Analysis of the Driving Factors in Film Production

Introduction: Film

- Webster’s dictionary : A recording of moving images that tells a story and that people watch on a screen or television

- With the growing influence of Film in our daily lives, we aim perform an analysis of the creative and non creative elements in Film Production and its driving factors

Dataset

Credits

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / B...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

Movies

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"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
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788

https://www.kaggle.com/tmdb/tmdb-movie-metadata

Issues With the Dataset

- **Problems**
  - Columns are in dictionary format
  - Column Names of dataframes are not convenient to perform joins
  - Features aren’t very useful in their current state. For example, Release date, popularity, vote count, vote average[ratings] and etc.
  - There is no target Variable

- **Solution**: Feature Engineering

Feature Engineering

1) Mapping Dictionary like columns to dataframes and re-naming columns

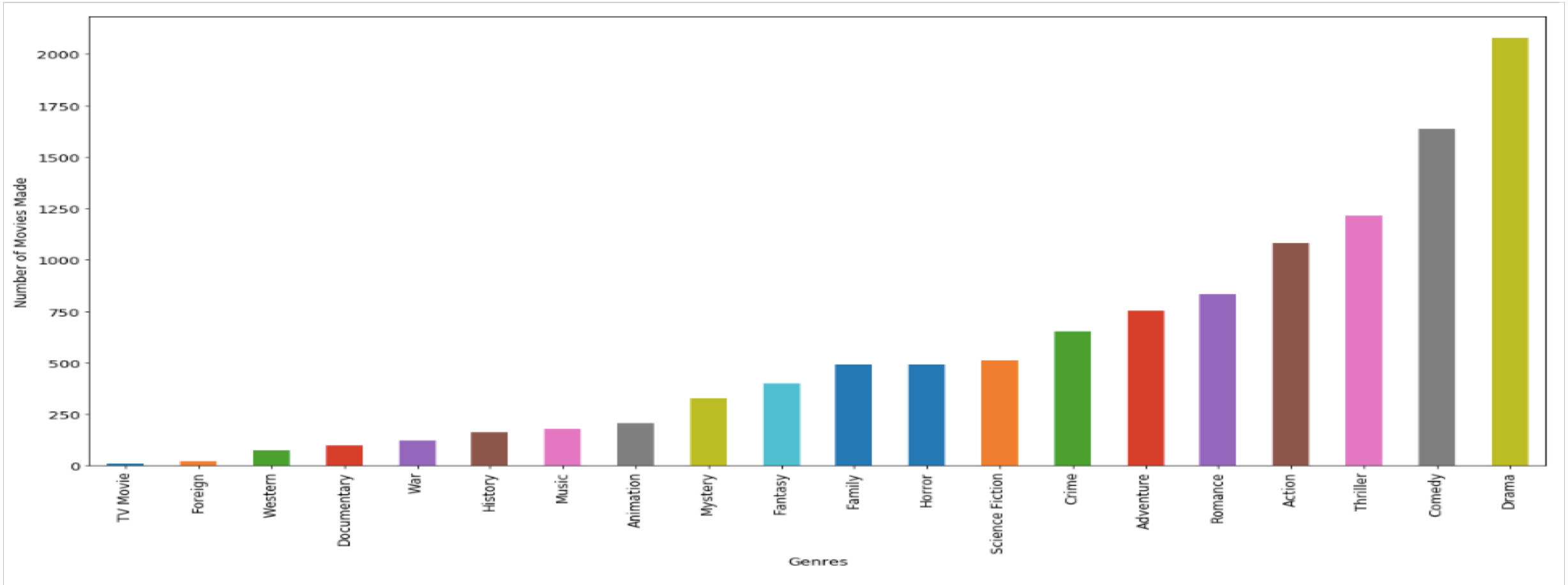
```
#Clean the Dataset
#Create new dataframes from Dictionary like columns
def mapper(df, col, new_col):
    temp = []
    for n, row in df.iterrows():
        [temp.append(tuple(i.values())+(row[new_col],)) for i in json.loads(row[col])]
    new_df = pd.DataFrame(temp)
    new_df.columns = list(json.loads(row[col])[0].keys()) + ['movie_id']

    return new_df
```

```
#Extracting the Data from json like columns
credit_cast = mapper(credits, 'cast', 'movie_id')
credit_crew = mapper(credits, 'crew', 'movie_id')
movie_genres = mapper(movies, 'genres', 'id')
movie_keywords = mapper(movies, 'keywords', 'id')
movie_production_companies = mapper(movies, 'production_companies', 'id')
movie_spoken_languages = mapper(movies, 'spoken_languages', 'id')
movie_production_countries = mapper(movies, 'production_countries', 'id')
```

```
#Renaming the columns for convinience
credit_cast.columns = ['cast_id', 'character', 'credit_id', 'gender', 'actor_id', 'actor_name', 'order','movie_id']
credit_crew.columns = ['credit_id', 'department', 'gender', 'job_id', 'job', 'crew_member', 'movie_id']
movie_genres.columns = ['genre_id','genre','movie_id']
movie_production_companies.columns = ['studio', 'studio_id', 'movie_id']
movie_spoken_languages.columns = ['iso_639_1', 'language', 'movie_id']
movie_production_countries.columns = ['iso_3166_1', 'country', 'movie_id']
```

Feature Engineering

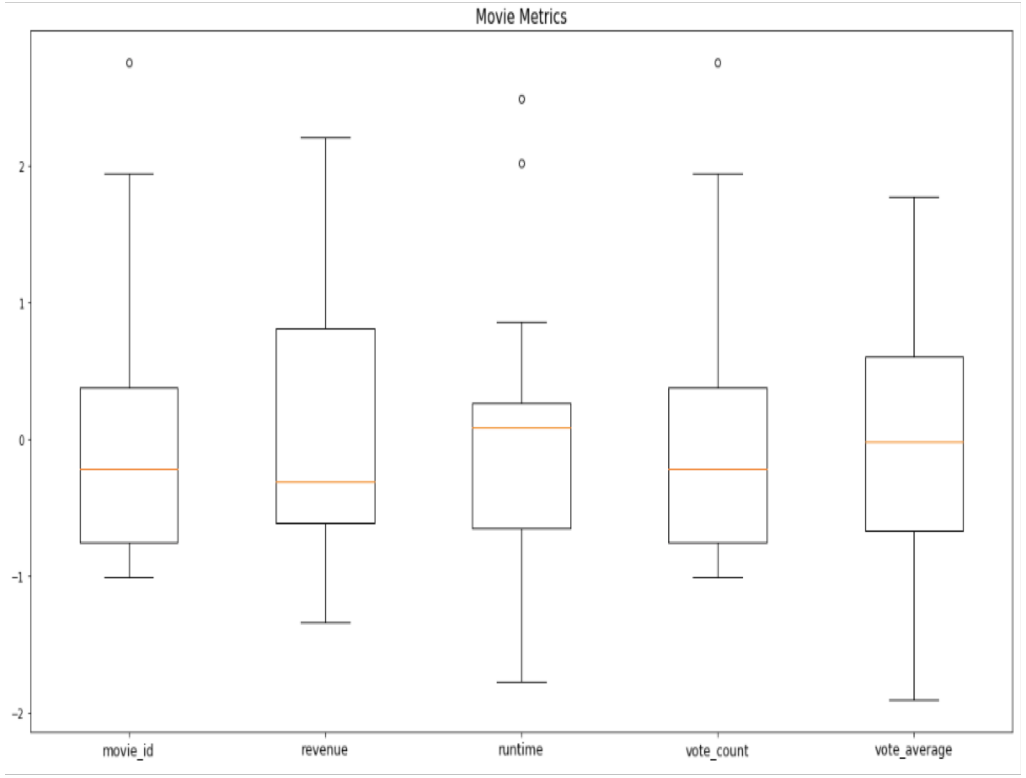
2) Creating new and more meaningful Features
   - Genre Based Weights attached to each entity: actor, crew, studio



Feature Engineering

Creating Genre Based weights for entities and scoring them.

```
#Creating a Weights dataframe to attach genre-wise weights to the movies, These weights are decided by various aspectes
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
metrics = movies[['id','popularity','revenue','runtime','vote_count','vote_average']]
metrics.columns = ['movie_id','popularity','revenue','runtime','vote_count','vote_average']
weights = pd.merge(movie_genres, metrics, on='movie_id',how='inner')
weights = weights.groupby('genre').agg({'movie_id':'count', 'revenue':'mean','runtime':'mean',
                        'vote_count':'count','vote_average':'mean'})
plt.figure(figsize=(20,10))
plt.boxplot(ss.fit_transform(weights))
plt.xticks([1,2,3,4,5],list(weights.columns), fontsize=13)
plt.title('Movie Metrics', fontsize=15)
plt.show()

weights = np.sum(ss.fit_transform(weights),axis=1)
```



Feature Engineering

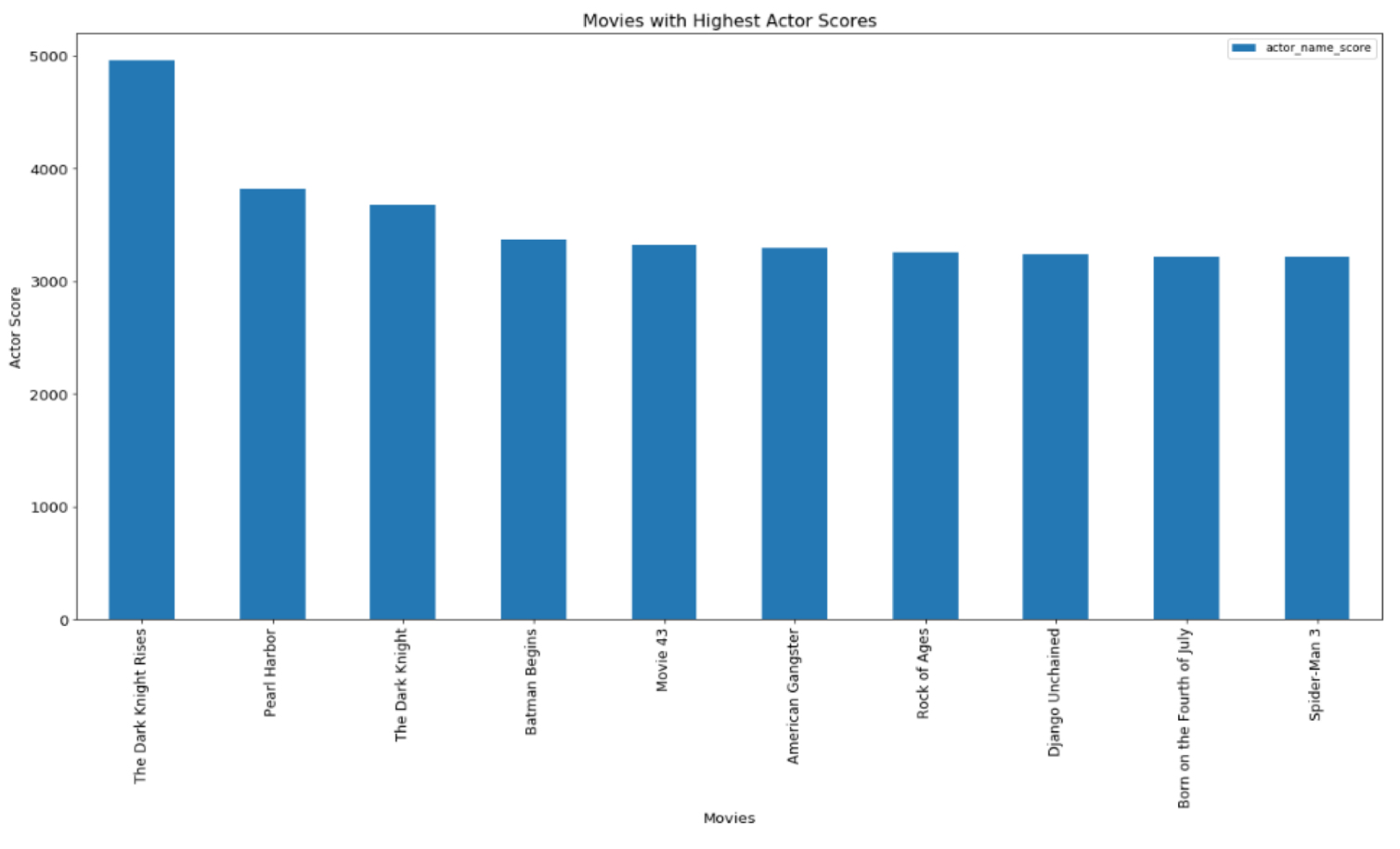
Converting Dates into categorical data, i.e., seasons

```
movies['release_month'] = movies.release_date.str.split('-').apply(lambda x:int(x[1]))
def season(row):
    if row >= 1 and row < 4:
        return 'winter'
    elif row >= 4 and row < 7:
        return 'spring'
    elif row >= 7 and row < 10:
        return 'summer'
    elif row >= 10 and row < 12:
        return 'autumn'
    else:
        return 'winter'

movies['season'] = movies.release_month.apply(season)
movies.columns
data = movies.drop(['genres', 'homepage', 'keywords', 'original_title', 'overview', 'production_companies',
                    'release_date', 'production_countries', 'spoken_languages', 'status', 'tagline',
                    'release_month'],axis=1)
data.columns = ['budget', 'movie_id', 'original_language', 'popularity', 'revenue', 'runtime', 'title', 'vote_average',
                'vote_count', 'season']
data = pd.merge(data, cast_score, on='movie_id', how='inner')
data = pd.merge(data, production_score, on='movie_id', how='inner')
data = pd.merge(data, crew_score, on='movie_id', how='inner')
data.head()
```

iginal_language	popularity	revenue	runtime	title	vote_average	vote_count	season	actor_name_score	studio_score	crew_member_score
en	150.437577	2787965087	162.0	Avatar	7.2	11800	winter	1329.677727	1562.806334	5512.893874
en	139.082615	961000000	169.0	Pirates of the Caribbean: At World's End	6.9	4500	spring	1793.666049	577.165604	2872.702440
en	107.376788	880674609	148.0	Spectre	6.3	4466	autumn	1400.459149	1074.022624	6200.243624
en	112.312950	1084939099	165.0	The Dark Knight Rises	7.6	9106	summer	4952.808625	2003.318157	9276.768018
en	43.926995	284139100	132.0	John Carter	6.1	2124	winter	1263.970055	448.560277	4357.590517

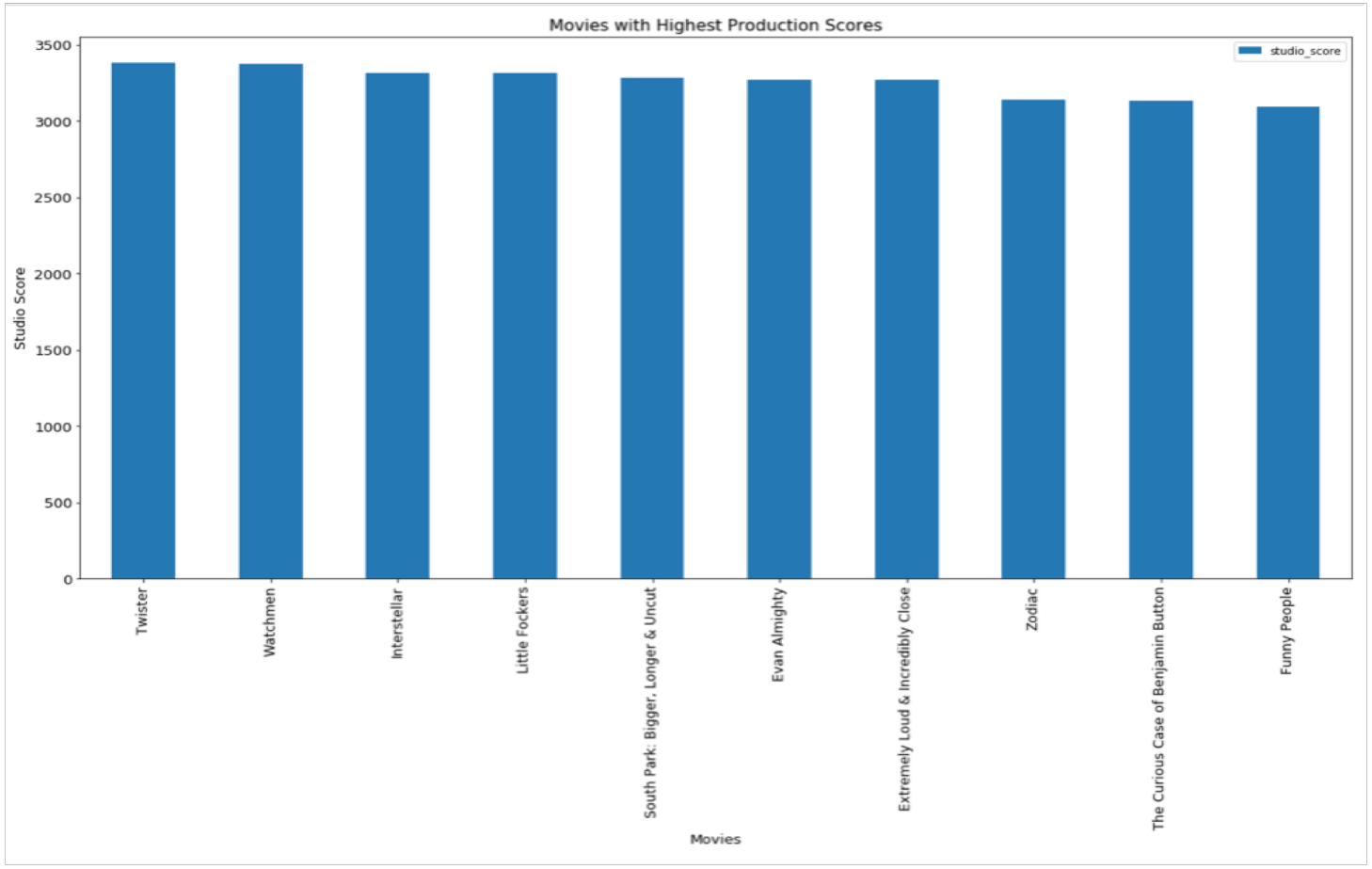
Feature Engineering

- Top Movies based on Actor scores are observed
- This score indicates the measure of an Actor in his craft



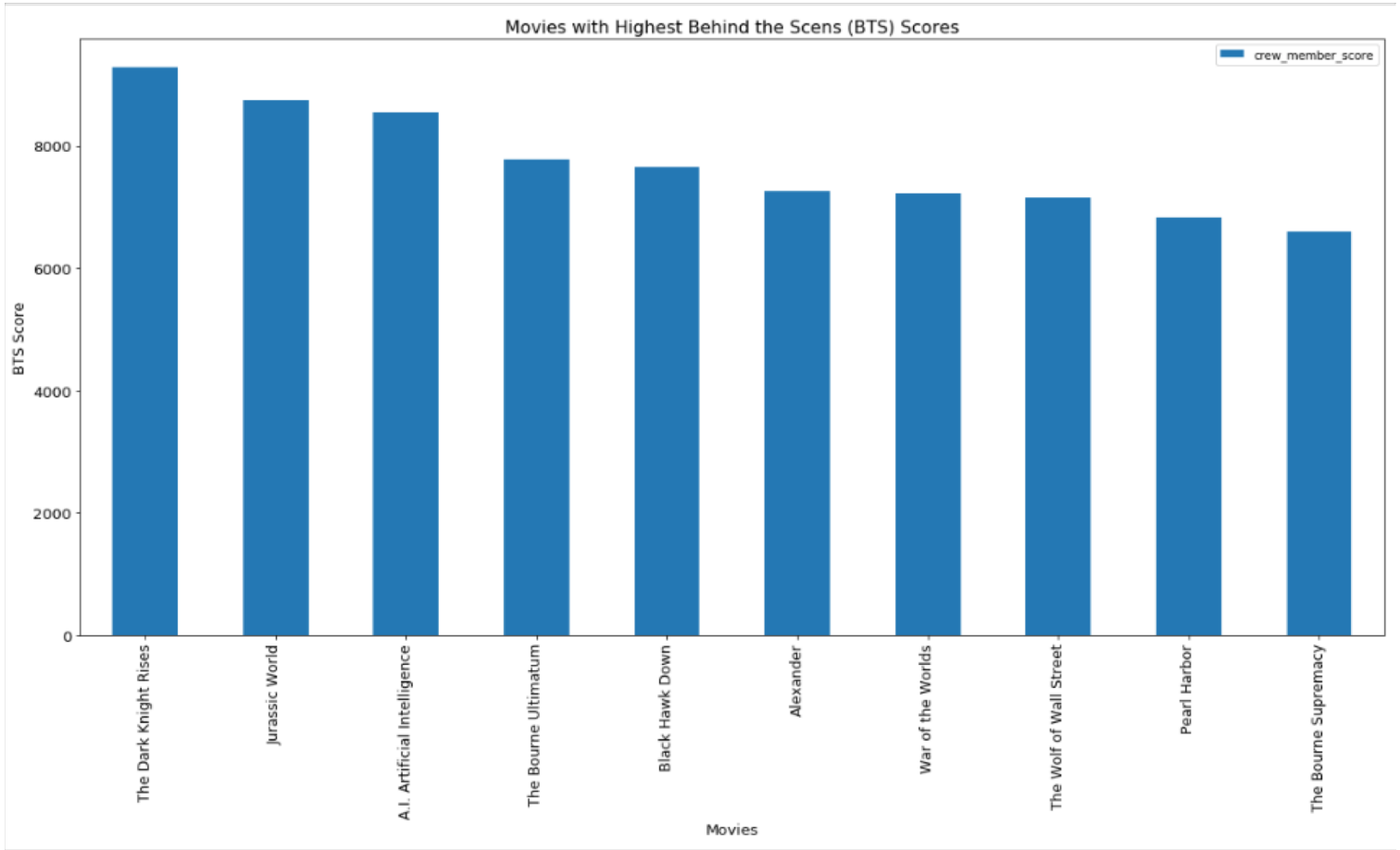
Feature Engineering

- Top Movies based on Studio scores are observed
- This score indicates the measure of a Studio's ability to produce a successful film



Feature Engineering

- Top Movies based on Crew scores are observed
- This score indicates the measure of a Crew's ability to creatively produce a film



Data Modelling

- Using GridSearchCV we find perform Hyper-Parameter tuning on random forest and Decision trees to identify the best parameter values
- Once Random Forest is determined as the best algorithm to model our data, we will used it to identify the main features for the Data

```
# Sort best_score_param_estimators in descending order of the best_score_
best_score_param_estimators = sorted(best_score_param_estimators, key=lambda x : x[0], reverse=True)

# For each [best_score_, best_params_, best_estimator_]
for best_score_param_estimator in best_score_param_estimators:
    # Print out [best_score_, best_params_, best_estimator_], where best_estimator_ is a pipeline
    # Since we only print out the type of classifier of the pipeline
    print([best_score_param_estimator[0], best_score_param_estimator[1], type(best_score_param_estimator[2].named_steps['cl
```

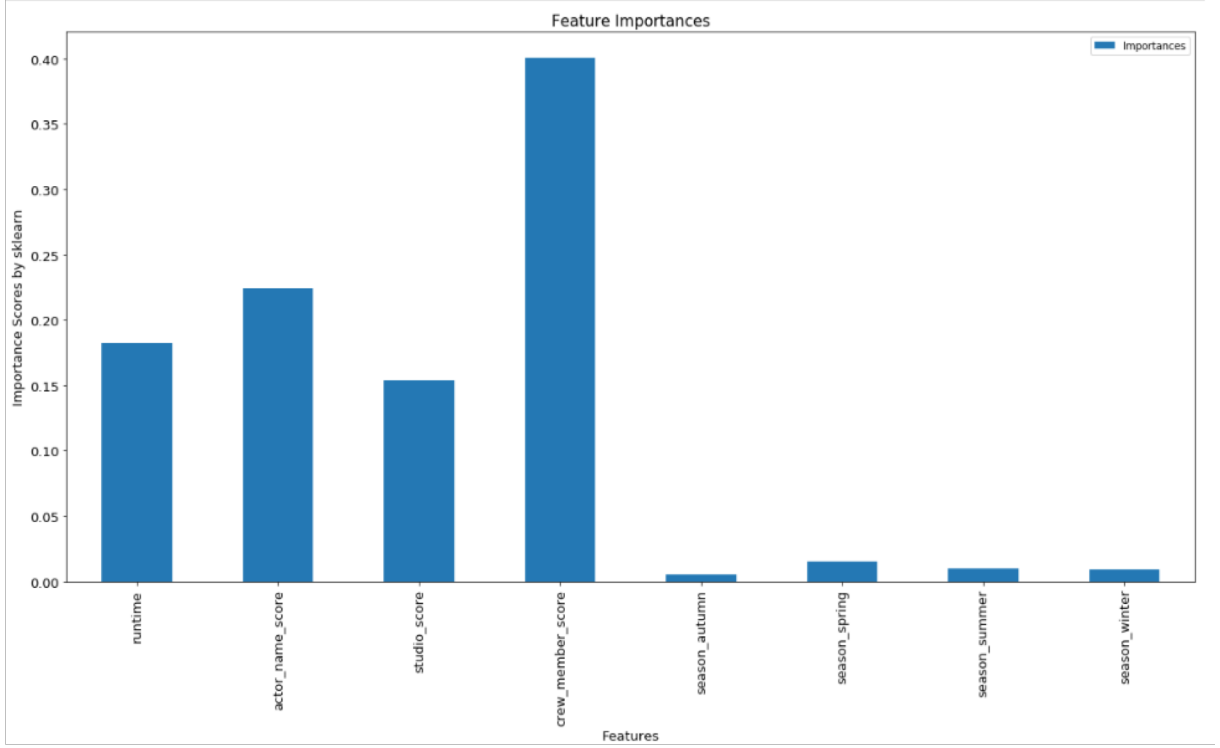
```
[0.8285223367697595, {'clf__min_samples_leaf': 10, 'clf__min_samples_split': 2, 'clf__n_estimators': 10}, <class 'sklearn.e
nsemble.forest.RandomForestClassifier'>]

[0.8175257731958763, {'clf__min_samples_leaf': 30, 'clf__min_samples_split': 2}, <class 'sklearn.tree.tree.DecisionTreeClas
sifier'>]
```

Conclusion

According to our findings,

- Behind the Scene score are most important, i.e., roles of Director, Producer, Set Producer, Costume Designer and etc.
- The second most important score in our analysis is the Actor Score
- The third most important score is the Studio Score.

```
eval_df = pd.DataFrame({'Feature':cols, 'Importances':rf.feature_importances_})
ax = eval_df.plot(kind='bar', figsize=(20,10), fontsize=13)
plt.title('Feature Importances', fontsize=15)
ax.set_xlabel('Features', fontsize=13)
ax.set_ylabel('Importance Scores by sklearn', fontsize=13)
plt.xticks(list(range(0,8)),cols)
plt.show()
```



Learning Process

- This opportunity has helped me learn and understand the following,
  - Python Programming Concepts
  - Machine Learning Algorithms: Random Forest and Decision Trees
  - Hyper Parameter tuning of Machine Learning Algorithms
  - Understanding of Entertainment Data

Thank you