Movie Revenue Prediction

Project overview

The client is a movie studio and they need to be able to predict movie revenue in order to greenlight the project and assign a budget to it. Most of the data is comprised of categorical variables. While the budget for the movie is known in the dataset it is often an unknown variable during the greenlighting process.

Prediction Basis

Since predicting a movie's revenue in itself is a challenge, additionally the budget amount remains elusive makes it a difficult business problem as there is no base-line to establish the prediction.

I- Executive Summary

Analyze available data to predict a movie revenue and finally assign a budget during green-lighting process.

II- Business Understanding

Define Organization

Client is a movie studio in the business of producing, sponsoring and financing movies.

Intended Stakeholders of Data

Movie producers, finance & budget managers.

Define Business Objectives

based on the input parameters, a movie's revenue is to be predicted.

Background

Business Objectives

• Which variables can help in predicting revenue figures? • determine w hich parameters had the most effect on a movie revenue? • Limiting the problem to predicting just the revenue amounts. • Evaluate and cross-validate the revenue figures.

Business Success Criteria

Predicting the movie revenue as accurately as possible, and making the **REVENUE** as the **TARGET VARIABLE** or **OUTCOME**. Since the revenue is a w hole-number, a Regression w ill be developed.

Assumptions, and Constraints

Assumptions: data is accurate and reliable.

Terminology - Code book - Data Dictionary

A lead is a person w ho has indicated interest in your company's product or service in some w ay, shape, or form.

- title - title of the movie
- tagline - few w ords for movie presentation
- revenue - revenue generated by the movie
- budget - planned expenditure
- genres - categorical group of the movie
- homepage - movie promotional w ebsite
- id - movie id
- keyw ords - tags associated w ith the movie
- original_language - original movie language
- overview - movie synopsis
- production_companies - sponsoring and producing comapnies
- production_countries - locations of the movie made
- release_date - movie available for view ing date.
- runtime - movie duration
- spoken_languages - spoken languages in the movie
- status - movie status for view ing

Project Plan

gets updated as per the succeding stages.

modelling planned to use: Multiple Linear Regression, SVM,

III- Data Understanding

In [24]:

```
# Importing the libraries
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import seaborn as sns
import ast


from sklearn.model_selection import train_test_split
from sklearn import model_selection
import xgboost as xgb
from sklearn.linear_model import LinearRegression
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error


%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

plt.rcParams["figure.figsize"] = (20,10)
```

In [2]:

```
# Run multiple commands and get multiple outputs within a single cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

Collect Initial Data

In [6]:
```
# Load CSV Using Python Standard Library
mvrevenue = pd.read_csv('1-MovieRevenue-WorkBook.csv')

mvrevenue.head(1)
```

Out[6]:

	title	tagline	genres	homepage	id	keywords	original_language	overview	p
0	Avatar	Enter the World of Pandora.	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	In the 22nd century, a paraplegic Marine is di...	[

Understanding the Data: Data quality report

In [7]:
```
mvrevenue.iloc[0]
```

Out[7]:
```
title                                                                   Avata
r
tagline                                           Enter the World of Pandor
a.
genres                   [{"id": 28, "name": "Action"}, {"id": 12, "na
m...
homepage                                          http://www.avatarmovie.com
/
id                                                                       1999
5
keywords                 [{"id": 1463, "name": "culture clash"}, {"i
d":...
original_language                                                           e
n
overview                 In the 22nd century, a paraplegic Marine is d
i...
production_companies     [{"name": "Ingenious Film Partners", "id": 28
9...
production_countries     [{"iso_3166_1": "US", "name": "United States
o...
release_date                                                         12/10/200
9
runtime                                                                    16
2
spoken_languages         [{"iso_639_1": "en", "name": "English"}, {"is
o...
status                                                                Release
d
budget                                                                23700000
0
revenue                                                              278796508
7
Name: 0, dtype: object
```

In [8]:
```
# check the column type
element = mvrevenue.iloc[0]['genres']
print(type(element))
print(element)
```

```
<class 'str'>
[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 1
4, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]
```

Since some cols looks like in dictionary format but in actual its string format

This can be handled as first convert the type string into dictionary format and then extract the values.

In [9]:
```
# Create a function to convert string into dict

dict_columns = [ 'genres', 'production_companies','production_countries',
'spoken_languages', 'keywords']

def text_to_dict(df):
    for column in dict_columns:
        df[column] = df[column].apply(lambda x: {} if pd.isna(x) else ast.
literal_eval(x) )
    return df
```

In [10]:
```
#  Convert string into dict
con_data = text_to_dict(mvrevenue)
con_data.head(1)
```

Out[10]:

	title	tagline	genres	homepage	id	keywords	original_language	overview	p
0	Avatar	Enter the World of Pandora.	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	http://www.avatarmovie.com/	19995	[{'id': 1463, 'name': 'culture clash'}, {'id':...	en	In the 22nd century, a paraplegic Marine is di...	[

In [11]:
```
# checking for dict format
element = con_data.iloc[0]['genres']
print(type(element))
print(element)
```

```
<class 'list'>
[{'id': 28, 'name': 'Action'}, {'id': 12, 'name': 'Adventure'}, {'id': 1
4, 'name': 'Fantasy'}, {'id': 878, 'name': 'Science Fiction'}]
```

Initial Data Collection Report

Describing Data at High Level

In [12]:
```
# rowsXcolumns format
con_data.shape

# missing values
con_data.info()
```

Out[12]:
```
(4803, 16)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 16 columns):
title                   4803 non-null object
tagline                 3959 non-null object
genres                  4803 non-null object
homepage                1712 non-null object
id                      4803 non-null int64
keywords                4803 non-null object
original_language       4803 non-null object
overview                4800 non-null object
production_companies    4803 non-null object
production_countries    4803 non-null object
release_date            4802 non-null object
runtime                 4801 non-null float64
spoken_languages        4803 non-null object
status                  4803 non-null object
budget                  4803 non-null int64
revenue                 4803 non-null int64
dtypes: float64(1), int64(3), object(12)
memory usage: 600.5+ KB
```

In [16]:
```
# dataframe bckup copy
conv_data
conv_data = con_data.copy()
```

Out[16]:

	title	tagline	genres	homepage	id	
0	Avatar	Enter the World of Pandora.	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	http://www.avatarmovie.com/	19995	clas
1	Pirates of the Caribbean: At World's End	At the end of the world, the adventure begins.	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	http://disney.go.com/disneypictures/pirates/	285	'oc
2	Spectre	A Plan No One Escapes	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	http://www.sonypictures.com/movies/spectre/	206647	'nai
3	The Dark Knight Rises	The Legend Ends	[{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...	http://www.thedarkknightrises.com/	49026	cor
4	John Carter	Lost in our world, found in another.	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	http://movies.disney.com/john-carter	49529	'nar
5	Spider-Man 3	The battle within.	[{'id': 14, 'name': 'Fantasy'}, {'id': 28, 'na...	http://www.sonypictures.com/movies/spider-man3/	559	'n ider
6	Tangled	They're taking adventure to new lengths.	[{'id': 16, 'name': 'Animation'}, {'id': 10751...	http://disney.go.com/disneypictures/tangled/	38757	{'i
7	Avengers: Age of Ultron	A New Age Has Come.	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	http://marvel.com/movies/movie /193/avengers_ag...	99861	cc
8	Harry Potter and the Half-Blood Prince	Dark Secrets Revealed	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	http://harrypotter.warnerbros.com/harrypottera...	767	'v
9	Batman v Superman: Dawn of Justice	Justice or revenge	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	http://www.batmanvsupermandawnofjustice.com/	209112	cor
10	Superman Returns	NaN	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	http://www.superman.com	1452	wc
11	Quantum of Solace	For love, for hate, for justice, for revenge.	[{'id': 12, 'name': 'Adventure'}, {'id': 28, '...	http://www.mgm.com/view/movie/234/Quantum-of-S...	10764	'k
12	Pirates of the Caribbean: Dead Man's Chest	Jack is back!	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	http://disney.go.com/disneypictures/pirates/	58	'v
13	The Lone Ranger	Never Take Off the Mask	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	http://disney.go.com/the-lone-ranger/	57201	'te

In [13]:
```
# Checking the null values
con_data.isnull().sum()
```

Out[13]:
```
title                      0
tagline                  844
genres                     0
homepage                3091
id                         0
keywords                   0
original_language          0
overview                   3
production_companies       0
production_countries       0
release_date               1
runtime                    2
spoken_languages           0
status                     0
budget                     0
revenue                    0
dtype: int64
```

Since homepage, tagline has the most missing values and dont add any w eight in building prediction model, these can be dropped.

In [17]:
```
con_data = con_data.drop(['tagline', 'homepage'], axis=1)
```

In [18]:
```
# Dropped the rows with missing values since they are few
con_data = con_data.dropna(axis = 0, how ='any')
```

In [19]:
```
con_data.columns
```

Out[19]:
```
Index(['title', 'genres', 'id', 'keywords', 'original_language', 'overvi
ew',
       'production_companies', 'production_countries', 'release_date',
       'runtime', 'spoken_languages', 'status', 'budget', 'revenue'],
      dtype='object')
```

In [20]:
```
# Calculate Correlation
con_data.describe().T
```

Out[20]:

	count	mean	std	min	25%	50%	75%	max
id	4799.0	5.689992e+04	8.823650e+04	5.0	9012.5	14623.0	58461.5	4.470270e+05
runtime	4799.0	1.069031e+02	2.256131e+01	0.0	94.0	103.0	118.0	3.380000e+02
budget	4799.0	2.906593e+07	4.073251e+07	0.0	800000.0	15000000.0	40000000.0	3.800000e+08
revenue	4799.0	8.232920e+07	1.629076e+08	0.0	0.0	19184015.0	92956519.0	2.787965e+09

In [22]:

```
# correcting the dates columns in a standard format
con_data[['release_month','release_day','release_year']]=con_data['release_date'].str.split('/',expand=True).replace(np.nan, -1).astype(int)

#getting the month year and day using the string split function and the / as a delimiter; eg: 5/25/2015 -> month 5/ day 25 / year 2015
con_data.loc[ (con_data['release_year'] <= 19) & (con_data['release_year'] < 100), "release_year"] += 2000

## some rows have 4 digits for the year instead of 2, so the release year < 100 and > 100 is checking that
con_data.loc[ (con_data['release_year'] > 19)  & (con_data['release_year'] < 100), "release_year"] += 1900

releaseDate = pd.to_datetime(con_data['release_date'])

con_data.head()
```

Out[22]:

	title	genres	id	keywords	original_language	overview	production_companies	produc
0	Avatar	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	19995	[{'id': 1463, 'name': 'culture clash'}, {'id':...	en	In the 22nd century, a paraplegic Marine is di...	[{'name': 'Ingenious Film Partners', 'id': 289...	[{'is 'name
1	Pirates of the Caribbean: At World's End	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	285	[{'id': 270, 'name': 'ocean'}, {'id': 726, 'na...	en	Captain Barbossa, long believed to be dead, ha...	[{'name': 'Walt Disney Pictures', 'id': 2}, {'...	[{'is 'name
2	Spectre	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	206647	[{'id': 470, 'name': 'spy'}, {'id': 818, 'name...	en	A cryptic message from Bond's past sends him o...	[{'name': 'Columbia Pictures', 'id': 5}, {'nam...	[{'is
3	The Dark Knight Rises	[{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...	49026	[{'id': 849, 'name': 'dc comics'}, {'id': 853,...	en	Following the death of District Attorney Harve...	[{'name': 'Legendary Pictures', 'id': 923}, {'...	[{'is 'name
4	John Carter	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	49529	[{'id': 818, 'name': 'based on novel'}, {'id':...	en	John Carter is a war-weary, former military ca...	[{'name': 'Walt Disney Pictures', 'id': 2}]	[{'is 'name

In [ ]:

Data Description Report

Issues found:

- dates column
- text to dict column

Exploratory Data Analysis

This section handles the graphs and plots for data exploration.

checking for outliers & anamolies in runtime, budget and revenue

- Univariate visualization

Univariate analysis looks at one feature at a time. When we analyze a feature independently, we are usually mostly interested in the distribution of its values and ignore other features in the dataset.

Below, we will consider different statistical types of features and the corresponding tools for their individual visual analysis.

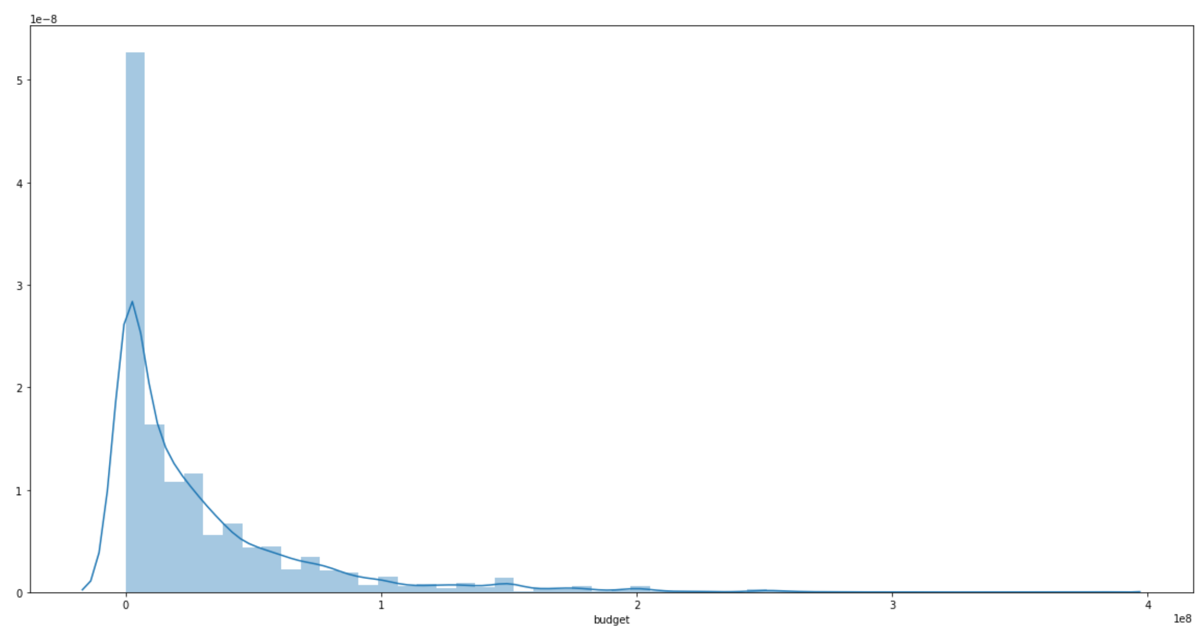
- Quantitative features

Quantitative features take on ordered numerical values. Those values can be discrete, like integers, or continuous, like real numbers, and usually express a count or a measurement.
- Frequency distributions and class distributions

Plotting distplot to check the distribution.

In [25]:
```
sns.distplot(con_data['budget'])
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1a554d43908>



Analysis

the above plots show that the variables contain outliers, with distribution as positively skewed and a thin kurtosis.

In [28]:
```
# creating a df for genres to be compared against revenue, budget, runtime and status
gen = con_data.loc[con_data['genres'].str.len()==1][
['genres','revenue','budget','runtime','status']].reset_index(drop = True)
gen['genres']= gen.genres.apply(lambda x :x[0]['name'])
```

In [29]:
```
genres = gen.groupby(gen.genres).agg('mean')
```

In [30]:
```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.barplot(genres['revenue'],genres.index)

plt.subplot(2,2,2)
sns.barplot(genres['budget'],genres.index)

plt.subplot(2,2,3)
sns.barplot(genres['runtime'],genres.index)
```

Out[30]:
```
<Figure size 1080x720 with 0 Axes>
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a558c3c860>
```

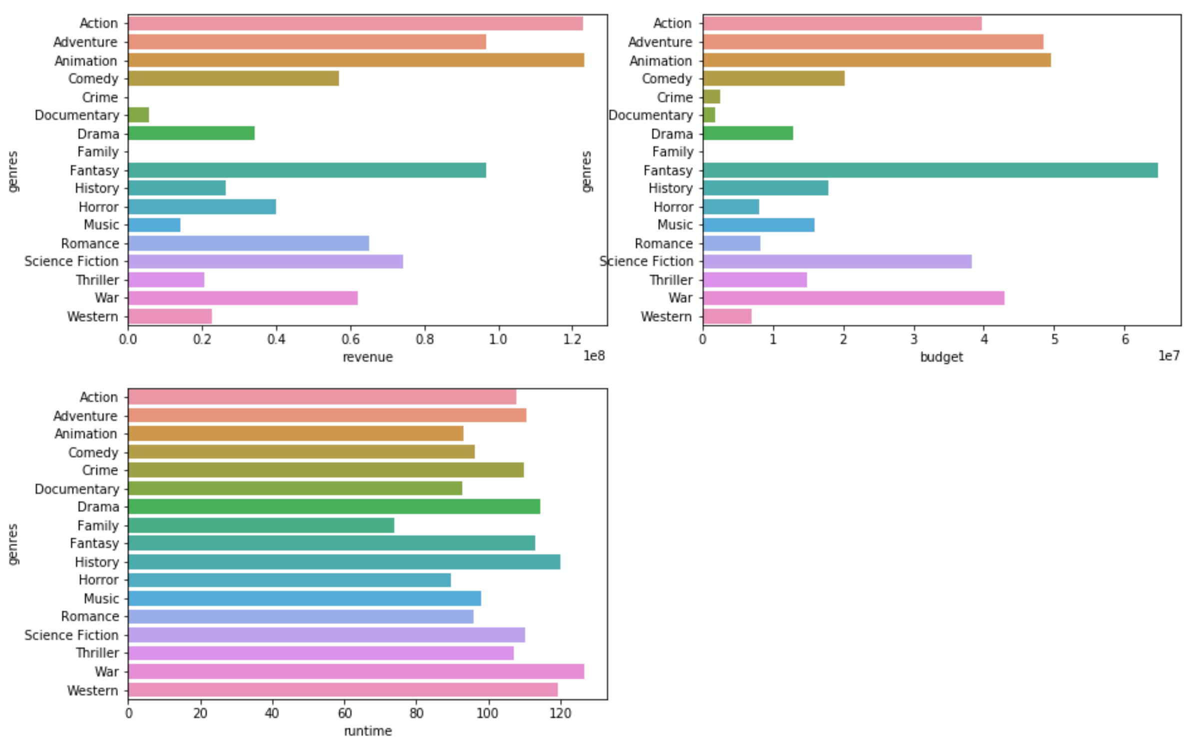
Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a558c3c860>
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a558980390>
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a558980390>
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a558a1f240>
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a558a1f240>
```



In [32]:
```
# Skewness value
con_data.skew()
```

Out[32]:
```
id                2.071986
runtime           0.739876
budget            2.436115
revenue           4.443129
release_month    -0.153424
release_day       0.022664
release_year     -2.170769
dtype: float64
```

In [34]:
```
# Class Distributions genres
class_counts = con_data.groupby('original_language').size()
class_counts
```

Out[34]:
```
original_language
af       1
ar       2
cn      12
cs       2
da       7
de      26
el       1
en    4503
es      32
fa       4
fr      70
he       3
hi      19
hu       1
id       2
is       1
it      13
ja      16
ko      11
ky       1
nb       1
nl       4
no       1
pl       1
ps       1
pt       9
ro       2
ru      11
sl       1
sv       5
ta       2
te       1
th       3
tr       1
vi       1
xx       1
zh      27
dtype: int64
```

In [35]:
```
con_data.original_language.unique()
```

Out[35]:
```
array(['en', 'ja', 'fr', 'zh', 'es', 'de', 'hi', 'ru', 'ko', 'te', 'cn',
       'it', 'nl', 'ta', 'sv', 'th', 'da', 'xx', 'hu', 'cs', 'pt', 'is',
       'tr', 'nb', 'af', 'pl', 'he', 'ar', 'vi', 'ky', 'id', 'ro', 'fa',
       'no', 'sl', 'ps', 'el'], dtype=object)
```

- Data Preparation

DF backup before further modification

In [36]:
```
con_data2 = con_data
con_data.head(1)
```

Out[36]:

	title	genres	id	keywords	original_language	overview	production_companies	production_cou
0	Avatar	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	19995	[{'id': 1463, 'name': 'culture clash'}, {'id':...	en	In the 22nd century, a paraplegic Marine is di...	[{'name': 'Ingenious Film Partners', 'id': 289...	[{'iso_3166_1 'name': 'United

In [37]:
```
element = con_data2.iloc[0]['genres']
element
```

Out[37]:
```
[{'id': 28, 'name': 'Action'},
 {'id': 12, 'name': 'Adventure'},
 {'id': 14, 'name': 'Fantasy'},
 {'id': 878, 'name': 'Science Fiction'}]
```

Converting the distionary columsn to extract the values

In [38]:
```
def parse_dict(raw_dict):
    return [d['name']  for d in raw_dict ]
```

In [39]:
```
def parse_dict(raw_dict):
    return [d['name']  for d in raw_dict ]
```

In [40]:
```
con_data2
```

Out[40]:

	title	genres	id	keywords	original_language	overview	production_cc
0	Avatar	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	19995	[{'id': 1463, 'name': 'culture clash'}, {'id':...	en	In the 22nd century, a paraplegic Marine is di...	[{'name': 'Inger Partners',
1	Pirates of the Caribbean: At World's End	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	285	[{'id': 270, 'name': 'ocean'}, {'id': 726, 'na...	en	Captain Barbossa, long believed to be dead, ha...	[{'name': 'Wa Pictures', 'i
2	Spectre	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	206647	[{'id': 470, 'name': 'spy'}, {'id': 818, 'name...	en	A cryptic message from Bond's past sends him o...	[{'name': Pictures', 'id': 5
3	The Dark Knight Rises	[{'id': 28, 'name': 'Action'}, {'id': 80, 'nam...	49026	[{'id': 849, 'name': 'dc comics'}, {'id': 853,...	en	Following the death of District Attorney Harve...	[{'name': 'L Pictures', 'id':
4	John Carter	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	49529	[{'id': 818, 'name': 'based on novel'}, {'id':...	en	John Carter is a war-weary, former military ca...	[{'name': 'Wa Picture
5	Spider-Man 3	[{'id': 14, 'name': 'Fantasy'}, {'id': 28, 'na...	559	[{'id': 851, 'name': 'dual identity'}, {'id': ...	en	The seemingly invincible Spider-Man goes up ag...	[{'name': Pictures', 'id': 5
6	Tangled	[{'id': 16, 'name': 'Animation'}, {'id': 10751...	38757	[{'id': 1562, 'name': 'hostage'}, {'id': 2343,...	en	When the kingdom's most wanted-and most charmi...	[{'name': 'Wa Pictures', 'i
7	Avengers: Age of Ultron	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	99861	[{'id': 8828, 'name': 'marvel comic'}, {'id': ...	en	When Tony Stark tries to jumpstart a dormant p...	[{'name Studios',
8	Harry Potter and the Half-Blood Prince	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	767	[{'id': 616, 'name': 'witch'}, {'id': 2343, 'n...	en	As Harry begins his sixth year at Hogwarts, he...	[{'name': 'Warr 'id': 6194},
9	Batman v Superman: Dawn of Justice	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	209112	[{'id': 849, 'name': 'dc comics'}, {'id': 7002...	en	Fearing the actions of a god-like Super Hero l...	[{'name': 'DC 'id': 429}, {'na
10	Superman Returns	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	1452	[{'id': 83, 'name': 'saving the world'}, {'id'...	en	Superman returns to discover his 5-year absenc...	[{'name': 'DC 'id': 429}, {'na
11	Quantum of Solace	[{'id': 12, 'name': 'Adventure'}, {'id': 28, '...	10764	[{'id': 627, 'name': 'killing'}, {'id': 1568, ...	en	Quantum of Solace continues the adventures of ...	[{'na Productions', '
12	Pirates of the Caribbean: Dead Man's Chest	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	58	[{'id': 616, 'name': 'witch'}, {'id': 663, 'na...	en	Captain Jack Sparrow works his way out of a bl...	[{'name': 'Wa Pictures', 'i
13	The Lone Ranger	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	57201	[{'id': 1556, 'name': 'texas'}, {'id': 2673, '...	en	The Texas Rangers chase down a gang of outlaws...	[{'name': 'Wa Pictures', 'i

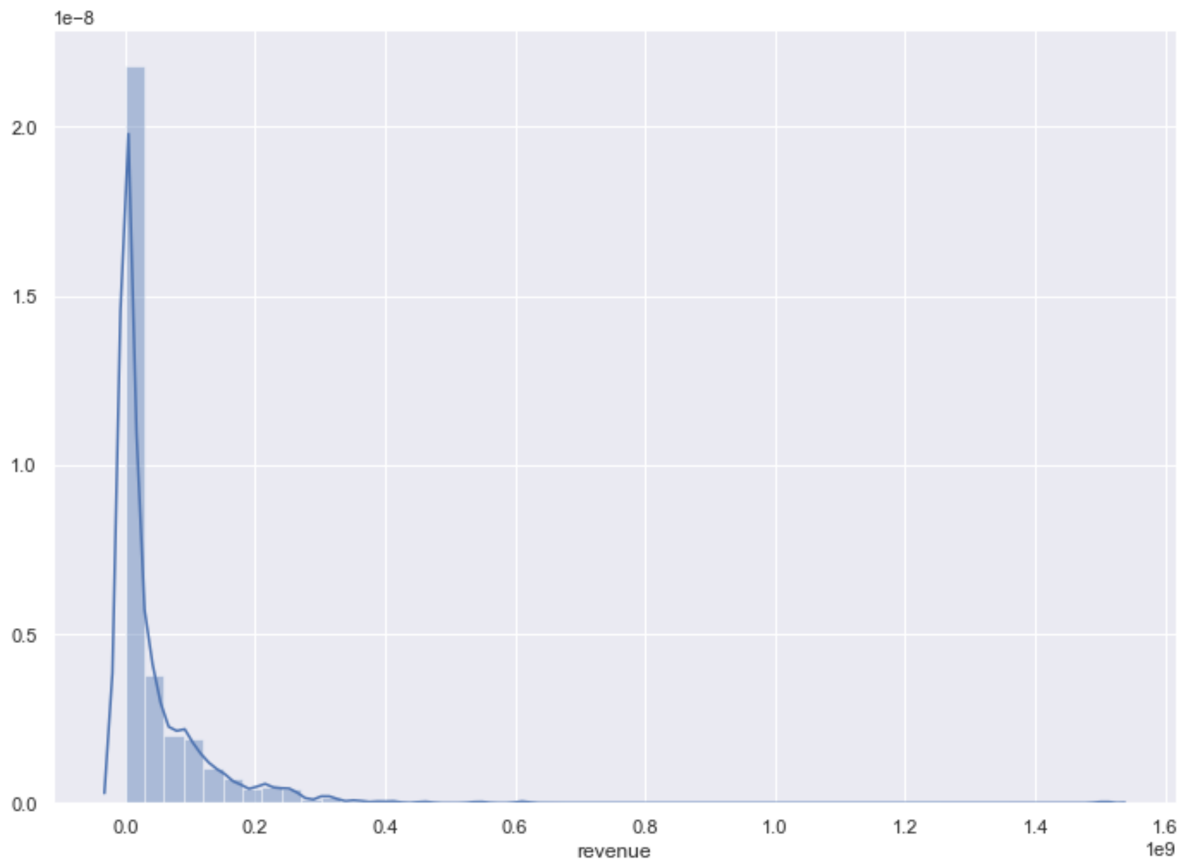
In [41]:
```
# One-Hot-Encoding for all nominal data

df = pd.get_dummies(gen)
```

In [42]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})


sns.distplot(df['revenue'])
```

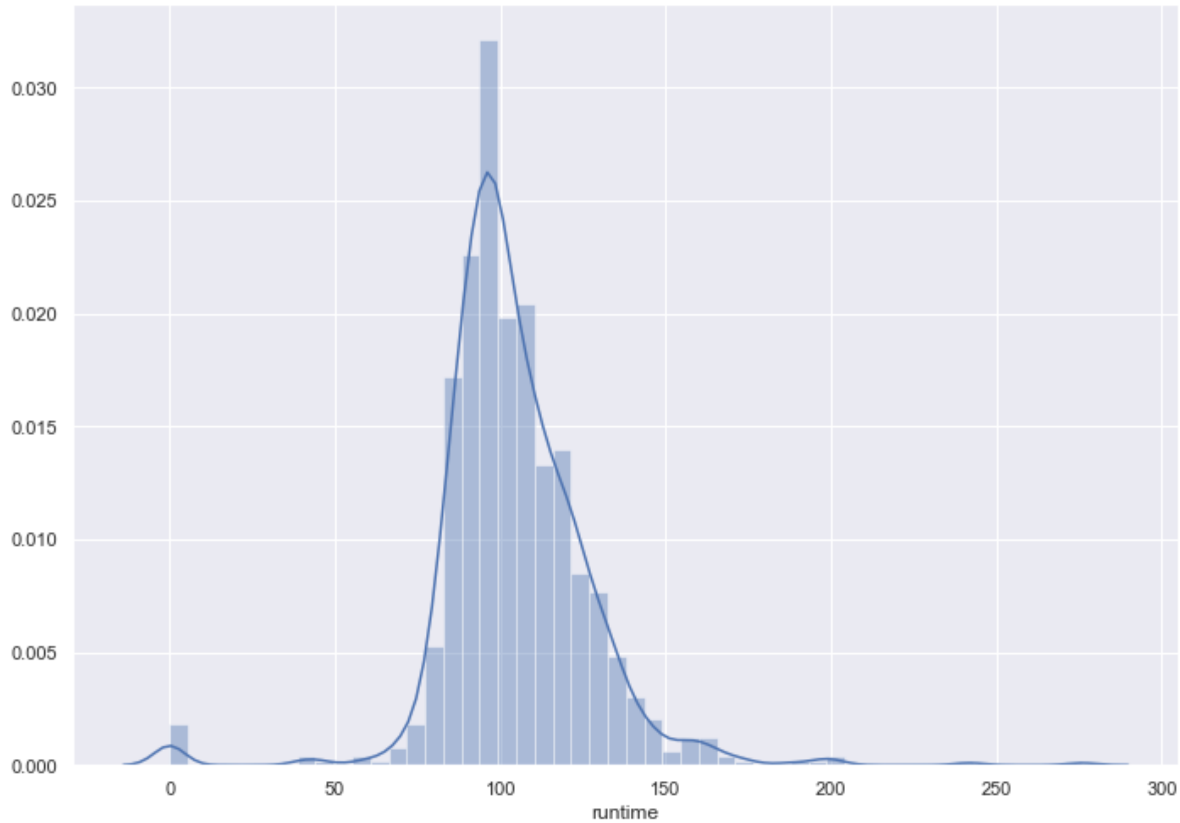
Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x1a55a0dd2e8>



In [43]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})


sns.distplot(df['runtime'])
```

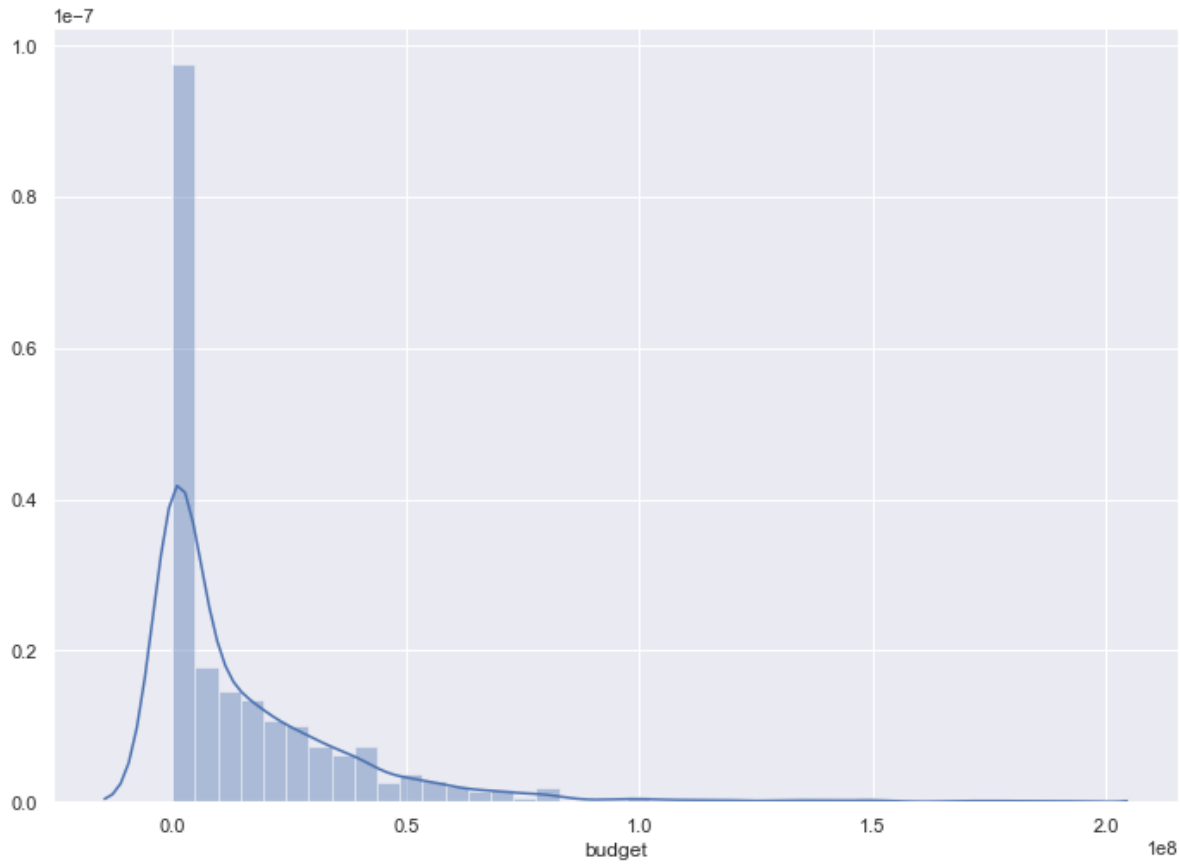
Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x1a55a1f3278>



In [44]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})


sns.distplot(df['budget'])
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x1a55a2c5518>



Modeling

In [46]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('max_colwidth', -1)
```

In [47]:
```
df.head()
```

Out[47]:

	revenue	budget	runtime	genres_Action	genres_Adventure	genres_Animation	genres_Comedy
0	1506249360	190000000	137.0	1	0	0	0
1	543934787	178000000	144.0	0	0	0	0
2	299370084	170000000	113.0	0	0	0	0
3	301000000	150000000	99.0	0	1	0	0
4	202026112	100000000	90.0	0	0	0	1

In [48]:
```
df.columns
```

Out[48]:
```
Index(['revenue', 'budget', 'runtime', 'genres_Action', 'genres_Adventur
e',
       'genres_Animation', 'genres_Comedy', 'genres_Crime',
       'genres_Documentary', 'genres_Drama', 'genres_Family', 'genres_Fa
ntasy',
       'genres_History', 'genres_Horror', 'genres_Music', 'genres_Romanc
e',
       'genres_Science Fiction', 'genres_Thriller', 'genres_War',
       'genres_Western', 'status_Post Production', 'status_Released',
       'status_Rumored'],
      dtype='object')
```

In [49]:
```
df = df[['budget', 'runtime', 'genres_Action', 'genres_Adventure',
       'genres_Animation', 'genres_Comedy', 'genres_Crime',
       'genres_Documentary', 'genres_Drama', 'genres_Family', 'genres_Fant
asy',
       'genres_Horror', 'genres_Science Fiction', 'genres_Thriller',
       'genres_Western', 'status_Released','revenue']]
```

Split the data into training set and testing set using train_test_split

using scikit learn split the data-set

In [50]:
```
df = df[['budget', 'runtime', 'genres_Action', 'genres_Adventure',
       'genres_Animation', 'genres_Comedy', 'genres_Crime',
       'genres_Documentary', 'genres_Drama', 'genres_Family', 'genres_Fant
asy',
       'genres_Horror', 'genres_Science Fiction', 'genres_Thriller',
       'genres_Western', 'status_Released','revenue']]
```

In [51]:
```
# Split-out validation dataset

X = df.drop('revenue', axis = 1)
Y = df.revenue
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test
_size=validation_size, random_state=seed)
```

In [52]:
```
print(X_train)
print(X_validation)
print(Y_train)
print(Y_validation)
```

```
        budget  runtime  genres_Action  genres_Adventure  genres_Animati
on  genres_Comedy  genres_Crime  genres_Documentary  genres_Drama  genre
s_Family  genres_Fantasy  genres_Horror  genres_Science Fiction  genres_
Thriller  genres_Western  status_Released
264  35000000   114.0    0              0                 0
1              0              0                   0             0
0               0              0                      0
0               1
22   79000000   91.0     0              0                 0
1              0              0                   0             0
0               0              0                      0
0               1
815  400000     95.0     0              0                 0
1              0              0                   0             0
0               0              0                      0
0               1
888  27000      92.0     0              0                 0
1              0              0                   0             0
0               0              0                      0
0               1
835  0          95.0     0              0                 0
0              0              0                   1             0
0               0              0                      0
0               1
232  27000000   113.0    0              0                 0
1              0              0                   0             0
0               0              0                      0
0               1
880  50000      111.0    0              0                 0
0              0              1                   0             0
0               0              0                      0
0               1
270  20000000   127.0    0              0                 0
0              0              0                   1             0
0               0              0                      0
0               1
669  35000000   94.0     0              0                 0
0              0              0                   0             0
0               1              0                      0
0               1
316  18000000   111.0    0              0                 0
0              0              0                   1             0
0               0              0                      0
0               1
465  11000000   98.0     0              0                 0
1              0              0                   0             0
0               0              0                      0
0               1
347  26000000   125.0    0              0                 0
0              0              0                   1             0
0               0              0                      0
0               1
142  35000000   91.0     0              0                 0
1              0              0                   0             0
0               0              0                      0
0               1
50   58000000   103.0    0              0                 0
1              0              0                   0             0
0               0              0                      0
0               1
709  1500000    111.0    0              0                 0
0              0              0                   1             0
0               0              0                      0
0               1
```

In [53]:

```
# Test options and evaluation metric
num_folds = 10
num_instances = len(X_train)
seed = 7

# Listing the possible scoring matrix
exv_score = 'explained_variance'
## metrics.explained_variance_score
me_score = 'max_error'
## metrics.max_error
nmeaae_score = 'neg_mean_absolute_error'
## metrics.mean_absolute_error
nmse_score = 'neg_mean_squared_error'
## metrics.mean_squared_error
nsle_score = 'neg_mean_squared_log_error'
## metrics.mean_squared_log_error
nmedae_score = 'neg_median_absolute_error'
## metrics.median_absolute_error
score_score = 'r2'
## metrics.r2_score

# Initiating the score matrix
scoring =  nmedae_score
```

In [54]:

```
random_seed = 12
outcome = []
model_names = []
models = []
models.append(('LR', LinearRegression()))
models.append(('LASSO', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('KNN', KNeighborsRegressor()))
models.append(('CART', DecisionTreeRegressor()))
models.append(('SVR', SVR()))
models.append(('XGB', xgb.XGBRegressor(objective="reg:squarederror")))
```

In [55]:

```
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits =10, random_state = random_see
d)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train,
cv = kfold, scoring = scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: -17967932.655045 (4219441.006266)
LASSO: -17967930.965429 (4219439.547439)
EN: -15040000.004560 (4472987.695526)
KNN: -16229267.990000 (7725781.534653)
CART: -13916194.200000 (7271114.313782)
SVR: -10271698.870695 (2284528.821899)
XGB: -17941190.300000 (6964612.118534)
```

In [56]:

```
# Validating  LR Score on Testing Set
LR_model = LinearRegression()
LR_model.fit(X_train, Y_train)
Y_validation = LR_model.predict(X_validation)
print('Accuracy LR:', LR_model.score(X_train, Y_train))


# Validating  LASSO Score on Testing Set
LASSO_model = Lasso()
LASSO_model.fit(X_train, Y_train)
Y_validation = LR_model.predict(X_validation)
print('Accuracy LASSO:', LASSO_model.score(X_train, Y_train))

# Validating  EN Score on Testing Set
EN_model = ElasticNet()
EN_model.fit(X_train, Y_train)
Y_validation = EN_model.predict(X_validation)
print('Accuracy EN:', EN_model.score(X_train, Y_train))

# Validating  KNN Score on Testing Set
KNN_model = KNeighborsRegressor()
KNN_model.fit(X_train, Y_train)
Y_validation = KNN_model.predict(X_validation)
print('Accuracy KNN:', KNN_model.score(X_train, Y_train))

# Validating  CART Score on Testing Set
CART_model = DecisionTreeRegressor()
CART_model.fit(X_train, Y_train)
Y_validation = CART_model.predict(X_validation)
print('Accuracy CART:', CART_model.score(X_train, Y_train))

# Validating  SVR Score on Testing Set
SVR_model = SVR()
SVR_model.fit(X_train, Y_train)
Y_validation = SVR_model.predict(X_validation)
print('Accuracy SVR:', SVR_model.score(X_train, Y_train))

# Validating  XGB Score on Testing Set
xgb_model = xgb.XGBRegressor(objective="reg:squarederror")
xgb_model.fit(X_train, Y_train)
Y_validation = xgb_model.predict(X_validation)
print('Accuracy XGB:', xgb_model.score(X_train, Y_train))
```

Out[56]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normaliz
e=False)
```

```
Accuracy LR: 0.40166788154135435
```

Out[56]:
```
Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=No
ne,
      selection='cyclic', tol=0.0001, warm_start=False)
```

```
Accuracy LASSO: 0.40166788154067684
```

Out[56]:
```
ElasticNet(alpha=1.0, copy_X=True, fit_intercept=True, l1_ratio=0.5,
           max_iter=1000, normalize=False, positive=False, precompute=Fa
lse,
           random_state=None, selection='cyclic', tol=0.0001, warm_star
t=False)
```

```
Accuracy EN: 0.38668308907801063
```

Out[56]:
```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform')
```

```
Accuracy KNN: 0.5303015056679152
```

Out[56]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=Non
e,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

```
Accuracy CART: 0.9970428790994333
```

Out[56]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
    gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
    tol=0.001, verbose=False)
```

```
Accuracy SVR: -0.15821293478315646
```

Out[56]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=
0,
             max_depth=3, min_child_weight=1, missing=None, n_estimator
s=100,
             n_jobs=1, nthread=None, objective='reg:squarederror',
             random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weigh
t=1,
             seed=None, silent=None, subsample=1, verbosity=1)
```

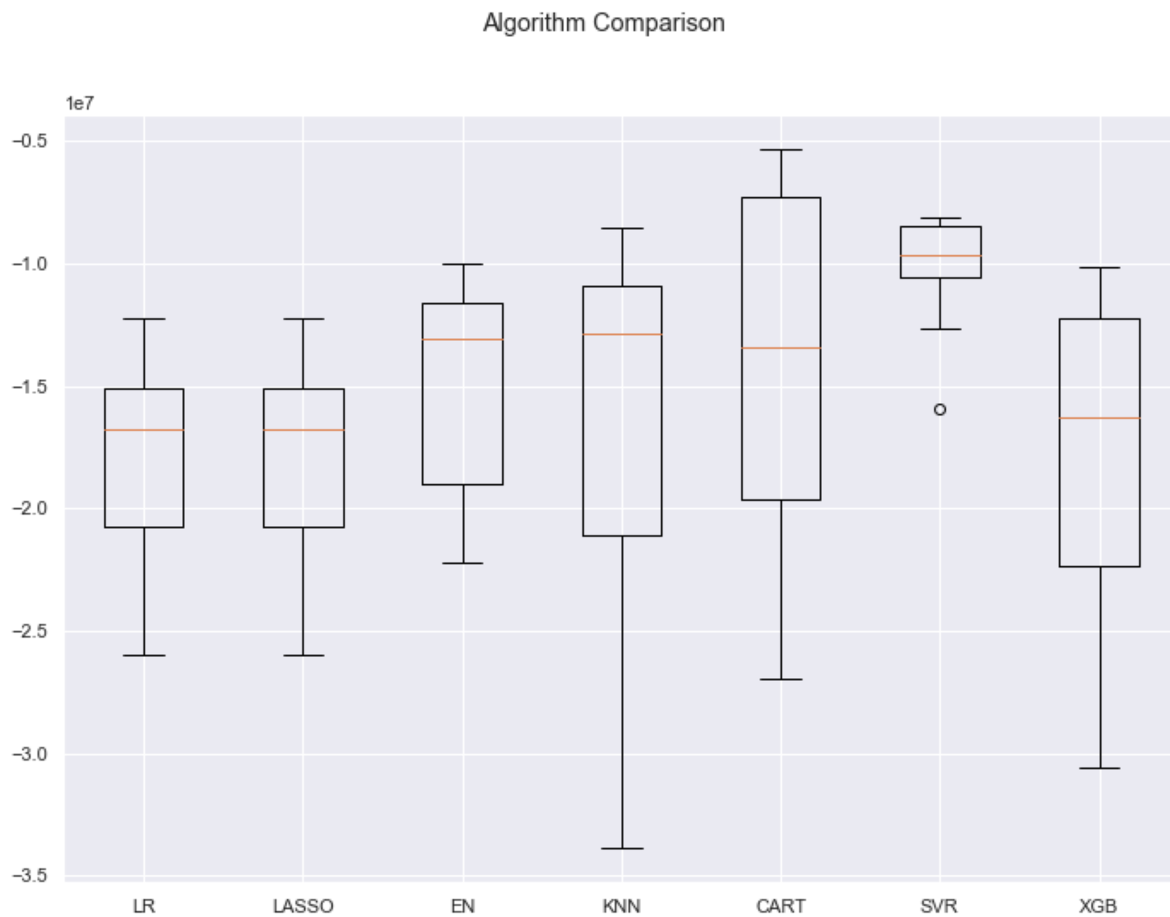
```
Accuracy XGB: 0.7233631870047701
```

In [57]:
```
# Compare Algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
plt.rcParams['figure.figsize'] = 20,10
```

```
Out[57]: Text(0.5, 0.98, 'Algorithm Comparison')

Out[57]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a55bb5cb38>,
           <matplotlib.lines.Line2D at 0x1a55bb5ceb8>,
           <matplotlib.lines.Line2D at 0x1a55bb76438>,
           <matplotlib.lines.Line2D at 0x1a55bb767b8>,
           <matplotlib.lines.Line2D at 0x1a55bb80cf8>,
           <matplotlib.lines.Line2D at 0x1a55bb80e10>,
           <matplotlib.lines.Line2D at 0x1a55bb955f8>,
           <matplotlib.lines.Line2D at 0x1a55bb95978>,
           <matplotlib.lines.Line2D at 0x1a55bba1eb8>,
           <matplotlib.lines.Line2D at 0x1a55bba1fd0>,
           <matplotlib.lines.Line2D at 0x1a55bbb47b8>,
           <matplotlib.lines.Line2D at 0x1a55bbb4b38>,
           <matplotlib.lines.Line2D at 0x1a55bbbfe10>,
           <matplotlib.lines.Line2D at 0x1a55bbcb438>],
          'caps': [<matplotlib.lines.Line2D at 0x1a55bb5cfd0>,
           <matplotlib.lines.Line2D at 0x1a55bb6b5f8>,
           <matplotlib.lines.Line2D at 0x1a55bb76b38>,
           <matplotlib.lines.Line2D at 0x1a55bb76eb8>,
           <matplotlib.lines.Line2D at 0x1a55bb8b438>,
           <matplotlib.lines.Line2D at 0x1a55bb8b7b8>,
           <matplotlib.lines.Line2D at 0x1a55bb95cf8>,
           <matplotlib.lines.Line2D at 0x1a55bb95e10>,
           <matplotlib.lines.Line2D at 0x1a55bbab5f8>,
           <matplotlib.lines.Line2D at 0x1a55bbab978>,
           <matplotlib.lines.Line2D at 0x1a55bbb4eb8>,
           <matplotlib.lines.Line2D at 0x1a55bbb4fd0>,
           <matplotlib.lines.Line2D at 0x1a55bbcb7b8>,
           <matplotlib.lines.Line2D at 0x1a55bbcbb38>],
          'boxes': [<matplotlib.lines.Line2D at 0x1a55bb5c710>,
           <matplotlib.lines.Line2D at 0x1a55bb6bdd8>,
           <matplotlib.lines.Line2D at 0x1a55bb80978>,
           <matplotlib.lines.Line2D at 0x1a55bb8bf98>,
           <matplotlib.lines.Line2D at 0x1a55bba1b38>,
           <matplotlib.lines.Line2D at 0x1a55bbb4438>,
           <matplotlib.lines.Line2D at 0x1a55bbbfcf8>],
          'medians': [<matplotlib.lines.Line2D at 0x1a55bb6b978>,
           <matplotlib.lines.Line2D at 0x1a55bb76fd0>,
           <matplotlib.lines.Line2D at 0x1a55bb8bb38>,
           <matplotlib.lines.Line2D at 0x1a55bba1438>,
           <matplotlib.lines.Line2D at 0x1a55bbabcf8>,
           <matplotlib.lines.Line2D at 0x1a55bbbf5f8>,
           <matplotlib.lines.Line2D at 0x1a55bbcbeb8>],
          'fliers': [<matplotlib.lines.Line2D at 0x1a55bb6bcf8>,
           <matplotlib.lines.Line2D at 0x1a55bb805f8>,
           <matplotlib.lines.Line2D at 0x1a55bb8beb8>,
           <matplotlib.lines.Line2D at 0x1a55bba17b8>,
           <matplotlib.lines.Line2D at 0x1a55bbabe10>,
           <matplotlib.lines.Line2D at 0x1a55bbbf978>,
           <matplotlib.lines.Line2D at 0x1a55bbcbfd0>],
          'means': []}

Out[57]: [Text(0, 0, 'LR'),
          Text(0, 0, 'LASSO'),
          Text(0, 0, 'EN'),
          Text(0, 0, 'KNN'),
          Text(0, 0, 'CART'),
          Text(0, 0, 'SVR'),
          Text(0, 0, 'XGB')]
```


Algorithm Comparison
1e7
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
LR
LASSO
EN
KNN
CART
SVR
XGB

In [58]:

```
# Standardize the dataset
pipelines = []
pipelines.append(('ScaledLR', Pipeline([('Scaler', StandardScaler()),('LR
', LinearRegression())])))
pipelines.append(('ScaledLASSO', Pipeline([('Scaler', StandardScaler()),('
LASSO', Lasso())])))
pipelines.append(('ScaledEN', Pipeline([('Scaler', StandardScaler()),('EN
', ElasticNet())])))
pipelines.append(('ScaledKNN', Pipeline([('Scaler', StandardScaler()),('KN
N', KNeighborsRegressor())])))
pipelines.append(('ScaledCART', Pipeline([('Scaler', StandardScaler()),('C
ART', DecisionTreeRegressor())])))
pipelines.append(('ScaledSVR', Pipeline([('Scaler', StandardScaler()),('SV
R', SVR())])))
results = []
names = []


for name, model in pipelines:
    kfold = model_selection.KFold(n_splits =10, random_state = random_see
d)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train,
cv = kfold, scoring = scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

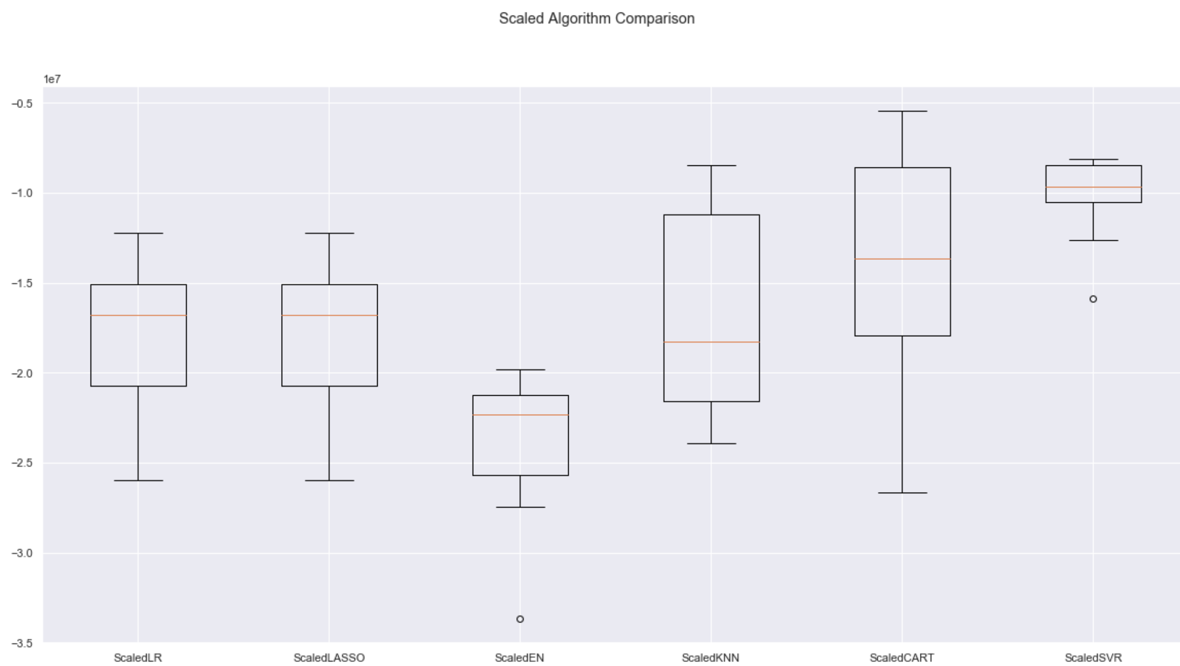
```
ScaledLR: -17967932.655045 (4219441.006266)
ScaledLASSO: -17967932.228152 (4219440.409011)
ScaledEN: -23935946.178337 (4046652.711388)
ScaledKNN: -16744828.980000 (5465308.707168)
ScaledCART: -13686137.225000 (6420439.461538)
ScaledSVR: -10271711.421294 (2284521.757456)
```

In [59]:
```
# Compare Algorithms
fig = plt.figure()
fig.suptitle('Scaled Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
plt.rcParams['figure.figsize'] = 20,10
```

```
Out[59]: Text(0.5, 0.98, 'Scaled Algorithm Comparison')
```

```
Out[59]: {'whiskers': [<matplotlib.lines.Line2D at 0x1a55bc1a4e0>,
            <matplotlib.lines.Line2D at 0x1a55bc1a8d0>,
            <matplotlib.lines.Line2D at 0x1a55bc25e10>,
            <matplotlib.lines.Line2D at 0x1a55bc25f28>,
            <matplotlib.lines.Line2D at 0x1a55bc38710>,
            <matplotlib.lines.Line2D at 0x1a55bc38a90>,
            <matplotlib.lines.Line2D at 0x1a55bc43fd0>,
            <matplotlib.lines.Line2D at 0x1a55bc43f60>,
            <matplotlib.lines.Line2D at 0x1a55bc598d0>,
            <matplotlib.lines.Line2D at 0x1a55bc59c50>,
            <matplotlib.lines.Line2D at 0x1a55bc65f28>,
            <matplotlib.lines.Line2D at 0x1a55bc6f550>],
           'caps': [<matplotlib.lines.Line2D at 0x1a55bc1ac50>,
            <matplotlib.lines.Line2D at 0x1a55bc1afd0>,
            <matplotlib.lines.Line2D at 0x1a55bc2e550>,
            <matplotlib.lines.Line2D at 0x1a55bc2e8d0>,
            <matplotlib.lines.Line2D at 0x1a55bc38e10>,
            <matplotlib.lines.Line2D at 0x1a55bc38f28>,
            <matplotlib.lines.Line2D at 0x1a55bc50710>,
            <matplotlib.lines.Line2D at 0x1a55bc50a90>,
            <matplotlib.lines.Line2D at 0x1a55bc59fd0>,
            <matplotlib.lines.Line2D at 0x1a55bc59f60>,
            <matplotlib.lines.Line2D at 0x1a55bc6f8d0>,
            <matplotlib.lines.Line2D at 0x1a55bc6fc50>],
           'boxes': [<matplotlib.lines.Line2D at 0x1a55bc1a080>,
            <matplotlib.lines.Line2D at 0x1a55bc25ac8>,
            <matplotlib.lines.Line2D at 0x1a55bc2ef60>,
            <matplotlib.lines.Line2D at 0x1a55bc43c50>,
            <matplotlib.lines.Line2D at 0x1a55bc59550>,
            <matplotlib.lines.Line2D at 0x1a55bc65e10>],
           'medians': [<matplotlib.lines.Line2D at 0x1a55bc1af60>,
            <matplotlib.lines.Line2D at 0x1a55bc2ec50>,
            <matplotlib.lines.Line2D at 0x1a55bc43550>,
            <matplotlib.lines.Line2D at 0x1a55bc50e10>,
            <matplotlib.lines.Line2D at 0x1a55bc65710>,
            <matplotlib.lines.Line2D at 0x1a55bc6ffd0>],
           'fliers': [<matplotlib.lines.Line2D at 0x1a55bc25710>,
            <matplotlib.lines.Line2D at 0x1a55bc2efd0>,
            <matplotlib.lines.Line2D at 0x1a55bc438d0>,
            <matplotlib.lines.Line2D at 0x1a55bc50f28>,
            <matplotlib.lines.Line2D at 0x1a55bc65a90>,
            <matplotlib.lines.Line2D at 0x1a55bc6ff60>],
           'means': []}
```

```
Out[59]: [Text(0, 0, 'ScaledLR'),
          Text(0, 0, 'ScaledLASSO'),
          Text(0, 0, 'ScaledEN'),
          Text(0, 0, 'ScaledKNN'),
          Text(0, 0, 'ScaledCART'),
          Text(0, 0, 'ScaledSVR')]
```



In [60]:

```
# ensembles
ensembles = []
ensembles.append(('ScaledAB', Pipeline([('Scaler', StandardScaler()),('AB
', AdaBoostRegressor())])))
ensembles.append(('ScaledGBM', Pipeline([('Scaler', StandardScaler()),('GB
M', GradientBoostingRegressor())])))
ensembles.append(('ScaledRF', Pipeline([('Scaler', StandardScaler()),('RF
', RandomForestRegressor())])))
ensembles.append(('ScaledET', Pipeline([('Scaler', StandardScaler()),('ET
', ExtraTreesRegressor())])))
results = []
names = []
for name, model in ensembles:
    kfold = model_selection.KFold(n_splits =10, random_state = random_see
d)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train,
cv = kfold, scoring = scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
ScaledAB: -23116544.313322 (15052863.399744)
ScaledGBM: -16843782.030088 (6860993.908853)
ScaledRF: -14978530.800000 (7501442.295760)
ScaledET: -14995719.135000 (8446222.354777)
```

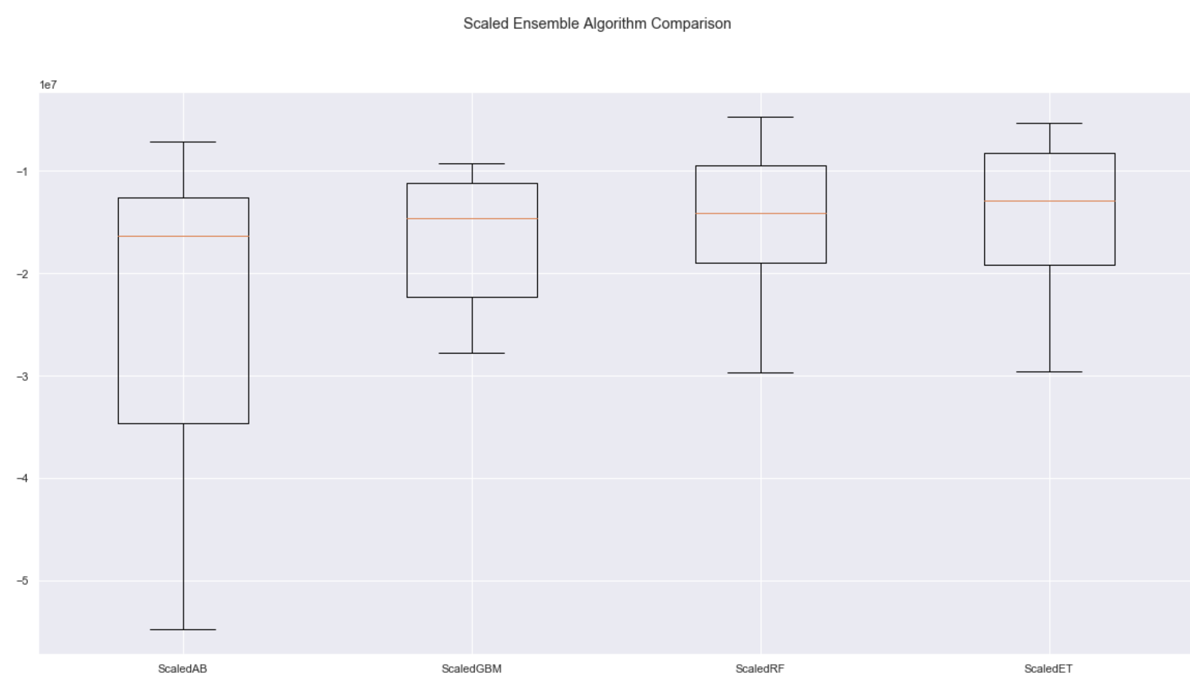
In [61]:

```
# Compare Algorithms
fig = plt.figure()
fig.suptitle('Scaled Ensemble Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
plt.rcParams['figure.figsize'] = 20,10
```

Out[61]:
```
Text(0.5, 0.98, 'Scaled Ensemble Algorithm Comparison')
```

Out[61]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1a55bcf5ac8>,
  <matplotlib.lines.Line2D at 0x1a55bcf5eb8>,
  <matplotlib.lines.Line2D at 0x1a55bd08438>,
  <matplotlib.lines.Line2D at 0x1a55bd087b8>,
  <matplotlib.lines.Line2D at 0x1a55bd13cf8>,
  <matplotlib.lines.Line2D at 0x1a55bd13e10>,
  <matplotlib.lines.Line2D at 0x1a55bd265f8>,
  <matplotlib.lines.Line2D at 0x1a55bd26978>],
 'caps': [<matplotlib.lines.Line2D at 0x1a55bcf5fd0>,
  <matplotlib.lines.Line2D at 0x1a55bcfd5f8>,
  <matplotlib.lines.Line2D at 0x1a55bd08b38>,
  <matplotlib.lines.Line2D at 0x1a55bd08eb8>,
  <matplotlib.lines.Line2D at 0x1a55bd1d438>,
  <matplotlib.lines.Line2D at 0x1a55bd1d7b8>,
  <matplotlib.lines.Line2D at 0x1a55bd26cf8>,
  <matplotlib.lines.Line2D at 0x1a55bd26e10>],
 'boxes': [<matplotlib.lines.Line2D at 0x1a55bcf5668>,
  <matplotlib.lines.Line2D at 0x1a55bcfddd8>,
  <matplotlib.lines.Line2D at 0x1a55bd13978>,
  <matplotlib.lines.Line2D at 0x1a55bd1df98>],
 'medians': [<matplotlib.lines.Line2D at 0x1a55bcfd978>,
  <matplotlib.lines.Line2D at 0x1a55bd08fd0>,
  <matplotlib.lines.Line2D at 0x1a55bd1db38>,
  <matplotlib.lines.Line2D at 0x1a55bd30438>],
 'fliers': [<matplotlib.lines.Line2D at 0x1a55bcfdcf8>,
  <matplotlib.lines.Line2D at 0x1a55bd135f8>,
  <matplotlib.lines.Line2D at 0x1a55bd1deb8>,
  <matplotlib.lines.Line2D at 0x1a55bd307b8>],
 'means': []}
```

Out[61]:
```
[Text(0, 0, 'ScaledAB'),
 Text(0, 0, 'ScaledGBM'),
 Text(0, 0, 'ScaledRF'),
 Text(0, 0, 'ScaledET')]
```



Running the Linear Regression Model

In [62]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
def rmsle(y,y0): return np.sqrt(np.mean(np.square(np.log1p(y)-np.log1p(y
0))))
model = reg.fit(X,Y)
y_pred = reg.predict(X)
rmsle = rmsle(y_pred, Y)
print("The linear model has intercept : {}, and coefficients : {}, and the
rmsle is {} ".format(model.intercept_, model.coef_, rmsle) )
```

```
The linear model has intercept : -14644620.058988929, and coefficients :
[ 2.11893237e+00  2.61006990e+05  2.78379824e+07 -1.74944018e+07
  1.13518567e+07  6.30172914e+06 -1.66119971e+07 -5.00613574e+06
 -5.61654625e+06 -1.91841449e+06 -5.27157457e+07  1.68289014e+07
 -1.82836835e+07 -2.13612411e+07 -5.81668017e+06 -2.75148271e+06], and t
he rmsle is 9.36410874773492
```

In [63]:
```
# Build and fit linear regression model
reg_lm = LinearRegression(normalize=True)
reg_lm.fit(X_train, Y_train)

# Calculate the predictions on the test set
pred = reg_lm.predict(X_validation)
```

Out[63]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normaliz
e=True)
```

In [64]:
```
pred
```

Out[64]:
```
array([-2.19615413e+06,  1.17301722e+08,  1.60152199e+07,  5.17659998e+0
6,
        4.75559258e+07, -4.62763531e+06,  1.10256610e+07,  4.09233613e+0
7,
        3.00847413e+07,  3.63919353e+07,  1.65019210e+06,  3.50653250e+0
7,
        5.67876503e+07,  1.24129311e+07,  4.40054836e+05, -1.38925141e+0
7,
       -5.13377653e+05,  1.33745176e+07,  3.95415235e+07,  2.77168851e+0
7,
        2.60362459e+06,  1.07141608e+08,  2.82808808e+07, -2.31106618e+0
7,
        6.85688571e+07,  2.21896516e+07,  4.64732552e+07,  1.04095993e+0
8,
        1.23267064e+08,  4.46069634e+07,  1.13076000e+07,  7.81273325e+0
7,
       -2.72981013e+05,  4.48208905e+05,  5.56320122e+07,  7.11336784e+0
7,
        7.02052020e+07,  7.59018404e+07, -1.23456757e+06,  1.38553109e+0
7,
        2.04842147e+07,  2.77263359e+06,  5.18494580e+06,  3.65717899e+0
7,
        3.91229167e+07,  1.07701943e+08,  6.00144998e+07, -1.64157930e+0
7,
        5.01360861e+07,  2.07852125e+08,  5.01574505e+06,  4.40054836e+0
5,
        1.89058874e+06,  8.07910864e+07,  1.70889233e+08,  7.10370263e+0
7,
        1.30520506e+08,  6.77681255e+07,  1.43361042e+07,  2.45709576e+0
7,
        9.33473049e+06,  8.92271203e+07,  2.53183898e+07,  6.69036746e+0
6,
       -1.54144356e+07,  1.01588371e+08,  2.08041531e+07,  8.62169465e+0
6,
       -1.47496421e+06,  6.05146839e+07,  1.34510597e+08,  1.01131692e+0
8,
        5.02571702e+07,  5.76525847e+07,  3.60548865e+07,  5.01574505e+0
6,
        1.03134406e+08,  9.15359678e+07,  2.84402123e+06,  3.46304071e+0
7,
        1.72208639e+07,  1.42809185e+07,  7.59481855e+07,  1.00490043e+0
8,
        1.20982411e+08,  1.26533277e+07,  2.52523766e+07,  1.02131753e+0
7,
        1.48168975e+07, -3.45417082e+06,  1.84833891e+07,  1.52334880e+0
7,
        2.60714367e+07,  1.48168975e+07,  3.48529033e+07,  2.53782911e+0
7,
        1.36149143e+07, -2.81135082e+05,  5.48889091e+07,  5.44697999e+0
7,
        5.97733161e+06,  1.50572941e+07,  5.89756519e+07,  2.53782635e+0
7,
        1.16917412e+07,  5.53148109e+07,  1.57784840e+07,  1.25493541e+0
7,
        5.73693497e+06,  1.36149143e+07,  4.52315011e+07,  6.36014497e+0
7,
        3.80560779e+06, -2.21765900e+07,  2.30716964e+07,  1.55551978e+0
7,
       -9.94170932e+05,  1.76184547e+07,  8.02136409e+07,  2.37138202e+0
6,
       -3.90644539e+06,  6.16200150e+07,  1.12167277e+08,  3.80560779e+0
6,
        9.15762410e+07,  1.16591552e+07,  2.37892256e+07,  5.82366044e+0
7,
```

In [68]:
```
print('*************Y Shape***************')
Y_validation.shape

print('*************Pred Shape***************')
pred.shape
```

```
*************Y Shape***************
```

Out[68]:
```
(180,)
```

```
*************Pred Shape***************
```

Out[68]:
```
(180,)
```

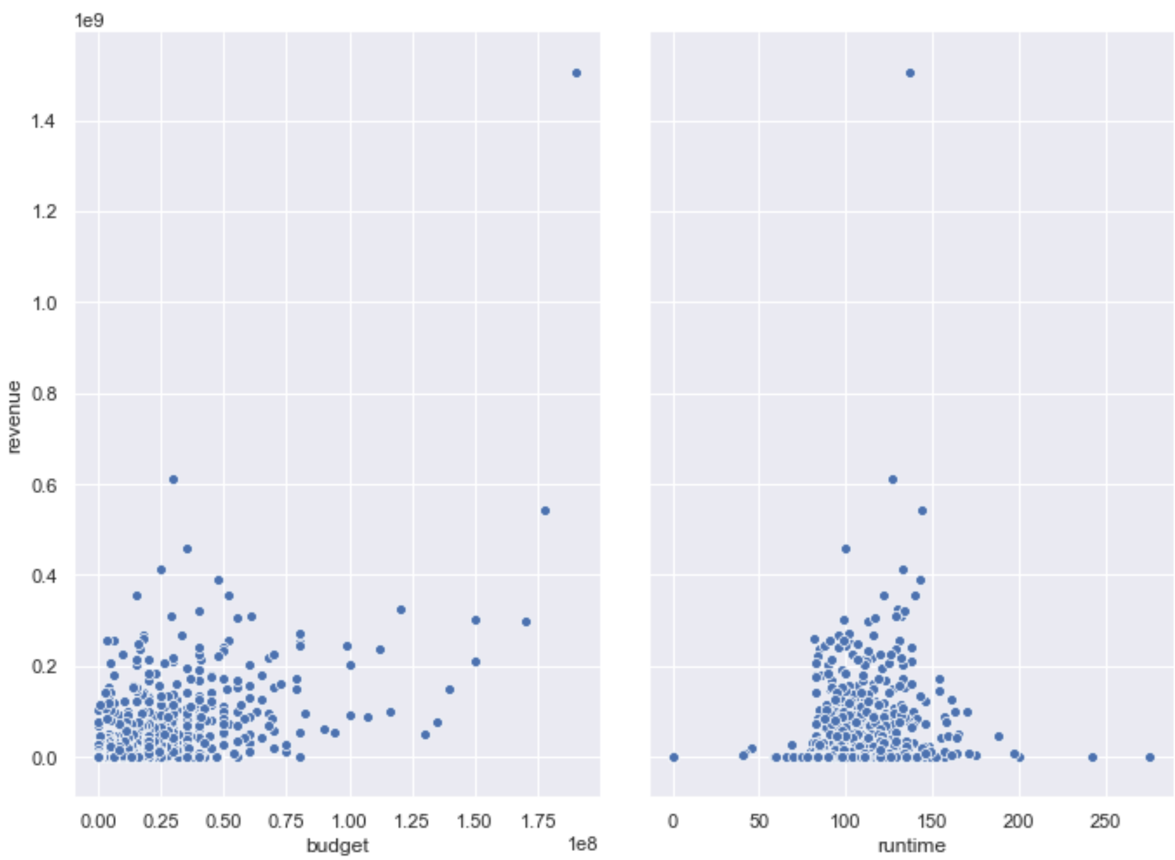
Assess Model

In [69]:
```
# Evaluate the performance using the RMSE
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(Y_validation, pred))
print('RMSE: {:.3f}'.format(rmse))
```

```
RMSE: 19463959.735
```

In [70]:
```
# visualize the relationship between the features and the response using s
catterplots
# sns.pairplot(mvrevenue, x_vars=['budget','release_year','runtime'], y_va
rs='revenue', size=7, aspect=0.7)
sns.pairplot(df, x_vars=[ 'budget', 'runtime'], y_vars='revenue', size=7,
aspect=0.7)
```

Out[70]:
```
<seaborn.axisgrid.PairGrid at 0x1a55bd61160>
```



In [71]:
```
# create X and y
feature_cols = ['budget']
X = df[feature_cols]
y = df.revenue

# instantiate and fit
lm2 = LinearRegression()
lm2.fit(X, y)

# print the coefficients
print(lm2.intercept_)
print(lm2.coef_)
```

Out[71]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normaliz
e=False)

8544030.1107831
[2.17718383]
```

Interpreting Model Coefficients

Interpreting the budget coefficient (β1)

A "unit" increase in budget is associated w ith a 3.34162454 "unit" increase in revenue Or more clearly: An additional $1,000 spent on budget is associated w ith an increase in sales of 2865.28004 w idgets Note here that the coefficients represent associations, not causations

Plotting the Least Squares Line

In [72]:
```
#sns.pairplot(mvrevenue, x_vars=['budget', 'runtime'], y_vars='revenue', s
ize=7, aspect=0.7, kind='reg')
```

In [73]:
```
# create X and y
feature_cols = ['budget','runtime']
X = df[feature_cols]
y = df.revenue

# # instantiate and fit
lm2 = LinearRegression()
lm2.fit(X, y)

# # print the coefficients
print(lm2.intercept_)
print(lm2.coef_)
```

Out[73]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normaliz
e=False)

-5309987.732538827
[2.14442863e+00 1.37603060e+05]
```

In [74]:
```
# # instantiate and fit
lm2 = LinearRegression()
lm2.fit(X, y)
```

Out[74]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normaliz
e=False)
```

Predict with Linear Regression

In [75]:
```
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test
_size=validation_size, random_state=seed)
```

In [76]:
```
#  Remove table meta data, column names   to use values for prediction.

# train_x = df_train_x.values
# train_y = df_train_y.values
# test_x  = df_test_x.values
# X_train = X_train.values
# Y_train = Y_train.values
#  Calculate the coefficients of the linear regression
reg = LinearRegression().fit(X_train, Y_train)

#   Using linear regression model on the prepared test data
Y_validation = reg.predict(X_validation)

# Accuracy
print('Accuracy Linear Regression:', reg.score(X_validation, Y_validatio
n))
```

```
Accuracy Linear Regression: 1.0
```

Predicting with XGBOOST

In [77]:
```
import xgboost as xgb

xgb_model = xgb.XGBRegressor(objective="reg:squarederror", random_state=7)

xgb_model.fit(X_train, Y_train)

Y_validation = xgb_model.predict(X_validation)

print('Accuracy XGB:', xgb_model.score(X_train, Y_train))
```

Out[77]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=
0,
             max_depth=3, min_child_weight=1, missing=None, n_estimator
s=100,
             n_jobs=1, nthread=None, objective='reg:squarederror',
             random_state=7, reg_alpha=0, reg_lambda=1, scale_pos_weigh
t=1,
             seed=None, silent=None, subsample=1, verbosity=1)
```

```
Accuracy XGB: 0.7066468166703885
```

Review Process Review of Process

In [ ]:
```

```

Determine Next Steps

In [ ]:

List of Possible Actions Decision

In [ ]:

Review Recommendations to Organization

In [ ]:

In [ ]: