CPSC_392_Final_Project

May 15, 2023

[ ]:
```
import warnings
warnings.filterwarnings('ignore')


import pandas as pd
import numpy as np
from plotnine import *

from sklearn.linear_model import LinearRegression # Linear Regression Model
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
#model evaluation
from sklearn.model_selection import train_test_split

%matplotlib inline
```

[ ]:
```
# Data
movies = pd.read_csv("https://raw.githubusercontent.com/ryanking916/Data/main/
movies.csv")

movies = movies.dropna()

movies.head()
```

[ ]:

	name	rating	genre	year	released	score	votes	director
0	The Shining	R	Drama	1980	June 13, 1980 (United States)	8.4	927000.0	Stanley Kubrick
1	The Blue Lagoon	R	Adventure	1980	July 2, 1980 (United States)	5.8	65000.0	Randal Kleiser
2	Star Wars: Episode V - The Empire Strikes Back	PG	Action	1980				
3	Airplane!	PG	Comedy	1980				
4	Caddyshack	R	Comedy	1980				

```
2  June 20, 1980 (United States)    8.7  1200000.0   Irvin Kershner
3   July 2, 1980 (United States)    7.7   221000.0     Jim Abrahams
4  July 25, 1980 (United States)    7.3   108000.0     Harold Ramis

                  writer            star         country      budget  \
0            Stephen King  Jack Nicholson  United Kingdom  19000000.0
1  Henry De Vere Stacpoole  Brooke Shields   United States   4500000.0
2          Leigh Brackett     Mark Hamill   United States  18000000.0
3            Jim Abrahams     Robert Hays   United States   3500000.0
4       Brian Doyle-Murray     Chevy Chase   United States   6000000.0

         gross             company  runtime
0   46998772.0        Warner Bros.    146.0
1   58853106.0   Columbia Pictures    104.0
2  538375067.0           Lucasfilm    124.0
3   83453539.0  Paramount Pictures     88.0
4   39846344.0      Orion Pictures     98.0
```

1 Question 1

1.0.1 Of the variables year, gross, votes, budget, runtime and the various movie genres, which ones have the strongest relationship with a movie's IMDb score?

[ ]:
```
movies = pd.get_dummies(movies, columns = ["genre", "rating"])
pd.set_option('display.max_columns', None)
movies.head()
```

[ ]:
```
                                       name  year  \
0                                The Shining  1980
1                             The Blue Lagoon  1980
2  Star Wars: Episode V - The Empire Strikes Back  1980
3                                   Airplane!  1980
4                                  Caddyshack  1980

                     released  score      votes         director  \
0  June 13, 1980 (United States)    8.4   927000.0  Stanley Kubrick
1   July 2, 1980 (United States)    5.8    65000.0   Randal Kleiser
2  June 20, 1980 (United States)    8.7  1200000.0   Irvin Kershner
3   July 2, 1980 (United States)    7.7   221000.0     Jim Abrahams
4  July 25, 1980 (United States)    7.3   108000.0     Harold Ramis

                  writer            star         country      budget  \
0            Stephen King  Jack Nicholson  United Kingdom  19000000.0
1  Henry De Vere Stacpoole  Brooke Shields   United States   4500000.0
2          Leigh Brackett     Mark Hamill   United States  18000000.0
3            Jim Abrahams     Robert Hays   United States   3500000.0
```

```
4       Brian Doyle-Murray     Chevy Chase   United States   6000000.0

         gross             company  runtime  genre_Action  genre_Adventure  \
0   46998772.0        Warner Bros.    146.0             0                0
1   58853106.0   Columbia Pictures    104.0             0                1
2  538375067.0           Lucasfilm    124.0             1                0
3   83453539.0  Paramount Pictures     88.0             0                0
4   39846344.0      Orion Pictures     98.0             0                0

   genre_Animation  genre_Biography  genre_Comedy  genre_Crime  genre_Drama  \
0                0                0             0            0            1
1                0                0             0            0            0
2                0                0             0            0            0
3                0                0             1            0            0
4                0                0             1            0            0

   genre_Family  genre_Fantasy  genre_Horror  genre_Mystery  genre_Romance  \
0             0              0             0              0              0
1             0              0             0              0              0
2             0              0             0              0              0
3             0              0             0              0              0
4             0              0             0              0              0

   genre_Sci-Fi  genre_Thriller  genre_Western  rating_Approved  rating_G  \
0             0               0              0                0         0
1             0               0              0                0         0
2             0               0              0                0         0
3             0               0              0                0         0
4             0               0              0                0         0

   rating_NC-17  rating_Not Rated  rating_PG  rating_PG-13  rating_R  \
0             0                 0          0             0         1
1             0                 0          0             0         1
2             0                 0          1             0         0
3             0                 0          1             0         0
4             0                 0          0             0         1

   rating_TV-MA  rating_Unrated  rating_X
0             0               0         0
1             0               0         0
2             0               0         0
3             0               0         0
4             0               0         0
```

[ ]:
```
predictors = ["year", "gross", "votes", "budget", "runtime",␣
  ↪"genre_Action",        "genre_Adventure",        "genre_Animation",        "genre_Biography
continuous_predictors = ["year", "gross", "votes", "budget", "runtime"]
```

```
X = movies[predictors]
y = movies["score"]

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.1)

z = StandardScaler()

X_train[continuous_predictors] = z.fit_transform(X_train[continuous_predictors])
X_test[continuous_predictors] = z.transform(X_test[continuous_predictors])

lr = LinearRegression()

lr.fit(X_train, y_train)

print("TRAIN MSE: ", mean_squared_error(y_train, lr.predict(X_train)))
print("TEST MSE: ", mean_squared_error(y_test, lr.predict(X_test)))
print()
print("TRAIN R^2: ", r2_score(y_train, lr.predict(X_train)))
print("TEST R^2: ", r2_score(y_test, lr.predict(X_test)))

coefficients = pd.DataFrame({"Coef": lr.coef_,
                             "Names": predictors})
coefficients
```

```
TRAIN MSE:  0.5624710473882857
TEST MSE:  0.457394975291859

TRAIN R^2:  0.39930241904657804
TEST R^2:  0.45334652745633564
```

[ ]:

	Coef	Names
0	-0.011191	year
1	0.037567	gross
2	0.439780	votes
3	-0.249383	budget
4	0.303563	runtime
5	-0.201144	genre_Action
6	-0.125346	genre_Adventure
7	0.698392	genre_Animation
8	0.405999	genre_Biography
9	-0.078102	genre_Comedy
10	0.135155	genre_Crime
11	0.148599	genre_Drama
12	0.115107	genre_Family
13	-0.234628	genre_Fantasy
14	-0.420124	genre_Horror
15	-0.273161	genre_Mystery

```
16  0.232677    genre_Romance
17  0.018717     genre_Sci-Fi
18 -0.221618   genre_Thriller
19 -0.200526    genre_Western
```

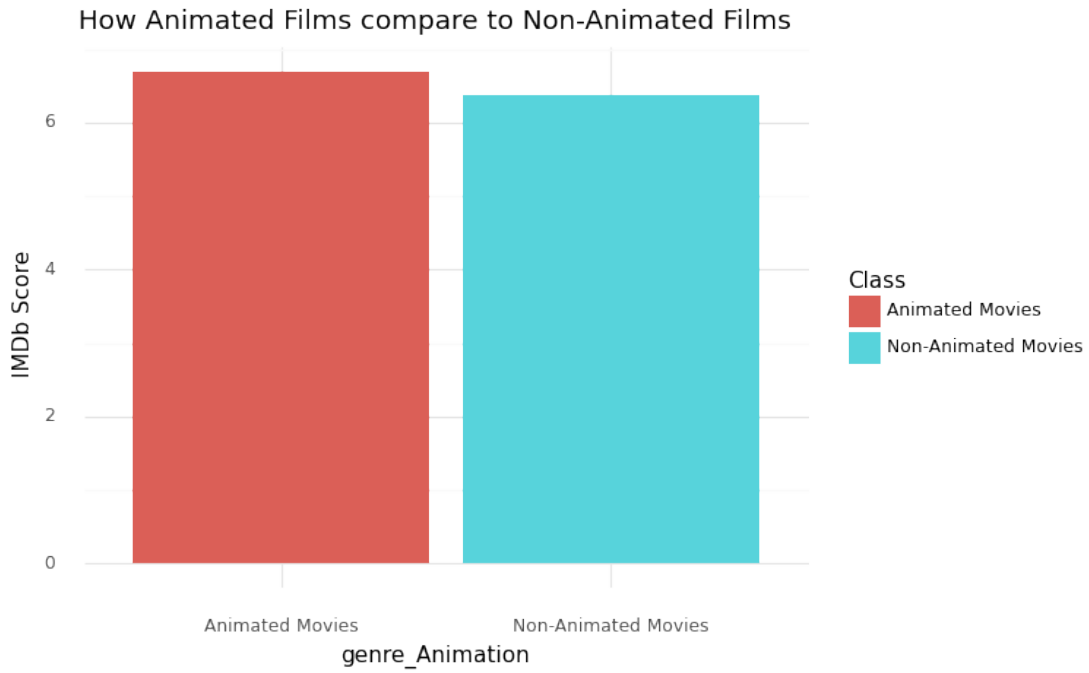
[ ]:
```
(ggplot(coefficients, aes(x = "Names", y = "Coef", fill = "Names" )) +␣
 ↪geom_bar(stat = "identity") + theme_minimal()+ ggtitle("Coefficient Values"))
```

[ ]:
```
<ggplot: (8737242674822)>
```

Graph of the coefficient values produced

[ ]:
```
non_animated_movies = movies[(movies["genre_Animation"] == 0)]
na_movies_scores = non_animated_movies['score'].mean()
animated_movies = movies[(movies["genre_Animation"] == 1)]
a_movies_scores = animated_movies['score'].mean()

temp_df = pd.DataFrame({'Class': ['Animated Movies', 'Non-Animated Movies'],␣
 ↪'IMDb Score': [a_movies_scores,na_movies_scores]})
```

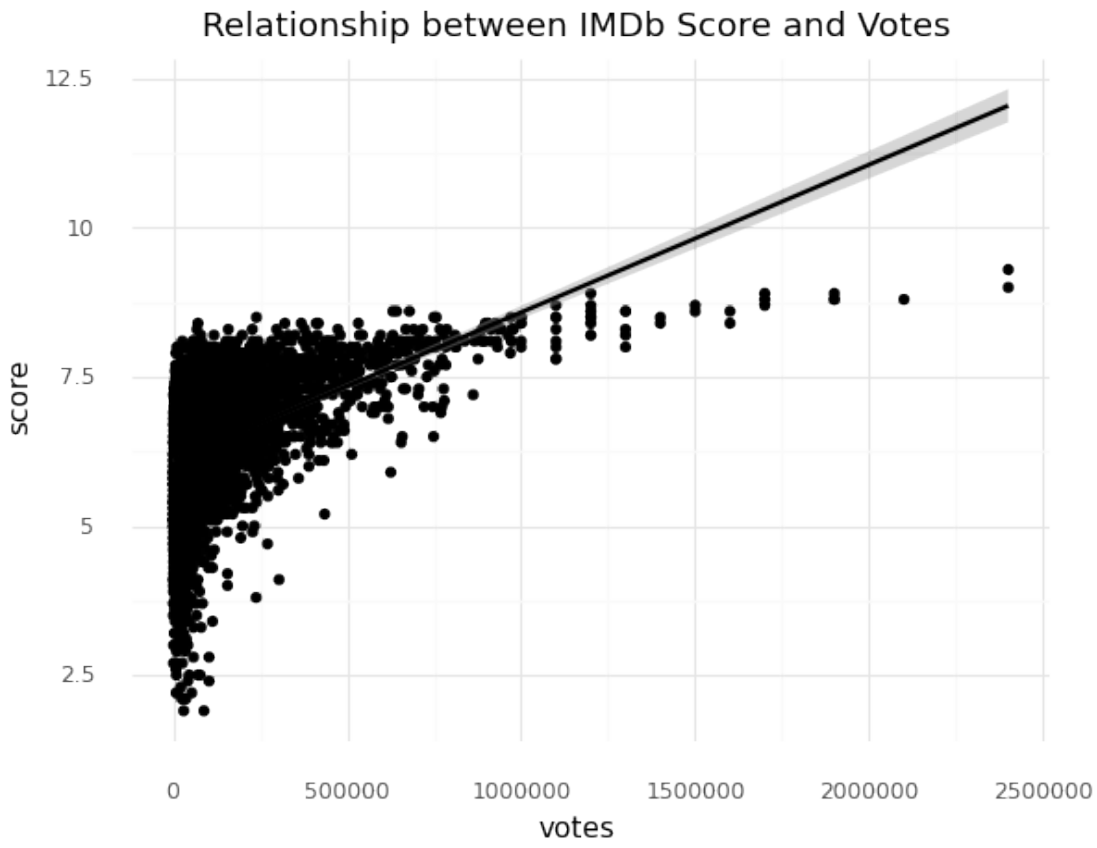
```
(ggplot(temp_df, aes("Class", "IMDb Score")) + geom_bar(aes(fill = "Class"),␣
 ↪stat = "identity") +
 labs(y= "IMDb Score", x = "genre_Animation") +
 theme_minimal() + ggtitle("How Animated Films compare to Non-Animated Films"))
```



[ ]:
```
<ggplot: (8737234338129)>
```

Graph displaying the average IMDb Score of animated movies vs non-animated movies

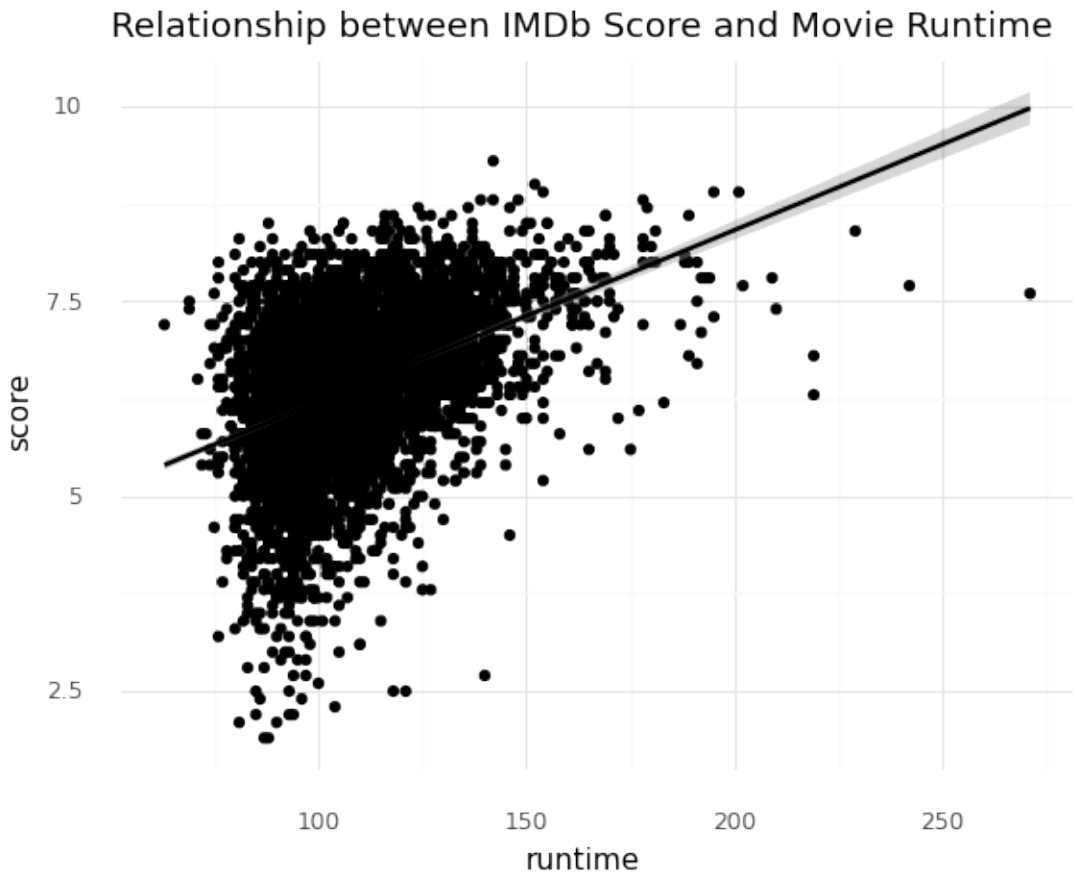
[ ]:
```
(ggplot(movies, aes(x='votes', y='score')) + geom_point() + theme_minimal()+␣
 ↪geom_smooth() + ggtitle("Relationship between IMDb Score and Votes"))
```



[ ]:
```
<ggplot: (8737243059759)>
```

Graph displaying the relationship between IMDb Score and Total Number of Votes

[ ]:
```
(ggplot(movies, aes(x='runtime', y='score')) + geom_point() + theme_minimal()+␣
↪geom_smooth() + ggtitle("Relationship between IMDb Score and Movie Runtime"))
```



[ ]: <ggplot: (8737246074232)>

Graph displaying relationship between IMDb Score and Movie Runtime

1.0.2 Changes Made from Part 3:

I limited the variables used because director, star, company, and country had too many dummy variables and that made the model not as efficient.

1.0.3 Question 1 Answer

Of the variables year, gross, votes, budget, runtime, and the various movie genres the ones with the strongest relationship with a movie's IMDb score are genre_Animation, votes, genre_Horror, genre_Biography, and runtime. This means that holding all other variables constant, if a movie is in the animation genre, it is expected to increase the IMDb score by about 0.698. Holding other variables constant, an increase of one unit in the number of votes is expected to increase a movie's IMDb score by about 0.439. Holding all other variables constant, if a movie is in the horror genre, it is expected to decrease the IMDb score by about 0.420. Holding all other variables constant, if a movie is a biography we can expect to see the IMDb score increase by about 0.405. Finally, holding other variables constant, an increase of one unit in the runtime is expected to increase the

IMDb score by about 0.304.

In the graph titled, "How Animated Films compare to Non-Animated Films", we can see that animated films produce slightly better IMDb scores than non-animated films.

The graph titled "Relationship between IMDb Score and Votes" shows that movies with more total user votes tend to have higher IMDb scores. This is most likely because viewers see movies that they really like then feel inclined to share their opinions and vote on IMDb. The two variables have a positive relationship.

The graph titled "Relationship between IMDb Score and Movie Runtime" shows a positive relationship between the two variables. As the movie runtimes increase the IMDb scores also increae.

2 Question 2

2.0.1 When comparing a model using PCA on all the continuous variables other than score in the dataset and retaining enough PCs to keep 90% of the variance, to a model using all the continuous variables besides score, what is the difference in the mean squared error when predicting the IMDb score of a movie

[ ]:
```
predictors2 = ["year", "gross", "votes", "budget", "runtime"]
X2 = movies[predictors2]
y2 = movies["score"]

X_train2, X_test2, y_train2, y_test2 = train_test_split(X2,y2, test_size=0.2)
z2 = StandardScaler()

X_train2[continuous_predictors] = z2.
  ↪fit_transform(X_train2[continuous_predictors])
X_test2[continuous_predictors] = z2.transform(X_test2[continuous_predictors])

lr = LinearRegression()

lr.fit(X_train2, y_train2)

print("TRAIN MSE: ", mean_squared_error(y_train2, lr.predict(X_train2)))
print("TEST MSE: ", mean_squared_error(y_test2, lr.predict(X_test2)))
```

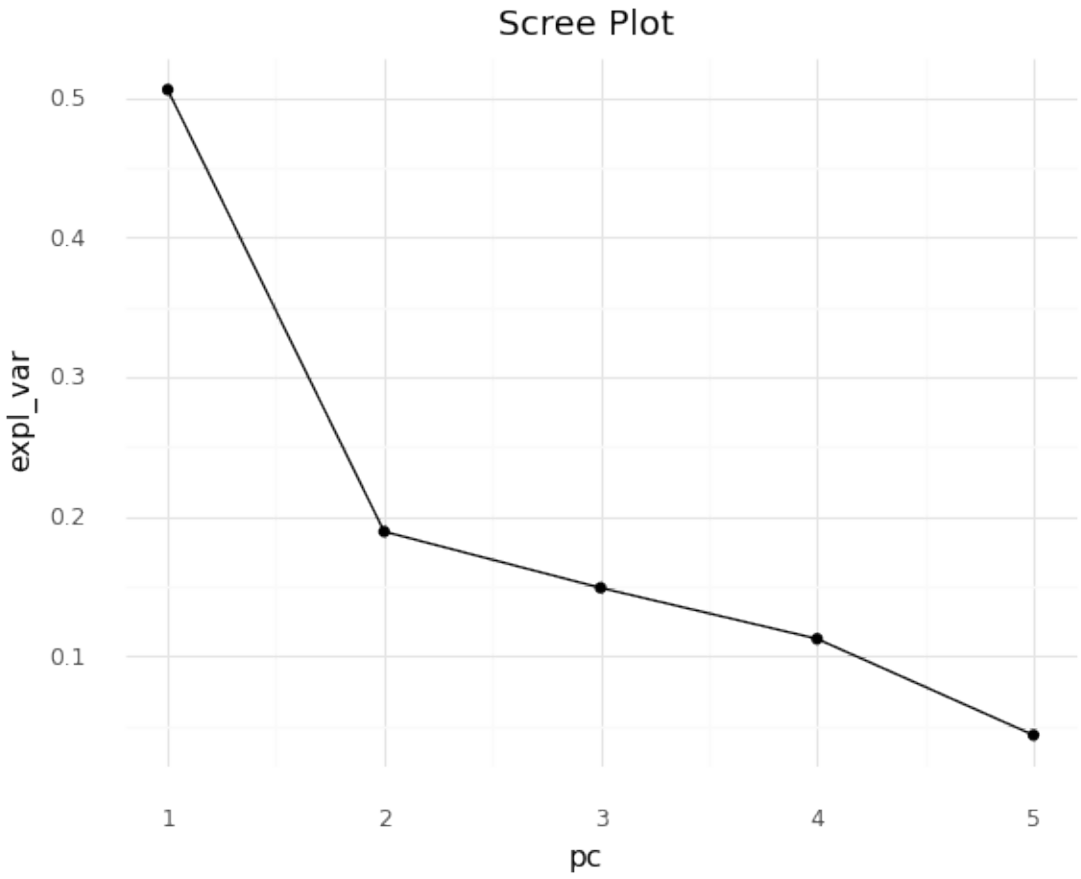
```
TRAIN MSE:  0.6107541050402419
TEST MSE:  0.6193670999904308
```

[ ]:
```
pca = PCA()
pca.fit(X_train2)

pca2 = pd.DataFrame({"expl_var" : pca.explained_variance_ratio_, "pc" :␣
  ↪range(1,6), "cum_var": pca.explained_variance_ratio_.cumsum()})
pca2.head()
```

[ ]:

	expl_var	pc	cum_var
0	0.505921	1	0.505921
1	0.189268	2	0.695190
2	0.148953	3	0.844142
3	0.112417	4	0.956559
4	0.043441	5	1.000000

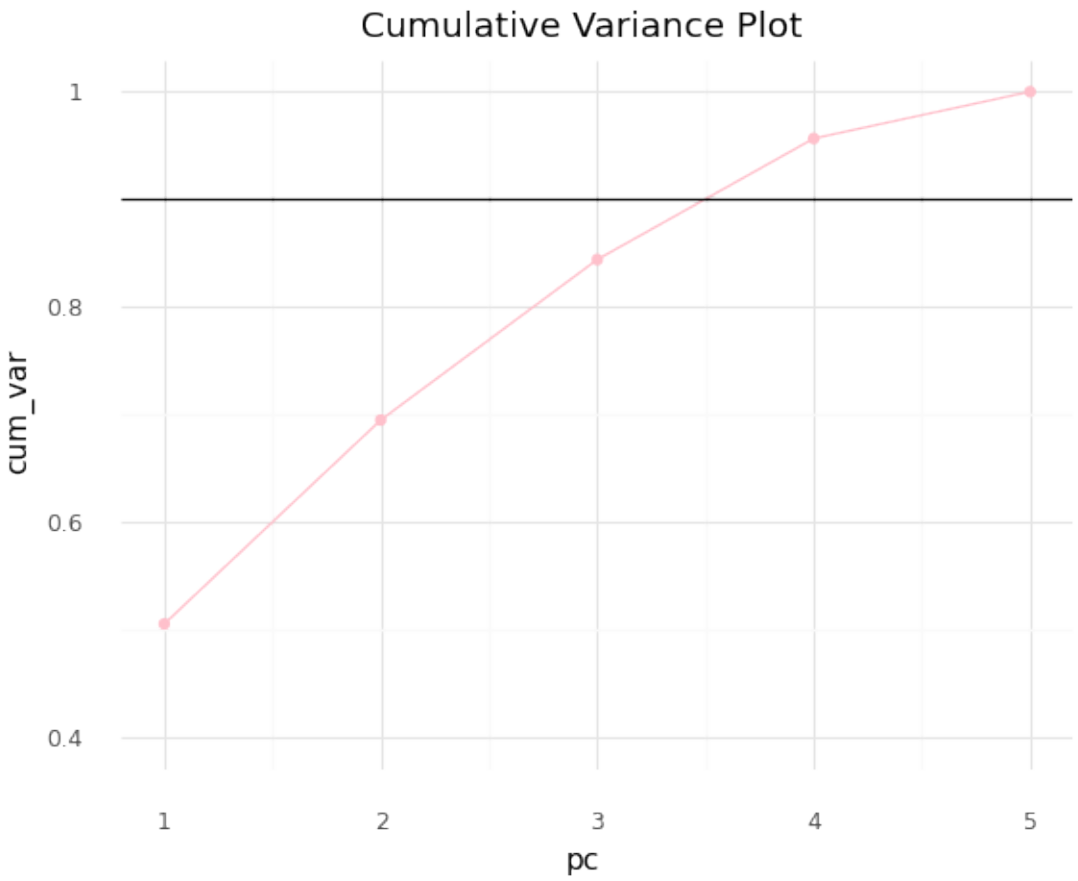
[ ]:
```
(ggplot(pca2, aes(x = "pc", y = "expl_var")) + geom_line() + geom_point() +␣
↪ggtitle("Scree Plot")+ theme_minimal())
```



[ ]:
```
<ggplot: (8737248200747)>
```

Scree plot showing the explained variance values for each individual prinicipal component

[ ]:
```
(ggplot(pca2, aes(x = "pc", y = "cum_var")) + geom_line(color = "pink") +
geom_point(color = "pink") + geom_hline(yintercept = 0.9) +␣
↪scale_y_continuous(limits=(0.4,1.0)) + theme_minimal() + ggtitle("Cumulative␣
↪Variance Plot"))
```



[ ]: <ggplot: (8737242924801)>

Cumulative Variance plot showing amount of the original data explained by each principal component

[ ]:
```
pcomps2 = pca.transform(X2)
pcomps2 = pd.DataFrame(pcomps2[:,0:2])


X_train3, X_test3, y_train3, y_test3 = train_test_split(pcomps2,␣
 ↪movies["score"],test_size = 0.2)

z3 = StandardScaler()

pcomps2 = z3.fit_transform(pcomps2)
lr3 = LinearRegression()
lr3.fit(X_train3, y_train3)
```

```
print("PComps2 Train MSE: ", mean_squared_error(y_train3, lr3.
 ↪predict(X_train3)))
print("PComps2 Test MSE: ", mean_squared_error(y_test3, lr3.predict(X_test3)))
print()
pcomps4 = pca.transform(X2)
pcomps4 = pd.DataFrame(pcomps4[:,0:4])

X_train4, X_test4, y_train4, y_test4 = train_test_split(pcomps4,␣
 ↪movies["score"],test_size = 0.2)

z4 = StandardScaler()

pcomps4 = z4.fit_transform(pcomps4)
lr4 = LinearRegression()
lr4.fit(X_train4, y_train4)

print("PComps4 Train MSE: ", mean_squared_error(y_train4, lr4.
 ↪predict(X_train4)))
print("PComps4 Test MSE: ", mean_squared_error(y_test4, lr4.predict(X_test4)))
```

```
PComps2 Train MSE:  0.8652197672481654
PComps2 Test MSE:  0.8420134327239729

PComps4 Train MSE:  0.6544560958669987
PComps4 Test MSE:  0.6827738352522837
```

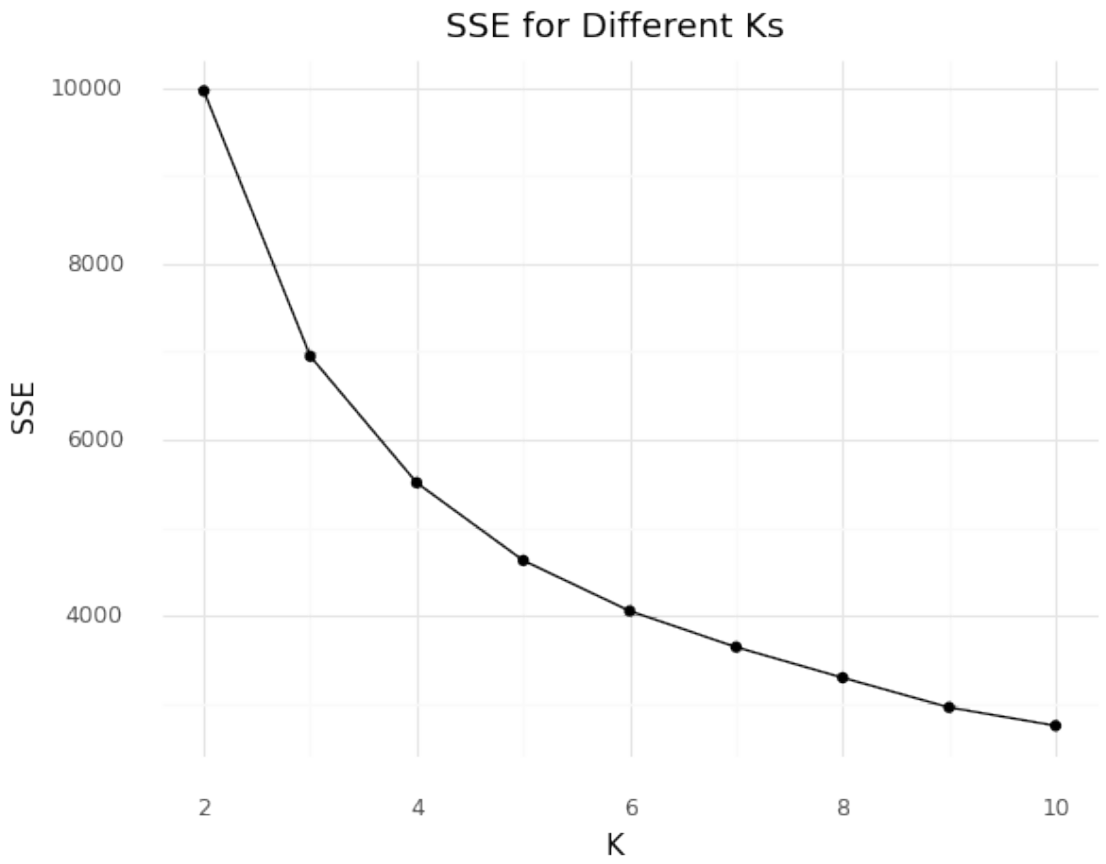
2.0.2 Question 2 Answer

To answer this question, I first created the original model to compare the PCA model to. That original model used all continuous variables except score and it produced train and test MSE values around 0.61. Next, I created a scree plot and a cumulative variance plot to figure out how many PCs I would need to use. Using the elbow method on the scree plot, it revealed that 2 PCs would be the best amount but the cumulative variance plot showed something different. The cumulative variance value passes the 90% mark at 3.5 PCs so 4 PCs would be needed. I then created linear regression models for both and calculated their MSE values. Using 2 PCs resulted in increased train and test MSE at 0.865 and 0.842. Using 4 PCs resulted in a train MSE of 0.654 and a test MSE of 0.683. Both of these are higher than the original model, so the original model is actually performing better than the PCA. The lower the MSE value indicates it is better fit to the data, and in this case the original model is the best fit.

3 Question 3

3.0.1 When considering the variables movie gross, score and budget, what clusters are shown, and describe what those clusters mean for those groups of movies.

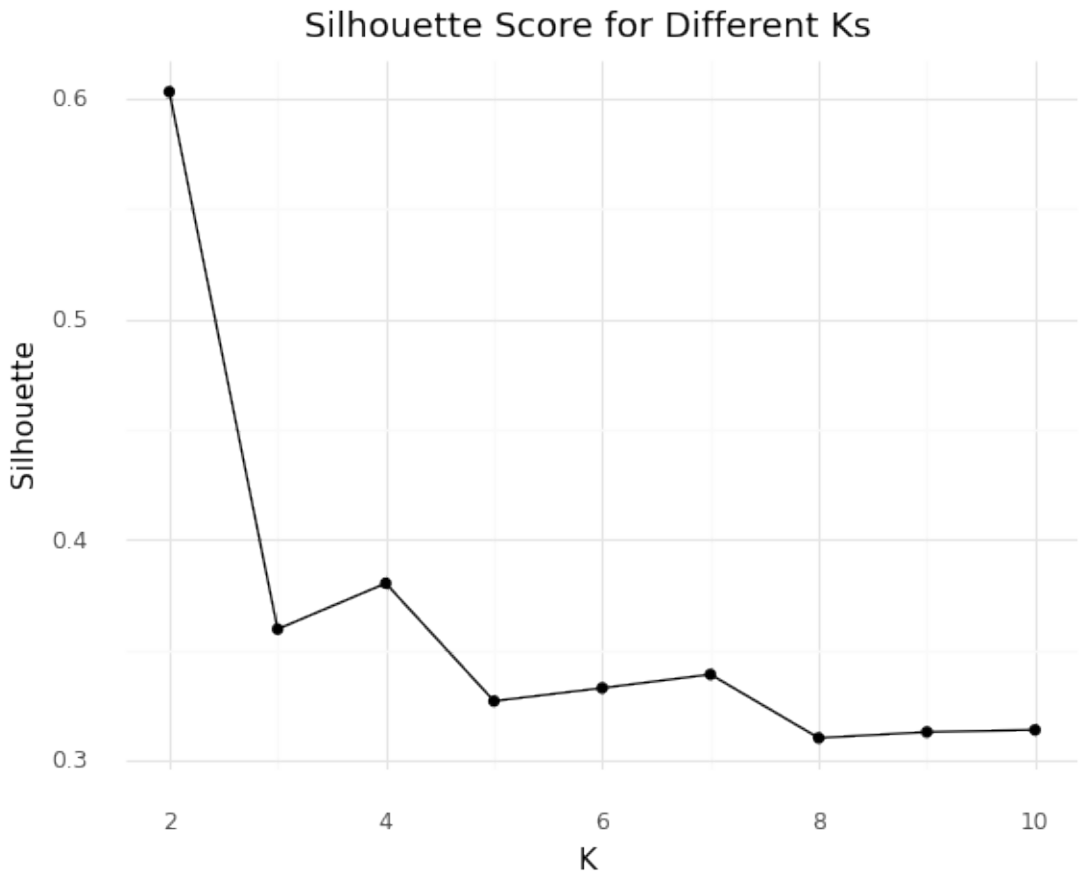
[ ]:
```
new_predictors = ["gross", "budget", "score"]
X = movies[new_predictors]

# Z-score
z = StandardScaler()
X[new_predictors] = z.fit_transform(X)

ks = [2,3,4,5,6,7,8,9,10]
sse = []
sils = []

for k in ks:
  km = KMeans(n_clusters = k)
  km.fit(X[new_predictors])

  sse.append(km.inertia_)
  sils.append(silhouette_score(X[new_predictors], km.
 ↪predict(X[new_predictors])))

sse_df = pd.DataFrame({"K": ks, "SSE": sse, "Silhouette": sils})
(ggplot(sse_df, aes(x = "K", y = "SSE")) + geom_point() + geom_line() +␣
 ↪theme_minimal() +
labs(title = "SSE for Different Ks"))
```



[ ]: <ggplot: (8737239843168)>

Plot of Sum of Squared Error values for different K values

[ ]:
```
(ggplot(sse_df, aes(x = "K", y = "Silhouette")) + geom_point() +
geom_line() +
theme_minimal() +
labs(title = "Silhouette Score for Different Ks"))
```



[ ]: <ggplot: (8737242674978)>

Plot of silhouette scores for different K values

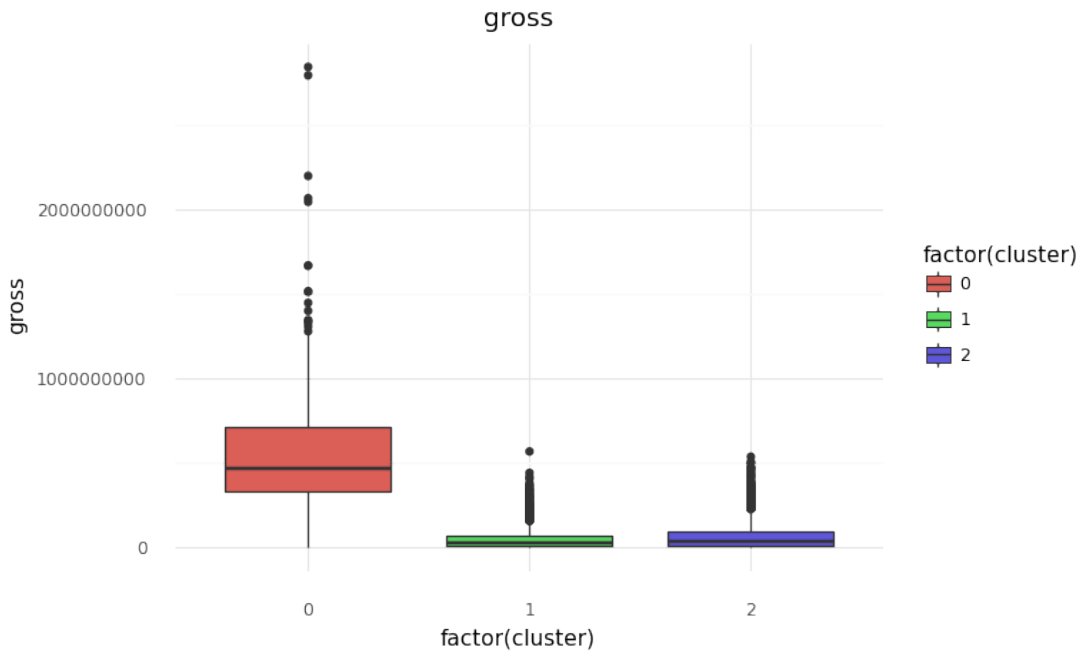
[ ]:
```
km = KMeans(n_clusters = 3)
km.fit(X[new_predictors])

movies["cluster"] = km.predict(X[new_predictors])
movies.head()
```

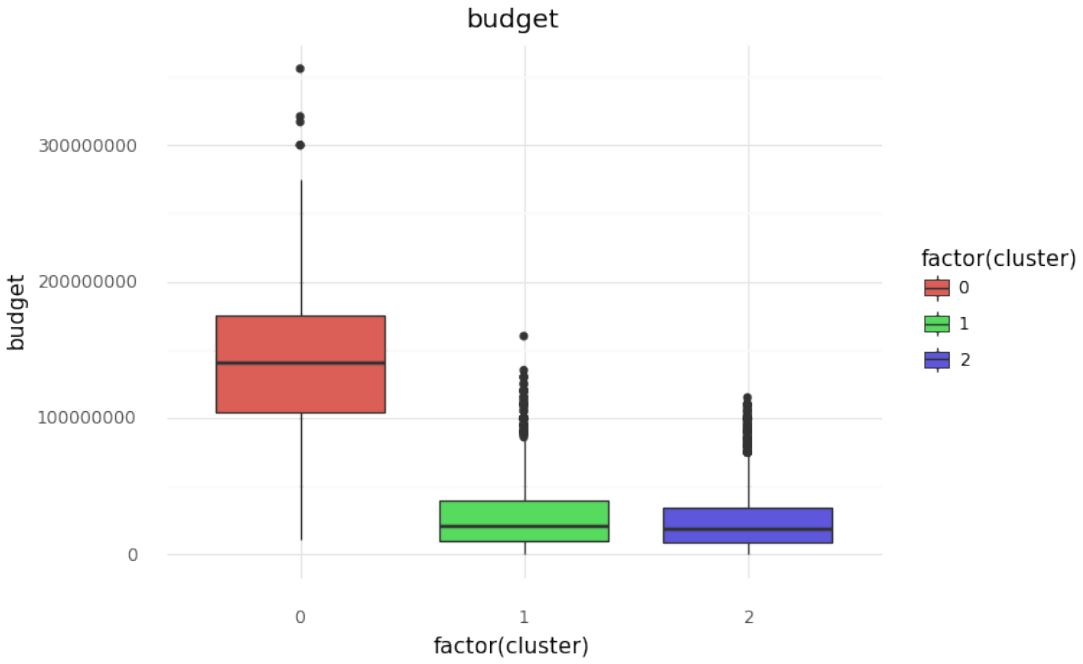
[ ]:
```
                                        name  year  \
0                                The Shining  1980
1                              The Blue Lagoon  1980
2  Star Wars: Episode V - The Empire Strikes Back  1980
3                                   Airplane!  1980
4                                  Caddyshack  1980

                        released  score      votes         director  \
0  June 13, 1980 (United States)    8.4   927000.0  Stanley Kubrick
1   July 2, 1980 (United States)    5.8    65000.0   Randal Kleiser
```

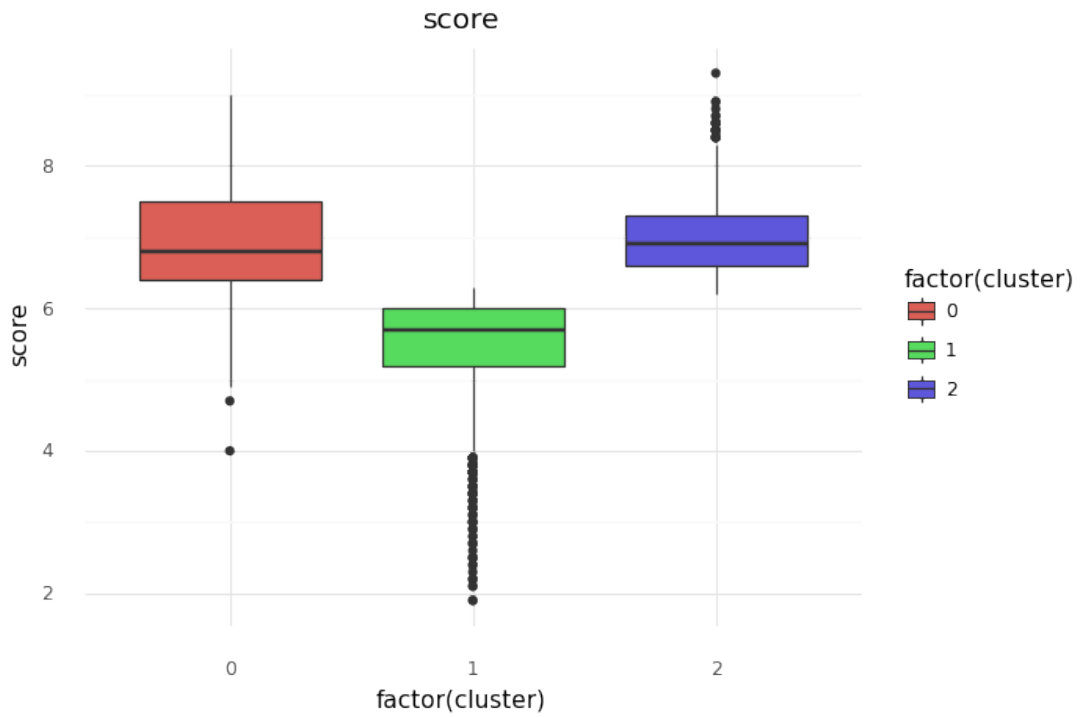
```
2  June 20, 1980 (United States)    8.7  1200000.0   Irvin Kershner
3   July 2, 1980 (United States)    7.7   221000.0     Jim Abrahams
4  July 25, 1980 (United States)    7.3   108000.0     Harold Ramis

                  writer            star         country      budget  \
0            Stephen King  Jack Nicholson  United Kingdom  19000000.0
1  Henry De Vere Stacpoole  Brooke Shields   United States   4500000.0
2          Leigh Brackett     Mark Hamill   United States  18000000.0
3            Jim Abrahams     Robert Hays   United States   3500000.0
4       Brian Doyle-Murray     Chevy Chase   United States   6000000.0

         gross             company  runtime  genre_Action  genre_Adventure  \
0   46998772.0        Warner Bros.    146.0             0                0
1   58853106.0   Columbia Pictures    104.0             0                1
2  538375067.0           Lucasfilm    124.0             1                0
3   83453539.0  Paramount Pictures     88.0             0                0
4   39846344.0      Orion Pictures     98.0             0                0

   genre_Animation  genre_Biography  genre_Comedy  genre_Crime  genre_Drama  \
0                0                0             0            0            1
1                0                0             0            0            0
2                0                0             0            0            0
3                0                0             1            0            0
4                0                0             1            0            0

   genre_Family  genre_Fantasy  genre_Horror  genre_Mystery  genre_Romance  \
0             0              0             0              0              0
1             0              0             0              0              0
2             0              0             0              0              0
3             0              0             0              0              0
4             0              0             0              0              0

   genre_Sci-Fi  genre_Thriller  genre_Western  rating_Approved  rating_G  \
0             0               0              0                0         0
1             0               0              0                0         0
2             0               0              0                0         0
3             0               0              0                0         0
4             0               0              0                0         0

   rating_NC-17  rating_Not Rated  rating_PG  rating_PG-13  rating_R  \
0             0                 0          0             0         1
1             0                 0          0             0         1
2             0                 0          1             0         0
3             0                 0          1             0         0
4             0                 0          0             0         1

   rating_TV-MA  rating_Unrated  rating_X  clusters  cluster
```

0	0	0	0	2	2
1	0	0	0	1	1
2	0	0	0	3	2
3	0	0	0	0	2
4	0	0	0	0	2

[ ]:
```
for p in new_predictors:
    print(ggplot(movies, aes(x = "factor(cluster)", y = p,
                             fill = "factor(cluster)")) +
          geom_boxplot() + theme_minimal() +
          labs(title = p))
```

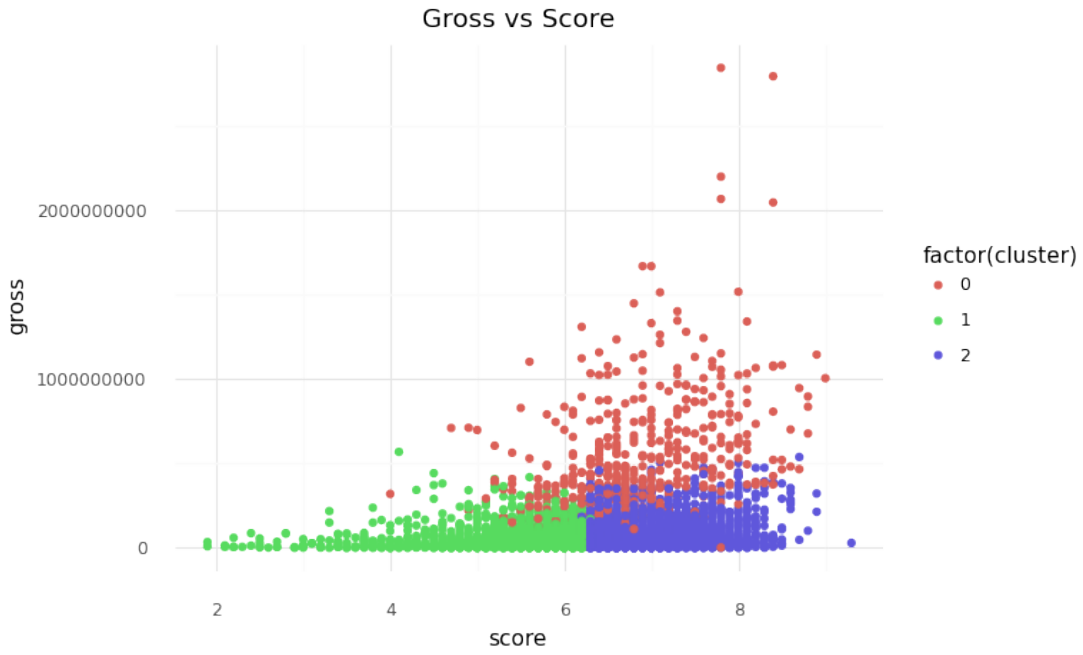



budget
300000000
200000000
100000000
0
budget
0
1
2
factor(cluster)
factor(cluster)
0
1
2


score
8
6
4
2
score
0
1
2
factor(cluster)
factor(cluster)
0
1
2

Boxplots representing the clusters that make up each variable

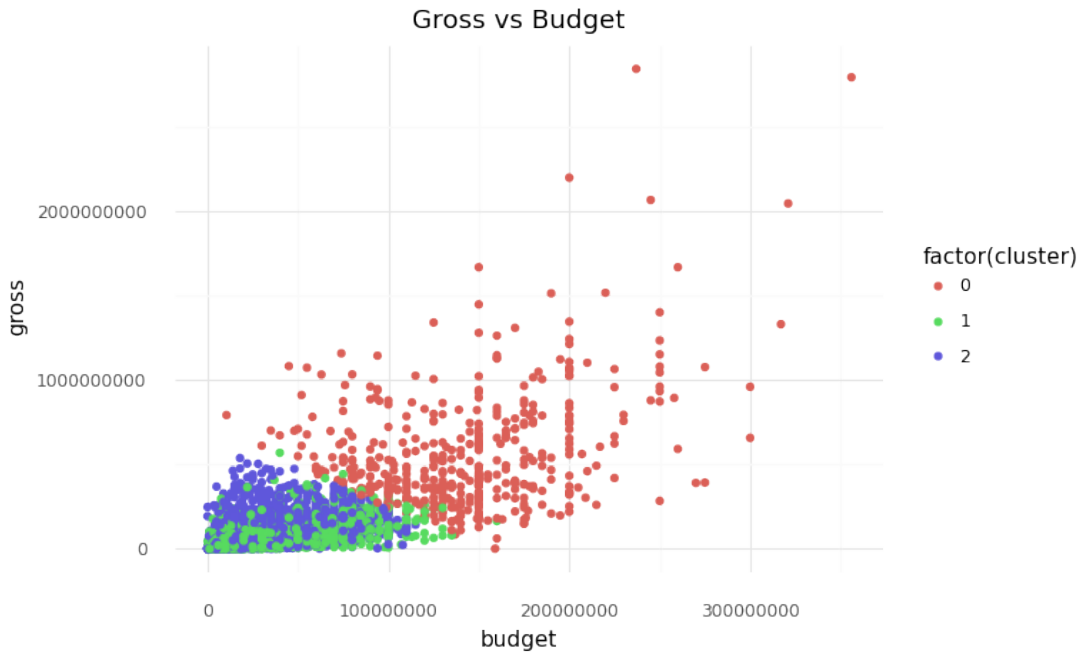
```
[ ]: (ggplot(movies, aes(x ="score", y="gross", color = "factor(cluster)"))+␣
     ↪geom_point()+theme_minimal() + labs(x="score", y="gross", title = "Gross vs␣
     ↪Score"))
```




```
[ ]: <ggplot: (8737309741579)>
```

Graph displaying the clusters in the relationship between Gross and Score

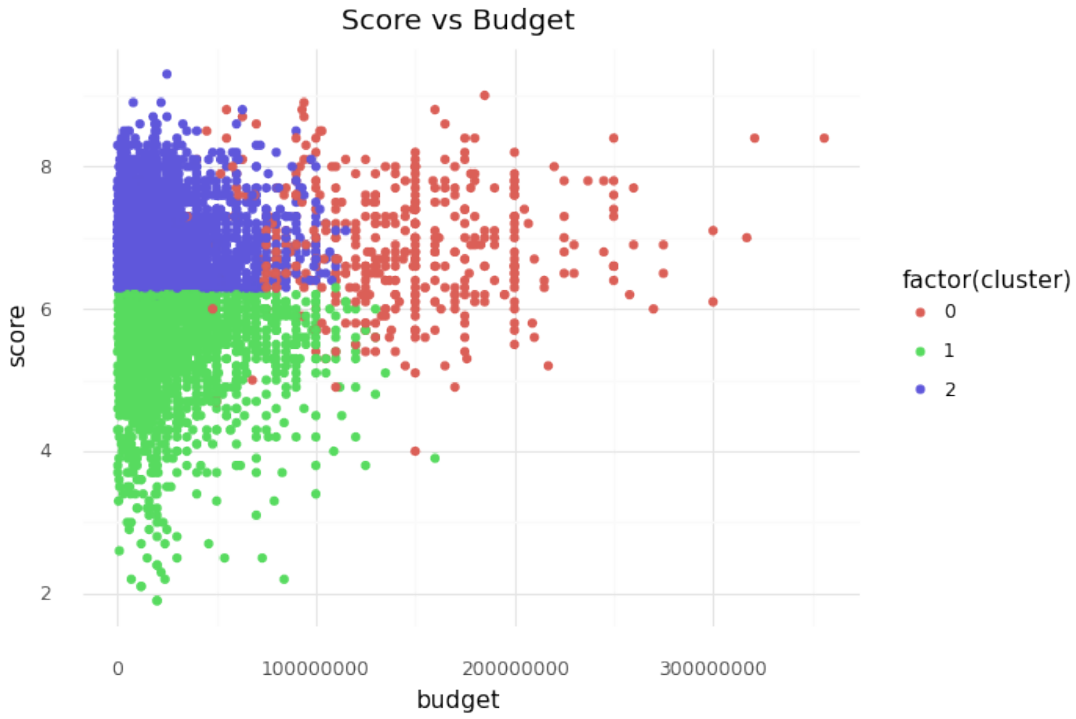
```
[ ]: (ggplot(movies, aes(x ="budget", y="gross", color = "factor(cluster)"))+␣
     ↪geom_point()+ theme_minimal() + labs(x="budget", y="gross", title = "Gross␣
     ↪vs Budget"))
```



[ ]: <ggplot: (8737243196023)>

Graph displaying the clusters in the relationship between Gross and Budget

[ ]:
```
(ggplot(movies, aes(x ="budget", y="score", color = "factor(cluster)"))+␣
↪geom_point()+ theme_minimal() + labs(x="budget", y="score", title = "Score␣
↪vs Budget"))
```



[ ]: <ggplot: (8737243429255)>

Graph displaying the clusters in the relationship between Score and Budget

3.0.2 Question 3 Answer

The three graphs above display the clusters created for the variables movie gross, score, and budget.

In the first graph we see Gross being compared to Score. Cluster zero appears to be the major blockbuster hit movies. In this cluster, these movies have high IMDb scores and high box office grosses. This is the cluster where the major movie studios would like to see their big budget movies be in. Cluster one appears to be movies that have low IMDb scores and low box office grosses. These movies would be considered flops and studios would not want to see their movies in this one. Cluster two appears to be movies that have very good IMDb scores but somewhat low grosses. These movies would be considered well-received by audiences and critics, but not financial success for the studios

The second graph shows movie gross compared to the budget. Cluster zero appears to be the major blockbuster hit movies with large budgets. These movies would be considered major successes for movie studios because they grossed much more money than it cost to produce the movie. Cluster one has movies that did not gross much money and had decently high budgets. The movies in this cluster would be considered flops. Cluster two has movies that performed decently and did not have that high of budgets. These movies would be considered successes for movie studios because it looks like most of these broke even and some made more than it cost to make.

The third graph shows IMDb scores compared to movie budgets. Cluster zero contains movies

that have high budgets that produced relatively good IMDb scores. These would be considered a success because the money put into the movie has resulted in audiences enjoying it. Cluster one consists of movies that have low IMDb scores and decent sized budgets. These movies would be considered flops because the money put into the movie did not result in audiences enjoying it. Cluster 2 consists of movies that have decent sized budgets but very good IMDb scores. These movies would be considered great successes amongst movie studios because they did not cost as much to make as the blockbusters but audiences really enjoyed watching them.

[2]:
```
# doesn't show this cells output when downloading PDF
!pip install gwpy &> /dev/null

# installing necessary files
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!sudo apt-get update
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-plain-generic

# installing pypandoc
!pip install pypandoc

# connecting your google drive
from google.colab import drive
drive.mount('/content/drive')

# copying your file over. Change "Class6-Completed.ipynb" to whatever your file␣
 ↪is called (see top of notebook)
!cp "drive/My Drive/Colab Notebooks/CPSC_392_Final_Project.ipynb" ./

# Again, replace "Class6-Completed.ipynb" to whatever your file is called (see␣
 ↪top of notebook)
!jupyter nbconvert --to PDF "CPSC_392_Final_Project.ipynb"
```

```
Reading package lists… Done
Building dependency tree
Reading state information… Done
pandoc is already the newest version (2.5-3build2).
texlive is already the newest version (2019.20200218-1).
texlive-latex-extra is already the newest version (2019.202000218-1).
texlive-xetex is already the newest version (2019.20200218-1).
0 upgraded, 0 newly installed, 0 to remove and 37 not upgraded.
Hit:1 https://cloud.r-project.org/bin/linux/ubuntu focal-cran40/ InRelease
Hit:2 http://archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Hit:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64
InRelease
Get:5 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:7 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu focal InRelease [18.1
```

```
kB]
Hit:8 http://ppa.launchpad.net/cran/libgit2/ubuntu focal InRelease
Hit:9 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu focal InRelease
Hit:10 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu focal InRelease
Get:11 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages
[2,681 kB]
Hit:12 http://ppa.launchpad.net/ubuntugis/ppa/ubuntu focal InRelease
Get:13 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu focal/main amd64
Packages [1,215 kB]
Get:14 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages
[1,046 kB]
Fetched 5,296 kB in 2s (2,251 kB/s)
Reading package lists… Done
Reading package lists… Done
Building dependency tree
Reading state information… Done
texlive-fonts-recommended is already the newest version (2019.20200218-1).
texlive-plain-generic is already the newest version (2019.20200218-1).
texlive-xetex is already the newest version (2019.20200218-1).
0 upgraded, 0 newly installed, 0 to remove and 37 not upgraded.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: pypandoc in /usr/local/lib/python3.10/dist-
packages (1.11)
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
[NbConvertApp] Converting notebook CPSC_392_Final_Project.ipynb to PDF
[NbConvertApp] Support files will be in CPSC_392_Final_Project_files/
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Making directory ./CPSC_392_Final_Project_files
[NbConvertApp] Writing 106025 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
```

```
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 458759 bytes to CPSC_392_Final_Project.pdf
```