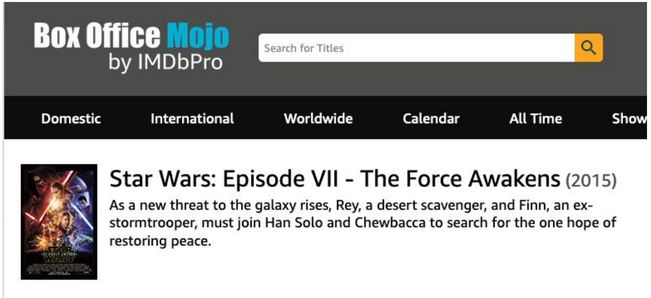
Domestic lifetime gross prediction for PG-13 movies

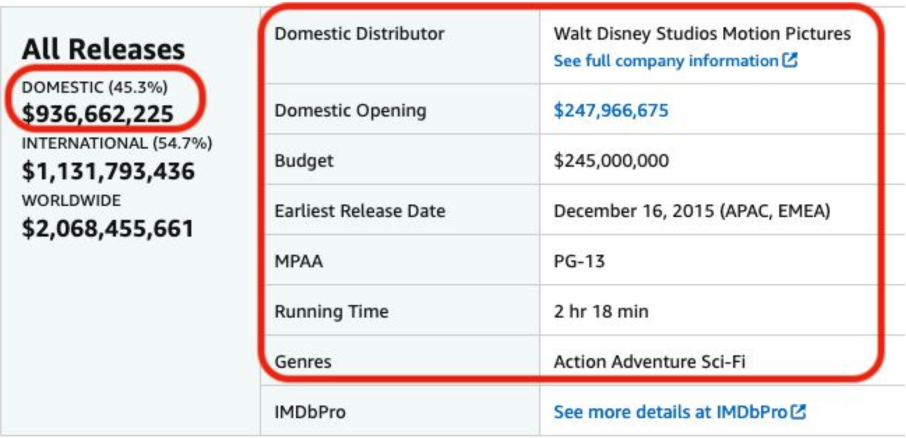
-- Xufei Li

Background story

Our client is a movie investing firm who is going to invest in several PG13 movies in the coming year. They want us to help them predict the domestic lifetime gross for the movies that they are interested in, they want to know which movies can bring them the highest return.

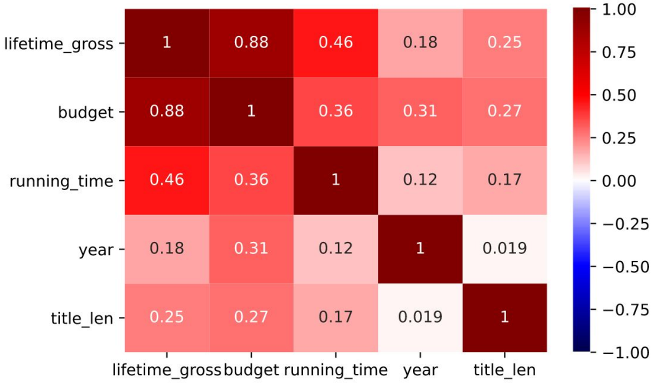
Data Collection- Web Scraping

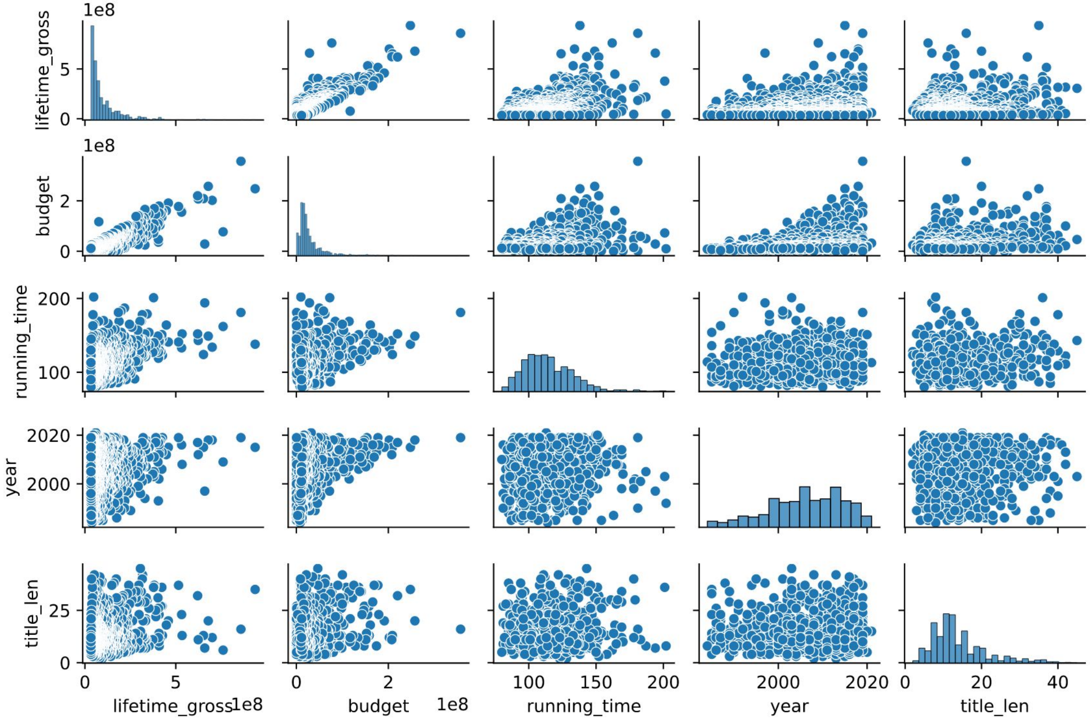




Total: 1000 data points

Data Cleaning & EDA(Numerical)





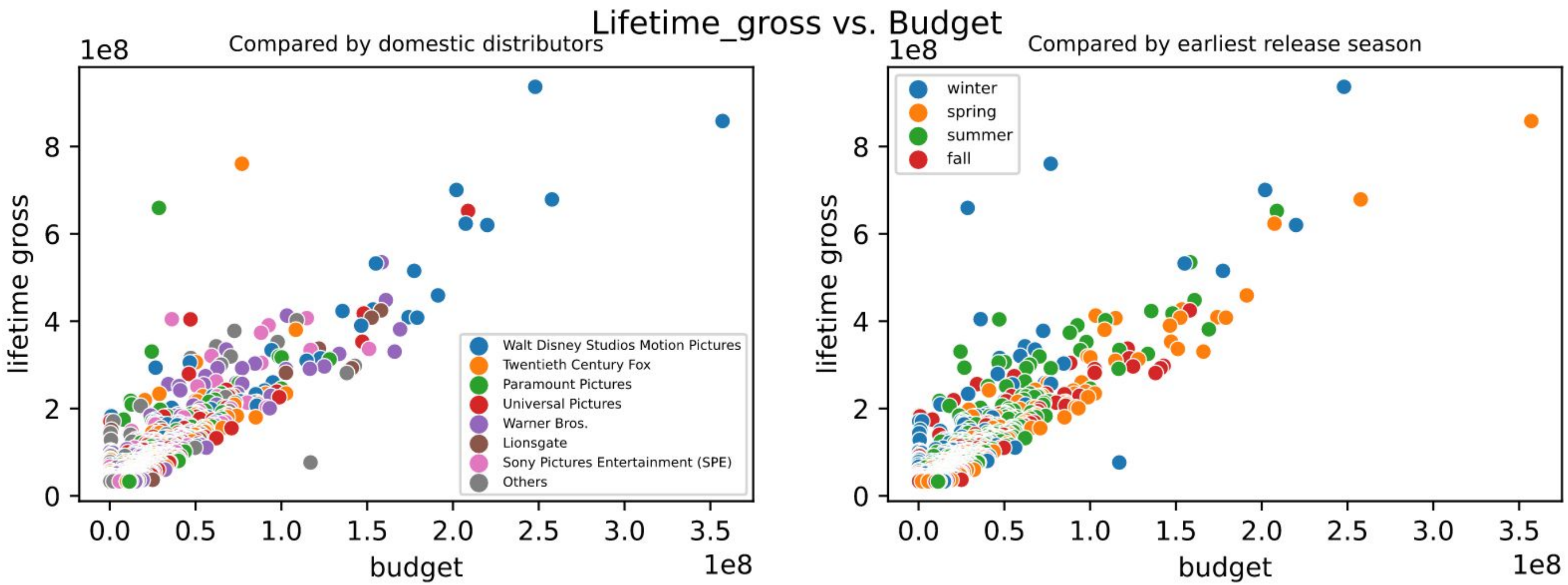
Q: Why 'title_len' ?

Target: lifetime_gross (right skewed)

Methodology:

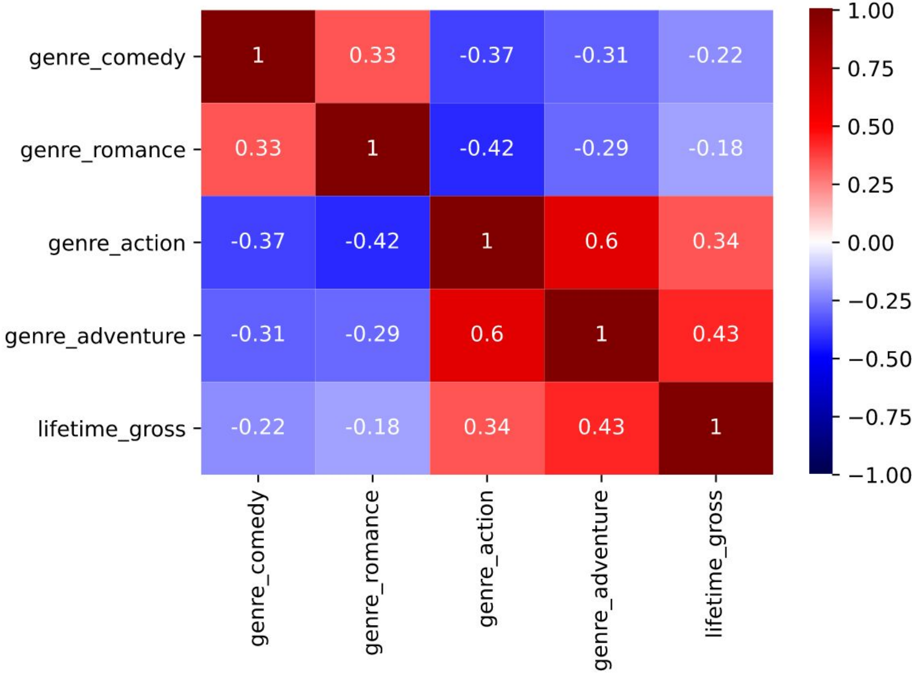
- **Log Transformation**
- **Removing outliers (968 observations)**

EDA(Categorical)



Graph1 : No pattern
Graph2 : Summer & Winter are higher

EDA(Categorical-continue)



Genres:
Adventure - Not significant(why?)
Action - Not significant (why?)
Comedy - Significant
Romance - Not significant

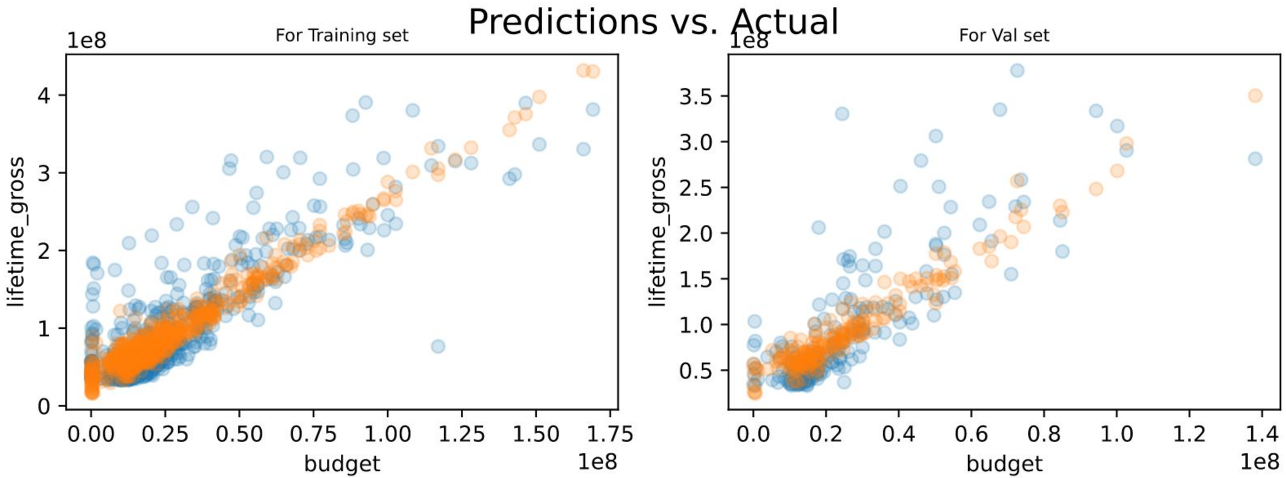
Modeling - Base model (5 folds CV)

Baseline Linear Regression Model:

R2: 0.7328

Features: 'budget', 'running_time', 'year'

Target: ‘lifetime_gross’



Feature Engineering

New model with R2: 0.7481 (+1.53%)

Feature: 'budget', 'running_time', 'year', 'genre_comedy','earliest_release_season_spring',

'earliest_release_season_summer', 'earliest_release_season_winter'

Target: 'lifetime_gross'

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.954e+09	2.88e+08	6.783	0.000	1.39e+09	2.52e+09
budget	2.4319	0.051	47.685	0.000	2.332	2.532
running_time	7.353e+05	7.06e+04	10.419	0.000	5.97e+05	8.74e+05
year	-1.004e+06	1.43e+05	-7.001	0.000	-1.29e+06	-7.23e+05
genre_comedy	5.251e+06	2.54e+06	2.064	0.039	2.59e+05	1.02e+07
earliest_release_season_spring	-6.374e+06	3.38e+06	-1.888	0.059	-1.3e+07	2.53e+05
earliest_release_season_summer	8.626e+06	3.27e+06	2.640	0.008	2.21e+06	1.5e+07
earliest_release_season_winter	1.138e+07	3.37e+06	3.375	0.001	4.76e+06	1.8e+07

Omnibus:	327.143	**Durbin-Watson:**	1.515
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2323.327
Skew:	1.358	**Prob(JB):**	0.00
Kurtosis:	10.087	**Cond. No.**	9.41e+09

Candidate models (5 folds CV)

Baseline model: 0.7328

Model	Val score	RMSE	MAE
1. Simple linear regression	0.748 +- 0.049	41,538,596	25,389,385
2. +Polynomial (degree 2)	0.764 +- 0.043	39,440,843	24,326,173
3. +LassoCV with Polynomial(degree 2)	0.764 +- 0.041	39,049,303	24,228,014
4. Random Forest	0.770 +- 0.048	39,064,999	23,901,101

Selected Model: Random Forest

Retrain model (train & validation) → Test score: 0.7764

Final model & Conclusion

Train RF model on Entire dataset (1000 observations)

(RF - good at dealing with missing values & Outliers)

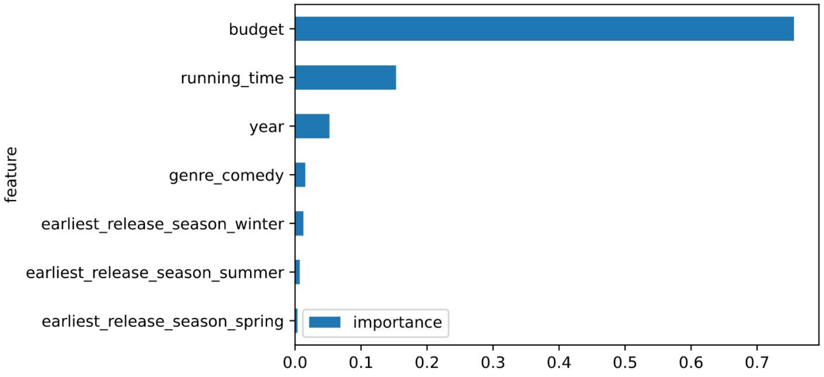
Training score: 0.8463

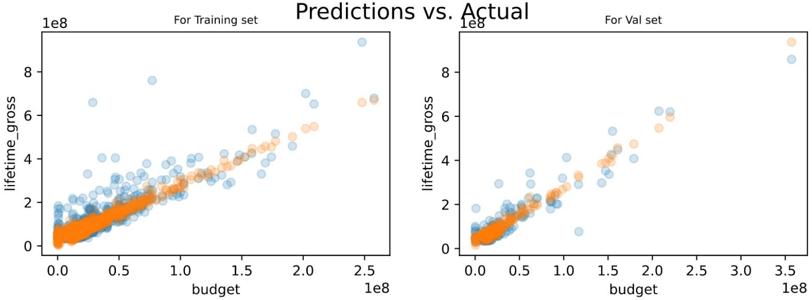
Validation Score: 0.8393 (base+10.65%)

Test Score: **0.8393**

```
rf.feature_importances_
```

```
array([0.75584413, 0.15299028, 0.05215988, 0.01550407, 0.0036398 ,
       0.007141  , 0.01272083])
```





Movies	Prediction	Actual
‘The River Wild’	48,979,015	46,816,343
‘Norbit’	109,733,112	95,673,607

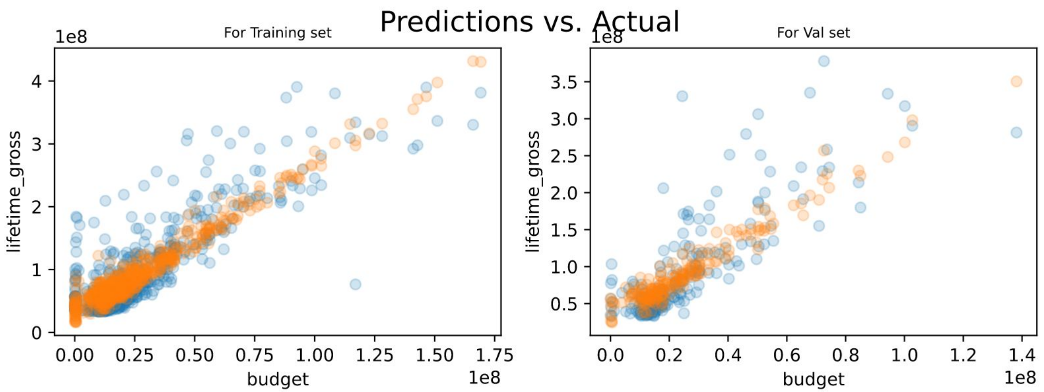
MAE for validation: 24,873,153

Future work

1. Getting more data to improve our model:
   directors, writers, etc...

2. Linear Model:

   Ordinary Least Square vs Weight Least Square



Appendix 1 -
Tuning hyper-parameter for Random Forest

```
#Grid Search with Cross Validation
from sklearn.model_selection import GridSearchCV# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
# Create a based model
rf = RandomForestRegressor()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                          cv = 3, n_jobs = -1, verbose = 2)
# Fit the grid search to the data
grid_search.fit(X, y)
```

```
Fitting 3 folds for each of 288 candidates, totalling 864 fits

GridSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=-1,
             param_grid={'bootstrap': [True], 'max_depth': [80, 90, 100, 110],
                         'max_features': [2, 3], 'min_samples_leaf': [3, 4, 5],
                         'min_samples_split': [8, 10, 12],
                         'n_estimators': [100, 200, 300, 1000]},
             verbose=2)
```

```
grid_search.best_params_
```

```
{'bootstrap': True,
 'max_depth': 110,
 'max_features': 3,
 'min_samples_leaf': 3,
 'min_samples_split': 8,
 'n_estimators': 1000}
```

Appendix 2:
LassoCV best alpha & Coefficient

```
lm_lasso.alpha_
```

```
1.552225357427048
```

```
lm_lasso.coef_
```

```
array([        0.        ,  64354616.09868296,  18202987.78807636,
       -18285172.71546404,   3159226.69210198,  -5795874.33500904,
        -2207334.42675896,   2601987.91235384, -19525676.88308844,
        21580001.36912451,  -7865004.79409379,   -333390.20366841,
         2447389.28738019,  10158093.32193002,  -2643302.69607428,
          905896.90672173, -11115756.42693536, -15937789.46516543,
       -23005907.15027149,   5935353.81863886,  24336698.22583675,
         8535029.87492859,  16689538.06061841,  30527042.53809172,
       -10688674.27126923, -16669995.1843216 ,  -1658338.97828754,
         -376503.0711751 ,    153656.30873463,    918162.59020638,
        -6528270.45307485,         0.        ,         0.        ,
         1157136.68346927,         0.        ,  -3780732.30448808])
```