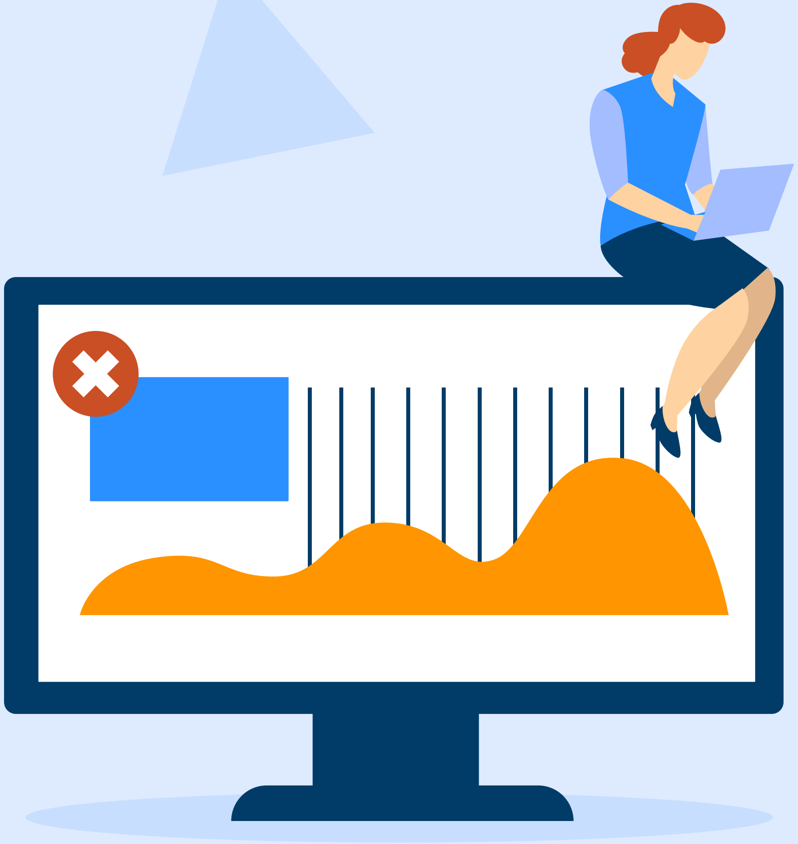
DSAI Mini Project

REPA Group 1
Jordon Kho Junyang
Riley Ang Xile
Truong Quang Duc



Outline

01 Introduction

02 Data Processing

03 EDA

04 Baseline Models

05 Improvements to Baseline Models

06 Discussions & Conclusions

01

Introduction

Dataset used

- IMDB 5000 Movie Dataset from Kaggle
- Contains the data for movies including: Reviews, Information, Gross...

Motivation

- Present the data found in a meaningful manner
- Find a model to predict the relationships between gross of movie and other factors
- Learning new model types and techniques to solve this problem

02

Data Processing

1. Handling missing values

- Replace missing values in numerical features with their medians and ones in categorical features with its mode.
- Exceptions are made to certain columns (e.g Director)

```
for c in null_features_list:
    # replacing missing values for categorical features
    if df[c].dtype.name == 'category':
        df[c].fillna(df[c].mode().iloc[0], inplace = True)
    # replacing missing values for numerical features
    else:
        df[c].fillna(df[c].median(), inplace=True)

null_features = getNull(df)
```

2. Removing duplicate values

- Use DataFrame.drop_duplicates()
- Reset index afterward

```
df.drop_duplicates(inplace=True) # drops all duplicates and updates the dataframe
```

```
df.reset_index(drop=True, inplace=True)
```

3. Feature Engineering: Movie count

- Add a more meaningful statistic: number of movies that the director/actor is featured in
- Fill in zero if the director/actor is missing
- Save the count as a categorical type

4. Feature Engineering: Genre separation

- Genres are represented by a string separated with "|" and should be divided
- Use one hot coding (pd.get_dummies) to put each genre as one column for machine learning later on

26	Action	4998 non-null	category
27	Adventure	4998 non-null	category
28	Animation	4998 non-null	category
29	Biography	4998 non-null	category
30	Comedy	4998 non-null	category
31	Crime	4998 non-null	category
32	Documentary	4998 non-null	category
33	Drama	4998 non-null	category
34	Family	4998 non-null	category
35	Fantasy	4998 non-null	category
36	Film-Noir	4998 non-null	category
37	Game-Show	4998 non-null	category

5. Feature Engineering: Main keyword

- The “keywords” column also contains multiple keywords like the “genres” column
- We only need to keep the main keyword and how many times it appeared
- Main keyword is defined as the first keyword

6. Dropping unnecessary columns

- Some columns are not necessary for analysis e.g "movie_title", "movie_imdb_link"...
- Dropping these will improve the EDA and the machine learning
- Re-categorize the columns to finish data processing

```
print("Data type : ", type(df))
print("Data dims : ", df.shape)
```

```
Data type :  <class 'pandas.core.frame.DataFrame'>
Data dims :  (4998, 53)
```

03

EDA

Exploratory Data Analysis & Preparation for ML

Gross

1 response variable

Numerical

Statistical Summary + Distribution plots

Categorical

Count plots + Box plots

ML Model

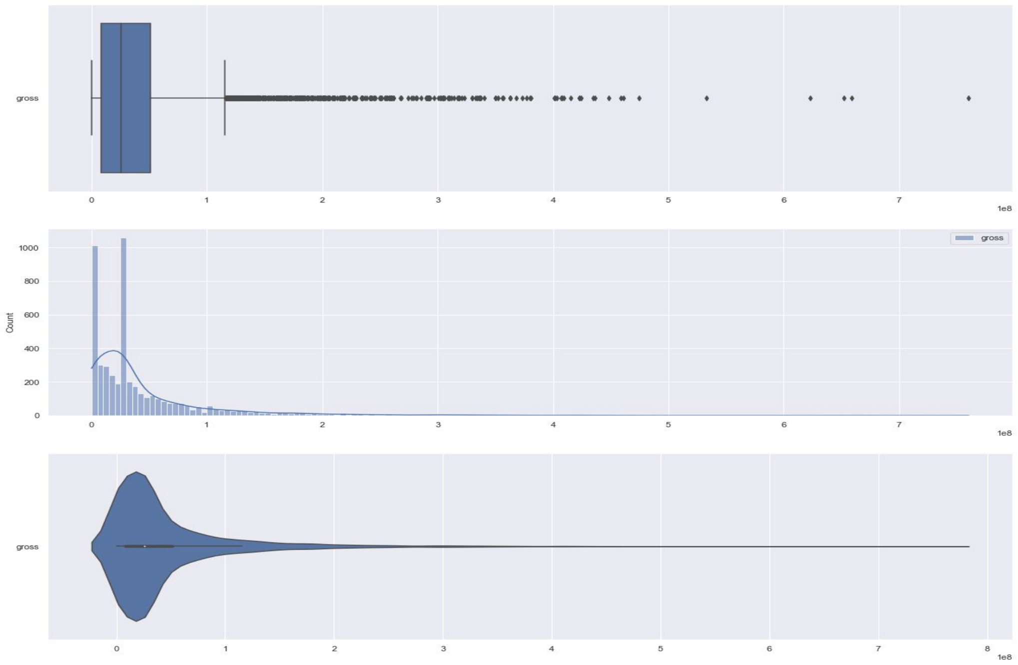
Dimensionality Reduction + Scaling

Gross - Univariate Visualization

Statistical Summary

	gross
count	4.998000e+03
mean	4.433719e+07
std	6.234076e+07
min	1.620000e+02
25%	8.382841e+06
50%	2.551750e+07
75%	5.137692e+07
max	7.605058e+08

Box Plot, Histogram & Violin Plot

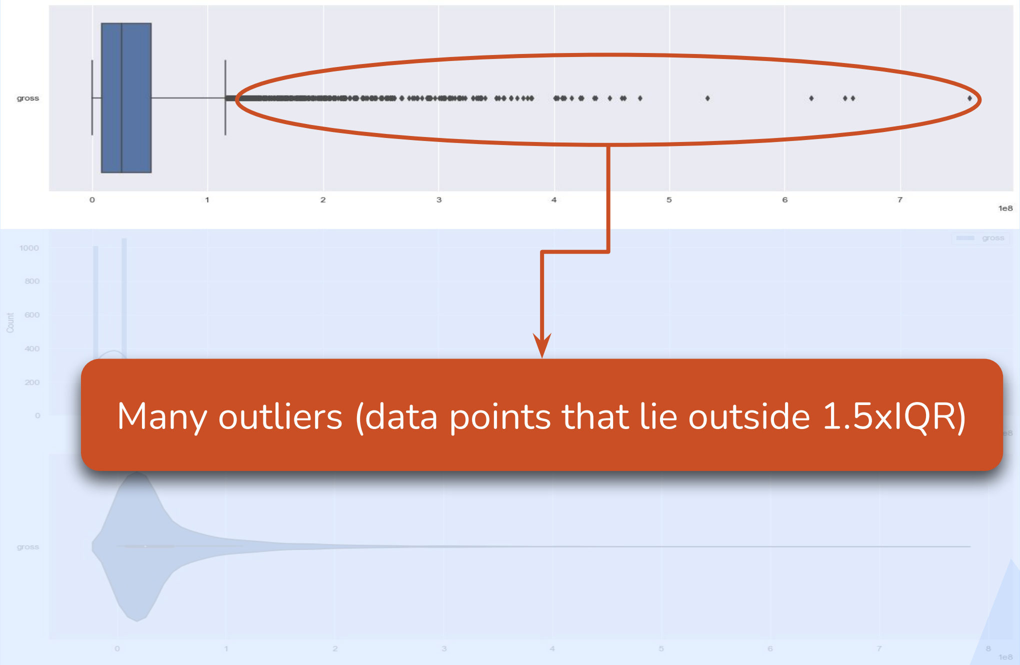


Gross - Univariate Visualization

Statistical Summary

	gross
count	4.998000e+03
mean	4.433719e+07
std	6.234076e+07
min	1.620000e+02
25%	8.382841e+06
50%	2.551750e+07
75%	5.137692e+07
max	7.605058e+08

Box Plot, Histogram & Violin Plot

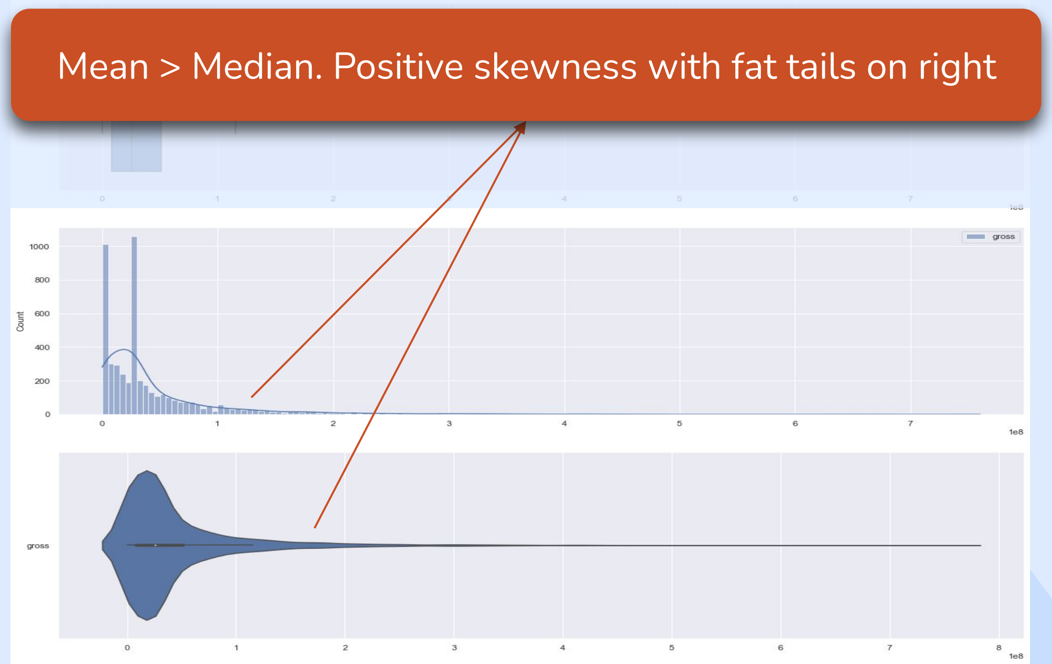


Gross - Univariate Visualization

Statistical Summary

	gross
count	4.998000e+03
mean	4.433719e+07
std	6.234076e+07
min	1.620000e+02
25%	8.382841e+06
50%	2.551750e+07
75%	5.137692e+07
max	7.605058e+08

Box Plot, Histogram & Violin Plot



Numerical - Univariate Visualization

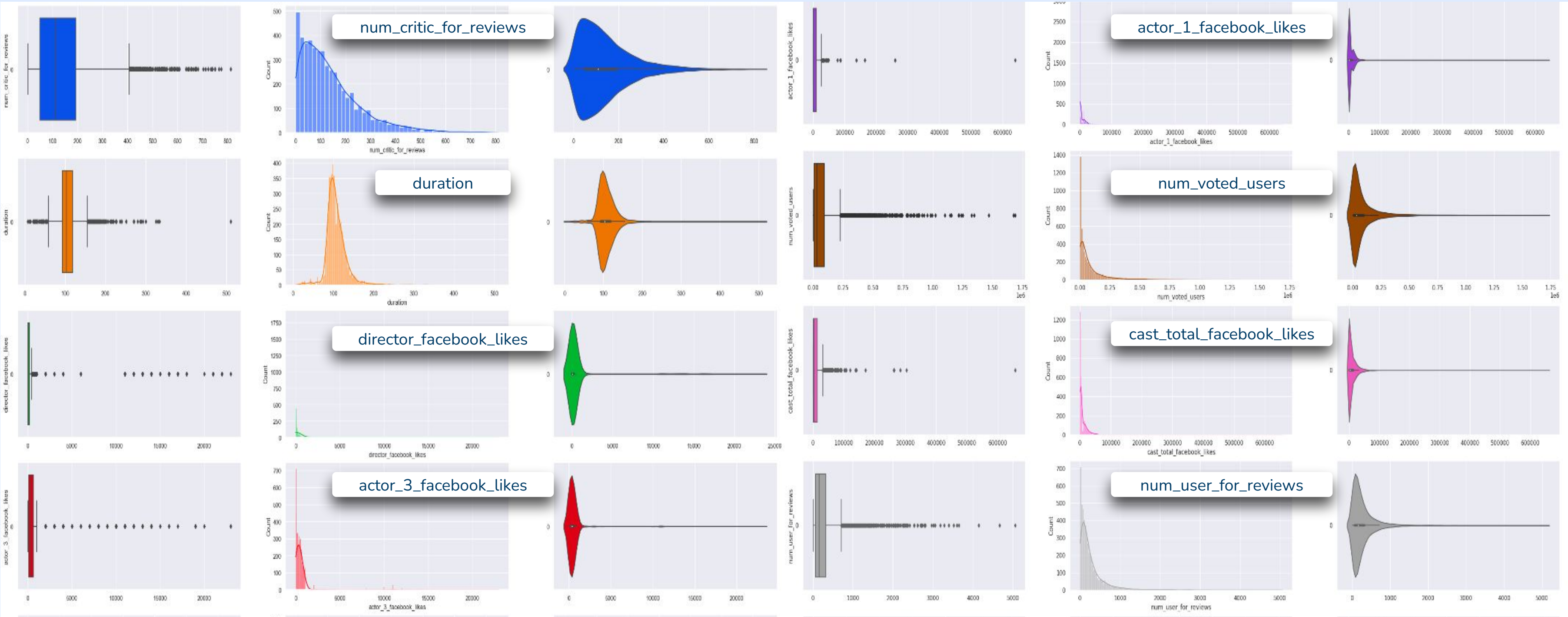
Statistical Summary

	count	mean	std	min	25%	50%	75%	max
num_critic_for_reviews	4998.0	1.395970e+02	1.209164e+02	1.0	50.00	110.0	193.00	8.130000e+02
duration	4998.0	1.072007e+02	2.521190e+01	7.0	93.00	103.0	118.00	5.110000e+02
director_facebook_likes	4998.0	6.754964e+02	2.793896e+03	0.0	7.00	49.0	189.00	2.300000e+04
actor_3_facebook_likes	4998.0	6.386658e+02	1.639613e+03	0.0	134.00	371.5	634.75	2.300000e+04
actor_1_facebook_likes	4998.0	6.549140e+03	1.505247e+04	0.0	613.00	986.0	11000.00	6.400000e+05
gross	4998.0	4.433719e+07	6.234076e+07	162.0	8382841.25	25517500.0	51376923.25	7.605058e+08
num_voted_users	4998.0	8.347020e+04	1.380866e+05	5.0	8560.00	34260.5	96120.75	1.689764e+06
cast_total_facebook_likes	4998.0	9.676941e+03	1.816540e+04	0.0	1405.50	3085.5	13740.50	6.567300e+05
num_user_for_reviews	4998.0	2.715272e+02	3.770563e+02	1.0	65.00	156.0	323.00	5.060000e+03
budget	4998.0	3.782366e+07	1.967122e+08	218.0	7000000.00	20000000.0	40000000.00	1.221550e+10
actor_2_facebook_likes	4998.0	1.640273e+03	4.026032e+03	0.0	281.00	595.0	912.75	1.370000e+05
imdb_score	4998.0	6.441056e+00	1.124107e+00	1.6	5.80	6.6	7.20	9.500000e+00
movie_facebook_likes	4998.0	7.487430e+03	1.929073e+04	0.0	0.00	162.5	3000.00	3.490000e+05

Variables have vastly different ranges

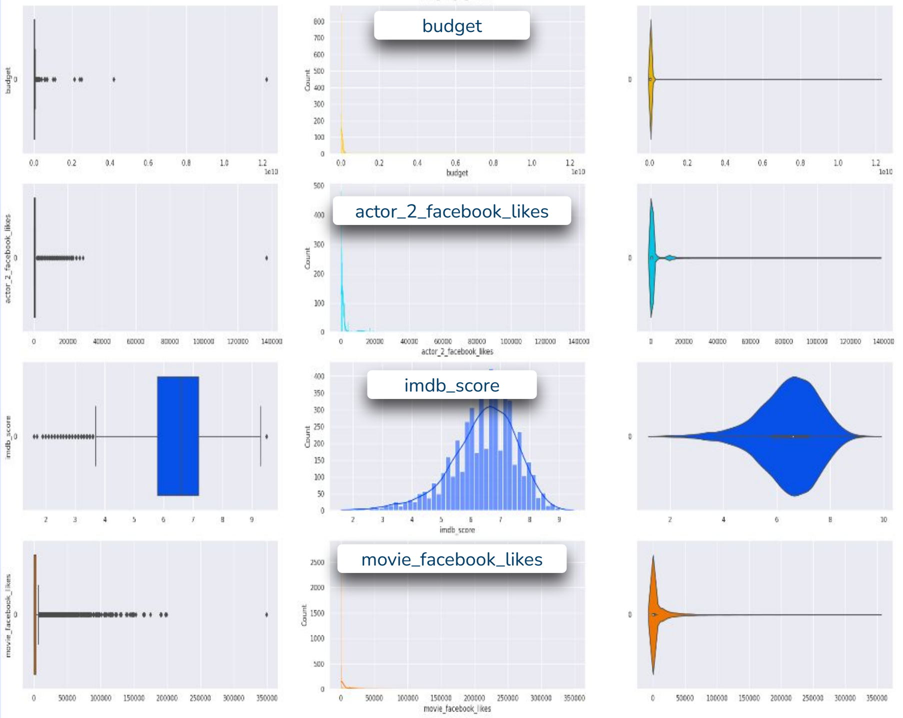
Numerical - Univariate Visualization

Box plots, Histogram & Violin Plots for each feature



Numerical - Univariate Visualization

Box plots, Histogram & Violin Plots for each feature

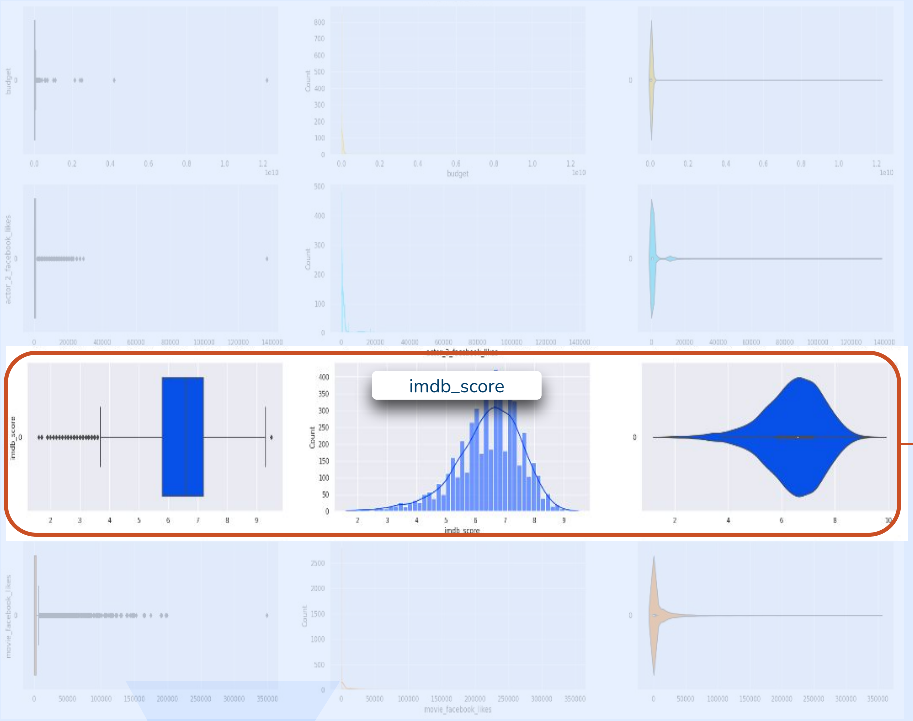


Numerical - Univariate Visualization

Box plots, Histogram & Violin Plots for each feature

Numerical - Univariate Visualization

Box plots, Histogram & Violin Plots for each feature

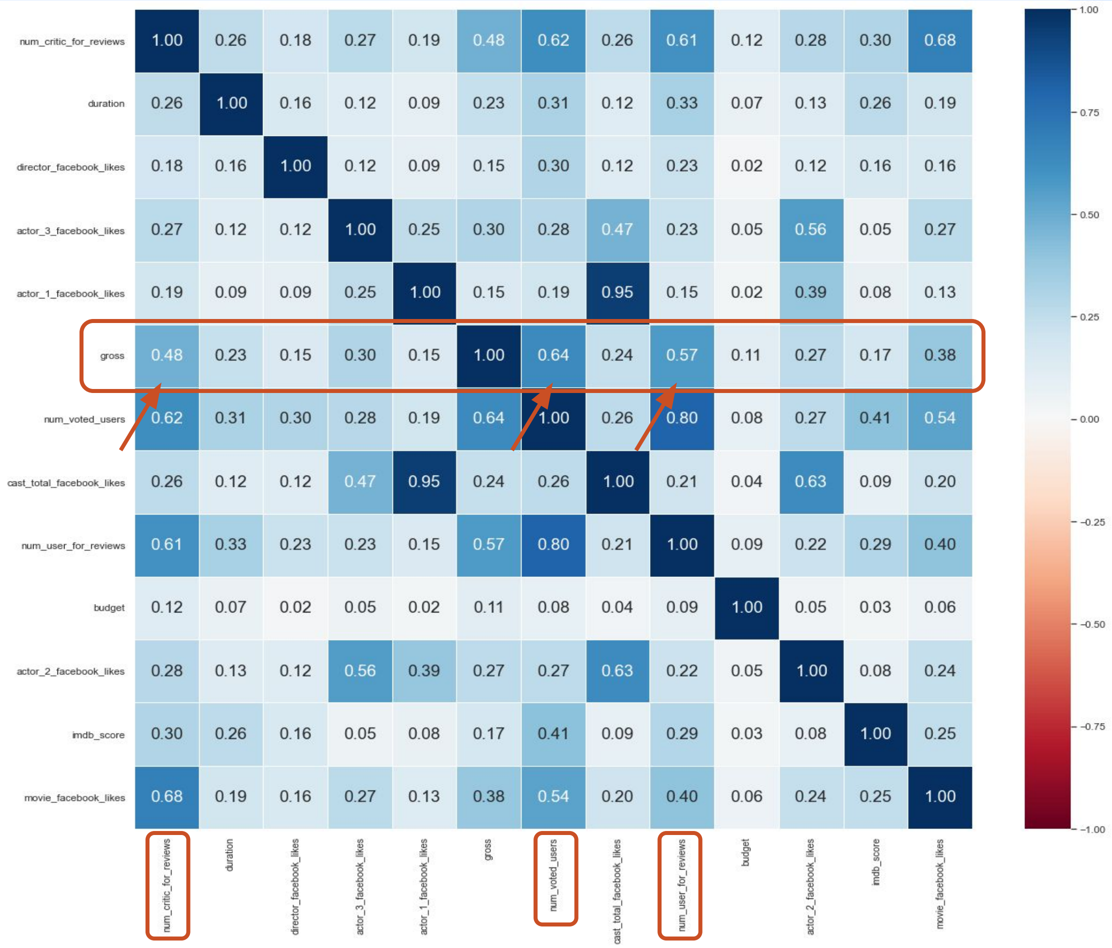


imdb_score resembles normal distribution

Numerical - Bivariate Visualization

Correlation Matrix Heatmap

- 3 features out of 12 numerical features are rather positively correlated with our response variable gross
    1. num_voted_users 0.64
    2. num_user_for_reviews 0.57
    3. num_critic_for_reviews 0.48



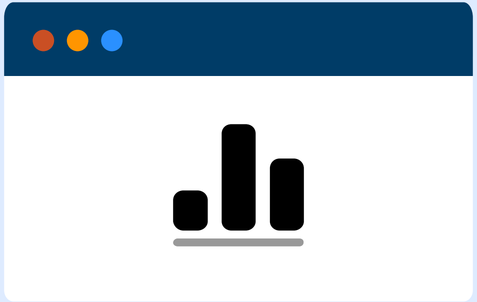
Numerical - Bivariate Visualization

Scatter Plots

3 features with highest positive correlation with gross showed the most obvious positive linear relationship

1. num_voted_users 0.64
2. num_user_for_reviews 0.57
3. num_critic_for_reviews 0.48

Categorical - Data Visualization

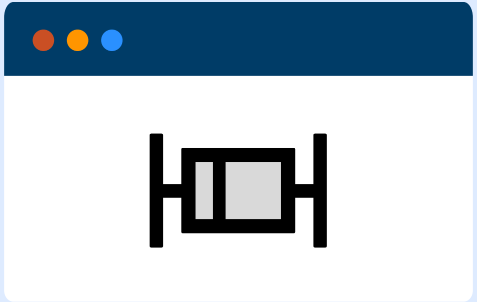


Count Plots

To explore the distribution of each categorical feature

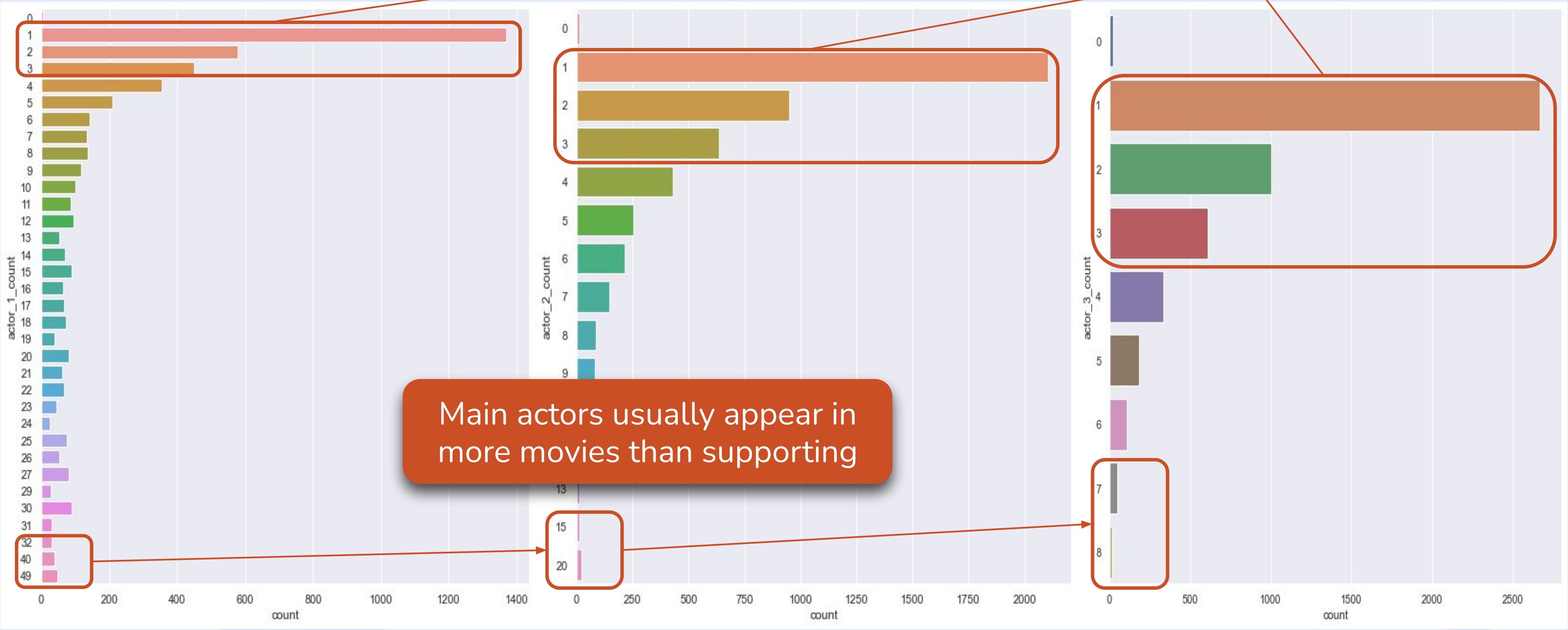
Box Plots

To explore how these categorical features relate with our response variable, gross



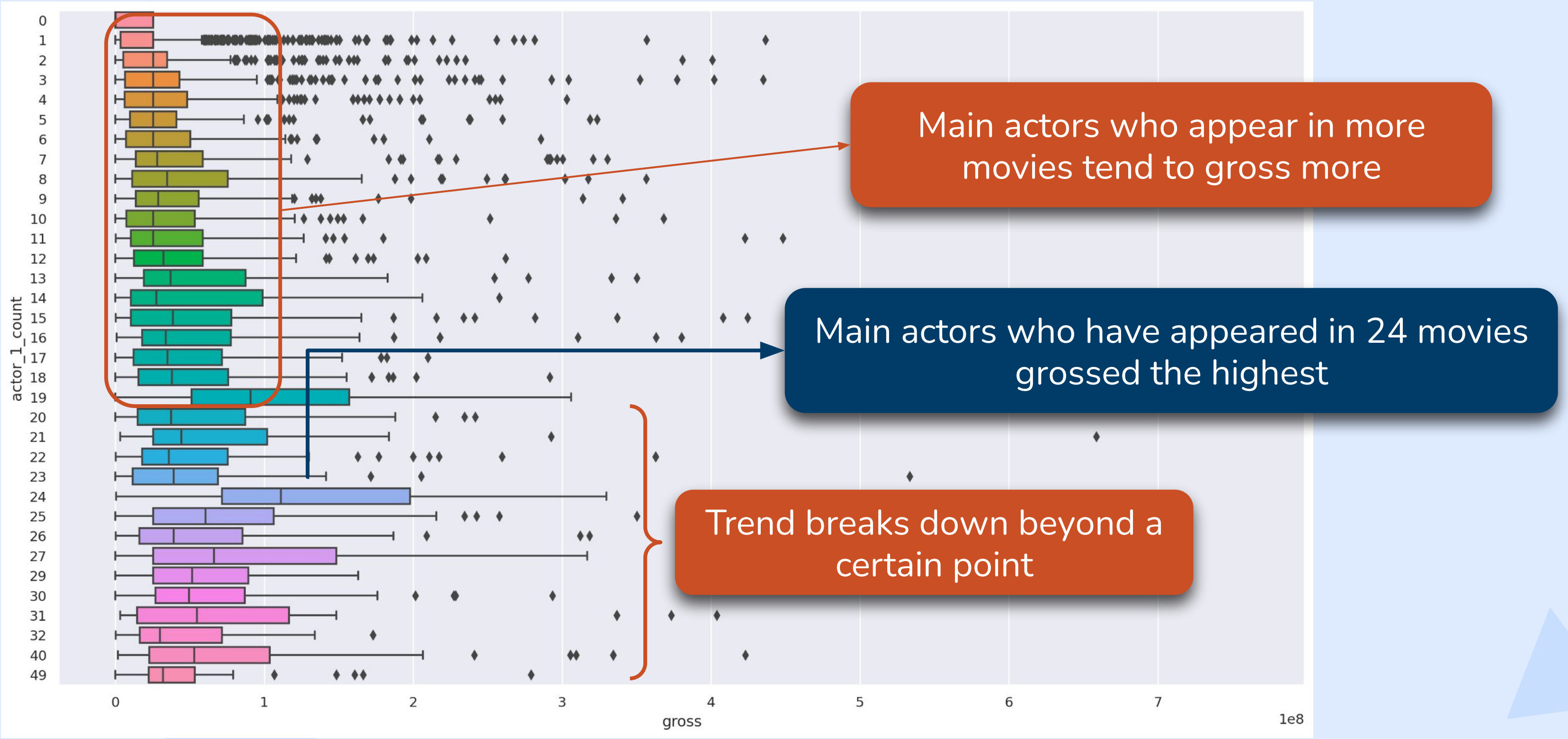
actor_1/2/3_count

Count Plot



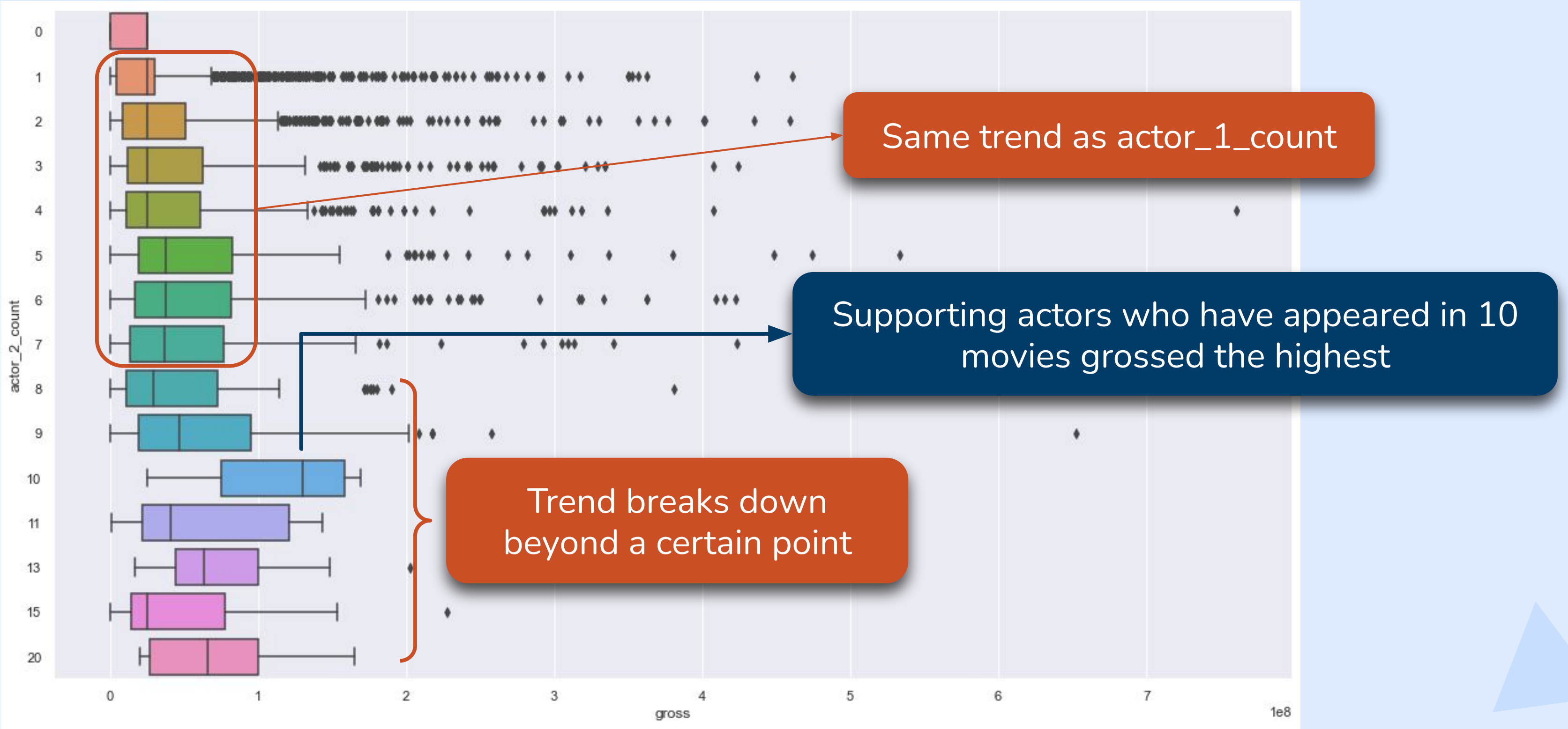
actor_1_count

Box Plot



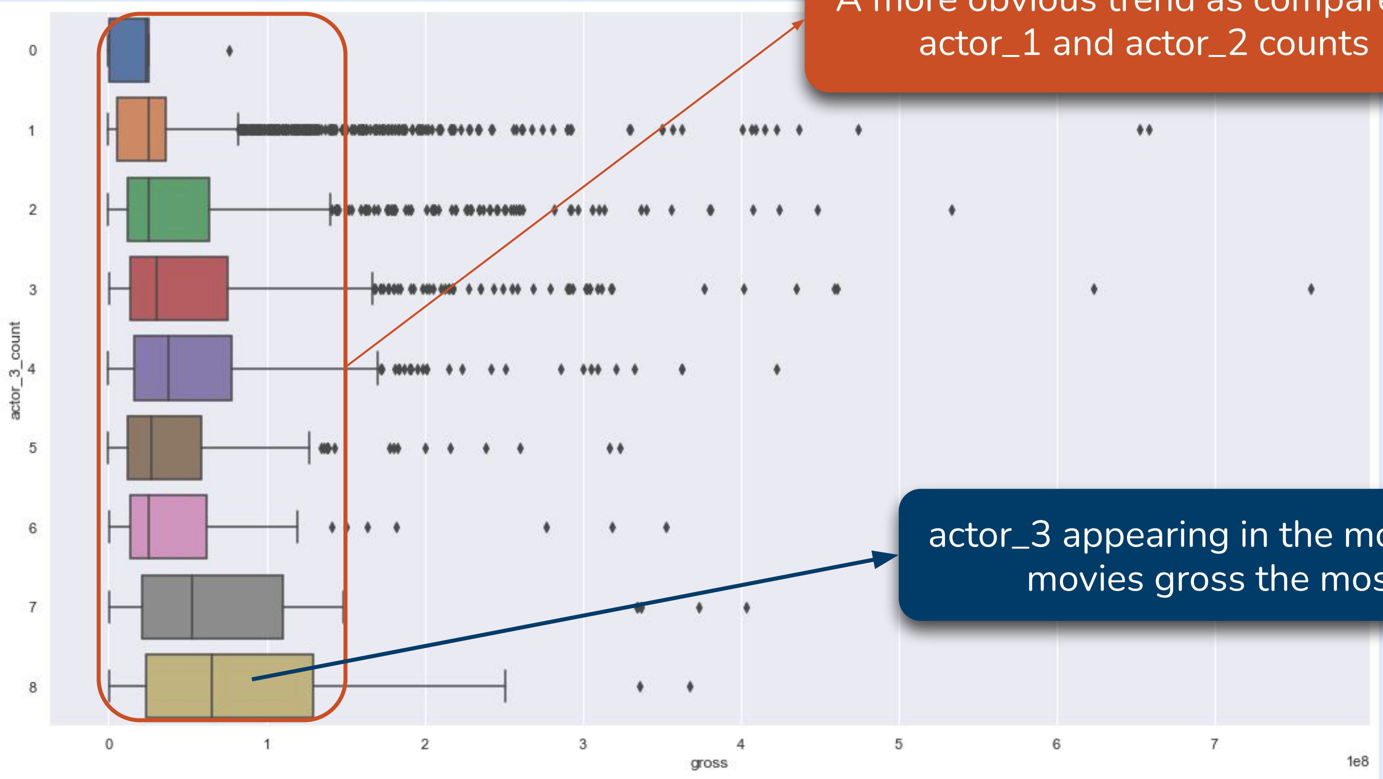
actor_2_count

Box Plot



actor_3_count

Box Plot

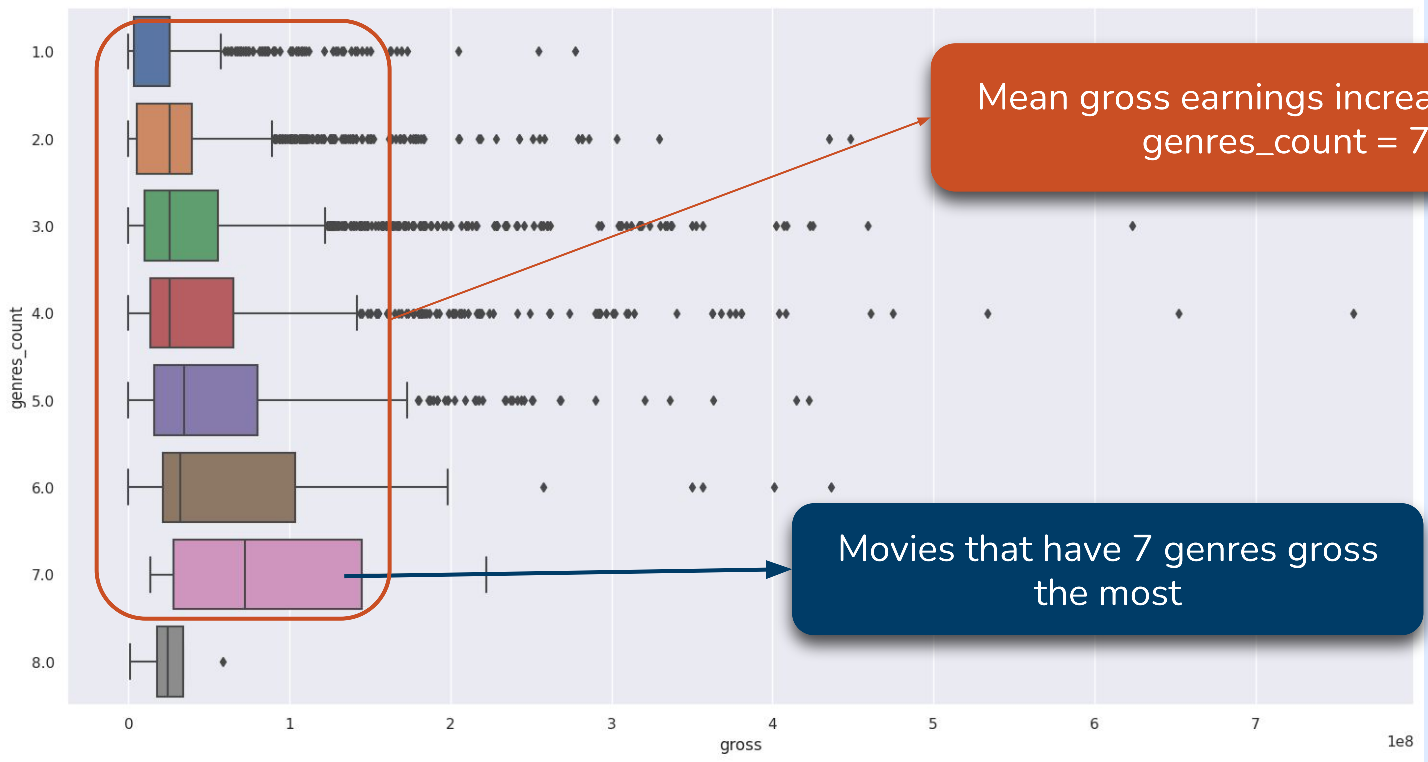


A more obvious trend as compared to actor_1 and actor_2 counts

actor_3 appearing in the most # of movies gross the most

genres_count

Box Plot

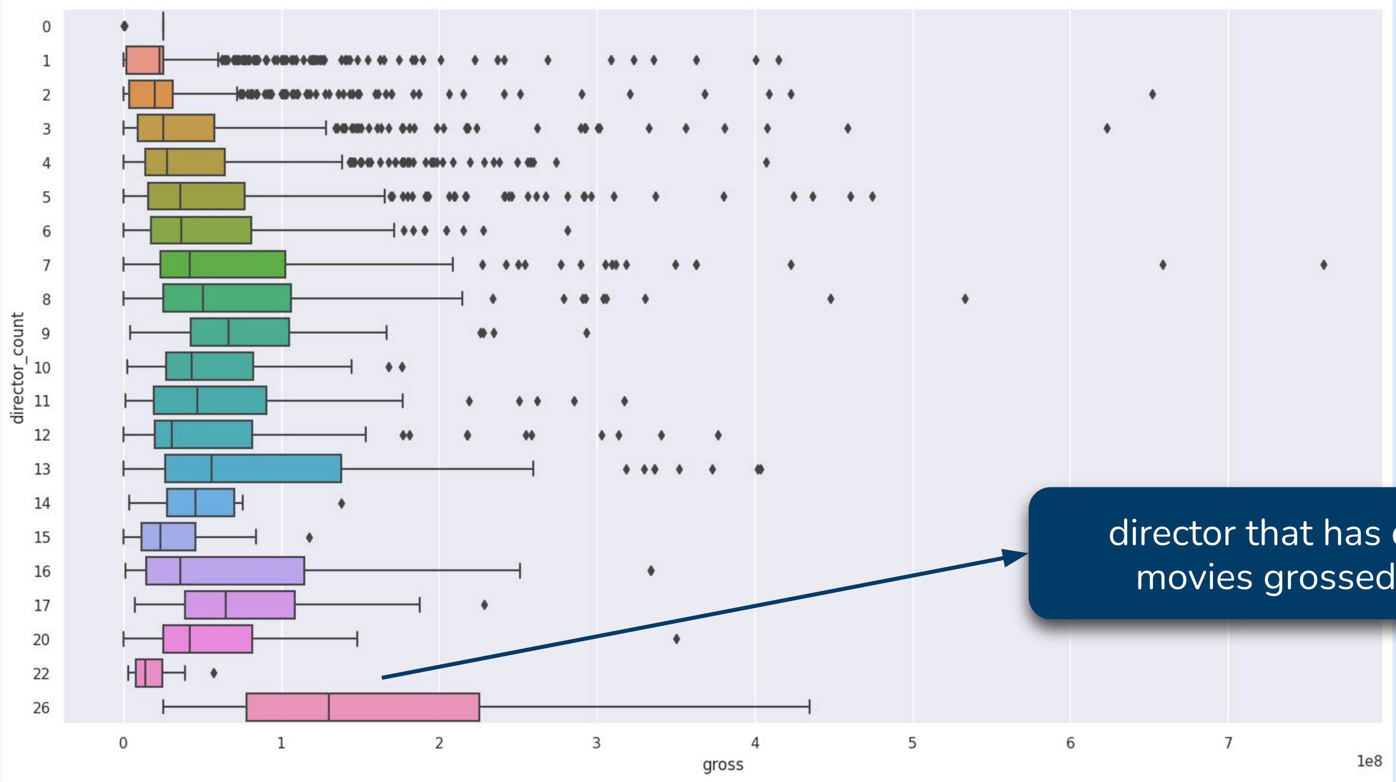


Mean gross earnings increases up till genres_count = 7

Movies that have 7 genres gross the most

director_count

Box Plot



director that has directed 26 movies grossed the most

Data Preparation for ML Models

01

Dimensionality Reduction

For Categorical features

02

Robust Scaler Transformation

For Numerical features

1. Dimensionality Reduction

	count	unique	top	freq
main_keyword_count	4998.0	35.0	1.0	1382.0
actor_3_count	4998	9	1	2669
language	4998	47	English	4674
title_year	4998.0	91.0	2009.0	365.0
content_rating	4998	18	R	2399
main_keyword	4998	2064	No keywords	152
actor_2_count	4998	15	1	2104
actor_1_count	4998	34	1	1369
director_count	4998	21	1	1513
aspect_ratio	4998.0	22.0	2.35	2664.0
country	4998	65	USA	3778
color	4998	2	Color	4791
facenumber_in_poster	4998.0	19.0	0.0	2149.0
genres_count	4998.0	8.0	3.0	1613.0

Features with high cardinality may lead to problems w/ OHE

1. Dimensionality Reduction

Since most categorical features suffer from high cardinality and class imbalance, we implement a function that aggregates dimensions (or unique values) with lower frequency into a 'catch-all' value like 'Other' to reduce high cardinality

```
from collections import Counter

def dimsReduction(column, threshold=0.70, return_categories_list=True):
  #Find the threshold value using the percentage and number of instances in the column
  threshold_value=int(threshold*len(column))
  #Initialise an empty list for our new minimised categories
  categories_list=[]
  #Initialise a variable to calculate the sum of frequencies
  s=0
  #Create a counter dictionary of the form unique_value: frequency
  counts=Counter(column)

  #Loop through the category name and its corresponding
  #frequency after sorting the categories by descending order of frequency
  for i,j in counts.most_common():
    #Add the frequency to the global sum
    s+=dict(counts)[i]
    #Append the category name to the list
    categories_list.append(i)
    #Check if the global sum has reached the threshold value, if so break the loop
    if s>=threshold_value:
      break

  #Append the category Other to the list
  categories_list.append('Other')

  #Replace all instances not in our new categories by Other
  new_column = column.apply(lambda x: x if x in categories_list else 'Other')

  #Return transformed column and unique values if return_categories=True
  if(return_categories_list):
    return pd.Series(new_column), categories_list
  #Return only the transformed column if return_categories=False
  else:
    return pd.Series(new_column)
```

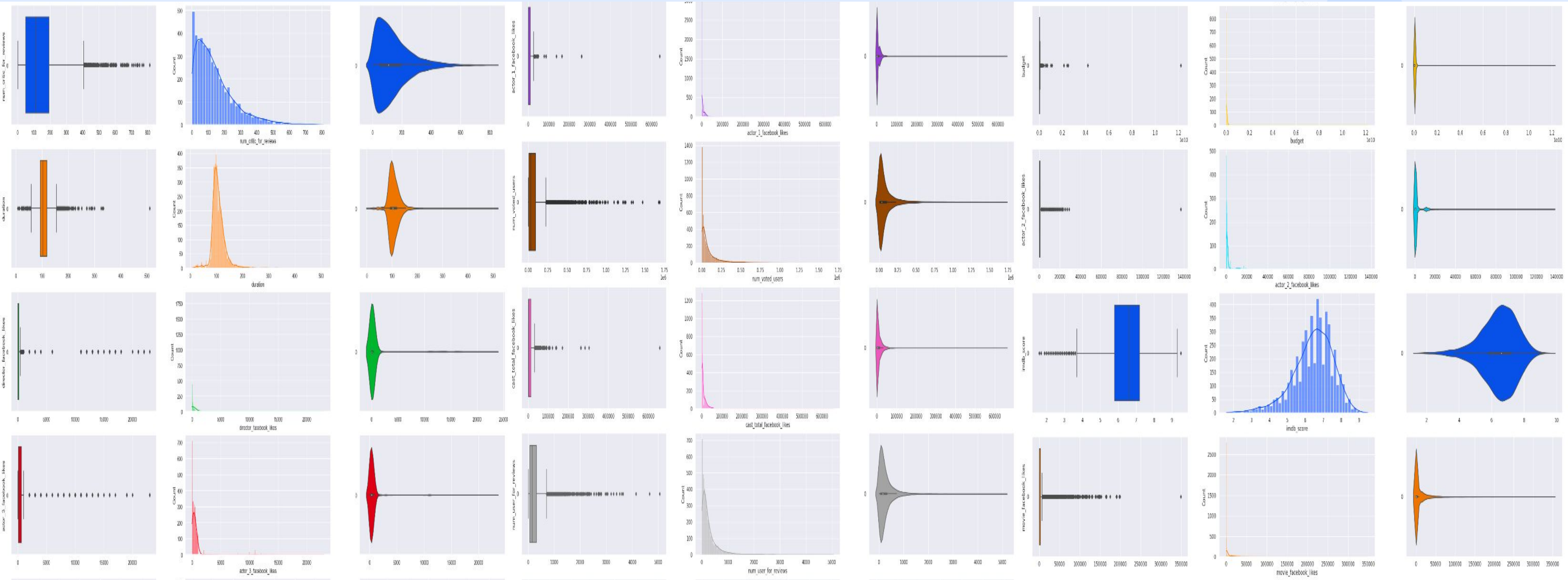
1. Dimensionality Reduction

	count	unique	top	freq
main_keyword_count	4998.0	35.0	1.0	1382.0
actor_3_count	4998	9	1	2669
language	4998	47	English	4674
title_year	4998.0	91.0	2009.0	365.0
content_rating	4998	18	R	2399
main_keyword	4998	2064	No keywords	152
actor_2_count	4998	15	1	2104
actor_1_count	4998	34	1	1369
director_count	4998	21	1	1513
aspect_ratio	4998.0	22.0	2.35	2664.0
country	4998	65	USA	3778
color	4998	2	Color	4791
facenumber_in_poster	4998.0	19.0	0.0	2149.0
genres_count	4998.0	8.0	3.0	1613.0

Aggregating function to reduce cardinality

	count	unique	top	freq
main_keyword_count	4998	11	Other	1415
actor_3_count	4998	3	1	2669
language	4998	2	English	4674
title_year	4998	17	Other	1430
content_rating	4998	3	R	2399
main_keyword	4998	34	Other	3996
actor_2_count	4998	4	1	2104
actor_1_count	4998	11	Other	1403
director_count	4998	6	1	1513
aspect_ratio	4998.0	3.0	2.35	2664.0
country	4998	2	USA	3778
color	4998	2	Color	4791
facenumber_in_poster	4998.0	4.0	0.0	2149.0
genres_count	4998.0	4.0	3.0	1613.0

2. Robust Scaler Transformation



2. Robust Scaler Transformation

	count	mean	std	min	25%	50%	75%	max
num_critic_for_reviews	4998.0	1.395970e+02	1.209164e+02	1.0	50.00	110.0	193.00	8.130000e+02
duration	4998.0	1.072007e+02	2.521190e+01	7.0	93.00	103.0	118.00	5.110000e+02
director_facebook_likes	4998.0	6.754964e+02	2.793896e+03	0.0	7.00	49.0	189.00	2.300000e+04
actor_3_facebook_likes	4998.0	6.386658e+02	1.639613e+03	0.0	134.00	371.5	634.75	2.300000e+04
actor_1_facebook_likes	4998.0	6.549140e+03	1.505247e+04	0.0	613.00	986.0	11000.00	6.400000e+05
gross	4998.0	4.433719e+07	6.234076e+07	162.0	8382841.25	25517500.0	51376923.25	7.605058e+08
num_voted_users	4998.0	8.347020e+04	1.380866e+05	5.0	8560.00	34260.5	96120.75	1.689764e+06
cast_total_facebook_likes	4998.0	9.676941e+03	1.816540e+04	0.0	1405.50	3085.5	13740.50	6.567300e+05
num_user_for_reviews	4998.0	2.715272e+02	3.770563e+02	1.0	65.00	156.0	323.00	5.060000e+03
budget	4998.0	3.782366e+07	1.967122e+08	218.0	7000000.00	20000000.0	40000000.00	1.221550e+10
actor_2_facebook_likes	4998.0	1.640273e+03	4.026032e+03	0.0	281.00	595.0	912.75	1.370000e+05
imdb_score	4998.0	6.441056e+00	1.124107e+00	1.6	5.80	6.6	7.20	9.500000e+00
movie_facebook_likes	4998.0	7.487430e+03	1.929073e+04	0.0	0.00	162.5	3000.00	3.490000e+05

Variables have vastly different ranges

2. Robust Scaler Transformation

Standardization (Z-score)

$$\frac{x - mean}{sd}$$

Transforms input variable to a standard Gaussian but becomes skewed in the presence of outliers

Normalization (0-1)

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

Transforms distribution of input variable to a range of [0,1] but still does not account for outliers

Robust Scaler Transform

$$\frac{x_i - x_{\text{med}}}{x_{75} - x_{25}}$$

Robust Scaler Transform ignores the outliers (beyond IQR) from the calculation

2. Robust Scaler Transformation

	count	mean	std	min	25%	50%	75%	max
num_critic_for_reviews	3748.0	0.208511	0.850200	-0.750430	-0.419966	0.0	0.580034	4.839931
duration	3748.0	0.163065	0.995204	-4.000000	-0.416667	0.0	0.583333	9.458333
[illegible]	748.0	3.725224	16.171095	-0.270345	-0.231724	0.0	0.768276	126.626207
[illegible]	748.0	0.546674	3.338499	-0.737525	-0.474052	0.0	0.525948	45.170659
[illegible]	748.0	0.553517	1.588564	-0.095256	-0.036852	0.0	0.963148	61.484653
num_voted_users	3748.0	0.581992	1.626931	-0.388589	-0.291053	0.0	0.708947	18.798543
cast_total_facebook_likes	3748.0	0.545468	1.575494	-0.249945	-0.137419	0.0	0.862581	52.657751
num_user_for_reviews	3748.0	0.448900	1.438544	-0.593870	-0.352490	0.0	0.647510	17.283525
budget	3748.0	0.567793	6.524865	-0.606027	-0.393939	0.0	0.606061	3[illegible]
actor_2_facebook_likes	3748.0	1.645307	6.553330	-0.932237	-0.492362	0.0	0.507638	213.717195
imdb_score	3748.0	-0.108782	0.803186	-3.500000	-0.571429	0.0	0.428571	2.071429
movie_facebook_likes	3748.0	2.520312	6.646545	-0.058833	-0.058833	0.0	0.941167	116.274500

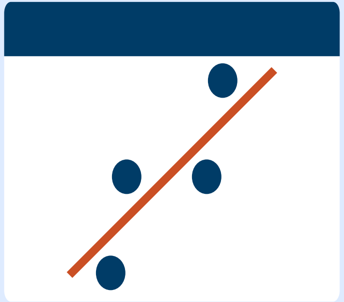
SD significantly reduced and mostly close to 1

Median = 0

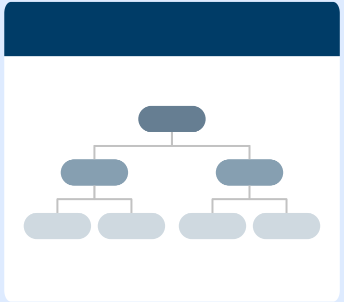
04

Baseline Models

Types of Baseline Models



1. Linear Regression



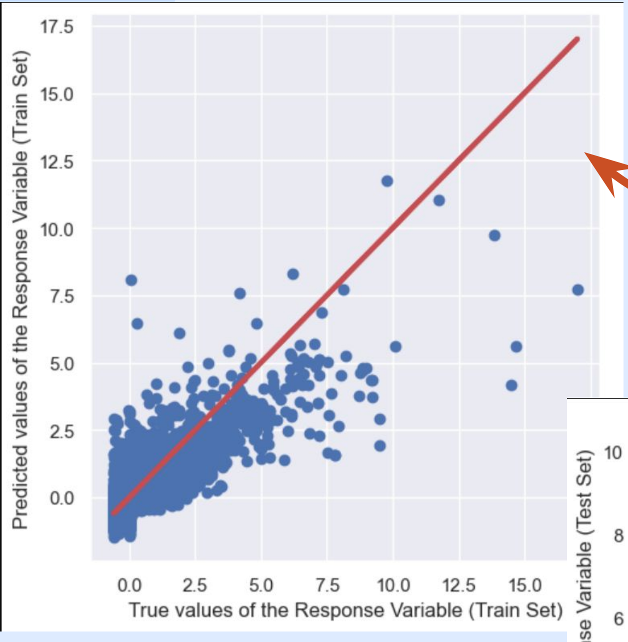
2. Decision Tree

- Prone to overfitting
- Expected to perform worst

3. Random Forest

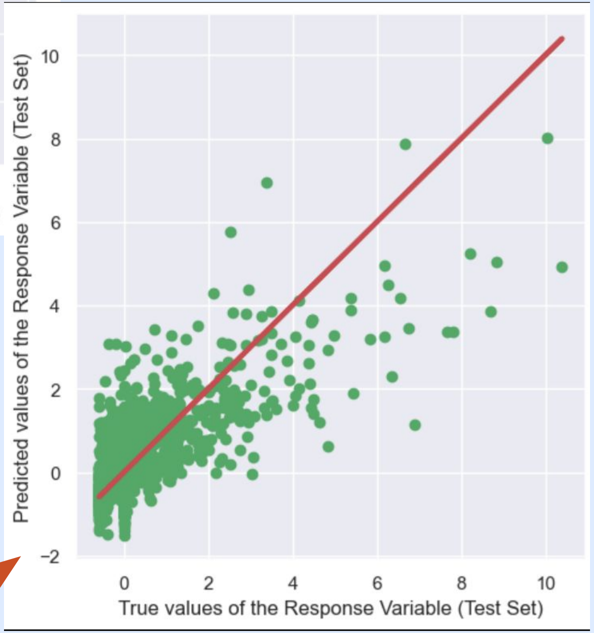
4. Gradient Boosting

- Ensemble learning and boosting
- Expected to perform best



Training Set

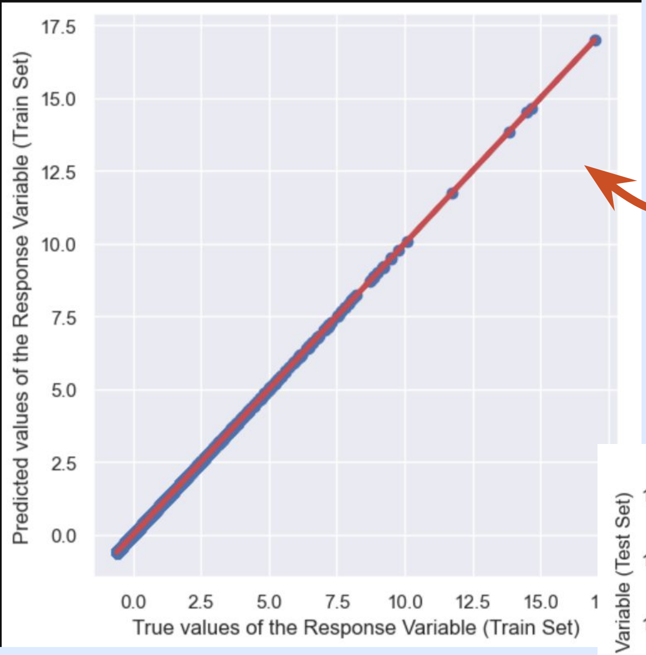
1. Linear Regression



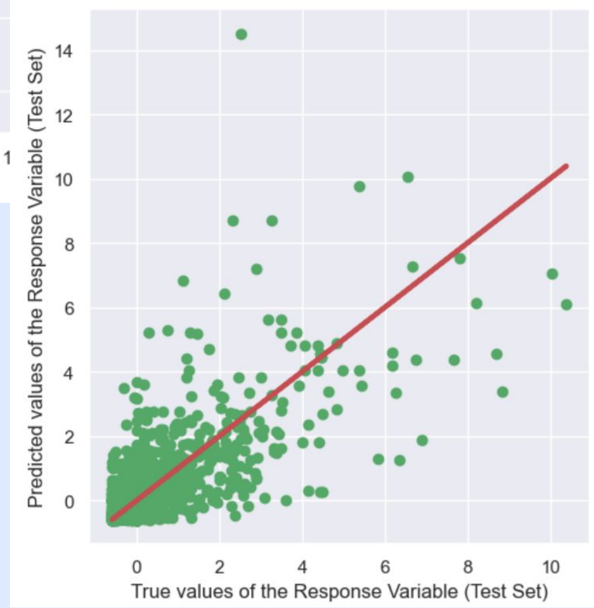
Training Set:
R^2 = 0.600
RMSE = 0.938

Test Set:
R^2 = 0.528
RMSE = 0.903

Test Set



Training Set

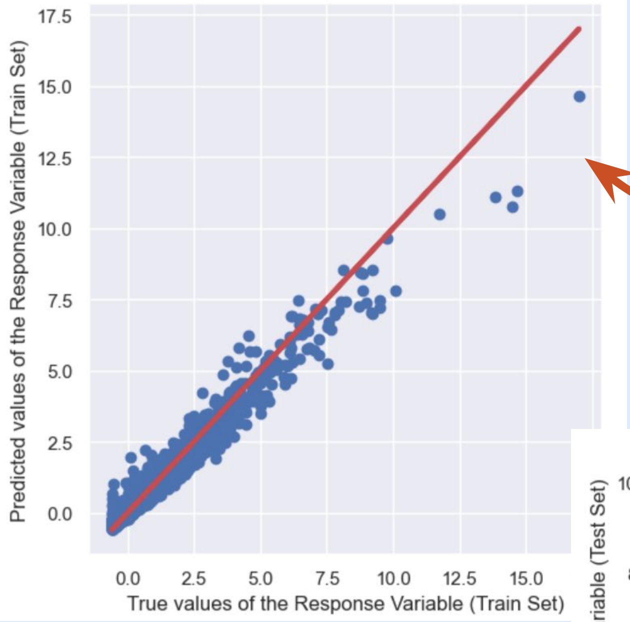


Test Set

2. Decision Tree

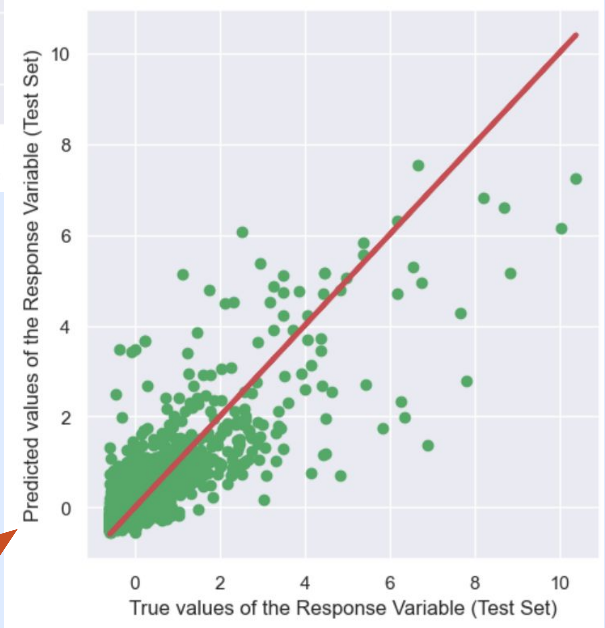
Training Set:
R^2 = 1.000
RMSE = 1.256

Test Set:
R^2 = 0.378
RMSE = 1.037



Training Set

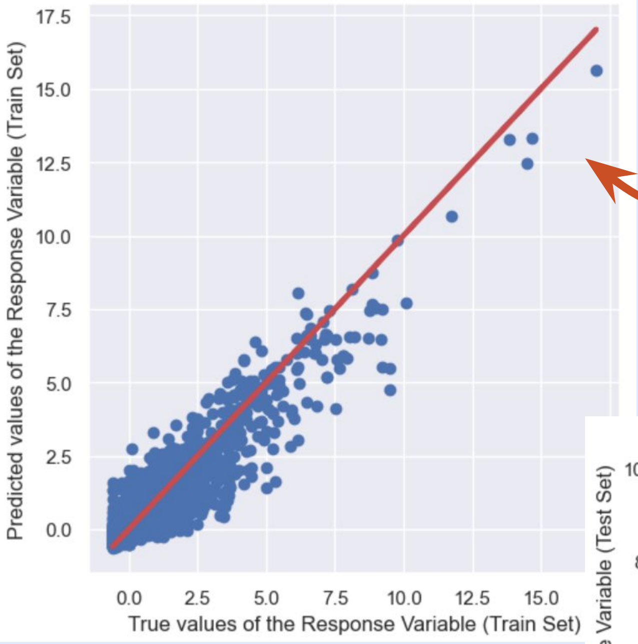
3. Random Forest



Test Set

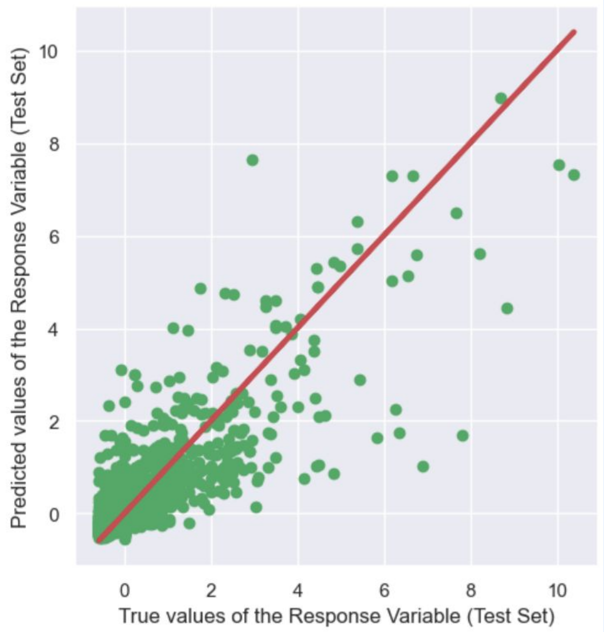
Training Set:
R^2 = 0.960
RMSE = 0.295

Test Set:
R^2 = 0.657
RMSE = 0.770



Training Set

4. Gradient Boosting



Training Set:
R^2 = 0.839
RMSE = 0.594

Test Set:
R^2 = 0.659
RMSE = 0.767

Test Set

Performance of Baseline Models

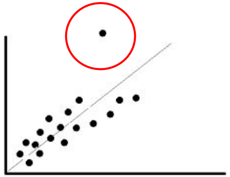
- Best: Gradient Boosting
- Worst: Decision Tree

	Model	Baseline RMSE
0	Linear Regression	0.903338
1	Decision Tree	1.036676
2	Random Forest	0.770219
3	Gradient Boosting	0.767328

05

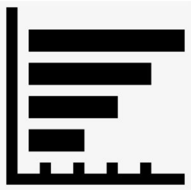
Improvements to Baseline Models

Techniques Used



1. Dealing with Outliers

Removing outliers from numerical features



2. Feature Selection

Feature importances scoring with MDI

3. Hyperparameter Tuning

GridSearchCV with 5-fold cross-validation

1. Dealing with Outliers

- IQR measured for each numerical feature
- Data points which lie outside 3 * IQR are considered as outliers

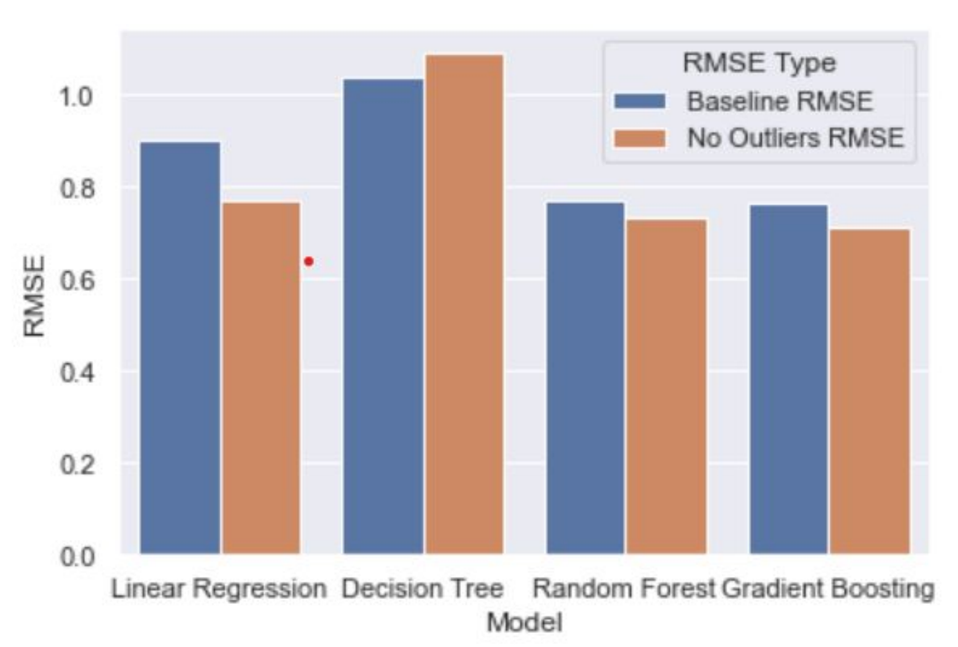
```
Number of data points dropped for num_critic_for_reviews: 206
Number of data points dropped for duration: 217
Number of data points dropped for director_facebook_likes: 523
Number of data points dropped for actor_3_facebook_likes: 106
Number of data points dropped for actor_1_facebook_likes: 96
Number of data points dropped for gross: 235
Number of data points dropped for num_voted_users: 83
Number of data points dropped for cast_total_facebook_likes: 55
Number of data points dropped for num_user_for_reviews: 65
Number of data points dropped for budget: 102
Number of data points dropped for actor_2_facebook_likes: 166
Number of data points dropped for imdb_score: 107
Number of data points dropped for movie_facebook_likes: 378
```

1. Dealing with Outliers

Results

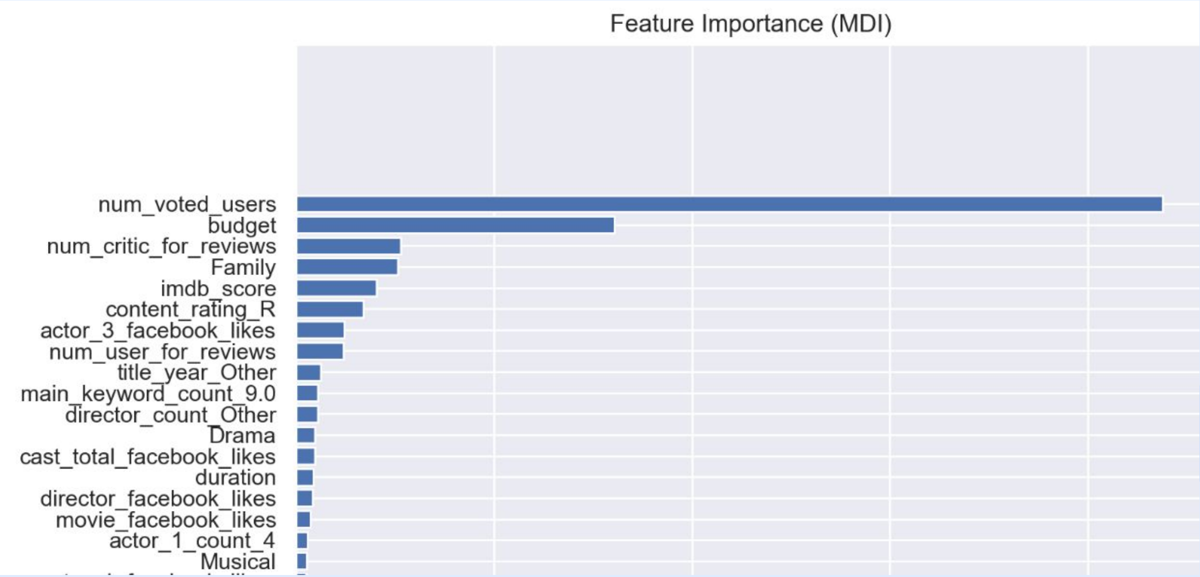
- Drop in RMSE for all models except Decision Tree
- Likely due to overfitting

	Model	Baseline RMSE	No Outliers RMSE
0	Linear Regression	0.903338	0.771406
1	Decision Tree	1.036676	1.090749
2	Random Forest	0.770219	0.731088
3	Gradient Boosting	0.767328	0.713894



2. Feature Selection

- Ranking features using feature importance scores (MDI)
- Higher the MDI → better at reducing impurity
- Gradient Boosting used to rank features as it performed best
- Top 30 features were picked



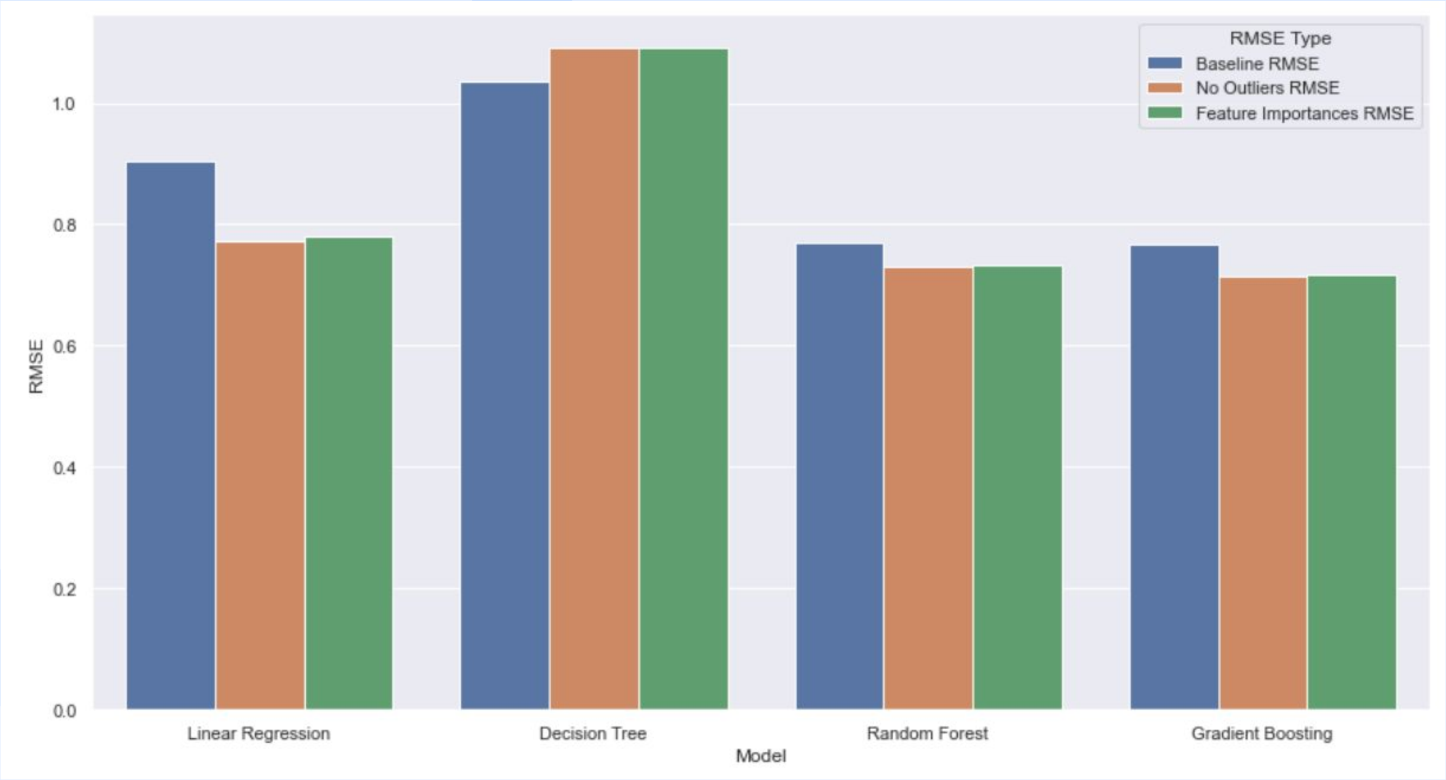
2. Feature Selection

Results

- Drop in RMSE compared to baseline models
- Using top 30 features is equivalent to using entire feature set

	Model	Baseline RMSE	No Outliers RMSE	Feature Importances RMSE
0	Linear Regression	0.903338	0.771406	0.781240
1	Decision Tree	1.036676	1.090749	1.090080
2	Random Forest	0.770219	0.731088	0.731599
3	Gradient Boosting	0.767328	0.713894	0.717653

2. Feature Selection



3. Hyperparameter Tuning

- Only key parameters tuned for each model
- 5-fold cross-validation used
- Expected Decision Tree to improve since overfitting can be resolved by tuning max_depth

```
# Decision Tree Regressor model
model = DecisionTreeRegressor()

# Evaluation
cv = RepeatedKFold(random_state = 0)

# define parameters search space
space = dict()
space['max_depth'] = [2,5,10]
space['min_samples_split'] = [2,5,10]
space['min_samples_leaf'] = [2,5,10]

search = GridSearchCV(model, space, scoring = 'neg_root_mean_squared_error', n_jobs = -1, cv = cv)
result = search.fit(X_train, y_train)
```

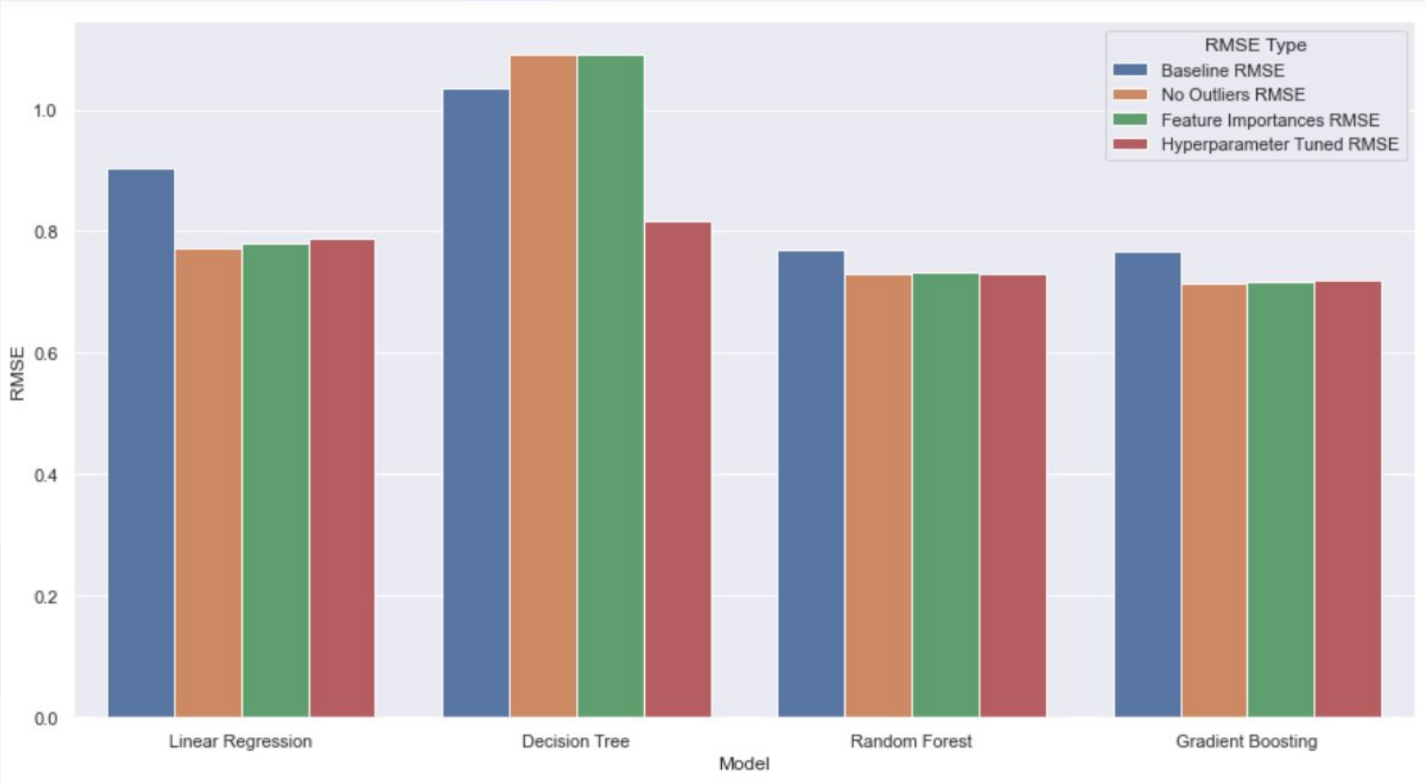
3. Hyperparameter Tuning

Results

- Drop in RMSE compared to baseline models
- Decision Tree improved significantly

	Model	Baseline RMSE	No Outliers RMSE	Feature Importances RMSE	Hyperparameter Tuned RMSE
0	Linear Regression	0.903338	0.771406	0.781240	0.787387
1	Decision Tree	1.036676	1.090749	1.090080	0.818121
2	Random Forest	0.770219	0.731088	0.731599	0.729424
3	Gradient Boosting	0.767328	0.713894	0.717653	0.718928

3. Hyperparameter Tuning



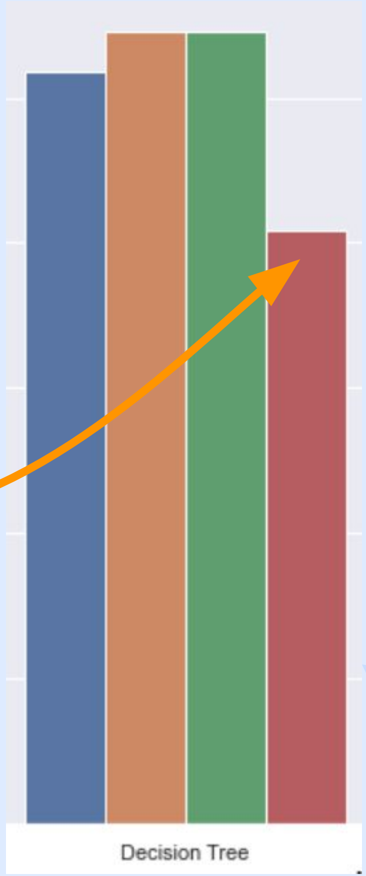
06

Discussions & Conclusions

1. Poor Performance of Decision Tree

- No stopping criteria → split till all children nodes are pure → overfitting

- Only improved after hyperparameter tuning



2. Why Gradient Boosting is the best

Decision Tree

- Prone to overfitting

Random Forest

- Ensemble learning
- More accurate
- More efficient

Gradient Boosting

- Ensemble learning AND boosting
- Focuses on past errors and seeks to minimise them

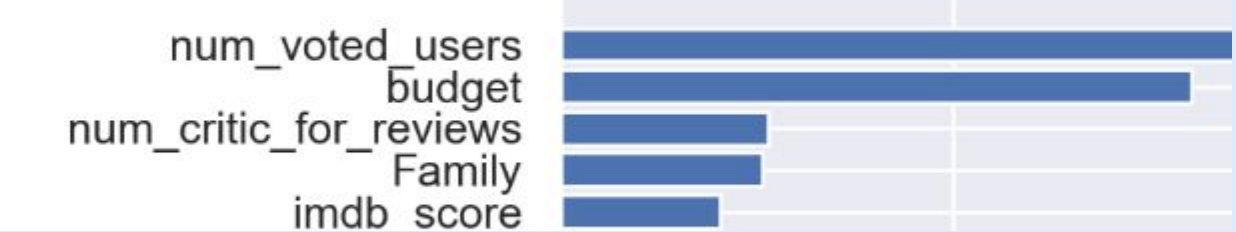
Conclusion

Gradient Boosting

- Best tree-based model to predict top movie revenue
- Performs best in theory
- Results in line with theory

Conclusion

Top Features to Predict Movie Revenue



1. Number of users who voted
2. Budget
3. Number of critic reviews
4. Whether the movie is of the genre ‘Family’
5. IMDB Score

Thank you for your attention!