Using Python for Research Homework: Week 5, Case Study Part 1

The movie dataset on which this case study is based is a database of 5000 movies catalogued by The Movie Database (TMDb). The information available about each movie is its budget, revenue, rating, actors and actresses, etc. In this case study, we will use this dataset to determine whether any information about a movie can predict the total revenue of a movie. We will also attempt to predict whether a movie's revenue will exceed its budget.

In Part 1, we will inspect, clean, and transform the data.

Exercise 1

First, we will import several libraries. `scikit-learn` (**sklearn**) contains helpful statistical models, and we'll use the `matplotlib.pyplot` library for visualizations. Of course, we will use `numpy` and `pandas` for data manipulation throughout.

Instructions

- Read and execute the given code.
- Call `df.head()` to take a look at the data.

In [1]:
```
import pandas as pd
import numpy as np

from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score

import matplotlib.pyplot as plt

df = pd.read_csv("https://courses.edx.org/asset-v1:HarvardX+PH526x+2T2019+type@asse

df.head()
```

Out[1]:

	budget	genres	homepage	id	keywords	original_la
0	237000000	Action, Adventure, Fantasy, Science Fiction	http://www.avatarmovie.com/	19995	culture clash, future, space war, space colony...	
1	300000000	Adventure, Fantasy, Action	http://disney.go.com/disneypictures/pirates/	285	ocean, drug abuse, exotic island, east india t...	
2	245000000	Action, Adventure, Crime	http://www.sonypictures.com/movies/spectre/	206647	spy, based on novel, secret agent, sequel, mi6...	
3	250000000	Action, Crime, Drama, Thriller	http://www.thedarkknightrises.com/	49026	dc comics, crime fighter, terrorist, secret id...	
4	260000000	Action, Adventure, Science Fiction	http://movies.disney.com/john-carter	49529	based on novel, mars, medallion, space travel,...	

5 rows × 22 columns

Exercise 2

In this exercise, we will define the regression and classification outcomes. Specifically, we will use the `revenue` column as the target for regression. For classification, we will construct an indicator of profitability for each movie.

Instructions

- Create a new column in `df` called `profitable`, defined as 1 if the movie `revenue` is greater than the movie `budget`, and 0 otherwise.
- Next, define and store the outcomes we will use for regression and classification.
  - Define `regression_target` as the string `'revenue'`.
  - Define `classification_target` as the string `'profitable'`.

In [8]:
```
df['profitable'] = np.where(df['revenue'] > df['budget'], 1, 0)
regression_target = 'revenue'
classification_target = 'profitable'
```

```
df[df['profitable']==1]
print(df['profitable'].value_counts()[1])
```

```
2585
```

Exercise 3

For simplicity, we will proceed by analyzing only the rows without any missing data. In this exercise, we will remove rows with any infinite or missing values.

Instructions

- Use `df.replace()` to replace any cells with type `np.inf` or `-np.inf` with `np.nan`.
- Drop all rows with any `np.nan` values in that row using `df.dropna()`. Do any further arguments need to be specified in this function to remove rows with any such values?

In [11]:
```
df.replace([np.inf,-np.inf],np.nan)
df.dropna()
```

Out[11]:

	budget	genres	homepage	id	keyword
0	237000000	Action, Adventure, Fantasy, Science Fiction	http://www.avatarmovie.com/	19995	culture clash future, space wa space colony
1	300000000	Adventure, Fantasy, Action	http://disney.go.com/disneypictures/pirates/	285	ocean, dru abuse, exot island, east indi t
2	245000000	Action, Adventure, Crime	http://www.sonypictures.com/movies/spectre/	206647	spy, based o novel, secre agent, seque mi6
3	250000000	Action, Crime, Drama, Thriller	http://www.thedarkknightrises.com/	49026	dc comics, crim fighter, terroris secret id
4	260000000	Action, Adventure, Science Fiction	http://movies.disney.com/john-carter	49529	based on nove mars, medallio space travel,
...	...	...	...	...	
4758	4000000	Thriller, Science Fiction	https://www.facebook.com/thesignalfilm	242095	hacke supernatur powers, road tri indepe
4766	0	Documentary, Music	http://www.mgm.com/#/our-titles/1092/The-Last-...	13963	1970s, mus
4773	27000	Comedy	http://www.miramax.com/movie/clerks/	2292	salesclerk, lose aftercreditsstinge
4791	13	Horror	http://tincanmanthemovie.com/	157185	home invasio
4796	7000	Science Fiction, Drama, Thriller	http://www.primermovie.com	14337	distrust, garag identity crisi time travel

1406 rows × 23 columns

Exercise 4

Many of the variables in our dataframe contain the names of genre, actors/actresses, and keywords. Let's add indicator columns for each genre.

Instructions

- Determine all the genres in the genre column. Make sure to use the `strip()` function on each genre to remove trailing characters.
- Next, include each listed genre as a new column in the dataframe. Each element of these genre columns should be 1 if the movie belongs to that particular genre, and 0 otherwise. Keep in mind, a movie may belong to several genres at once.
- Call `df[genres].head()` to view your results.

In [48]:
```
df = df.dropna(how="any")
list_genres = df.genres.apply(lambda x: x.split(","))
genres = []
for row in list_genres:
    row = [genre.strip() for genre in row]
    for genre in row:
        if genre not in genres:
            genres.append(genre)

print(genres)
```

```
['Action', 'Adventure', 'Fantasy', 'Science Fiction', 'Crime', 'Drama', 'Thrille
r', 'Animation', 'Family', 'Western', 'Comedy', 'Romance', 'Horror', 'Mystery', 'W
ar', 'History', 'Music', 'Documentary', 'TV Movie', 'Foreign']
```

Exercise 5

Some variables in the dataset are already numeric and perhaps useful for regression and classification. In this exercise, we will store the names of these variables for future use. We will also take a look at some of the continuous variables and outcomes by plotting each pair in a scatter plot. Finally, we will evaluate the skew of each variable.

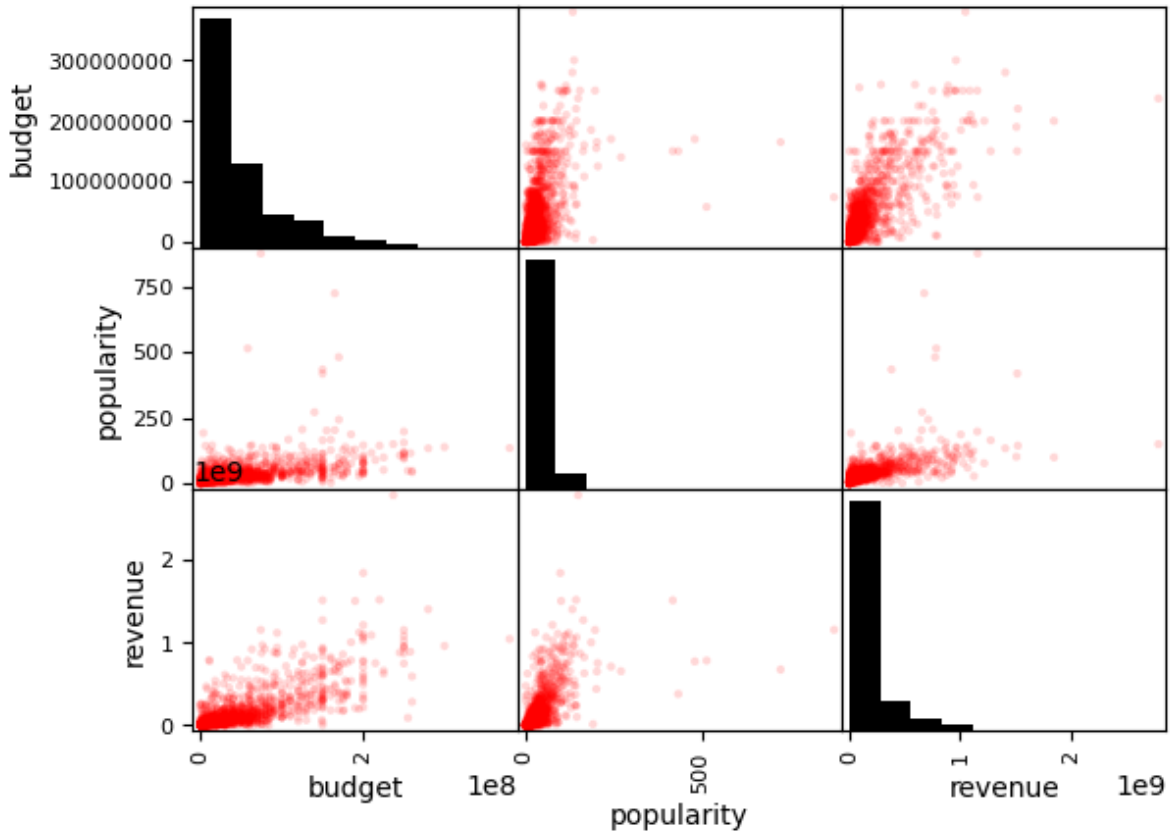
Instructions

- Call `plt.show()` to observe the plot below.
  - Which of the covariates and/or outcomes are correlated with each other?
- Call `skew()` on the columns `outcomes_and_continuous_covariates` in df.
  - Is the skew above 1 for any of these variables?

In [50]:
```
continuous_covariates = ['budget', 'popularity', 'runtime', 'vote_count', 'vote_ave
outcomes_and_continuous_covariates = continuous_covariates + [regression_target, cl
plotting_variables = ['budget', 'popularity', regression_target]

axes = pd.plotting.scatter_matrix(df[plotting_variables], alpha=0.15, \
       color=(0,0,0), hist_kwds={"color":(0,0,0)}, facecolor=(1,0,0))
plt.show()

print(df[outcomes_and_continuous_covariates].skew())
```




```
budget          1.754872
popularity      7.968139
runtime         1.059804
vote_count      2.461041
vote_average   -1.080038
revenue         3.084680
profitable     -1.081030
dtype: float64
```

Exercise 6

It appears that the variables `budget`, `popularity`, `runtime`, `vote_count`, and `revenue` are all right-skewed. In this exercise, we will transform these variables to eliminate this skewness. Specifically, we will use the `np.log10()` method. Because some of these variable values are exactly 0, we will add a small positive value to each to ensure it is defined; this is necessary because log(0) is negative infinity.

Instructions

- For each above-mentioned variable in `df`, transform value `x` into `np.log10(1+x)`.

In [64]:

```
for covariate in ['budget', 'popularity', 'runtime', 'vote_count', 'revenue']:
    df[covariate] = df[covariate].apply(lambda x: np.log10(1+x))

    print(df[outcomes_and_continuous_covariates].skew())
```

```
budget          -3.281368
popularity      -1.664154
runtime          0.429287
vote_count      -2.682463
vote_average    -1.080038
revenue         -2.559239
profitable      -1.081030
dtype: float64
budget          -3.281368
popularity      -1.664304
runtime          0.429287
vote_count      -2.682463
vote_average    -1.080038
revenue         -2.559239
profitable      -1.081030
dtype: float64
budget          -3.281368
popularity      -1.664304
runtime          0.000000
vote_count      -2.682463
vote_average    -1.080038
revenue         -2.559239
profitable      -1.081030
dtype: float64
budget          -3.281368
popularity      -1.664304
runtime          0.000000
vote_count      -2.682826
vote_average    -1.080038
revenue         -2.559239
profitable      -1.081030
dtype: float64
budget          -3.281368
popularity      -1.664304
runtime          0.000000
vote_count      -2.682826
vote_average    -1.080038
revenue         -2.559249
profitable      -1.081030
dtype: float64
```

Exercise 7

Let's now save our dataset.

Instructions

- Use `to_csv()` to save the `df` object as `movies_clean.csv`.

In [68]:
```
df.to_csv('movies_clean')
```

In [ ]:
```

```

In [ ]:
```

```