In This notebook, we will perform the Regression Analysis of the Movie data.

In [497…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

movies = pd.read_csv('./data/tn_moviesource2
```

In [498…
```
def clean_currency(x):
    """ If the value is a string, then remov
    otherwise, the value is numeric and can b
    """
    if isinstance(x, str):
        return(x.replace('$', '').replace(',
    return(x)
```

In [499…
```
colstoclean = ['ProductionBudget', 'Opening W
       'MaximumTheaters', 'TheatricalEngageme
       'DomesticBox Office', 'Infl. Adj. Dom
       'InternationalBox Office', 'Worldwide
for col in colstoclean:
    movies[col] = movies[col].apply(clean_cu
```

In [ ]:
```

```

Need to merge IMDB movie data with Actor

Director influence

Need to merge Bass Model Coeffiecients with Movie data

Need to merge with IMDB movie data including actors and directors

Merge IMDB movie data with Actor/Director influence

In [16]:
```
titleswprins_df = pd.read_csv('./data/titlesv
```

In [32]:
```
act_actress_df = titleswprins_df[(titleswprii
```

In [33]:
```
director_df = titleswprins_df[(titleswprins_(
```

In [35]:
```
director_df.Released.value_counts()
```

Out[35]:
```
2015    326
2016    284
2011    282
2013    264
2012    258
2017    254
2014    251
2010    246
2018    217
2019    211
2020     80
2021     34
Name: Released, dtype: int64
```

In [68]:
```
bigact_wicar_df = pd.read_csv('./data/bigact
```

In [41]:
```
bigdir_wicar_df = pd.read_csv('./data/bigdir_
```

In [707…
```
bigdir_wicar_df[bigdir_wicar_df.director ==
```

Out[707…

	index	mid	title	released	direc
19879	1777	tt0082971	Raiders of the Lost Ark	06-12-1981	Ste Spielb
19880	280	tt0083866	E.T. the Extra-Terrestrial	06-11-1982	Ste Spielb
19881	2894	tt0087469	Indiana Jones and the Temple of Doom	05-23-1984	Ste Spielb

	index	mid	title	released	direc
19882	2647	tt0088939	The Color Purple	02-07-1986	Ste Spielb
19883	1522	tt0092965	Empire of the Sun	12-25-1987	Ste Spielb
19884	201	tt0097576	Indiana Jones and the Last Crusade	05-24-1989	Ste Spielb
19885	2350	tt0096794	Always	12-22-1989	Ste Spielb
19886	897	tt0102057	Hook	12-11-1991	Ste Spielb
19887	1631	tt0107290	Jurassic Park	06-11-1993	Ste Spielb
19888	1865	tt0108052	Schindler's List	02-04-1994	Ste Spielb
19889	954	tt0119567	The Lost World: Jurassic Park	05-23-1997	Ste Spielb
19890	4026	tt0118607	Amistad	12-25-1997	Ste Spielb
19891	3560	tt0120815	Saving Private Ryan	07-24-1998	Ste Spielb

	index	mid	title	released	direc
19892	393	tt0212720	A.I. Artificial Intelligence	06-29-2001	Ste Spielb
19893	416	tt0181689	Minority Report	06-21-2002	Ste Spielb
19894	699	tt0264464	Catch Me If You Can	12-25-2002	Ste Spielb
19895	2942	tt0362227	The Terminal	06-18-2004	Ste Spielb
19896	459	tt0407304	War of the Worlds	06-29-2005	Ste Spielb
19897	2486	tt0408306	Munich	01-06-2006	Ste Spielb
19898	3032	tt0367882	Indiana Jones and the Kingdom of the Crystal S...	05-22-2008	Ste Spielb
19899	182	tt0983193	The Adventures of Tintin	12-21-2011	Ste Spielb
19900	1012	tt1568911	War Horse	12-25-2011	Ste Spielb
19901	1459	tt0443272	Lincoln	11-16-2012	Ste Spielb

	index	mid	title	released	direc
19902	1012	tt3682448	Bridge of Spies	10-16-2015	Ste Spielb
19903	268	tt3691740	The BFG	07-01-2016	Ste Spielb
19904	2066	tt6294822	The Post	01-12-2018	Ste Spielb
19905	1006	tt1677720	Ready Player One	03-29-2018	Ste Spielb
19906	2041	tt3581652	West Side Story	12-10-2021	Ste Spielb

In [581...
```
bigdir_wicar_df[bigdir_wicar_df.title == 'Ja
```

Out[581...

	mid	title	released	director	boxoffice
189	tt0810913	Jack and Jill	11-11-2011	Dennis Dugan	$74,158,157

In [709...
```
lookup_dir_inf('tt1677720')
```

Out[709... 39212066.1

In [708…
```
bigdir_wicar_df.loc[bigdir_wicar_df.mid == '
```

Out[708…
```
39212066.1
```

In [579…
```
#lookup tconst in director_df lookup tconst=
#return direc influence
def lookup_dir_inf(x):
    inf_rat = 0
    try:
        inf_rat = bigdir_wicar_df.loc[bigdir_
        return inf_rat.nlargest(1).sum()
    except:
        return 0
```

In [580…
```
#lookup tconst in director_df lookup tconst=
#return top two actor influence
def lookup_act_inf(movieid):
    inf_rat = 0
    try:
        inf_rat = bigact_wicar_df.loc[bigact_
        return inf_rat.nlargest(2).sum()
    except:
         return 0
```

In [727…
```
#Test Lookup dir Influence
def test_lookup_dir_infl(mid,answer):
    inf = lookup_dir_inf(mid)
    if(inf == answer):
        return True
    else:
        print('Inf', inf, 'answer',answer)
        return False
```

```
#Test Lookup actor Influence
def test_lookup_actor_infl(mid, answer):
    inf = lookup_act_inf('mid')
    if(inf == answer):
        return True
    else:
        print('Inf', inf, 'answer',answer)
        return False
```

Lookup Act Influence

In [747…

```
#Actor Infl
#lookup actor influence for each movie in act
infl = []
for mid in act_actress_df['tconst']:
    rat = lookup_act_inf(mid)
    infl.append(rat)
act_actress_df['act_infl']  = infl
```

```
<ipython-input-747-a58871750136>:7: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a sl
ice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = val
ue instead

See the caveats in the documentation: http
s://pandas.pydata.org/pandas-docs/stable/user
_guide/indexing.html#returning-a-view-versus-
a-copy
  act_actress_df['act_infl']  = infl
```

Lookup Director Influence

In [740…
```
#Director infl
infl = []
for mid in director_df['tconst']:
    rat = lookup_dir_inf(mid)
    infl.append(rat)
director_df['dir_infl'] = infl
```

```
<ipython-input-740-f5f87b8e919e>:5: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a sl
ice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = val
ue instead

See the caveats in the documentation: http
s://pandas.pydata.org/pandas-docs/stable/user
_guide/indexing.html#returning-a-view-versus-
a-copy
  director_df['dir_infl'] = infl
```

Lookup tconst

In [664…
```
#lookup moviesname in titles with prins(acts
#return
def lookup_movie_tconst(moviename):
    mid = ''
    try:
        mid = titleswprins_df.loc[titleswpri
    except IndexError:
        mid = 'N/A'
    return mid
```

Lookup all Titles for the IMDB tcosnt to merge on later.

```
movies_clean.drop_duplicates(ignore_index=Tr
```

In [458...

Out[458...

	Released	ReleasedWorldwide	Title	Theat
0	Jun 21, 2019	Jun 20, 2019	Toy Story 4	
1	Jun 18, 2010	Jun 18, 2010	Toy Story 3	
2	Jun 21, 2013	Jun 21, 2013	Monsters University	
3	Jun 24, 2011	Jun 23, 2011	Cars 2	
4	Jun 16, 2017	Jun 16, 2017	Cars 3	
...	...	...	...	
1302	Oct 25, 2013	Oct 25, 2013	The Counselor	
1303	Jul 12, 2013	Jul 12, 2013	Fruitvale Station	
1304	Nov 1, 2013	Sep 4, 2013	About Time	
1305	Mar 15, 2013	Mar 6, 2013	Spring Breakers	

	Released	ReleasedWorldwide	Title	Theat
1306	Dec 6, 2013	Nov 6, 2013	Inside Llewyn Davis	

1307 rows × 20 columns

In [500…
```
movies['tconst'] = pd.Series([lookup_movie_t
```

In [501…
```
movies_clean = movies[movies['DomesticBox Of
movies_clean['Released_dt'] = pd.to_datetime
movies_clean['year'] = pd.DatetimeIndex(movi
movies_clean = movies_clean[movies_clean.yea
movies_clean.drop_duplicates(inplace=True,ig
movies_clean.dropna(inplace=True)
```

```
<ipython-input-501-5324765e70c1>:2: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a sl
ice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = val
ue instead

See the caveats in the documentation: http
s://pandas.pydata.org/pandas-docs/stable/user
_guide/indexing.html#returning-a-view-versus-
a-copy
  movies_clean['Released_dt'] = pd.to_datetim
e(movies_clean['Released'])
```

```
<ipython-input-501-5324765e70c1>:3: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a sl
ice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = val
ue instead

See the caveats in the documentation: http
s://pandas.pydata.org/pandas-docs/stable/user
_guide/indexing.html#returning-a-view-versus-
a-copy
  movies_clean['year'] = pd.DatetimeIndex(mov
ies_clean['Released_dt']).year
```

In [502…
```
movies_clean.year.value_counts()
```

Out[502…
```
2011    206
2010    199
2016    191
2012    178
2015    177
2013    175
2014    170
2017    161
2018    157
2019    136
Name: year, dtype: int64
```

In [485…
```
#Cleaned 200 movies Titles to match imdb Tit
movies.replace(to_replace='It: Chapter Two',
               value='It Chapter Two', inpla
```

In [671…
```
#final movies clean before merge with bass m
movies_clean = movies_clean[movies_clean.tco
```

In [516…
```
movies_clean['franchise']=0
```

In [ ]:
```
movies = movies[movies.MaximumTheaters != 0]
```

In [518…
```
movies_clean.to_excel('./data/movies_clean.x
```

In [673…
```
movies_clean = pd.read_csv('./data/movies_cl
```

In [750…
```
act_actress_slim_df = act_actress_df[['tcons
```

In [752…
```
director_slim_df = director_df[['tconst', 'T
```

In [751…
```
act_actress_slim_df=act_actress_slim_df[act_
```

Merge Actor and Director Infl to Movies based on tconst movie id

In [764…
```
#merge with act_actress_df and director_df o
merged_df = movies_clean.merge(act_actress_s
```

In [765…
```
#merge with act_actress_df and director_df o
merged_df = merged_df.merge(director_df,left_
```

In [768…
```
merged_df.drop_duplicates(subset='tconst',in
```

In [770…
```
merged_df.columns
```

Out[770…
```
Index(['Released_x', 'ReleasedWorldwide_x',
'Title_x',
       'TheatricalDistributor_x', 'Genre_x',
'Source_x', 'ProductionMethod_x',
       'CreativeType_x', 'ProductionBudget_
x', 'Opening WeekendTheaters_x',
```

```
       'MaximumTheaters_x', 'TheatricalEngage
ments_x',
       'Opening WeekendRevenue_x', 'DomesticB
ox Office_x',
       'Infl. Adj. Dom.Box Office_x', 'Intern
ationalBox Office_x',
       'WorldwideBox Office_x', 'tconst', 'Re
leased_dt', 'year', 'franchise',
       'Title_y', 'runtimeMinutes_x', 'act_in
fl', 'ordering_x', 'nconst_x',
       'Released_y', 'ReleasedWorldwide_y',
'Title', 'TheatricalDistributor_y',
       'Genre_y', 'Source_y', 'ProductionMeth
od_y', 'CreativeType_y',
       'ProductionBudget_y', 'Opening Weekend
Theaters_y', 'MaximumTheaters_y',
       'TheatricalEngagements_y', 'Opening We
ekendRevenue_y',
       'DomesticBox Office_y', 'Infl. Adj. Do
m.Box Office_y',
       'InternationalBox Office_y', 'Worldwid
eBox Office_y', 'TitleLower',
       'titleType', 'primaryTitle', 'original
Title', 'startYear', 'endYear',
       'runtimeMinutes_y', 'primaryTitleLowe
r', 'ordering_y', 'nconst_y',
       'category', 'job', 'characters', 'dir_
infl'],
      dtype='object')
```

In [771…

```
#Reorder Columna
merged_df = merged_df[['tconst', 'Title_x','
       'TheatricalDistributor_x', 'Genre_x',
       'CreativeType_x', 'ProductionBudget_x
       'MaximumTheaters_x', 'TheatricalEngag
       'Opening WeekendRevenue_x', 'Domestic
       'Infl. Adj. Dom.Box Office_x', 'Inter
```

```
        'WorldwideBox Office_x', 'Released_dt
        'Title_y', 'runtimeMinutes_x', 'act_i
        'Released_y', 'ReleasedWorldwide_y',
        'Genre_y', 'Source_y', 'ProductionMet
        'ProductionBudget_y', 'Opening Weeken
        'TheatricalEngagements_y', 'Opening W
        'DomesticBox Office_y', 'Infl. Adj. D
        'InternationalBox Office_y', 'Worldwi
        'titleType', 'primaryTitle', 'origina
        'runtimeMinutes_y', 'primaryTitleLowe
        'category', 'job', 'characters', 'dir
```

In [779…
```
#shrink df
merged_df = merged_df[['tconst', 'Title_x','
          'Source_x', 'ProductionMethod_x',
          'MaximumTheaters_x', 'DomesticBox
        'year', 'franchise', 'runtimeMinute
```

In [781…
```
merged_df.reset_index(inplace=True)
```

In [784…
```
merged_df.drop(columns='index', inplace=True
```

In [785…
```
merged_df
```

Out[785…

	tconst	Title_x	Released_x	TheatricalDist
0	tt2140507	The Current War: Director's Cut	Oct 25, 2019	1(

	tconst	Title_x	Released_x	TheatricalDist
1	tt1606378	A Good Day to Die Hard	Feb 14, 2013	20th Ce
2	tt1611224	Abraham Lincoln: Vampire Hunter	Jun 22, 2012	20th Ce
3	tt2935510	Ad Astra	Sep 20, 2019	20th Ce
4	tt2316204	Alien: Covenant	May 19, 2017	20th Ce
...	...	...	...	
1503	tt2404425	Woman in Gold	Apr 1, 2015	Wei
1504	tt0464154	Piranha 3D	Aug 20, 2010	Weinstein/D
1505	tt1262416	Scream 4	Apr 15, 2011	Weinstein/D
1506	tt1403177	Hesher	May 13, 2011	W Ente
1507	tt2402085	Jimi: All is By My Side	Sep 26, 2014	XLra

1508 rows × 17 columns

In [788…
```
merged_df.rename(inplace=True,columns={"Titl
                                       "TheatricalDistribu
                                       "Genre_x":"genre"
                                       "ProductionMethod_x
                                       "CreativeType_x":"
                                      "MaximumTheaters_x"
                                       "Infl. Adj. Dom.Box
```

In [789…
```
merged_df
```

Out[789…

	tconst	title	released	distribu
0	tt2140507	The Current War: Director's Cut	Oct 25, 2019	101 Stu
1	tt1606378	A Good Day to Die Hard	Feb 14, 2013	20th Century
2	tt1611224	Abraham Lincoln: Vampire Hunter	Jun 22, 2012	20th Century
3	tt2935510	Ad Astra	Sep 20, 2019	20th Century

	tconst	title	released	distribu
4	tt2316204	Alien: Covenant	May 19, 2017	20th Century
...	...	...	...	
1503	tt2404425	Woman in Gold	Apr 1, 2015	Weinstein
1504	tt0464154	Piranha 3D	Aug 20, 2010	Weinstein/Dimens
1505	tt1262416	Scream 4	Apr 15, 2011	Weinstein/Dimens
1506	tt1403177	Hesher	May 13, 2011	Wrekin Entertainm
1507	tt2402085	Jimi: All is By My Side	Sep 26, 2014	XLrator Me

1508 rows × 17 columns

In [794...
```
merged_df.to_csv('./data/merged_regress_df.cs
```

In [801...
```
merged_df['runtime_min'].replace(to_replace=
```

In [891...
```
merged_df['maxtheaters'].replace(to_replace=(
```

In [ ]:

In [817…

```
merged_df['act_infl'].replace(to_replace=0.00
merged_df['dir_infl'].replace(to_replace=0.00
```

In [803…

```
merged_df['runtime_min'] = pd.to_numeric(merg
```

In [959…

```
merged_df['distributor'] = merged_df.distribu
merged_df['genre'] = merged_df.genre.str.repl
merged_df['source'] = merged_df.source.str.re
merged_df['prodmethod'] = merged_df.prodmetho
merged_df['creativetype'] = merged_df.creativ

merged_df['distributor'] = merged_df.distribu
merged_df['genre'] = merged_df.genre.str.repl
merged_df['source'] = merged_df.source.str.re
merged_df['prodmethod'] = merged_df.prodmetho
merged_df['creativetype'] = merged_df.creativ

merged_df['distributor'] = merged_df.distribu
merged_df['genre'] = merged_df.genre.str.repl
merged_df['source'] = merged_df.source.str.re
merged_df['prodmethod'] = merged_df.prodmetho
merged_df['creativetype'] = merged_df.creativ

merged_df['distributor'] = merged_df.distribu
merged_df['genre'] = merged_df.genre.str.repl
merged_df['source'] = merged_df.source.str.re
merged_df['prodmethod'] = merged_df.prodmetho
merged_df['creativetype'] = merged_df.creativ

merged_df['distributor'] = merged_df.distribu
merged_df['genre'] = merged_df.genre.str.repl
merged_df['source'] = merged_df.source.str.re
```

```
merged_df['prodmethod'] = merged_df.prodmeth
merged_df['creativetype'] = merged_df.creati
```

In [927…

```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1508 entries, 0 to 1507
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   tconst            1508 non-null   object
 1   title             1508 non-null   object
 2   released          1508 non-null   object
 3   distributor       1508 non-null   object
 4   genre             1508 non-null   object
 5   source            1508 non-null   object
 6   prodmethod        1508 non-null   object
 7   creativetype      1508 non-null   object
 8   productionbudget  1508 non-null   int64
 9   maxtheaters       1508 non-null   int64
 10  dombox            1508 non-null   int64
 11  inf_dombox        1508 non-null   int64
 12  year              1508 non-null   int64
 13  franchise         1508 non-null   int64
 14  runtime_min       1508 non-null   int64
 15  act_infl          1508 non-null   float6
4
 16  dir_infl          1508 non-null   float6
4
dtypes: float64(2), int64(7), object(8)
memory usage: 200.4+ KB
```

In [960…

```
merged_df.distributor.unique()
```

Out[960… array(['101_Studios', '20th_Century_Fox', 'A24', 'Abramorama_Films',
'AFFRM', 'Alchemy', 'Amazon_Studios',
'Amazon_StudiosRoadside_Attractions', 'Anchor_Bay_Entertainment',
'Annapurna_Pictures', 'Apparition', 'Atlas_Distribution',
'Aviron_Pictures', 'BH_Tilt', 'Bleecker_Street',
'Briarcliffe_Entertainment', 'Broad_Green_Pictures', 'CBS_Films',
'Cinelou_Releasing', 'Clarius_Entertainment',
'Codeblack_Entertainment', 'Cohen_Media_Group', 'Drafthouse_Films',
'Echolight_Studios', 'Electric_Entertainment',
'Elevation_Pictures', 'Entertainment_One',
'Entertainment_Studios_Motion_Pictures', 'EuropaCorp',
'Exclusive_Releasing', 'Fathom_Events', 'FilmDistrict',
'Five_and_Two_Pictures', 'Focus_Features', 'Focus_World',
'Fox_Searchlight', 'Freestyle_Releasing',
'Fun_Academy_Motion_Pictures', 'FUNimation', 'GKIDS',
'Global_Road', 'Good_Deed_Entertainment', 'Gunpowder_and_Sky',
'GVN_Releasing', 'Hammond_Entertainment', 'High_Top_Releasing',
'IFC_Films', 'Image_Entertainment', 'Indican_Pictures',
'LD_Distribution', 'Lionsgate', 'LionsgateRoadside_Attractions',
'Magnet_Pictures', 'Magnolia_Picture

```
s', 'MGM', 'Miramax',
       'Mongrel_Media', 'Monterey_Media', 'Mu
sic_Box_Films', 'Neon',
       'New_Films_Cinema', 'OminFreestyle',
'Open_Road', 'Orion_Pictures',
       'Oscilloscope_Pictures', 'OTL_Releasin
g', 'Overture_Films',
       'Paladin', 'Paramount_Pictures', 'Para
mount_Vantage',
       'Peace_Film_LLC', 'Picturehouse', 'Pur
die_Distribution',
       'Pure_Flix_Entertainment', 'Radius',
'RADiUS_TWC', 'Relativity',
       'Roadside_Attractions', 'Rocky_Mountai
n_Pictures', 'Saban_Films',
       'Samuel_Goldwyn_Films', 'Smith_Global_
Media',
       'Smodshow_Productions', 'Sony_Picture
s', 'Sony_Pictures_Classics',
       'STX_Entertainment', 'Summit_Entertain
ment', 'Sundance_Selects',
       'Ten_Furlongs', 'The_Bubble_Factory',
'The_Orchard',
       'TriBeca_Films', 'United_Artists', 'Un
iversal', 'Variance_Films',
       'Vertical_Entertainment', 'Viva_Entert
ainment',
       'Vivendi_Entertainment', 'Walt_Disne
y', 'Warner_Bros',
       'Weinstein_Co', 'WeinsteinDimension',
'Wrekin_Hill_Entertainment',
       'XLrator_Media'], dtype=object)
```

In [961…

```
merged_df.distributor.unique()
```

Out[961…

```
array(['101_Studios', '20th_Century_Fox', 'A2
4', 'Abramorama_Films',
       'AFFRM', 'Alchemy', 'Amazon_Studios',
```

```
       'Amazon_StudiosRoadside_Attractions',
'Anchor_Bay_Entertainment',
       'Annapurna_Pictures', 'Apparition', 'A
tlas_Distribution',
       'Aviron_Pictures', 'BH_Tilt', 'Bleecke
r_Street',
       'Briarcliffe_Entertainment', 'Broad_Gr
een_Pictures', 'CBS_Films',
       'Cinelou_Releasing', 'Clarius_Entertai
nment',
       'Codeblack_Entertainment', 'Cohen_Medi
a_Group', 'Drafthouse_Films',
       'Echolight_Studios', 'Electric_Enterta
inment',
       'Elevation_Pictures', 'Entertainment_O
ne',
       'Entertainment_Studios_Motion_Picture
s', 'EuropaCorp',
       'Exclusive_Releasing', 'Fathom_Event
s', 'FilmDistrict',
       'Five_and_Two_Pictures', 'Focus_Featur
es', 'Focus_World',
       'Fox_Searchlight', 'Freestyle_Releasin
g',
       'Fun_Academy_Motion_Pictures', 'FUNima
tion', 'GKIDS',
       'Global_Road', 'Good_Deed_Entertainmen
t', 'Gunpowder_and_Sky',
       'GVN_Releasing', 'Hammond_Entertainmen
t', 'High_Top_Releasing',
       'IFC_Films', 'Image_Entertainment', 'I
ndican_Pictures',
       'LD_Distribution', 'Lionsgate', 'Lions
gateRoadside_Attractions',
       'Magnet_Pictures', 'Magnolia_Picture
s', 'MGM', 'Miramax',
       'Mongrel_Media', 'Monterey_Media', 'Mu
sic_Box_Films', 'Neon',
```

```
       'New_Films_Cinema', 'OminFreestyle',
'Open_Road', 'Orion_Pictures',
       'Oscilloscope_Pictures', 'OTL_Releasin
g', 'Overture_Films',
       'Paladin', 'Paramount_Pictures', 'Para
mount_Vantage',
       'Peace_Film_LLC', 'Picturehouse', 'Pur
die_Distribution',
       'Pure_Flix_Entertainment', 'Radius',
'RADiUS_TWC', 'Relativity',
       'Roadside_Attractions', 'Rocky_Mountai
n_Pictures', 'Saban_Films',
       'Samuel_Goldwyn_Films', 'Smith_Global_
Media',
       'Smodshow_Productions', 'Sony_Picture
s', 'Sony_Pictures_Classics',
       'STX_Entertainment', 'Summit_Entertain
ment', 'Sundance_Selects',
       'Ten_Furlongs', 'The_Bubble_Factory',
'The_Orchard',
       'TriBeca_Films', 'United_Artists', 'Un
iversal', 'Variance_Films',
       'Vertical_Entertainment', 'Viva_Entert
ainment',
       'Vivendi_Entertainment', 'Walt_Disne
y', 'Warner_Bros',
       'Weinstein_Co', 'WeinsteinDimension',
'Wrekin_Hill_Entertainment',
       'XLrator_Media'], dtype=object)
```

In [962…

```
merged_df.genre.unique()
```

Out[962…

```
array(['Drama', 'Action', 'Horror', 'Thriller
Suspense', 'Adventure',
       'Comedy', 'Musical', 'ConcertPerforman
ce', 'Romantic_Comedy',
       'Black_Comedy', 'Western', 'Documentar
y'], dtype=object)
```

In [963…
```
merged_df.source.unique()
```

Out[963…
```
array(['Based_on_Real_Life_Events', 'Original
_Screenplay',
       'Based_on_Fiction_BookShort_Story', 'B
ased_on_ComicGraphic_Novel',
       'Based_on_TV', 'Based_on_Game', 'Based
_on_Play',
       'Based_on_Factual_BookArticle', 'Remak
e',
       'Based_on_Religious_Text', 'Spin_Off',
'Based_on_Short_Film',
       'Based_on_Toy', 'Based_on_Movie',
       'Based_on_Folk_TaleLegendFairytale',
'Based_on_Musical_Group',
       'Based_on_Musical_or_Opera', 'Compilat
ion', 'Based_on_Song',
       'Based_on_Theme_Park_Ride'], dtype=obj
ect)
```

In [964…
```
merged_df.prodmethod.unique()
```

Out[964…
```
array(['Live_Action', 'AnimationLive_Action',
'Digital_Animation',
       'Stop_Motion_Animation', 'Hand_Animati
on',
       'Multiple_Production_Methods'], dtype=
object)
```

In [965…
```
merged_df.creativetype.unique()
```

Out[965…
```
array(['Live_Action', 'AnimationLive_Action',
'Digital_Animation',
       'Stop_Motion_Animation', 'Hand_Animati
on',
```

```
       'Multiple_Production_Methods'], dtype=
object)
```

In [966…
```
merged_df.describe().T
```

Out[966…

	count	mean	st
productionbudget	1508.0	4.986676e+07	5.762856e+0
maxtheaters	1508.0	2.441458e+03	1.294936e+0
dombox	1508.0	6.574160e+07	9.473145e+0
inf_dombox	1508.0	7.138618e+07	1.014798e+0
year	1508.0	2.014338e+03	2.821794e+0
franchise	1508.0	2.015915e-01	4.013218e-0
runtime_min	1508.0	1.083309e+02	1.781256e+0
act_infl	1508.0	2.248539e+08	3.863954e+0
dir_infl	1508.0	7.774019e+06	2.746204e+0

In [967…
```
del movies_ohe
del preprocessed
```

In [968…
```
continuous = ['productionbudget', 'maxtheate
categoricals = ['distributor', 'genre', 'sou
```

In [942…
```

```

In [969…

```
movies_cont = merged_df[continuous]

# log features
log_names = [f'{column}_log' for column in mo

movies_log = np.log(movies_cont)
movies_log.columns = log_names

# normalize (subract mean and divide by std)
def normalize(feature):
    return (feature - feature.mean()) / featu

movies_log_norm = movies_log.apply(normalize

# one hot encode categoricals
movies_ohe = pd.get_dummies(merged_df[categor
preprocessed = pd.concat([movies_log_norm,mer
```

Baseline Model- regression on inflation domestic boxoffice

In [970…

```
from statsmodels.formula.api import ols
outcome = 'inf_dombox_log'
predictor_variables = "+".join(preprocessed.c
formula = outcome + '~' + predictor_variables
print(predictor_variables)
print(formula)
```

```
productionbudget_log+maxtheaters_log+dombox_l
og+inf_dombox_log+runtime_min_log+act_infl_lo
g+dir_infl_log+franchise+distributor_20th_Cen
tury_Fox+distributor_A24+distributor_AFFRM+di
stributor_Abramorama_Films+distributor_Alchem
```

y+distributor_Amazon_Studios+distributor_Amaz
on_StudiosRoadside_Attractions+distributor_An
chor_Bay_Entertainment+distributor_Annapurna_
Pictures+distributor_Apparition+distributor_A
tlas_Distribution+distributor_Aviron_Pictures
+distributor_BH_Tilt+distributor_Bleecker_Str
eet+distributor_Briarcliffe_Entertainment+dis
tributor_Broad_Green_Pictures+distributor_CBS
_Films+distributor_Cinelou_Releasing+distribu
tor_Clarius_Entertainment+distributor_Codebla
ck_Entertainment+distributor_Cohen_Media_Grou
p+distributor_Drafthouse_Films+distributor_Ec
holight_Studios+distributor_Electric_Entertai
nment+distributor_Elevation_Pictures+distribu
tor_Entertainment_One+distributor_Entertainme
nt_Studios_Motion_Pictures+distributor_Europa
Corp+distributor_Exclusive_Releasing+distribu
tor_FUNimation+distributor_Fathom_Events+dist
ributor_FilmDistrict+distributor_Five_and_Two
_Pictures+distributor_Focus_Features+distribu
tor_Focus_World+distributor_Fox_Searchlight+d
istributor_Freestyle_Releasing+distributor_Fu
n_Academy_Motion_Pictures+distributor_GKIDS+d
istributor_GVN_Releasing+distributor_Global_R
oad+distributor_Good_Deed_Entertainment+distr
ibutor_Gunpowder_and_Sky+distributor_Hammond_
Entertainment+distributor_High_Top_Releasing+
distributor_IFC_Films+distributor_Image_Enter
tainment+distributor_Indican_Pictures+distrib
utor_LD_Distribution+distributor_Lionsgate+di
stributor_LionsgateRoadside_Attractions+distr
ibutor_MGM+distributor_Magnet_Pictures+distri
butor_Magnolia_Pictures+distributor_Miramax+d
istributor_Mongrel_Media+distributor_Monterey
_Media+distributor_Music_Box_Films+distributo
r_Neon+distributor_New_Films_Cinema+distribut
or_OTL_Releasing+distributor_OminFreestyle+di
stributor_Open_Road+distributor_Orion_Picture

s+distributor_Oscilloscope_Pictures+distributor_Overture_Films+distributor_Paladin+distributor_Paramount_Pictures+distributor_Paramount_Vantage+distributor_Peace_Film_LLC+distributor_Picturehouse+distributor_Purdie_Distribution+distributor_Pure_Flix_Entertainment+distributor_RADiUS_TWC+distributor_Radius+distributor_Relativity+distributor_Roadside_Attractions+distributor_Rocky_Mountain_Pictures+distributor_STX_Entertainment+distributor_Saban_Films+distributor_Samuel_Goldwyn_Films+distributor_Smith_Global_Media+distributor_Smodshow_Productions+distributor_Sony_Pictures+distributor_Sony_Pictures_Classics+distributor_Summit_Entertainment+distributor_Sundance_Selects+distributor_Ten_Furlongs+distributor_The_Bubble_Factory+distributor_The_Orchard+distributor_TriBeca_Films+distributor_United_Artists+distributor_Universal+distributor_Variance_Films+distributor_Vertical_Entertainment+distributor_Viva_Entertainment+distributor_Vivendi_Entertainment+distributor_Walt_Disney+distributor_Warner_Bros+distributor_WeinsteinDimension+distributor_Weinstein_Co+distributor_Wrekin_Hill_Entertainment+distributor_XLrator_Media+genre_Adventure+genre_Black_Comedy+genre_Comedy+genre_ConcertPerformance+genre_Documentary+genre_Drama+genre_Horror+genre_Musical+genre_Romantic_Comedy+genre_ThrillerSuspense+genre_Western+source_Based_on_Factual_BookArticle+source_Based_on_Fiction_BookShort_Story+source_Based_on_Folk_TaleLegendFairytale+source_Based_on_Game+source_Based_on_Movie+source_Based_on_Musical_Group+source_Based_on_Musical_or_Opera+source_Based_on_Play+source_Based_on_Real_Life_Events+source_Based_on_Religious_Text+source_Based_on_Short_Film+source_Based_on_Song+source_Based_on_TV+source_Based_on_Theme_Pa

```
rk_Ride+source_Based_on_Toy+source_Compilatio
n+source_Original_Screenplay+source_Remake+so
urce_Spin_Off+prodmethod_Digital_Animation+pr
odmethod_Hand_Animation+prodmethod_Live_Actio
n+prodmethod_Multiple_Production_Methods+prod
method_Stop_Motion_Animation+creativetype_Dig
ital_Animation+creativetype_Hand_Animation+cr
eativetype_Live_Action+creativetype_Multiple_
Production_Methods+creativetype_Stop_Motion_A
nimation
inf_dombox_log~productionbudget_log+maxtheate
rs_log+dombox_log+inf_dombox_log+runtime_min_
log+act_infl_log+dir_infl_log+franchise+distr
ibutor_20th_Century_Fox+distributor_A24+distr
ibutor_AFFRM+distributor_Abramorama_Films+dis
tributor_Alchemy+distributor_Amazon_Studios+d
istributor_Amazon_StudiosRoadside_Attractions
+distributor_Anchor_Bay_Entertainment+distrib
utor_Annapurna_Pictures+distributor_Apparitio
n+distributor_Atlas_Distribution+distributor_
Aviron_Pictures+distributor_BH_Tilt+distribut
or_Bleecker_Street+distributor_Briarcliffe_En
tertainment+distributor_Broad_Green_Pictures+
distributor_CBS_Films+distributor_Cinelou_Rel
easing+distributor_Clarius_Entertainment+dist
ributor_Codeblack_Entertainment+distributor_C
ohen_Media_Group+distributor_Drafthouse_Films
+distributor_Echolight_Studios+distributor_El
ectric_Entertainment+distributor_Elevation_Pi
ctures+distributor_Entertainment_One+distribu
tor_Entertainment_Studios_Motion_Pictures+dis
tributor_EuropaCorp+distributor_Exclusive_Rel
easing+distributor_FUNimation+distributor_Fat
hom_Events+distributor_FilmDistrict+distribut
or_Five_and_Two_Pictures+distributor_Focus_Fe
atures+distributor_Focus_World+distributor_Fo
x_Searchlight+distributor_Freestyle_Releasing
+distributor_Fun_Academy_Motion_Pictures+dist
```

```
ributor_GKIDS+distributor_GVN_Releasing+distr
ibutor_Global_Road+distributor_Good_Deed_Ente
rtainment+distributor_Gunpowder_and_Sky+distr
ibutor_Hammond_Entertainment+distributor_High
_Top_Releasing+distributor_IFC_Films+distribu
tor_Image_Entertainment+distributor_Indican_P
ictures+distributor_LD_Distribution+distribut
or_Lionsgate+distributor_LionsgateRoadside_At
tractions+distributor_MGM+distributor_Magnet_
Pictures+distributor_Magnolia_Pictures+distri
butor_Miramax+distributor_Mongrel_Media+distr
ibutor_Monterey_Media+distributor_Music_Box_F
ilms+distributor_Neon+distributor_New_Films_C
inema+distributor_OTL_Releasing+distributor_O
minFreestyle+distributor_Open_Road+distributo
r_Orion_Pictures+distributor_Oscilloscope_Pic
tures+distributor_Overture_Films+distributor_
Paladin+distributor_Paramount_Pictures+distri
butor_Paramount_Vantage+distributor_Peace_Fil
m_LLC+distributor_Picturehouse+distributor_Pu
rdie_Distribution+distributor_Pure_Flix_Enter
tainment+distributor_RADiUS_TWC+distributor_R
adius+distributor_Relativity+distributor_Road
side_Attractions+distributor_Rocky_Mountain_P
ictures+distributor_STX_Entertainment+distrib
utor_Saban_Films+distributor_Samuel_Goldwyn_F
ilms+distributor_Smith_Global_Media+distribut
or_Smodshow_Productions+distributor_Sony_Pict
ures+distributor_Sony_Pictures_Classics+distr
ibutor_Summit_Entertainment+distributor_Sunda
nce_Selects+distributor_Ten_Furlongs+distribu
tor_The_Bubble_Factory+distributor_The_Orchar
d+distributor_TriBeca_Films+distributor_Unite
d_Artists+distributor_Universal+distributor_V
ariance_Films+distributor_Vertical_Entertainm
ent+distributor_Viva_Entertainment+distributo
r_Vivendi_Entertainment+distributor_Walt_Disn
ey+distributor_Warner_Bros+distributor_Weinst
```

```
einDimension+distributor_Weinstein_Co+distrib
utor_Wrekin_Hill_Entertainment+distributor_XL
rator_Media+genre_Adventure+genre_Black_Comed
y+genre_Comedy+genre_ConcertPerformance+genre
_Documentary+genre_Drama+genre_Horror+genre_M
usical+genre_Romantic_Comedy+genre_ThrillerSu
spense+genre_Western+source_Based_on_Factual_
BookArticle+source_Based_on_Fiction_BookShort
_Story+source_Based_on_Folk_TaleLegendFairyta
le+source_Based_on_Game+source_Based_on_Movie
+source_Based_on_Musical_Group+source_Based_o
n_Musical_or_Opera+source_Based_on_Play+sourc
e_Based_on_Real_Life_Events+source_Based_on_R
eligious_Text+source_Based_on_Short_Film+sour
ce_Based_on_Song+source_Based_on_TV+source_Ba
sed_on_Theme_Park_Ride+source_Based_on_Toy+so
urce_Compilation+source_Original_Screenplay+s
ource_Remake+source_Spin_Off+prodmethod_Digit
al_Animation+prodmethod_Hand_Animation+prodme
thod_Live_Action+prodmethod_Multiple_Producti
on_Methods+prodmethod_Stop_Motion_Animation+c
reativetype_Digital_Animation+creativetype_Ha
nd_Animation+creativetype_Live_Action+creativ
etype_Multiple_Production_Methods+creativetyp
e_Stop_Motion_Animation
```

In [971…

```
baseline_model = ols(formula = formula, data
baseline_model.summary()
```

Out[971…

OLS Regression Results

Dep. Variable:	inf_dombox_log	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	1.000
Method:	Least Squares	**F-statistic:**	2.986e+30

Date:	Thu, 09 Sep 2021	**Prob (F-statistic):**	0.00
Time:	01:54:12	**Log-Likelihood:**	49075.
No. Observations:	1508	**AIC:**	-9.786e+04
Df Residuals:	1363	**BIC:**	-9.709e+04
Df Model:	144		
Covariance Type:	nonrobust		

Intercept	6.6
productionbudget_log	-6.2
maxtheaters_log	-8.0
dombox_log	-3.0
inf_dombox_log	1.
runtime_min_log	-1.6
act_infl_log	1.9

dir_infl_log	-7.1
franchise	1.2
distributor_20th_Century_Fox	-2.9
distributor_A24	-6.6
distributor_AFFRM	7.7
distributor_Abramorama_Films	-3.7
distributor_Alchemy	-1.7
distributor_Amazon_Studios	-7.7
distributor_Amazon_StudiosRoadside_Attractions	-7.9
distributor_Anchor_Bay_Entertainment	-2.6
distributor_Annapurna_Pictures	2.7
distributor_Apparition	-2.8
distributor_Atlas_Distribution	2.

distributor_Aviron_Pictures	-8.8
distributor_BH_Tilt	-2.6
distributor_Bleecker_Street	-5.5
distributor_Briarcliffe_Entertainment	-3.9
distributor_Broad_Green_Pictures	-2.
distributor_CBS_Films	-8.8
distributor_Cinelou_Releasing	-1.
distributor_Clarius_Entertainment	-3.1
distributor_Codeblack_Entertainment	-3.3
distributor_Cohen_Media_Group	-2.7
distributor_Drafthouse_Films	-2.
distributor_Echolight_Studios	
distributor_Electric_Entertainment	-4.8

distributor_Elevation_Pictures	-2.
distributor_Entertainment_One	2.7
distributor_Entertainment_Studios_Motion_Pictures	-7.7
distributor_EuropaCorp	9.9
distributor_Exclusive_Releasing	-7.7
distributor_FUNimation	-2.1
distributor_Fathom_Events	-3.1
distributor_FilmDistrict	-2.1
distributor_Five_and_Two_Pictures	-2.6
distributor_Focus_Features	-2.9
distributor_Focus_World	-2.
distributor_Fox_Searchlight	-1.
distributor_Freestyle_Releasing	-1.2

distributor_Fun_Academy_Motion_Pictures	
distributor_GKIDS	-5.9
distributor_GVN_Releasing	-3.6
distributor_Global_Road	-2.0
distributor_Good_Deed_Entertainment	-2.
distributor_Gunpowder_and_Sky	-2.6
distributor_Hammond_Entertainment	-2.4
distributor_High_Top_Releasing	-1.2
distributor_IFC_Films	-1.8
distributor_Image_Entertainment	-2.
distributor_Indican_Pictures	2.
distributor_LD_Distribution	-2.1
distributor_Lionsgate	-1.8

distributor_LionsgateRoadside_Attractions	-2.5
distributor_MGM	-1.1
distributor_Magnet_Pictures	-1.9
distributor_Magnolia_Pictures	-2.3
distributor_Miramax	-1.5
distributor_Mongrel_Media	-1.
distributor_Monterey_Media	-4.6
distributor_Music_Box_Films	1.2
distributor_Neon	-2.1
distributor_New_Films_Cinema	4.4
distributor_OTL_Releasing	-2.0
distributor_OminFreestyle	-1.
distributor_Open_Road	-1.4

distributor_Orion_Pictures	-1.
distributor_Oscilloscope_Pictures	-7.7
distributor_Overture_Films	1.
distributor_Paladin	-2.8
distributor_Paramount_Pictures	-8.3
distributor_Paramount_Vantage	-1.
distributor_Peace_Film_LLC	2.
distributor_Picturehouse	4.4
distributor_Purdie_Distribution	-1.2
distributor_Pure_Flix_Entertainment	-1.1
distributor_RADiUS_TWC	-1.
distributor_Radius	-3.7
distributor_Relativity	-1.4

distributor_Roadside_Attractions	-2.7
distributor_Rocky_Mountain_Pictures	1.7
distributor_STX_Entertainment	-9.9
distributor_Saban_Films	-2.1
distributor_Samuel_Goldwyn_Films	-1.2
distributor_Smith_Global_Media	-2.4
distributor_Smodshow_Productions	-4.6
distributor_Sony_Pictures	-9.9
distributor_Sony_Pictures_Classics	-1.
distributor_Summit_Entertainment	-1.2
distributor_Sundance_Selects	-3.7
distributor_Ten_Furlongs	
distributor_The_Bubble_Factory	-4.8

distributor_The_Orchard	-2.1
distributor_TriBeca_Films	2.
distributor_United_Artists	-9.9
distributor_Universal	-1.
distributor_Variance_Films	-2.
distributor_Vertical_Entertainment	-1.
distributor_Viva_Entertainment	-5.3
distributor_Vivendi_Entertainment	-5.5
distributor_Walt_Disney	-1.
distributor_Warner_Bros	-2.6
distributor_WeinsteinDimension	-3.2
distributor_Weinstein_Co	-1.7
distributor_Wrekin_Hill_Entertainment	-6.6

distributor_XLrator_Media	-4.4
genre_Adventure	8.9
genre_Black_Comedy	-4.1
genre_Comedy	2.1
genre_ConcertPerformance	1.2
genre_Documentary	-1.6
genre_Drama	3.1
genre_Horror	1.3
genre_Musical	-7.7
genre_Romantic_Comedy	5.
genre_ThrillerSuspense	5.4
genre_Western	-4.1
source_Based_on_Factual_BookArticle	-3.3

source_Based_on_Fiction_BookShort_Story	2.2
source_Based_on_Folk_TaleLegendFairytale	-5.1
source_Based_on_Game	5.2
source_Based_on_Movie	-3.2
source_Based_on_Musical_Group	-3.3
source_Based_on_Musical_or_Opera	1.2
source_Based_on_Play	-4.3
source_Based_on_Real_Life_Events	-2.0
source_Based_on_Religious_Text	-3.2
source_Based_on_Short_Film	-3.6
source_Based_on_Song	-4.4
source_Based_on_TV	8.3
source_Based_on_Theme_Park_Ride	7.2

source_Based_on_Toy	4.1
source_Compilation	3.6
source_Original_Screenplay	9.0
source_Remake	-1.3
source_Spin_Off	3.3
prodmethod_Digital_Animation	5.
prodmethod_Hand_Animation	4.6
prodmethod_Live_Action	2.4
prodmethod_Multiple_Production_Methods	2.3
prodmethod_Stop_Motion_Animation	-6.1
creativetype_Digital_Animation	-5.6
creativetype_Hand_Animation	4.3
creativetype_Live_Action	-1.1

creativetype_Multiple_Production_Methods	2.6
creativetype_Stop_Motion_Animation	3.6

Omnibus:	264.321	**Durbin-Watson:**	0.717
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	944.642
Skew:	-0.832	**Prob(JB):**	7.47e-206
Kurtosis:	6.503	**Cond. No.**	2.24e+18

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.17e-33. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Observations: Conclusions from baseline model:

1. Drop **dombox_log** as its not a relevant feature
2. Drop **dir_infl_log** Director Influence not relevant
3. Part of a Franchise it self not relevant...maybe more secific would be
4. Can remove distributors as none are relevant

In [972…
```
X = preprocessed.drop(columns='inf_dombox_log
y = preprocessed['inf_dombox_log']
```

Perform a train-test split

In [989…
```
# Split the data into training and test sets
from sklearn.model_selection import train_tes
X_train, X_test, y_train, y_test = train_test
print(len(X_train), len(X_test), len(y_train
```

```
1131 377 1131 377
```

Apply model to train set

In [995…
```
# Import and initialize the linear regressio
from sklearn.linear_model import LinearRegres
baseline_linreg = LinearRegression()
# Fit the model to train data
baseline_linreg.fit(X_train, y_train)
```

Out[995…
```
LinearRegression()
```

Calculate predictions on training

and test sets

In [996…
```
# Calculate predictions on training and test
y_hat_train = baseline_linreg.predict(X_trai
y_pred = baseline_linreg.predict(X_test)
```

In [998…
```
# Calculate residuals
train_residuals = y_hat_train - y_train
test_residuals = y_pred - y_test
```

Calculate the Mean Squared Error (MSE)

A good way to compare overall performance is to compare the mean squarred error for the predicted values on the training and test sets.

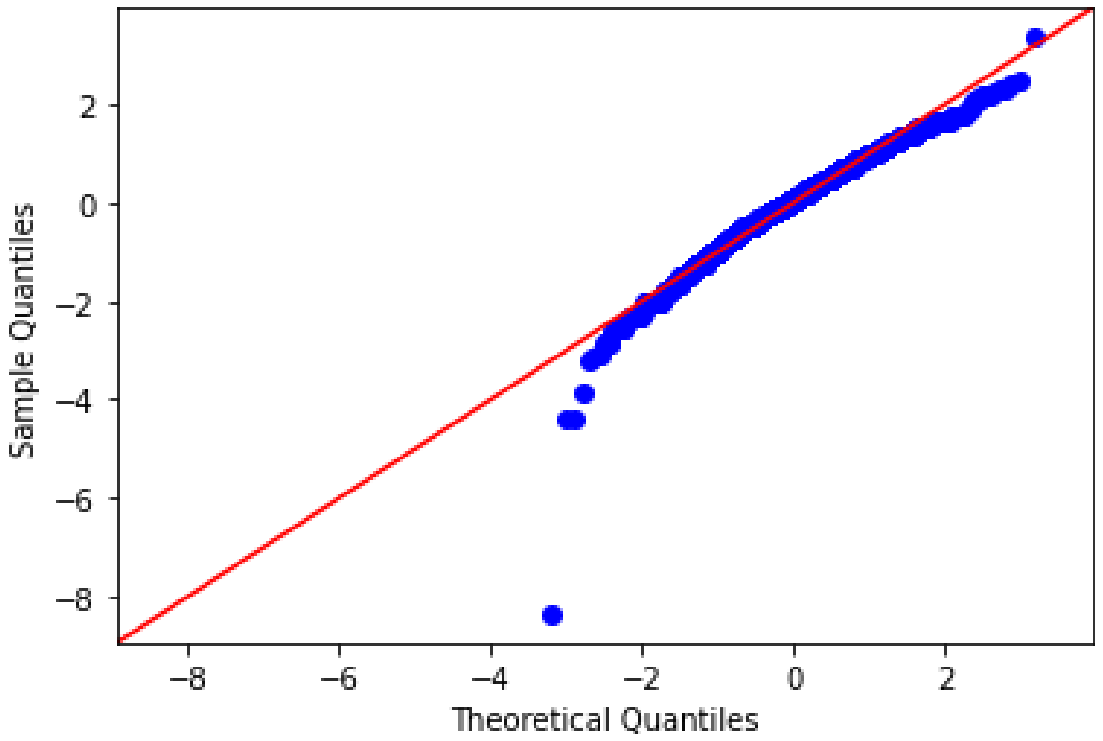
In [900…
```
# Import mean_squared_error from sklearn.meti
from sklearn.metrics import mean_squared_erro
```

In [100…
```
# Calculate training and test MSE
train_mse = mean_squared_error(y_train, y_hat
test_mse = mean_squared_error(y_test, y_pred
print('Train Mean Squarred Error:', train_mse
print('Test Mean Squarred Error:', test_mse)
```

```
Train Mean Squarred Error: 0.0006523654827221
526
Test Mean Squarred Error: 4.432120578640094e+
17
```

In [100…
```
import statsmodels.api as sm
import scipy.stats as stats
```

```
fig = sm.graphics.qqplot(baseline_model.resi
```



In [ ]:

In [ ]:

Model 2 - Remove Disto

In [101…

```
continuous = ['productionbudget', 'maxtheate
categoricals = ['genre', 'source', 'prodmeth
```

In [ ]:

In [101…

```
movies_cont = merged_df[continuous]

# log features
log_names = [f'{column}_log' for column in m
```

```
movies_log = np.log(movies_cont)
movies_log.columns = log_names

# normalize (subract mean and divide by std)
def normalize(feature):
    return (feature - feature.mean()) / featu

movies_log_norm = movies_log.apply(normalize

# one hot encode categoricals
movies_ohe = pd.get_dummies(merged_df[catego
preprocessed2 = pd.concat([movies_log_norm,m
```

In [101…

```
outcome = 'inf_dombox_log'
predictor_variables = "+".join(preprocessed2
formula = outcome + '~' + predictor_variable
print(predictor_variables)
print(formula)
```

```
productionbudget_log+maxtheaters_log+inf_domb
ox_log+runtime_min_log+act_infl_log+franchise
+genre_Adventure+genre_Black_Comedy+genre_Com
edy+genre_ConcertPerformance+genre_Documentar
y+genre_Drama+genre_Horror+genre_Musical+genr
e_Romantic_Comedy+genre_ThrillerSuspense+genr
e_Western+source_Based_on_Factual_BookArticle
+source_Based_on_Fiction_BookShort_Story+sour
ce_Based_on_Folk_TaleLegendFairytale+source_B
ased_on_Game+source_Based_on_Movie+source_Bas
ed_on_Musical_Group+source_Based_on_Musical_o
r_Opera+source_Based_on_Play+source_Based_on_
Real_Life_Events+source_Based_on_Religious_Te
xt+source_Based_on_Short_Film+source_Based_on
_Song+source_Based_on_TV+source_Based_on_Them
e_Park_Ride+source_Based_on_Toy+source_Compil
ation+source_Original_Screenplay+source_Remak
```

```
e+source_Spin_Off+prodmethod_Digital_Animatio
n+prodmethod_Hand_Animation+prodmethod_Live_A
ction+prodmethod_Multiple_Production_Methods+
prodmethod_Stop_Motion_Animation+creativetype
_Digital_Animation+creativetype_Hand_Animatio
n+creativetype_Live_Action+creativetype_Multi
ple_Production_Methods+creativetype_Stop_Moti
on_Animation
inf_dombox_log~productionbudget_log+maxtheate
rs_log+inf_dombox_log+runtime_min_log+act_inf
l_log+franchise+genre_Adventure+genre_Black_C
omedy+genre_Comedy+genre_ConcertPerformance+g
enre_Documentary+genre_Drama+genre_Horror+gen
re_Musical+genre_Romantic_Comedy+genre_Thrill
erSuspense+genre_Western+source_Based_on_Fact
ual_BookArticle+source_Based_on_Fiction_BookS
hort_Story+source_Based_on_Folk_TaleLegendFai
rytale+source_Based_on_Game+source_Based_on_M
ovie+source_Based_on_Musical_Group+source_Bas
ed_on_Musical_or_Opera+source_Based_on_Play+s
ource_Based_on_Real_Life_Events+source_Based_
on_Religious_Text+source_Based_on_Short_Film+
source_Based_on_Song+source_Based_on_TV+sourc
e_Based_on_Theme_Park_Ride+source_Based_on_To
y+source_Compilation+source_Original_Screenpl
ay+source_Remake+source_Spin_Off+prodmethod_D
igital_Animation+prodmethod_Hand_Animation+pr
odmethod_Live_Action+prodmethod_Multiple_Prod
uction_Methods+prodmethod_Stop_Motion_Animati
on+creativetype_Digital_Animation+creativetyp
e_Hand_Animation+creativetype_Live_Action+cre
ativetype_Multiple_Production_Methods+creativ
etype_Stop_Motion_Animation
```

In [101…

```
model2 = ols(formula = formula, data = prepr
model2.summary()
```

OLS Regression Results

Out[101...

Dep. Variable:	inf_dombox_log	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	1.000
Method:	Least Squares	**F-statistic:**	1.125e+31
Date:	Thu, 09 Sep 2021	**Prob (F-statistic):**	0.00
Time:	03:45:55	**Log-Likelihood:**	49054.
No. Observations:	1508	**AIC:**	-9.803e+04
Df Residuals:	1467	**BIC:**	-9.781e+04
Df Model:	40		
Covariance Type:	nonrobust		

	coef	
Intercept	-7.459e-17	2
productionbudget_log	-2.914e-16	8
maxtheaters_log	-1.527e-16	1
inf_dombox_log	1.0000	1

runtime_min_log	-1.7e-16	6
act_infl_log	-1.141e-16	5
franchise	5.239e-16	1
genre_Adventure	4.372e-16	2
genre_Black_Comedy	-9.437e-16	3
genre_Comedy	-7.98e-17	2
genre_ConcertPerformance	2.762e-15	9
genre_Documentary	-7.772e-16	9
genre_Drama	4.684e-16	1
genre_Horror	-1.804e-16	2
genre_Musical	1.193e-15	4
genre_Romantic_Comedy	1.013e-15	3
genre_ThrillerSuspense	1.093e-16	1

genre_Western	-6.384e-16	4
source_Based_on_Factual_BookArticle	3.331e-16	3
source_Based_on_Fiction_BookShort_Story	-2.29e-16	2
source_Based_on_Folk_TaleLegendFairytale	3.469e-17	4
source_Based_on_Game	7.633e-17	5
source_Based_on_Movie	-1.041e-16	6
source_Based_on_Musical_Group	-3.331e-15	2
source_Based_on_Musical_or_Opera	-1.665e-15	1
source_Based_on_Play	2.776e-16	5
source_Based_on_Real_Life_Events	4.857e-17	2
source_Based_on_Religious_Text	1.589e-15	8
source_Based_on_Short_Film	-1.166e-15	6
source_Based_on_Song	2.442e-15	1

source_Based_on_TV	-2.213e-15	3
source_Based_on_Theme_Park_Ride	-8.327e-16	1
source_Based_on_Toy	-3.886e-16	6
source_Compilation	1.11e-16	6
source_Original_Screenplay	-3.469e-17	2
source_Remake	-4.302e-16	3
source_Spin_Off	2.498e-16	5
prodmethod_Digital_Animation	-1.388e-17	1
prodmethod_Hand_Animation	-3.886e-16	5
prodmethod_Live_Action	3.192e-16	
prodmethod_Multiple_Production_Methods	6.939e-17	6
prodmethod_Stop_Motion_Animation	-3.886e-16	3
creativetype_Digital_Animation	4.163e-17	1

creativetype_Hand_Animation	-4.718e-16	5
creativetype_Live_Action	1.041e-16	
creativetype_Multiple_Production_Methods	-5.551e-17	6
creativetype_Stop_Motion_Animation	-3.608e-16	3

Omnibus:	100.187	**Durbin-Watson:**	0.895
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	122.340
Skew:	-0.630	**Prob(JB):**	2.72e-27
Kurtosis:	3.598	**Cond. No.**	1.28e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.97e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Conclusions from model 2:

1. genre_Documentary,genre_Comedy,genre_Horror
   source_Based_on_Song,source_Based_on_Theme_
   source_Compilation,source_Original_Screenplay,sc
   prodmethod_Hand_Animation,creativetype_Live_A
   no longerrelevant feature. Will need to drop it
   in the next model

In [101…
```
X = preprocessed2.drop(columns='inf_dombox_l
y = preprocessed2['inf_dombox_log']
```

Perform a train-test split

In [101…
```
# Split the data into training and test sets
from sklearn.model_selection import train_te
X_train, X_test, y_train, y_test = train_tes
print(len(X_train), len(X_test), len(y_train
```

```
1131 377 1131 377
```

Apply model to train set

In [101…
```
# Import and initialize the linear regressio
from sklearn.linear_model import LinearRegre
model2linreg = LinearRegression()
# Fit the model to train data
model2linreg.fit(X_train, y_train)
```

Out[101… LinearRegression()

Calculate predictions on training and test sets

In [102…
```
# Calculate predictions on training and test
y_hat_train = model2linreg.predict(X_train)
y_pred = model2linreg.predict(X_test)
```

In [102…
```
# Calculate residuals
train_residuals = y_hat_train - y_train
test_residuals = y_pred - y_test
```

In [ ]:
```
```

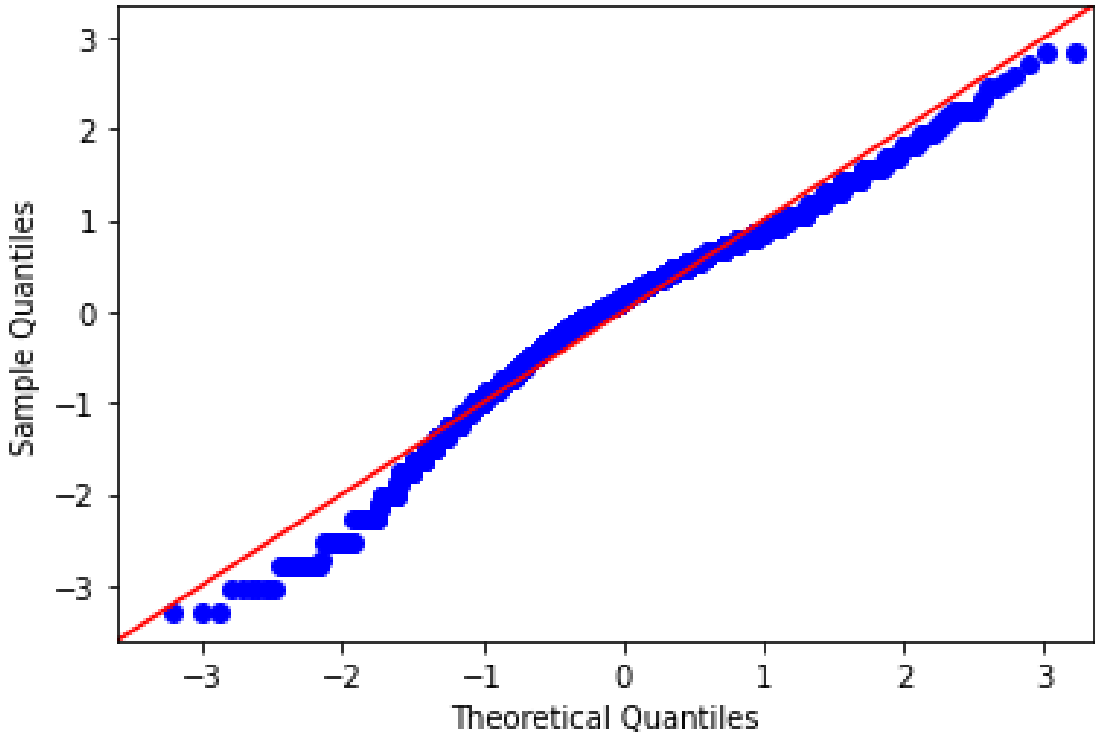
Calculate the Mean Squared Error (MSE)

A good way to compare overall performance is to compare the mean squarred error for the predicted values on the training and test sets.

In [102…
```
# Calculate training and test MSE
train_mse = mean_squared_error(y_train, y_ha
test_mse = mean_squared_error(y_test, y_pred
print('Train Mean Squarred Error:', train_ms
print('Test Mean Squarred Error:', test_mse)
```

```
Train Mean Squarred Error: 0.1732949415524894
5
Test Mean Squarred Error: 0.2060230345661356
```

In [102…
```
fig2 = sm.graphics.qqplot(model2.resid, dist=
```



In [ ]:

In [ ]:

Model 3 - Remove Prodbudget and other cols

In [102…
```
continuous = ['maxtheaters', 'inf_dombox','ru
categoricals = ['genre', 'source', 'prodmetho
```

In [ ]:

In [104…
```
movies_cont = merged_df[continuous]

# log features
```

```
log_names = [f'{column}_log' for column in mo

movies_log = np.log(movies_cont)
movies_log.columns = log_names

movies_log_norm = movies_log.apply(normalize

# one hot encode categoricals
movies_ohe = pd.get_dummies(merged_df[catego
movies_ohe.drop(columns=['genre_Documentary'
                         ,'genre_Western','s
                         'source_Based_on_So
'source_Compilation','source_Original_Screen
'prodmethod_Hand_Animation','creativetype_Li
                         'creativetype_Stop_
                        'genre_Musical','sou
'source_Based_on_Movie',
'source_Based_on_Musical_Group',
'source_Based_on_Musical_or_Opera',
'source_Based_on_Play',
'source_Based_on_Real_Life_Events',
'source_Based_on_Religious_Text',
'source_Based_on_Short_Film',
'source_Based_on_TV','prodmethod_Multiple_Pr
'prodmethod_Stop_Motion_Animation',
'creativetype_Hand_Animation',], inplace=Tru
preprocessed3 = pd.concat([movies_log_norm,mo
```

In [104…

```
outcome = 'inf_dombox_log'
predictor_variables = "+".join(preprocessed3
formula = outcome + '~' + predictor_variable
```

In [104…

```
model3 =ols(formula = formula, data = prepro
model3.summary()
```

Out[104...

OLS Regression Results

Dep. Variable:	inf_dombox_log	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	1.000
Method:	Least Squares	**F-statistic:**	7.897e+31
Date:	Thu, 09 Sep 2021	**Prob (F-statistic):**	0.00
Time:	09:10:26	**Log-Likelihood:**	49535.
No. Observations:	1508	**AIC:**	-9.905e+04
Df Residuals:	1496	**BIC:**	-9.898e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	
Intercept	1.37e-16	1.6
maxtheaters_log	4.51e-16	7.1
inf_dombox_log	1.0000	7.7

runtime_min_log	8.032e-16	3.9
act_infl_log	3.799e-16	3.6
franchise	-2.463e-16	1.0
genre_Adventure	-9.298e-16	1.3
genre_Drama	5.794e-16	8.6
genre_Romantic_Comedy	0	1.9
source_Based_on_Fiction_BookShort_Story	3.712e-16	9
prodmethod_Live_Action	-6.939e-17	1.5
creativetype_Digital_Animation	1.18e-16	1.9

Omnibus:	55.732	**Durbin-Watson:**	1.595
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	62.864
Skew:	0.448	**Prob(JB):**	2.24e-14
Kurtosis:	3.443	**Cond. No.**	11.1

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Conclusions from model 2:

1. genre_Documentary,genre_Comedy,genre_Horror source_Based_on_Song,source_Based_on_Theme_ source_Compilation,source_Original_Screenplay,sc prodmethod_Hand_Animation,creativetype_Live_A no longerrelevant feature. Will need to drop it in the next model

In [105…
```
X = preprocessed3.drop(columns='inf_dombox_l
y = preprocessed3['inf_dombox_log']
```

Perform a train-test split

In [105…
```
X_train, X_test, y_train, y_test = train_tes
print(len(X_train), len(X_test), len(y_train
```

```
1131 377 1131 377
```

Apply model to train set

In [103…
```
# Import and initialize the linear regressio
from sklearn.linear_model import LinearRegre
model3linreg = LinearRegression()
# Fit the model to train data
model3linreg.fit(X_train, y_train)
```

Out[103… LinearRegression()

Calculate predictions on training and test sets

In [104…
```
# Calculate predictions on training and test
y_hat_train = model3linreg.predict(X_train)
y_pred = model3linreg.predict(X_test)
```

In [104…
```
# Calculate residuals
train_residuals = y_hat_train - y_train
test_residuals = y_pred - y_test
```

In [ ]:
```

```

Calculate the Mean Squared Error (MSE)

A good way to compare overall performance is to compare the mean squarred error for the predicted values on the training and test sets.

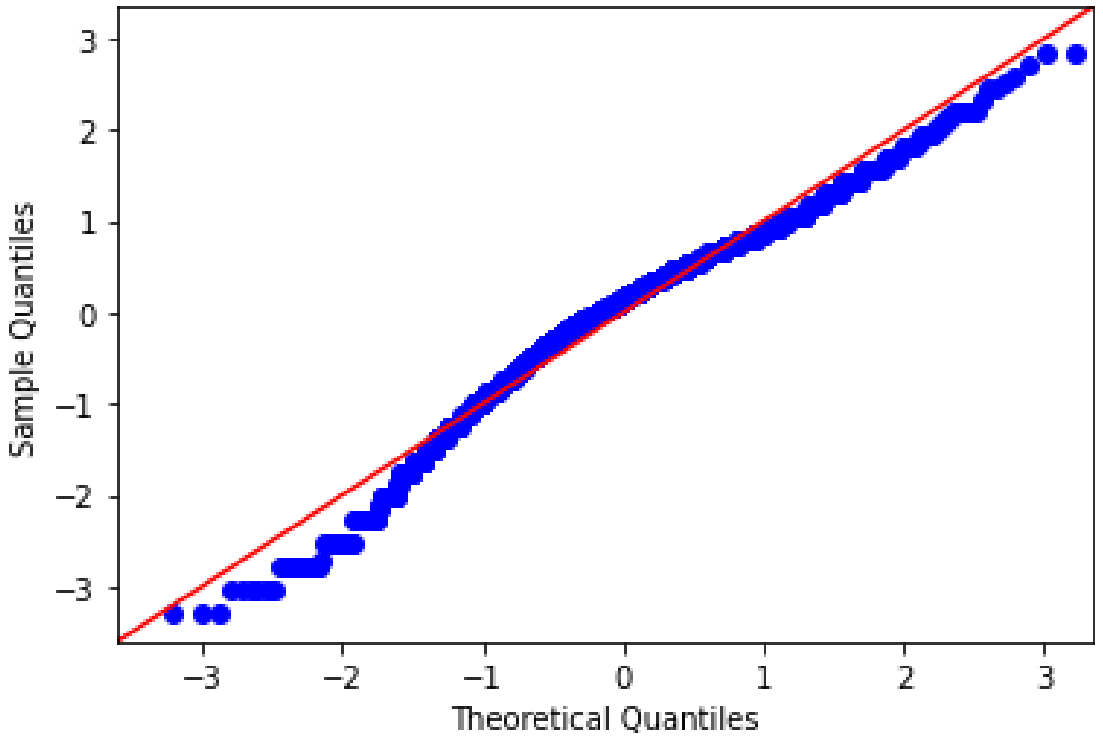
In [104…
```
# Calculate training and test MSE
train_mse = mean_squared_error(y_train, y_ha
```

```
test_mse = mean_squared_error(y_test, y_pred
print('Train Mean Squarred Error:', train_ms
print('Test Mean Squarred Error:', test_mse)
```

```
Train Mean Squarred Error: 0.1732949415524894
5
Test Mean Squarred Error: 0.2060230345661356
```

In [102…

```
fig2 = sm.graphics.qqplot(model2.resid, dist
```



In [ ]:

Regress for inference on M Q and P for Bass Model

In [105…

```
bass_df = pd.read_csv('./data/bass_model_valu
```

In [106…

```
bass_df.describe()
```

Out[106...

	year	M	p	
count	2381.000000	2.381000e+03	2381.000000	2381.
mean	2014.338933	5.163880e+06	0.116816	0.
std	3.111208	1.174091e+07	1.049731	0.
min	2009.000000	1.731849e+02	-27.190591	-4.
25%	2011.000000	2.996146e+04	0.067116	0.
50%	2014.000000	4.076424e+05	0.158372	0.
75%	2017.000000	5.989178e+06	0.395063	0.
max	2021.000000	2.849854e+08	11.955771	21.

In [106...
```
bass_merged_df = merged_df.merge(bass_df,lef
```

In [106...
```
bass_merged_df.describe()
```

Out[106...

	tconst	title	released	dist
0	tt1606378	A Good Day to Die Hard	Feb 14, 2013	20th_Cent
1	tt2935510	Ad Astra	Sep 20, 2019	20th_Cent
2	tt2316204	Alien: Covenant	May 19, 2017	20th_Cent

	tconst	title	released	dist
3	tt0437086	Alita: Battle Angel	Feb 14, 2019	20th_Cent
4	tt1727824	Bohemian Rhapsody	Nov 2, 2018	20th_Cent
...	...	...	...	
810	tt5362988	Wind River	Aug 4, 2017	Weins
811	tt2404425	Woman in Gold	Apr 1, 2015	Weins
812	tt0464154	Piranha 3D	Aug 20, 2010	WeinsteinDir
813	tt1262416	Scream 4	Apr 15, 2011	WeinsteinDir
814	tt1403177	Hesher	May 13, 2011	Wrekin_Hill_Entert

815 rows × 21 columns

In [ ]:
```
**Regress for M**
```

In [ ]:
```
```

In [ ]:
```
**Regress for P**
```

In [ ]:
```

```

In [ ]:
```
**Regress for Q**
```

Random Forest

In [106…
```
import pandas as pd
import numpy as np
np.random.seed(0)
import matplotlib.pyplot as plt
from sklearn.model_selection import train_te
from sklearn.metrics import accuracy_score,
from sklearn.tree import DecisionTreeClassif
from sklearn.ensemble import BaggingClassifi
```

In [107…
```
X = preprocessed.drop(columns='inf_dombox_lo
y = preprocessed['inf_dombox_log']
```

In [107…
```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_tes
print(len(X_train), len(X_test), len(y_train
```

```
1131 377 1131 377
```

Decision tree Baseline

In [107…
```
#Baseline a Decision Tree
tree_clf = DecisionTreeClassifier(criterion=
tree_clf.fit(X_train, y_train)
```

```
---------------------------------------------
-----------------------------
ValueError                                Tra
ceback (most recent call last)
<ipython-input-1073-4b283d00851f> in <module>
      1 #Baseline a Decision Tree
      2 tree_clf = DecisionTreeClassifier(cri
terion='gini', max_depth=5)
----> 3 tree_clf.fit(X_train, y_train)

~\anaconda3\envs\learn-env\lib\site-packages
\sklearn\tree\_classes.py in fit(self, X, y,
 sample_weight, check_input, X_idx_sorted)
    888         """
    889
--> 890         super().fit(
    891             X, y,
    892             sample_weight=sample_weig
ht,

~\anaconda3\envs\learn-env\lib\site-packages
\sklearn\tree\_classes.py in fit(self, X, y,
 sample_weight, check_input, X_idx_sorted)
    179
    180         if is_classification:
--> 181             check_classification_targ
ets(y)
    182             y = np.copy(y)
    183

~\anaconda3\envs\learn-env\lib\site-packages
\sklearn\utils\multiclass.py in check_classif
ication_targets(y)
    170     if y_type not in ['binary', 'mult
iclass', 'multiclass-multioutput',
    171                       'multilabel-ind
icator', 'multilabel-sequences']:
--> 172         raise ValueError("Unknown lab
```

```
el type: %r" % y_type)
    173
    174

ValueError: Unknown label type: 'continuous'
```

In [ ]:
```
DecisionTreeClassifier(class_weight=None, cr
                       max_features=None, ma
                       min_impurity_decrease
                       min_samples_leaf=1, m
                       min_weight_fraction_l
                       random_state=None, sp
```

Feature Importances

In [ ]:
```
# Feature importance
tree_clf.feature_importances_
```

In [ ]:
```
def plot_feature_importances(model):
    n_features = data_train.shape[1]
    plt.figure(figsize=(8,8))
    plt.barh(range(n_features), model.featur
    plt.yticks(np.arange(n_features), data_t
    plt.xlabel('Feature importance')
    plt.ylabel('Feature')

plot_feature_importances(tree_clf)
```

In [ ]:
```
# Test set predictions
pred = tree_clf.predict(data_test)

# Confusion matrix and classification report
print(confusion_matrix(target_test, pred))
print(classification_report(target_test, pre
```

In [ ]:
```
print("Testing Accuracy for Decision Tree Cl
```

In [ ]:
```

```

In [ ]:
```

```

Random Forest

In [ ]:
```
# Instantiate and fit a RandomForestClassifi
forest = RandomForestClassifier(n_estimators=
forest.fit(data_train, target_train)
```

In [ ]:
```
RandomForestClassifier(bootstrap=True, class_
                       max_depth=5, max_featu
                       min_impurity_decrease=
                       min_samples_leaf=1, mi
                       min_weight_fraction_le
                       n_jobs=None, oob_score
                       verbose=0, warm_start=
```

In [ ]:
```
# Training accuracy score
forest.score(data_train, target_train)
```

In [ ]:
```
# Test accuracy score
forest.score(data_test, target_test)
```

In [ ]:
```
plot_feature_importances(forest)
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: