



Regression

Internet Movie Database

IMDb

9 Dec 2021

Rahaf Alessa

Dalya Alanazi

*Dr.*Mejdal Alqahtani



Introduction

Here in this project we will predict the gross of movies from IMDB website (Internet Movie Database)

and we will apply the ML models to reach to the best result and best accuracy.

we will start with:

- DataStructure
- DataCleaning
- DataAnalysis
- Experimental ML Models
- Final Predict
- Conclusion



01- DataStructure

Here is the top 5 of the dataset before the cleaning .

5043 rows and 28 columns

df.head()

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_like
0	Color	James Cameron	723.0	178.0	0.0	855
1	Color	Gore Verbinski	302.0	169.0	563.0	1000
2	Color	Sam Mendes	602.0	148.0	0.0	161
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000
4	NaN	Doug Walker	NaN	NaN	131.0	Na

5 rows x 28 columns

```
#show number of rows and columns
print("DataSet dimension      :",df.shape)
```

DataSet dimension : (5043, 28)

```
#show information
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 28 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   color                                5024 non-null   object
1   director_name                        4939 non-null   object
2   num_critic_for_reviews               4993 non-null   float64
3   duration                             5028 non-null   float64
4   director_facebook_likes              4939 non-null   float64
5   actor_3_facebook_likes               5020 non-null   float64
6   actor_2_name                         5030 non-null   object
7   actor_1_facebook_likes               5036 non-null   float64
8   gross                               4159 non-null   float64
9   genres                              5043 non-null   object
10  actor_1_name                         5036 non-null   object
11  movie_title                          5043 non-null   object
12  num_voted_users                      5043 non-null   int64
13  cast_total_facebook_likes            5043 non-null   int64
14  actor_3_name                         5020 non-null   object
15  facenumber_in_poster                 5030 non-null   float64
16  plot_keywords                        4890 non-null   object
17  movie_imdb_link                      5043 non-null   object
18  num_user_for_reviews                 5022 non-null   float64
19  language                             5031 non-null   object
20  country                             5038 non-null   object
21  content_rating                       4740 non-null   object
22  budget                               4551 non-null   float64
23  title_year                           4935 non-null   float64
24  actor_2_facebook_likes               5030 non-null   float64
25  imdb_score                           5043 non-null   float64
26  aspect_ratio                         4714 non-null   float64
27  movie_facebook_likes                 5043 non-null   int64
dtypes: float64(13), int64(3), object(12)
memory usage: 1.1+ MB
```

```
#know the type of data
df.dtypes

color                                object
director_name                        object
num_critic_for_reviews               float64
duration                             float64
director_facebook_likes              float64
actor_3_facebook_likes               float64
actor_2_name                         object
actor_1_facebook_likes               float64
gross                               float64
genres                              object
actor_1_name                         object
movie_title                          object
num_voted_users                      int64
cast_total_facebook_likes            int64
actor_3_name                         object
facenumber_in_poster                 float64
plot_keywords                        object
movie_imdb_link                      object
num_user_for_reviews                 float64
language                             object
country                             object
content_rating                       object
budget                               float64
title_year                           float64
actor_2_facebook_likes               float64
imdb_score                           float64
aspect_ratio                         float64
movie_facebook_likes                 int64
dtype: object
```



02- DataCleaning

- We found missing values

- delete unnecessary columns and rows

- delete outliers

- after cleaning 3653 rows and 21 columns

```
[15]: df.shape  
[15]: (3653, 21)
```

```
#show all null value in columns  
df.isnull().sum()  
  
color 19  
director_name 104  
num_critic_for_reviews 50  
duration 15  
director_facebook_likes 104  
actor_3_facebook_likes 23  
actor_2_name 13  
actor_1_facebook_likes 7  
gross 884  
genres 0  
actor_1_name 7  
movie_title 0  
num_voted_users 0  
cast_total_facebook_likes 0  
actor_3_name 23  
facenumber_in_poster 13  
plot_keywords 153  
movie_imdb_link 0  
num_user_for_reviews 21  
language 12  
country 5  
content_rating 303  
budget 492  
title_year 108  
actor_2_facebook_likes 13  
imdb_score 0  
aspect_ratio 329  
movie_facebook_likes 0  
dtype: int64
```

```
df['gross'][df['gross']< 6.5] = None  
df.dropna(subset = ["gross"], inplace=True)
```

```
df['num_voted_users'][df['num_voted_users']< 4.4] = None  
df.dropna(subset = ["num_voted_users"], inplace=True)
```

```
#drop actor_1_facebook_likes column  
df.drop(['actor_1_facebook_likes'], axis = 1, inplace = True)
```

```
#drop num_user_for_reviews column  
df.drop(['num_user_for_reviews'], axis = 1, inplace = True)
```

```
#drop movie_facebook_likes column  
df.drop(['movie_facebook_likes'], axis = 1, inplace = True)
```

```
#drop num_critic_for_reviews column  
df.drop(['num_critic_for_reviews'], axis = 1, inplace = True)
```

```
#drop actor_2_facebook_likes column  
df.drop(['actor_2_facebook_likes'], axis = 1, inplace = True)
```

```
#drop actor_3_facebook_likes column  
df.drop(['actor_3_facebook_likes'], axis = 1, inplace = True)
```

```
#drop imdb_score column  
df.drop(['imdb_score'], axis = 1, inplace = True)
```

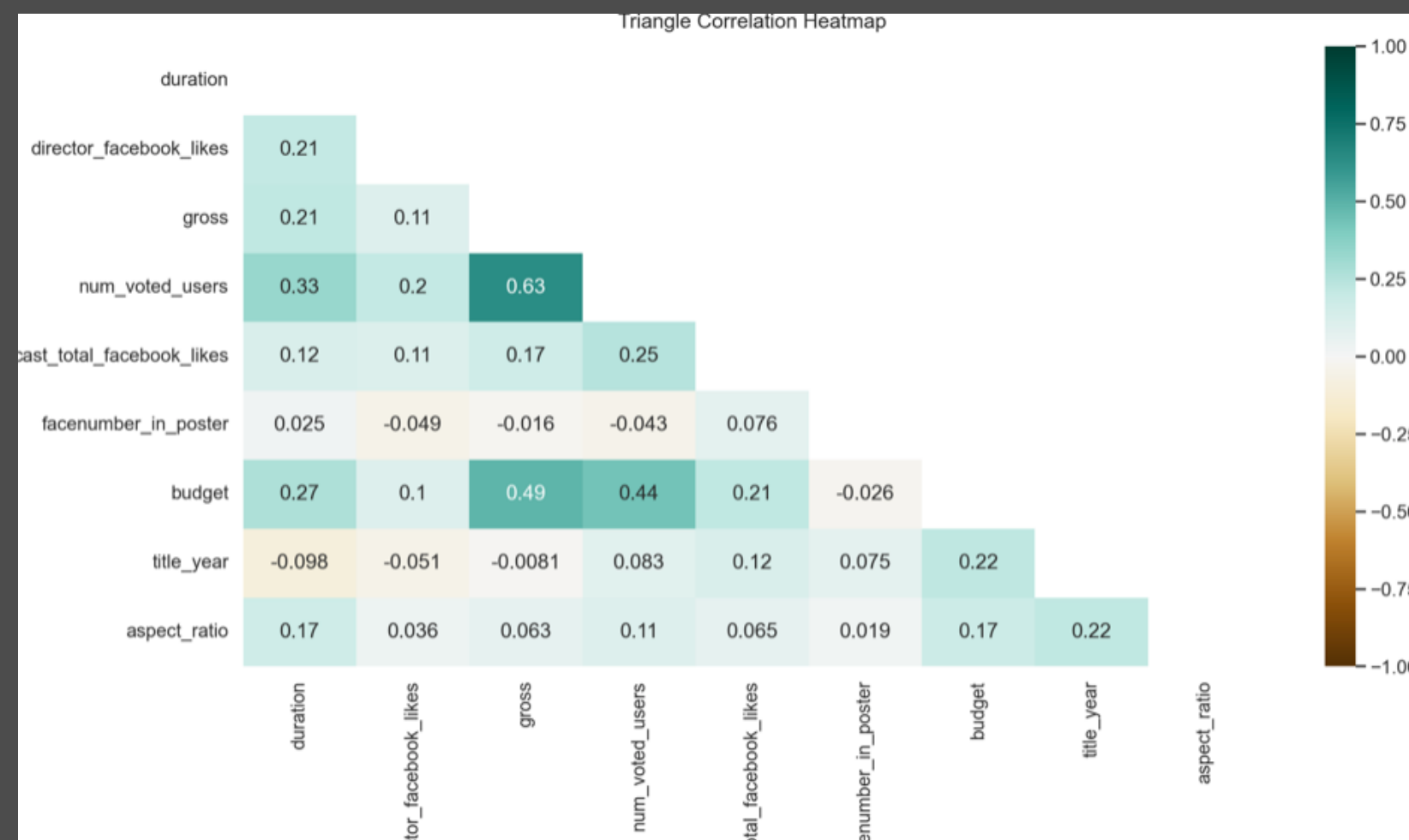
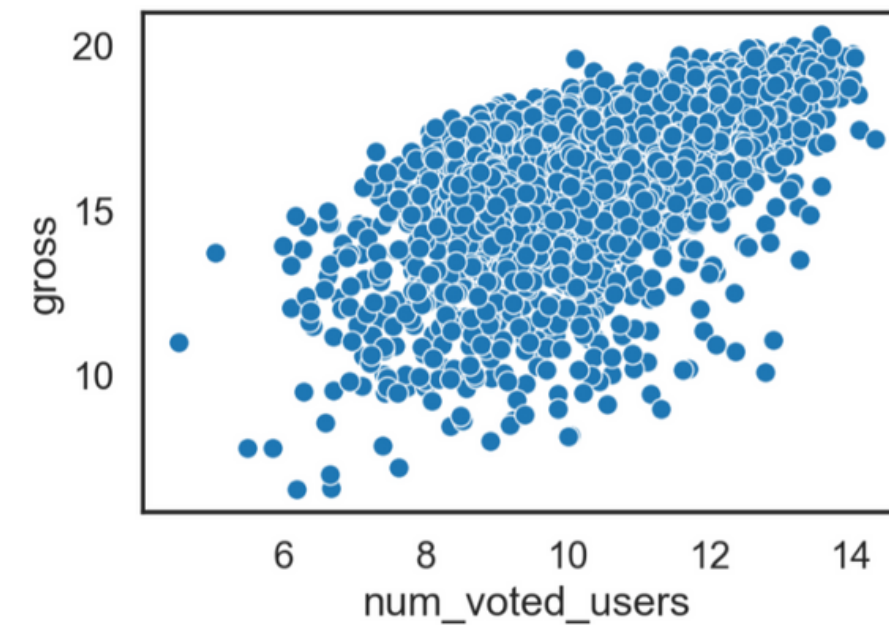

03- DataAnalysis



- We found there is positive relationship between number voted users and gross .

- Here is the corrections between features

```
sns.scatterplot(x = 'num_voted_users', y = 'gross', data = df);
```



04- Experimental ML Models



→ -Dummy's:

```
df_dummy=pd.get_dummies(df)
df_dummy.columns
```

```
Index(['duration', 'director_facebook_likes', 'gross', 'num_voted_users',
      'cast_total_facebook_likes', 'facenumber_in_poster', 'budget',
      'title_year', 'aspect_ratio', 'color_ Black and White',
      ...,
      'content_rating_GP', 'content_rating_M', 'content_rating_NC-17',
      'content_rating_Not Rated', 'content_rating_PG', 'content_rating_PG-13',
      'content_rating_Passed', 'content_rating_R', 'content_rating_Unrated',
      'content_rating_X'],
      dtype='object', length=20070)
```

```
x = df_dummy[['duration', 'director_facebook_likes', 'num_voted_users',
              'cast_total_facebook_likes', 'facenumber_in_poster',
              'budget', 'title_year',
              'aspect_ratio']]
y = df_dummy['gross']
```

df_dummy

	duration	director_facebook_likes	gross	num_voted_users	cast_total_facebook_likes	facenumber
12	106.00	395.00	18.94	12.71	2023	
22	156.00	0.00	18.47	12.26	3244	
24	113.00	129.00	18.07	11.91	24106	
26	194.00	0.00	20.31	13.58	45223	
29	124.00	365.00	20.30	12.94	8458	
...	
5027	90.00	397.00	13.42	8.42	5	
5029	111.00	62.00	11.46	8.75	115	
5033	77.00	291.00	12.96	11.19	368	
5035	81.00	0.00	14.53	10.86	147	
5042	90.00	16.00	11.35	8.36	163	

3703 rows × 20070 columns

04- Experimental ML Models



→ -Split data :

```
3]: from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=10)

4]: from sklearn import metrics
from sklearn.model_selection import cross_val_score

def cross_val(model):
    pred = cross_val_score(model, X, y, cv=5)
    return pred.mean()

def print_evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    print('MAE:', mae)
    print('MSE:', mse)
    print('RMSE:', rmse)
    print('R2 Square', r2_square)
    print('_____')

def evaluate(true, predicted):
    mae = metrics.mean_absolute_error(true, predicted)
    mse = metrics.mean_squared_error(true, predicted)
    rmse = np.sqrt(metrics.mean_squared_error(true, predicted))
    r2_square = metrics.r2_score(true, predicted)
    return mae, mse, rmse, r2_square
```



04- Experimental ML Models

Linear regression :

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train,y_train)
```

R2 Square in Test = 0.47

R2 Square in Test = 0.49

```
test_pred = lin_reg.predict(X_test)
train_pred = lin_reg.predict(X_train)

print('Test set evaluation:\n_____')
print_evaluate(y_test, test_pred)
print('Train set evaluation:\n_____')
print_evaluate(y_train, train_pred)
```

Test set evaluation:

MAE: 1.1237325546186263
MSE: 2.6085103096661646
RMSE: 1.615088328750525
R2 Square 0.47136891371734335

Train set evaluation:

MAE: 1.042646157916681
MSE: 2.2129910196619655
RMSE: 1.4876125233614987
R2 Square 0.4957116065139111

```
results_df = pd.DataFrame(data=[["Linear Regression", *evaluate(y_test, test_pred) ,
                                cross_val(Lin
```

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	1.12	2.61	1.62	0.47	-0.15

04- Experimental ML Models



Polynomial regression :

R2 Square in Test = 0.53

R2 Square in Test = 0.54

```
from sklearn.preprocessing import PolynomialFeatures

poly_reg = PolynomialFeatures(degree=2)

X_train_2_d = poly_reg.fit_transform(X_train)
X_test_2_d = poly_reg.transform(X_test)

lin_reg = LinearRegression(normalize=True)
lin_reg.fit(X_train_2_d, y_train)

test_pred = lin_reg.predict(X_test_2_d)
train_pred = lin_reg.predict(X_train_2_d)

print('Test set evaluation:\n_____')
print_evaluate(y_test, test_pred)
print('=====')
print('Train set evaluation:\n_____')
print_evaluate(y_train, train_pred)
```

Test set evaluation:

MAE: 1.0409578575698533
MSE: 2.318483091584654
RMSE: 1.5226565901688582
R2 Square 0.5301447609040801

Train set evaluation:

MAE: 0.9749047838239859
MSE: 1.9818434471296518
RMSE: 1.4077796159660971
R2 Square 0.5483846797324075



04- Experimental ML Models

- (Ridge) regression :

R2 Square in Test = 0.47

R2 Square in Test = 0.49

```
from sklearn.linear_model import Ridge

model = Ridge(alpha=100, solver='cholesky', tol=0.0001, random_state=42)
model.fit(X_train, y_train)
pred = model.predict(X_test)

test_pred = model.predict(X_test)
train_pred = model.predict(X_train)

print('Test set evaluation:\n_____')
print_evaluate(y_test, test_pred)
print('=====')
print('Train set evaluation:\n_____')
print_evaluate(y_train, train_pred)
```

Test set evaluation:

MAE: 1.1216994054790823
MSE: 2.612184769507832
RMSE: 1.6162254698858796
R2 Square 0.470624261227221

=====

Train set evaluation:

MAE: 1.0419562314393351
MSE: 2.2151833463127892
RMSE: 1.4883492017375457
R2 Square 0.4952120270420932



04- Experimental ML Models

- Lasso regression :

R2 Square in Test = 0.45

R2 Square in Test = 0.47

```
from sklearn.linear_model import Lasso

model = Lasso(alpha=0.1,
               precompute=True,
               # warm_start=True,
               positive=True,
               selection='random',
               random_state=42)
model.fit(X_train, y_train)

test_pred = model.predict(X_test)
train_pred = model.predict(X_train)

print('Test set evaluation:\n_____')
print_evaluate(y_test, test_pred)
print('=====')
print('Train set evaluation:\n_____')
print_evaluate(y_train, train_pred)
```

Test set evaluation:

MAE: 1.1389648759914734
MSE: 2.707774748267592
RMSE: 1.6455317524337207
R2 Square 0.45125234840699724

=====

Train set evaluation:

MAE: 1.0677804997566571
MSE: 2.2991322475263796
RMSE: 1.5162889723025685
R2 Square 0.47608205491306466



04- Experimental ML Models

- RandomForest regression :

R2 Square in Test = 0.58

R2 Square in Test = 0.95

```
from sklearn.ensemble import RandomForestRegressor

rf_reg = RandomForestRegressor(n_estimators=1000)
rf_reg.fit(X_train, y_train)

test_pred = rf_reg.predict(X_test)
train_pred = rf_reg.predict(X_train)

print('Test set evaluation:\n_____')
print_evaluate(y_test, test_pred)

print('Train set evaluation:\n_____')
print_evaluate(y_train, train_pred)
```

Test set evaluation:

MAE: 0.9596129441669611
MSE: 2.06400680921934
RMSE: 1.436665169487776
R2 Square 0.5817159864734989

Train set evaluation:

MAE: 0.32967788573087553
MSE: 0.23896883815356398
RMSE: 0.48884439053093776
R2 Square 0.945544645046004



04- Experimental ML Models

- Regression Models :

```
results_df_2 = pd.DataFrame(data=[[ "Random Forest", *evaluate(y_test, test_pred),  
                                     cross_val(RandomForestRegressor()) ]],  
                             columns=[ 'Model', 'MAE', 'MSE', 'RMSE', 'R2 Square',  
                                       'Cross Validation' ])  
results_df = results_df.append(results_df_2, ignore_index=True)  
results_df
```

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	1.12	2.61	1.62	0.47	-0.15
1	Polynomial Regression	1.04	2.32	1.52	0.53	0.00
2	Ridge Regression	1.12	2.61	1.62	0.47	-0.15
3	Lasso Regression	1.14	2.71	1.65	0.45	-0.52
4	Random Forest	0.96	2.06	1.44	0.58	0.29

We chose polynomial regression because it's the best model to fit our dataset

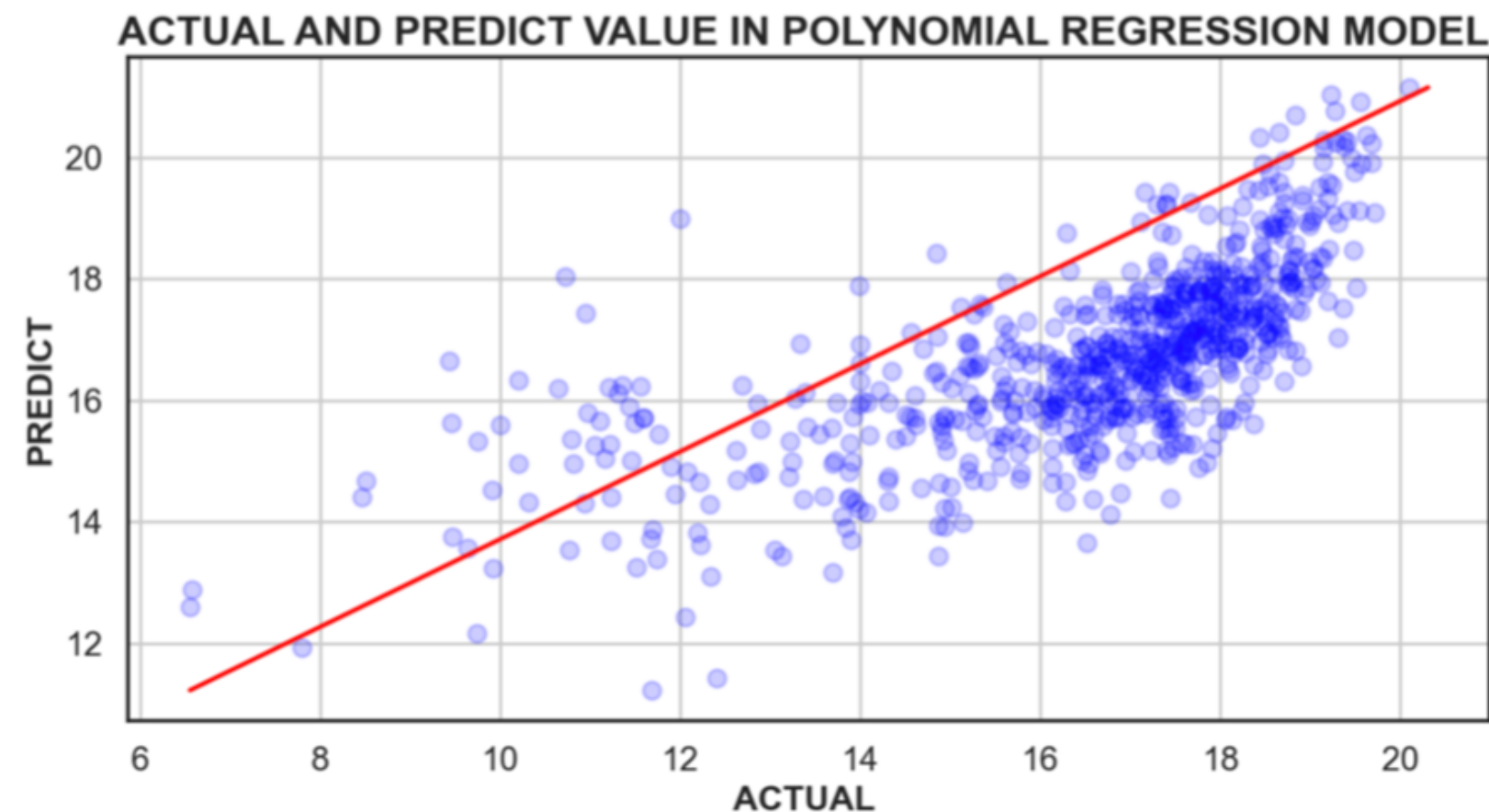




04- Experimental ML Models

-Prediction

```
plt.figure(figsize = [12,6])
sns.set_style("white")
x_plt_range = [y.min(),y.max()]
y_plt_range = [y_haat.min(),y_haat.max()]
plt.grid()
plt.scatter(y_test,y_haat,alpha=0.2,color='Blue',s=75)
plt.plot(x_plt_range,y_plt_range,c="r")
plt.title("ACTUAL AND PREDICT VALUE IN POLYNOMIAL REGRESSION MODEL "
          ,fontsize = 20, weight = 'bold')
plt.xlabel("ACTUAL ",fontsize = 17, weight = 'bold')
plt.ylabel('PREDICT',fontsize = 17, weight = 'bold');
plt.savefig('SDAIA.png')
```





Conclusion

We found the best model that will predict the response of our data

from a website such as predict the gross of a bunch of movies and see the correlation between each other.

