# CP1 - What movies make the most money at the box office
# Milestone Report 1

## Problem Statement

In a world  where movies made an estimated $41.7 billion in 2018, the film industry is more popular than ever. But what movies make the most money at the box office? How much does a director matter? Or the budget?
Can we build models, which will be able to accurately predict film revenue?

This project will help film production companies understand key features of having high revenue.

## Data Source

The major data source comes from the public dataset uploaded to Kaggle.com ([https://www.kaggle.com/c/tmdb-box-office-prediction/overview](https://www.kaggle.com/c/tmdb-box-office-prediction/overview)).

This dataset with metadata on over 7,000 past films from The Movie Database. Data points provided include cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries.

## Data Cleaning

There are 8 JSON-style columns. We will parse them and create categorical and dummy variables. For example, column “belongs_to_collection”:

**belongs_to_collection**

```python
for i, e in enumerate(master['belongs_to_collection'][:5]):
    print(i, e)
```

```
0 [{'id': 313576, 'name': 'Hot Tub Time Machine Collection', 'poster_path': '/iEhb00TGPucF0b4joM1ieyY026U.jpg', 'backdrop_p
ath': '/noeTVcgpBiD48fDjFVic1Vz7ope.jpg'}]
1 [{'id': 107674, 'name': 'The Princess Diaries Collection', 'poster_path': '/wt5AMbxPTS4Kfjx7Fgm149qPfZl.jpg', 'backdrop_p
ath': '/zSEtYD77pKRJlUPx34BJgUG9v1c.jpg'}]
2 nan
3 nan
4 nan
```

We create two new columns from column "belongs_to collection", first one is collection name and second one has collection or not. We assume that other information from this column we can’t use for future prediction.

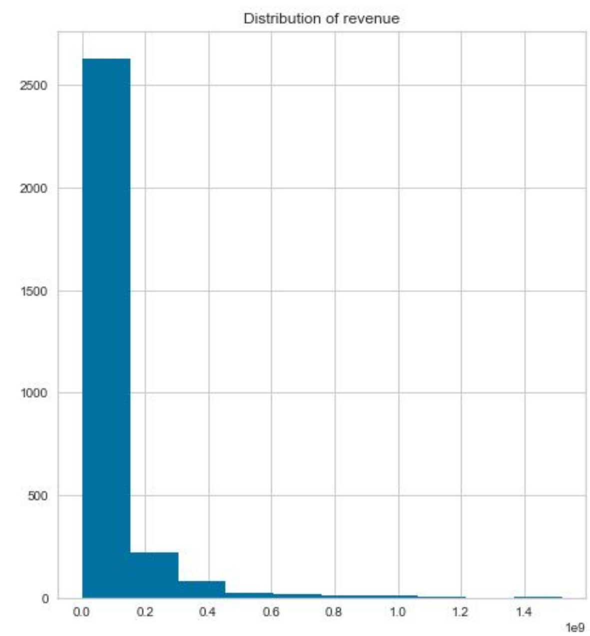
```python
master['collection_name'] = master['belongs_to_collection'].apply(lambda x: x[0]['name'] if x != {} else 0)
master['has_collection'] = master['belongs_to_collection'].apply(lambda x: len(x) if x != {} else 0)

master = master.drop(['belongs_to_collection'], axis=1)
```

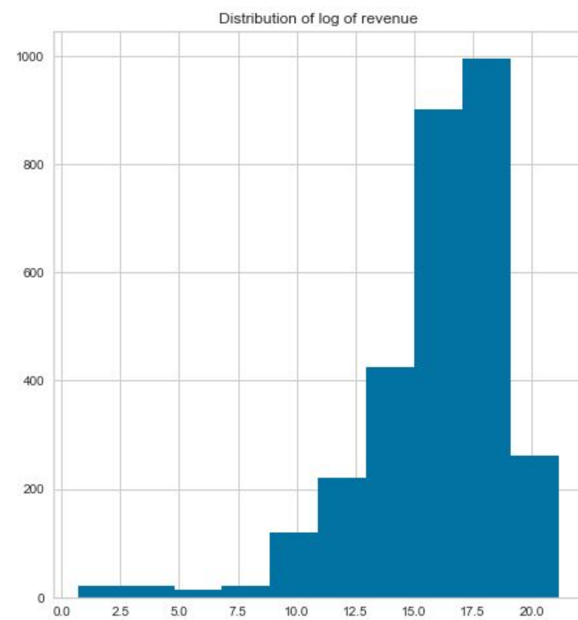
## Exploratory Analysis

### SOME INSIGHTS

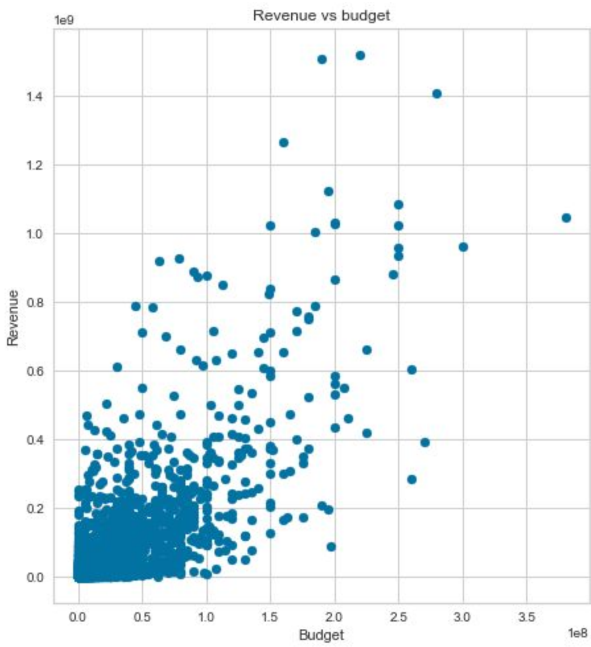
- Average length of a movie is about **107 minutes**
- The **longest** movie: *"Carlos"* is **5hrs 38min** long
- The **most popular** movies: *"Wonder Woman"* and *"Beauty and the Beast"*
- Movies with the **biggest budget**: *"Pirates of the Caribbean: On Stranger Tides"* and *"Pirates of the Caribbean: At World's End"*
- The **biggest** movie **producers**: *Warner Bros.* and *Universal Pictures*
- The most popular **genres**: *Drama* and *Comedy*

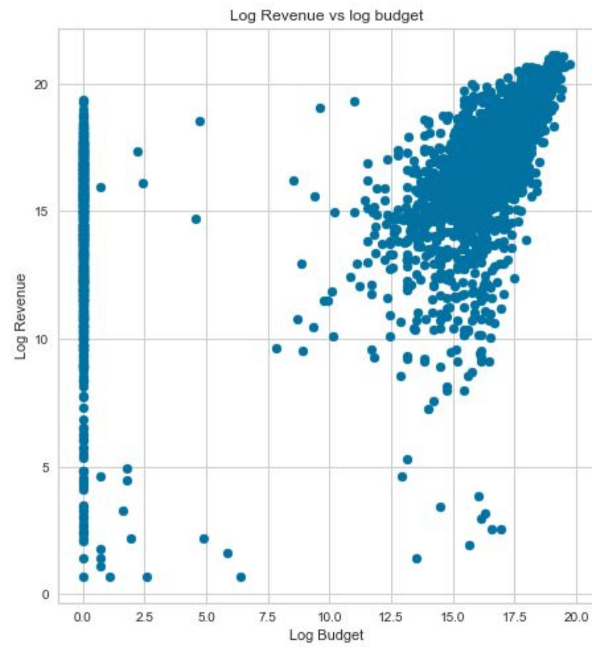
Let first look at the histograms of the target feature - **revenue**. As we can see revenue distribution has a high skewness. It is better to use log transformation of revenue.



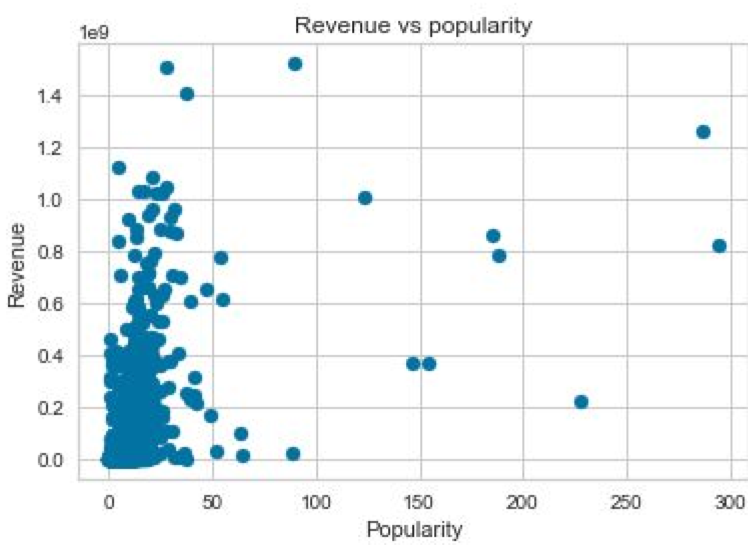



First feature we will look at is **budget**, intuitively the budget should be somehow correlated with revenue.

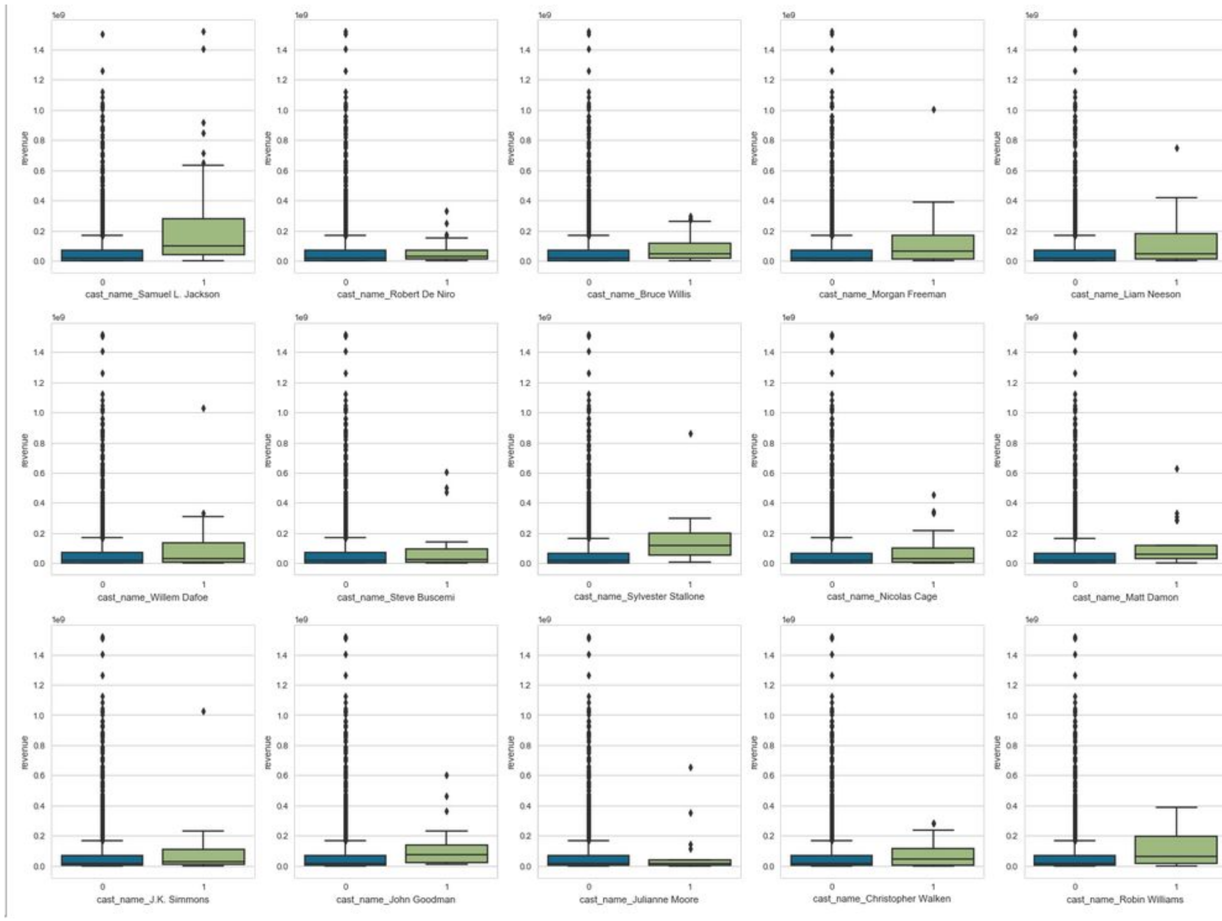



Let's look at popularity. We can see some clear trends that an increase in popularity tends to lead to higher revenue.




We have columns with most common cast members, so let's plot boxplots.




As you can see mostly films with these actors tend to have higher revenue.

## Machine Learning

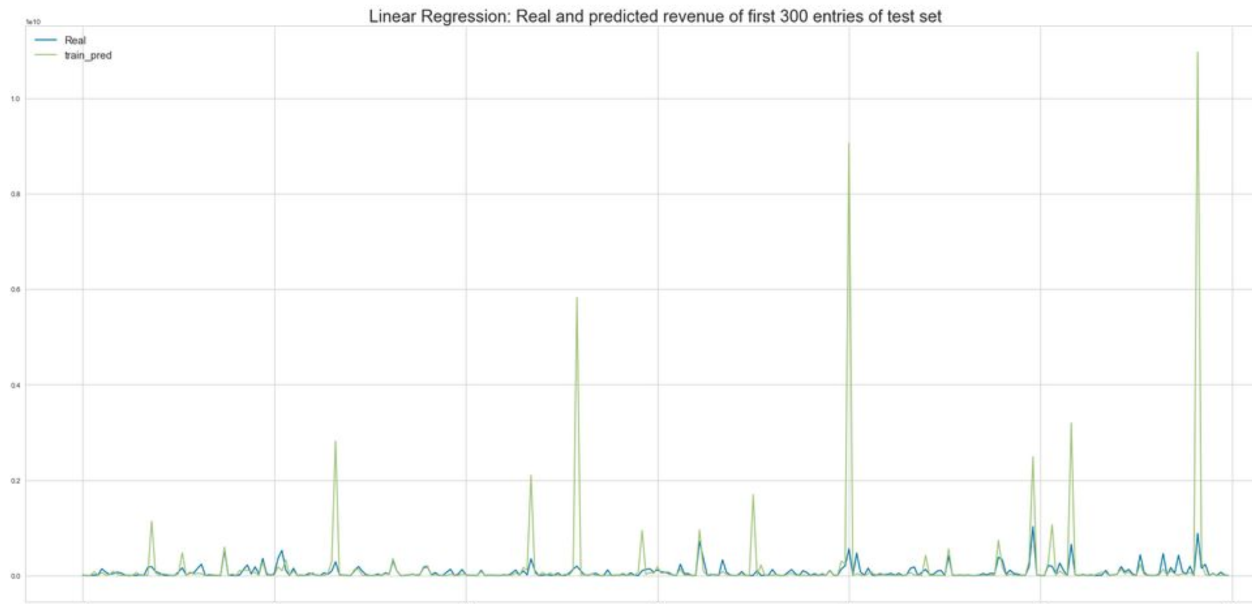
### 1. Linear Regression

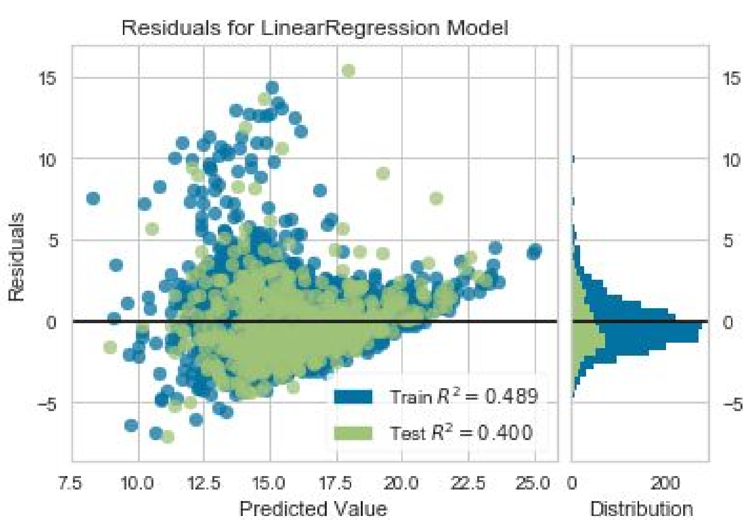
Once the data was transformed and usable for machine learning, the data was split into training and test sets with the neighbor players appended to the test set to make sure predictions applied to them.

```
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=42)
```

```
#Linear Regression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred=regressor.predict(X_test)
```

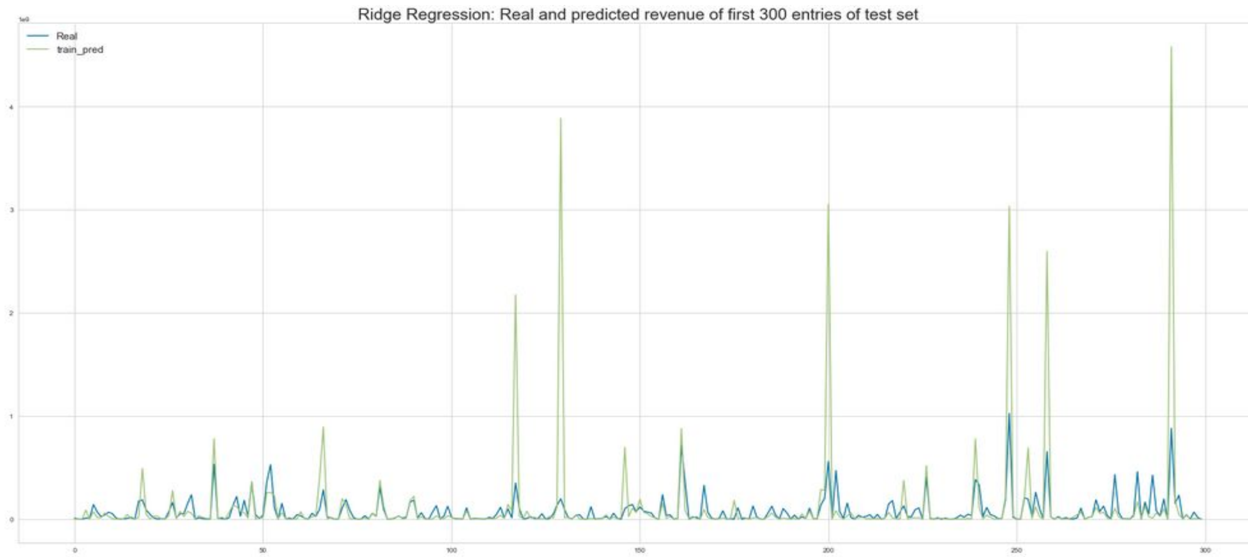



It was observed that the residuals between the predicted values and actual values roughly fit the same trends according to a Linear Regression model.
It was also observed that extremely higher revenues were underpredicted while lower values were predicted closer to their actual valuation.

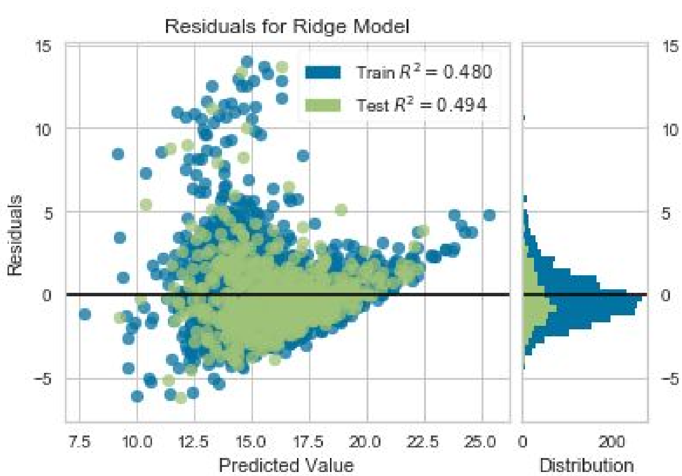
## 2. Ridge Regression

Lets try another model - Ridge Regression.




```
#Ridge Regression metrics
pd.set_option('display.float_format', lambda x: '%.3f' % x)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred_clf))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred_clf))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred_clf)))
print('Root Mean Squared Log Error:', np.sqrt(mean_squared_log_error( y_test, y_pred_clf )))
```

```
Mean Absolute Error: 1.4990212383799935
Mean Squared Error: 4.506682175538423
Root Mean Squared Error: 2.122894763180319
Root Mean Squared Log Error: 0.1905475484817239
```



As we can see results are pretty similar to linear regression with small improvement.

3. **XGBoost**

If things don't go your way in predictive modeling, use XGboost. The XGBoost algorithm has become the ultimate weapon of many data scientists. It's a highly sophisticated algorithm, powerful enough to deal with all sorts of irregularities of data.

Building a model using XGBoost is easy. But, improving the model using XGBoost is difficult. This algorithm uses multiple parameters. To improve the model, parameter tuning is must.

So using the code below I am trying to fine tune XGBoost parameters.

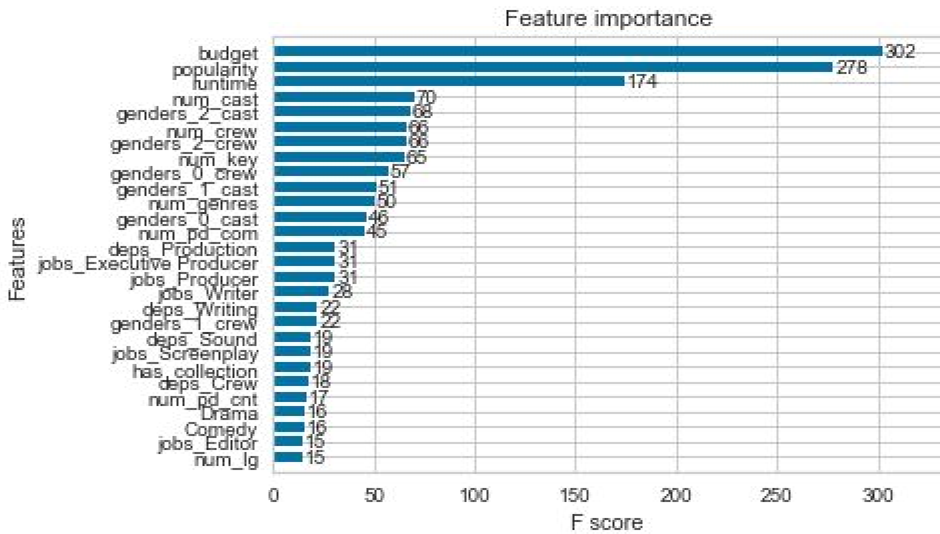
```
X_train, X_test, y_train, y_test = train_test_split(df_X_train, df_y_train, test_size=0.2, random_state=42)
# read in data
dtrain = xgb.DMatrix(X_train, label=y_train,feature_names=feature_names)
dtest = xgb.DMatrix(X_test, label=y_test,feature_names=feature_names)
```

```python
def algorithm_pipeline(X_train_data, X_test_data, y_train_data, y_test_data,
                       model, param_grid, cv=10, scoring_fit='neg_mean_squared_error',
                       do_probabilities = False):
    gs = GridSearchCV(
        estimator=model,
        param_grid=param_grid,
        cv=cv,
        n_jobs=-1,
        scoring=scoring_fit,
        verbose=2
    )
    fitted_model = gs.fit(X_train_data, y_train_data)

    if do_probabilities:
      pred = fitted_model.predict_proba(X_test_data)
    else:
      pred = fitted_model.predict(X_test_data)

    return fitted_model, pred

model = xgb.XGBRegressor()
param_grid = {
    'min_child_weight': [1,5,20,25,30],
    'max_depth': [4,5,6],
    'learning_rate': [0.01,0.05,0.1,0.3],
    'colsample_bytree': [0.9],
    'subsample': [0.9],
    'lambda': [1.],
    'nthread': [-1],
    'booster' : ['gbtree'],
    'silent': [1],
    'objective': ['reg:linear']
}

model, pred = algorithm_pipeline(X_train, X_test, y_train, y_test, model,
                                 param_grid, cv=[(slice(None), slice(None))])

# Root Mean Squared Error
print(np.sqrt(-model.best_score_))
print(model.best_params_)
```

So according to the code above we have following parameters:
max-depth - 6
learning rate - 0.3
min child weight - 1

We can investigate the importance of each feature, to understand what affects the revenue the most significantly. We can do this like this:

```
_=xgb.plot_importance(model_2, max_num_features=28, height=0.7)
```




According to the feature importance plot we have next top 3 features : budget, popularity and runtime. Below I create a dataframe with these features and revenue to show how it looks like.

Also below how these features look for 5 films from the dataset:

| | budget | popularity | runtime | num_cast | genders_2_cast | genders_2_crew | num_crew | num_key | genders_0_crew | genders_1_cast | num_genres |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3500000 | 0.556 | 90.000 | 11 | 3 | 2 | 2 | 2 | 0 | 5 | 1 |
| 1 | 0 | 2.087 | 100.000 | 7 | 2 | 5 | 13 | 7 | 8 | 1 | 2 |
| 2 | 2000000 | 1.189 | 89.000 | 3 | 2 | 0 | 1 | 1 | 1 | 1 | 3 |
| 3 | 98000000 | 7.284 | 119.000 | 31 | 20 | 12 | 16 | 6 | 2 | 2 | 2 |
| 4 | 0 | 1.219 | 101.000 | 7 | 3 | 0 | 2 | 0 | 2 | 0 | 4 |

| genders_0_cast | num_pd_com | jobs_Producer | deps_Production | jobs_Executive Producer | jobs_Writer | genders_1_crew | deps_Writing | jobs_Screenplay | deps_Sound |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 4 | 2 | 2 | 2 | 0 | 2 | 0 | 2 | 0 | 3 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 4 | 4 | 6 | 1 | 0 | 2 | 4 | 2 | 0 |
| 4 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

| has_collection | deps_Crew | num_pd_cnt | Comedy | Drama | jobs_Editor | num_lg | revenue | predicted_revenue | title |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 16.040 | 14.881 | Ringmaster |
| 0 | 0 | 1 | 0 | 0 | 0 | 3 | 2.079 | 13.255 | He-Man and She-Ra: The Secret of the Sword |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 10.425 | 12.766 | Cowboys & Angels |
| 0 | 0 | 4 | 0 | 0 | 0 | 2 | 16.120 | 18.869 | Cutthroat Island |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 16.003 | 14.164 | We Are from the Future 2 |

# Conclusion

This project demonstrates which features are important to produce high revenue film. Film companies can use these models to predict their revenue.

As I mentioned before, budget, popularity and runtime are top 3 important features, also you can see that the number of male in cast and crew also have some influence on revenue.

# Further Research Work

Though I have tried pretty much works of data exploration and model selection, it is only the introduction of the whole work and the accuracy of my model is not high enough. There are still so many gaps and now I'd like to study all capabilities of improvement one by one.

Possible parts to be improved are:

- **feature engineering**
    - there are lots of features to be created.
    - missing values
    - outliers
- **feature selection**
    - feature selection methods
    - consideration of validation of selected features
- **model selection**
    - other models
    - how to improve the performance of a model(specialize for the selected model)
    - model parameter tuning