

1) Target questions (assumption)

IMDb란 전 세계 최대 규모의 영화 사이트이며, IMDb의 점수는 대중성을 알기 수 있는 가장 객관적인 지표 중 하나라고 평가되고 있다. 이러한 IMDb에 데이터 사용 허가를 신청했고, 비 상업적인 데이터셋을 제공받을 수 있었다.

① 프로젝트 주제

영화 개봉 전, 영화의 장르, 상영시간, 배우 등의 영화 정보를 사용해서 IMDb 점수를 예측할 수 있을까?

② 활용 방안

최근 영화 값이 크게 올라 소비자들이 부담을 느끼고 있다. 영화가 개봉하기 전 대중성을 예측할 수 있다는 것은 관객들의 영화 관람 의사결정에 도움이 될 수 있을 것이다. 또한, 수십억대 제작비를 들인 국내 영화가 점점 흔하게 되어 흥행 실패에 대한 위험도 커지게 되었으며, 국내 영화뿐만 아니라 해외 영화들도 계속 끊임없이 영화를 제작하고 있다. 영화 제작 초기에 대중성을 예측한 것을 활용하여 리스크를 줄일 수 있을 것이다.

2) Data description

모든 데이터는 <https://developer.imdb.com/non-commercial-datasets/>에서 다운받았으며, 총 10246265개의 데이터를 포함하고 있다. 이 데이터는 gz 파일 형식으로 되어 있어 txt 형식으로 변환하여 살펴보았다.

- title.basics : 영화에 대한 기본적인 정보를 알려주는 데이터셋
- title.crew : 영화의 감독과 작가를 알려주는 데이터셋
- title.principals : 영화의 감독, 작가를 포함한 다른 역할들을 알려주는 데이터셋
- title.ratings : 영화의 투표수와 점수를 알려주는 데이터셋

위의 4가지 데이터셋에서 필요한 특성들만 선택하고 새로운 특성을 만들고 이를 합쳐 모델에 사용할 movies.csv를 만들었고, 전처리를 수행했다.

① 결측값 처리

startYear, runtimeMinutes, genres_num, first_director, first_writer, first_actor, first_actress 열에서 결측값이 있다는 것을 확인했다. startYear, runtimeMinutes는 평균값으로 대체했고, 나머지 열은 결측값이 있는 행을 삭제했다.

② 정수로 표현된 문자열을 정수형으로 변환

startYear, runtimeMinutes, first_director, first_writer, first_actor, first_actress 열은 object로 표현되어 있었다. 이를 astype(int)를 사용해 정수형으로 변환했다.

아래는 모델에 사용할 최종적인 movies 데이터의 모습이다.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 227877 entries, 0 to 349216
Data columns (total 19 columns):
```

#	Column	Non-Null Count	Dtype
0	tconst	227877 non-null	object
1	titleType	227877 non-null	int64
2	isAdult	227877 non-null	int64
3	startYear	227877 non-null	int32
4	runtimeMinutes	227877 non-null	int32
5	genres_num	227877 non-null	int64
6	genres_first_encoded	227877 non-null	int64
7	first_director	227877 non-null	int32
8	first_writer	227877 non-null	int32
9	directors_count	227877 non-null	int64
10	writers_count	227877 non-null	int64
11	actor_count	227877 non-null	int64
12	actress_count	227877 non-null	int64
13	crew_total	227877 non-null	int64
14	first_actor	227877 non-null	int32
15	first_actress	227877 non-null	int32
16	averageRating	227877 non-null	float64
17	numVotes	227877 non-null	int64
18	score	227877 non-null	int64

dtypes: float64(1), int32(6), int64(11), object(1)

	tconst	titleType	isAdult	startYear	runtimeMinutes	genres_num	genres_first_encoded	first_director	first_writer
0	tt0000009	0	0	1894	45	1	0	85156	85156
1	tt0000574	0	0	1906	70	3	1	846879	846879
2	tt0000591	0	0	1907	90	1	2	141150	141150
3	tt0000615	0	0	1907	1995	1	2	533958	92809
4	tt0000941	0	0	1909	45	1	2	63413	63413

directors_count	writers_count	actor_count	actress_count	crew_total	first_actor	first_actress	averageRating	numVotes	score
1	1	2	1	4	183823	63086	5.3	207	1
1	1	3	1	10	846894	846887	6.0	853	0
1	1	2	2	5	906197	1323543	5.0	21	1
1	2	5	1	9	3071427	218953	4.3	25	1
2	3	3	1	7	34453	294022	4.5	27	1

- tconst : 제목의 알파벳과 숫자로 이루어진 고유식별자
- titleType : 형식 또는 유형 (movie는 0, tvMovie는 1)
- isAdult : 성인용 콘텐츠 여부 (non-adult title은 0, adult title은 1)
- startYear (YYYY) : 발매 연도
- runtimeMinutes : 상영 시간 (분 단위)
- genres_num : 장르 개수 (최대 세 가지 장르를 포함한 문자열 배열인 title.basics 데이터셋의 genres 열을 이용하여 만든 장르의 개수)
- genres_first_encoded : 첫 번째 장르를 인코딩한 값 (title.basics 데이터셋의 genres 열을 이용하여 만든 첫 번째 장르를 숫자로 변환한 값)
- first_director : 첫 번째 감독의 식별자 (감독(들)을 나타내는 고유식별자 배열인 title.crew 데이터셋의 directors 열에서 가장 맨 앞의 식별자를 정수형으로 변환한 값)
- first_writer : 첫 번째 작가의 식별자 (작가(들)을 나타내는 고유식별자 배열인 title.crew 데이터셋의 writers 열에서 가장 맨 앞의 식별자를 정수형으로 변환한 값)
- directors_count : 감독의 수 (title.crew 데이터셋의 directors 열에서 고유식별자의 개수를 계산한 값)
- writers_count : 작가의 수 (title.crew 데이터셋의 writers 열에서 고유식별자의 개수를 계산한 값)

- actor_count : 남자 주연 배우의 수 (title_principals 데이터셋의 수행한 직업, 역할인 category 열을 이용하여 category가 actor일 때의 개수의 합)
- actress_count : 여자 주연 배우의 수 (title_principals 데이터셋의 category 열을 이용하여 category가 actress일 때의 개수의 합)
- crew_total : 스태프들의 수 ((title_principals 데이터셋의 category 열을 이용하여 category의 총 개수의 합)
- first_actor : 첫 번째 남자 주연 배우의 식별자 (title_principals 데이터셋에서 category가 actor일 때의 인물의 고유식별자인 nconst 열에서 제일 먼저 나온 식별자를 정수형으로 변환한 값)
- first_actress : 첫 번째 여자 주연 배우의 식별자 title_principals 데이터셋에서 category가 actress일 때의 nconst 열에서 제일 먼저 나온 식별자를 정수형으로 변환한 값)
- averageRating : 사용자 평점의 평균
- numVotes : 받은 투표(평점)의 수
- score : IMDb 점수 (averageRating을 정확히 예측하기는 어렵기 때문에 간을 나누어 종속변수로 사용할 score 특성을 만들었다)

averageRating	score	
1~3	2	Flop Movie
3~6	1	Average Movie
6~10	0	Hit Movie

3) Method & Result

① 분석 방법

높은 정확도의 예측을 위해서 다양한 학습 모델들을 선정하여 분석하기 위해 노력했다. 먼저 Sickit-Learn의 train_test_split을 사용하여 train 세트 30%, test 세트 30%로 분할하였고, 학습을 위해 사용할 모델로는 Logistic Regression, RandomForest, AdaBoostClassifier, BaggingClassifier, GradientBoostingClassifier, XGBClassifier를 선택하였다. 그리고 각 모델들의 최고 성능을 낼 수 있는 매개변수를 알아내기 위해 GridSearchCV를 이용하였고, 각 모델들의 성능 비교를 위해 정확도인 Accuracy score를 사용하였다.

```
[18] params = {
    'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
    'learning_rate': [0.0001, 0.001, 0.01, 0.1, 1.0]
}

adaboost = AdaBoostClassifier(random_state=10)
grid_adaboost = GridSearchCV(adaboost, param_grid=params, cv=2, scoring='accuracy')

best_model = grid_adaboost.fit(X_train, y_train)
```

```
[19] # 테스트 세트 점수
test_score = grid_adaboost.score(X_test, y_test)

print("테스트 세트 점수: {:.2f}".format(test_score))
print("최적 매개변수: {}".format(grid_adaboost.best_params_))
print("최고 교차 검증 점수: {:.2f}".format(grid_adaboost.best_score_))

테스트 세트 점수: 0.62
최적 매개변수: {'learning_rate': 1.0, 'n_estimators': 100}
최고 교차 검증 점수: 0.62
```

```
[20] #위의 결과로 나온 최적 하이퍼 파라미터로 다시 모델을 학습하여 테스트 세트 데이터에서 예측 성능을 측정
adaboost = AdaBoostClassifier(learning_rate = 1.0, n_estimators = 90, random_state=10)
adaboost.fit(X_train,y_train)
pred = adaboost.predict(X_test)
accuracy = accuracy_score(y_test,pred)
accuracy
```

데이터 분석의 예 (AdaBoost)

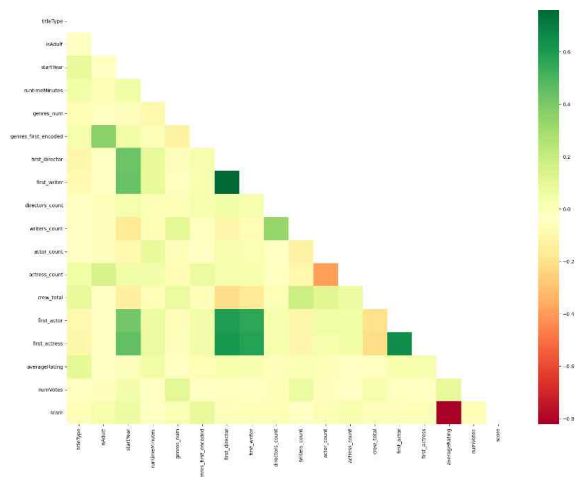
	Logistic Regression	Random Forest	AdaBoost
test 세트 정확도	0.56	0.64	0.62
최고 교차검증 점수	0.57	0.64	0.62
최적 매개변수	C= 0.1, penalty= 'l2', solver= 'liblinear'	max_depth= 12, min_samples_leaf=8, min_samples_split=20 n_estimators= 100	learning_rate = 0.1, n_estimators = 100

	BaggingClassifier	GradientBoosting Classifier	XGBClassifier
test 세트 정확도	0.64	0.63	0.67
최고 교차검증 점수	0.64	0.64	0.67
최적 매개변수	max_depth=10, min_samples_leaf=5, min_samples_split=5, max_features=0.5, max_samples=0.7, n_estimators= 20	learning_rate=0.01, max_depth=5, max_features=sqrt, criterion=mae, subsample=0.8, n_estimators=10	max_depth=5, min_child_weight=3, gamma=0, nthread=4, colsample_bytree=0.5

test 세트를 이용하여 정확도를 살펴보았을 때 6가지의 모델들 중 XGBClassifier가 정확도 67%의 가장 좋은 성능을 가진다는 것을 확인할 수 있었다.

② 분석 결과

예상과 달리 좋지 못한 65% 정도의 정확도를 보이는 것을 확인했다. 특성들 간의 상관관계가 두드러지게 보이지 않는 것이 가장 큰 이유라고 생각한다.



또한 score가 0인 Hit Movie와 2인 Flop Movie 간의 특성들을 평균(mean)과 중간값(median)으로 비교해본 결과 특성들 간에 큰 차이가 두드러지지 않는 것을 확인했으며, 이처럼 큰 차이가 없는 것도 낮은 정확도의 이유라고 생각한다.

1	flopMovie.median()
titleType	0.0
isAdult	0.0
startYear	2011.0
runtimeMinutes	90.0
genres_num	2.0
genres_first_encoded	7.0
first_director	1448482.0
first_writer	1784327.0
directors_count	1.0
writers_count	1.0
actor_count	3.0
actress_count	2.0
crew_total	10.0
first_actor	1357282.0
first_actress	1983032.0
averageRating	2.5
numVotes	163.0
score	2.0
dtype:	float64

1	flopMovie.mean().round(3)
titleType	0.071
isAdult	0.006
startYear	2003.766
runtimeMinutes	287.162
genres_num	1.799
genres_first_encoded	6.052
first_director	2597682.814
first_writer	3018091.002
directors_count	1.138
writers_count	1.697
actor_count	2.706
actress_count	1.888
crew_total	9.222
first_actor	2819802.810
first_actress	3319843.620
averageRating	2.391
numVotes	985.320
score	2.000
dtype:	float64

1	hitMovie.median()
titleType	0.0
isAdult	0.0
startYear	2000.0
runtimeMinutes	96.0
genres_num	2.0
genres_first_encoded	2.0
first_director	735305.0
first_writer	766499.0
directors_count	1.0
writers_count	2.0
actor_count	3.0
actress_count	2.0
crew_total	10.0
first_actor	681975.0
first_actress	765531.5
averageRating	6.8
numVotes	73.0
score	0.0
dtype:	float64


1	hitMovie.mean().round(3)
titleType	0.146
isAdult	0.006
startYear	1991.373
runtimeMinutes	253.274
genres_num	1.796
genres_first_encoded	4.646
first_director	1705807.000
first_writer	2013115.057
directors_count	1.100
writers_count	1.991
actor_count	2.554
actress_count	1.778
crew_total	9.564
first_actor	1872841.831
first_actress	2068449.721
averageRating	6.926
numVotes	6090.032
score	0.000
dtype:	float64

만약 영화의 예산, 예고편 길이, 제작사, 국가, 등급 등과 같은 특성들이 추가로 더 있었다면 더 좋은 예측을 할 수 있었을 것이다.


③ 분석 결과 활용

좋지 않은 정확도를 가졌지만 성능이 가장 좋았던 XGBClassifier 모델을 사용하여 개봉 예정인 영화들의 IMDb 점수를 예측해 보았다.

Dec 20, 2023



Kuolleet lehdet (2023)
Comedy · Drama
Alma Pöysti · Jussi Vatanen · Martti Suosalo · Alina Tomnikov



Noryang (2023)
Action · Biography · History
Yeo Jin-goo · Kim Yoon-seok · Jeong Jae-yeong · Ahn Seong-bong

Dec 27, 2023



Das Lehrerzimmer (2023)
Drama
Leonie Benesch · Leonard Stettnisch · Eva Löbau · Michael Klammer



20.000 especies de abejas (2023)
Drama
Sofía Otero · Patricia López Arnaiz · Ane Gabarain · Itziar Lazkano

```
1 release = pd.read_csv('release.csv')
2 release.head()

   tconst  titleType  isAdult  startYear  runtimeMinutes  genres_num  genres_first_encoded  first_director  first_writer  directors_count  writers_count  actors
0  tt21027780      0      0      2023           81         2              7         442454         442454              1              1          1
1  tt22813112      0      0      2023           90         3              3         3482943             0              1              1          1
2  tt26612950      0      0      2023           98         1              2         1921714         1672317              1              2          2
3  tt21113962      0      0      2023          128         1              4         5031184         5031184              1              1          1

1 pred = xgb.predict(release.iloc[:, 1:16])
2 pred

array([2, 0, 0, 0], dtype=int64)
```

‘Kuolleet lehdet’ 이외에 다른 세 영화들은 IMDb 점수가 좋을 것으로 예측하고 있다. 만약 12월 20일 이후에 영화를 보러가게 된다면 저 3개의 영화들을 고려해보면 좋을 것이다.

4) Troubleshooting & impression

사실 사용할 모델들 중 SVM 또한 선택했었지만 이해할 수 없는 런타임 오류가 계속 발생해 사용할 수 없게 되었고 이 과정에서 시간이 많이 소요되었다. BaggingClassifier와 GradientBoostingClassifier, XGBClassifier 모델을 GridSearchCV를 통해 파라미터를 구하고 모델을 훈련시키는 과정에서 또한 시간이 많이 소요되었다. 그리고 이 데이터에서는 감독과 작가, 배우들의 이름을 표현할 때 고유 식별자를 사용하고 있어 정확하게 누구인지 알 수 없다는 점과 영화마다 감독, 배우, 작가 이외에 역할들의 사용하는 이름이 달라 정확하게 인

원을 계산하고 새로운 특성을 만들 수 없는 것이 아쉬웠다.

마지막으로 프로젝트 이전까지는 깨끗한 데이터셋만을 이용해서 분석을 시도했었지만, 이번 프로젝트를 통해서 정리가 되어 있지 않은 대용량의 데이터셋을 전처리하고 예측하면서 이상치가 얼마나 데이터 분석에 있어서 중요하고 모델에 큰 영향을 끼친다는 것을 깨달았다. 원래의 목표처럼 IMDb에서 제공하는 API를 이용해서 형태가 일정하고 깨끗한 데이터셋을 만들었다면 더 좋은 성능을 가진 모델을 만들 수 있었을 것 같아 아쉽다.