values
   data_standard = scaler.fit_transform(data_imputed)
```

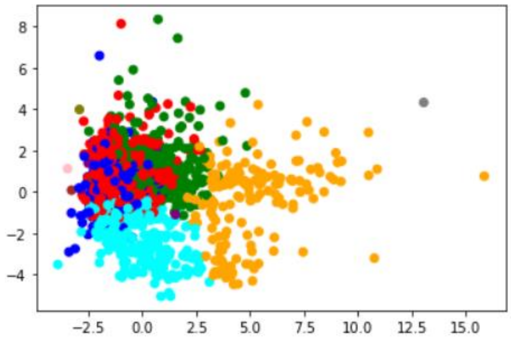
Then using the PCA method for dimensionality reduction, because we have so many features, then we need to reduce the complexity of the data and identify the most important multiple features. So in comparison PCA is more appropriate and more efficient.

7. we should take 95% variance Explained 's components. So component should be 70.

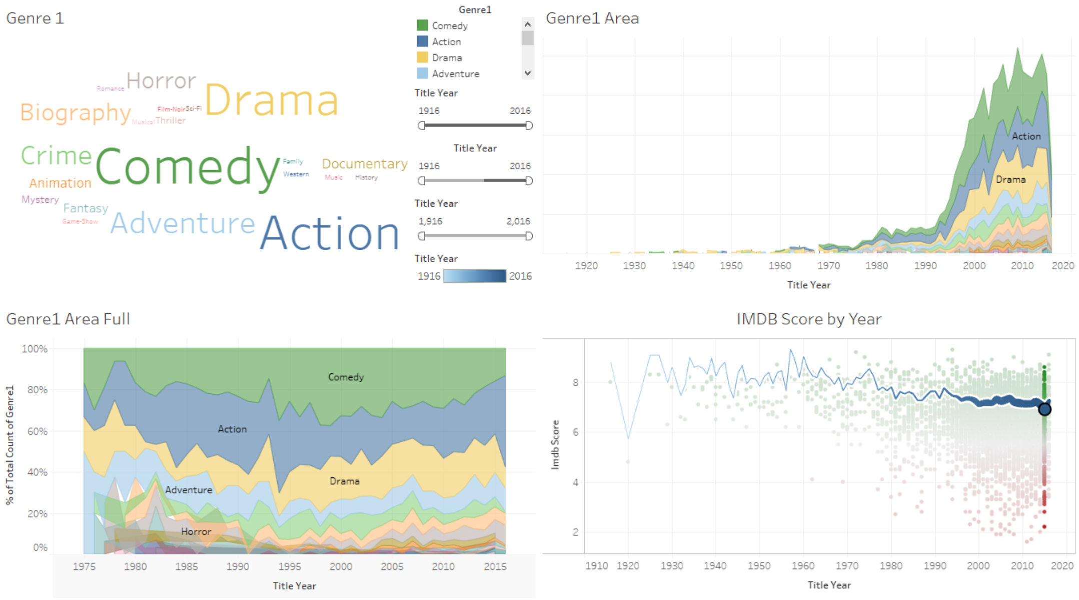


```
#we should take 95% variance Explained 's components.
for i in np.arange(5,75, 5):
    print('For {} components, explained variance:'.format(i),
          pca.explained_variance_ratio_[:i].sum())
scree_plot(pca, limit=70, figsize=(12, 8))

For 5 components, explained variance: 0.17495381354788403
For 10 components, explained variance: 0.2768637952281848
For 15 components, explained variance: 0.3635605735622592
For 20 components, explained variance: 0.44039741563339085
For 25 components, explained variance: 0.5107908381203672
For 30 components, explained variance: 0.5750393831893863
For 35 components, explained variance: 0.6364994328702571
For 40 components, explained variance: 0.6964945342343911
For 45 components, explained variance: 0.7536913731038501
For 50 components, explained variance: 0.8056936840844167
For 55 components, explained variance: 0.8517928243922281
For 60 components, explained variance: 0.891984684334245
For 65 components, explained variance: 0.9256773583611646
For 70 components, explained variance: 0.9555742325776041
```

8. Finally, the image is drawn using matplotlib.



9. Tableau dashboard
https://public.tableau.com/profile/ni.he#!/vizhome/IMDB5000MovieDataset_16089032498620/DashboardGenres?publish=yes


Genre 1
Romance
Horror
Drama
Biography
Musical
Film-Noir
Sci-Fi
Thriller
Crime
Comedy
Family
Western
Documentary
Music
History
Animation
Mystery
Fantasy
Game-Show
Adventure
Action
Genre1
Comedy
Action
Drama
Adventure
Title Year
1916
2016
Title Year
1916
2016
Title Year
1,916
2,016
Title Year
1916
2016
Genre1 Area
Action
Drama
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
Title Year
Genre1 Area Full
100%
80%
60%
40%
20%
0%
% of Total Count of Genre1
Comedy
Action
Adventure
Drama
Horror
1975
1980
1985
1990
1995
2000
2005
2010
2015
Title Year
IMDB Score by Year
8
6
4
2
Imdb Score
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
Title Year