In [30]:

```
import warnings
warnings.filterwarnings('ignore')

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

import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import mean_squared_error, r2_score, mean_a
bsolute_error
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LassoCV, RidgeCV
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score
import scipy.cluster.hierarchy as sch
from matplotlib import pyplot as plt

%matplotlib inline
```

Data Cleaning

In [53]:

```
pathDF = "/Users/cadystringer/Downloads/finalNetData.csv"
net = pd.read_csv(pathDF)
netflix = pd.read_csv(pathDF)
ratings_mean_votes = pd.read_csv("/Users/cadystringer/Downloads/
ratings_mean_votes.csv")
age_mean_votes = pd.read_csv("/Users/cadystringer/Downloads/ageL
vl_mean_votes.csv")
year_mean_votes = pd.read_csv("/Users/cadystringer/Downloads/rel
easeYear_mean_votes.csv")
scoreRating_mean_votes = pd.read_csv("/Users/cadystringer/Downlo
ads/scoreRating_mean_votes.csv")
TVandMovie = pd.read_csv("/Users/cadystringer/Downloads/netflix_
titles.csv")
```

In [49]:

```
net = net.rename(columns={'Centry21': 'century_21', 'agelvl': 'a
ge_level'})
net.head()
```

Out[49]:

	title	genre	director	weighted_average_vote	country	release_
0	¡Ay, mi madre!	Comedy	Frank Ariza	3.8	Spain	2
1	#Roxy	Multiple	Michael Kennedy	5.2	Canada	2
2	1 Chance 2 Dance	Multiple	Adam Deyoe	4.8	United States	2
3	1 Mile to You	Multiple	Leif Tilden	6.3	United States	2
4	10 jours en or	Multiple	Nicolas Brossette	5.8	France	2

In [50]:

```
#Creates factor levels for all the categorical variables we're g
oing to use in models later
le = LabelEncoder()
net["genre"] = le.fit_transform(net["genre"])
net["director"] = le.fit_transform(net["director"])
net["country"] = le.fit_transform(net["country"])
net["rating"] = le.fit_transform(net["rating"])
net["century_21"] = le.fit_transform(net["century_21"])
net["age_level"] = le.fit_transform(net["age_level"])
```

1. Does rating (PG-13, R, G, etc.) have an impact on IMDB score?

Rating does have an impact on IMDB score but the coefficient is pretty small. Interestingly, duration has the most impact on the IMDB score because it has the coefficient with the greatest magnitude.

In [74]:

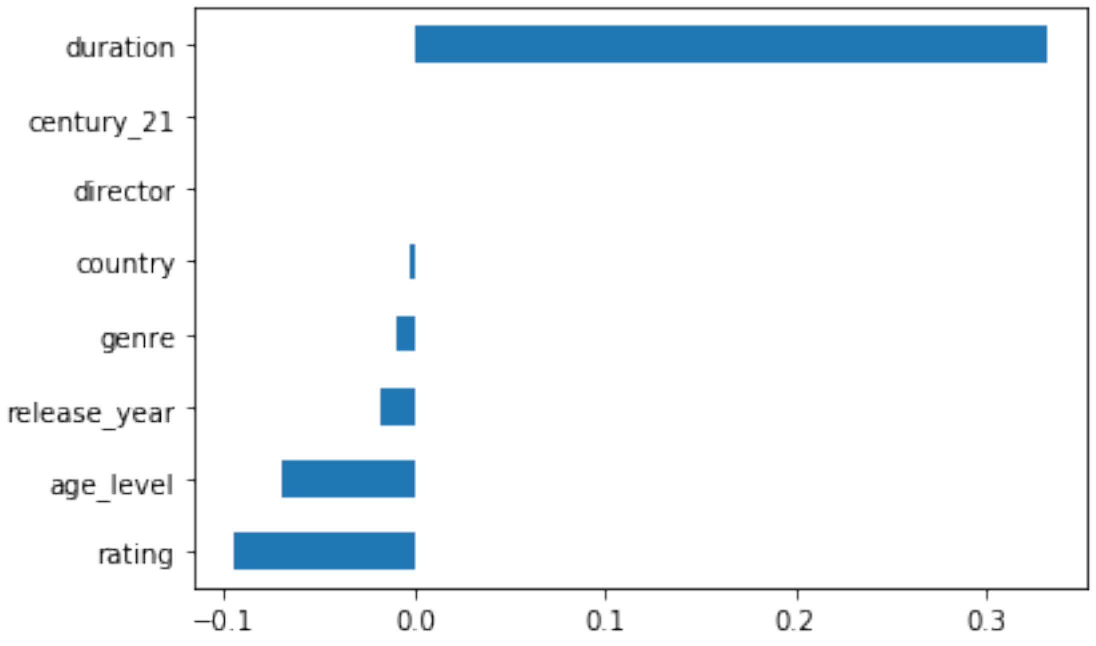
```
feat = ["genre", "director", "country", "release_year", "rating"
, "duration", "century_21", "age_level"]
cont_feat = ["rating","duration"]
X = net[feat]
y = net["weighted_average_vote"]

zscore = StandardScaler()
zscore.fit(X[cont_feat])
X[cont_feat] = zscore.transform(X[cont_feat])
X_train, X_test, y_train, y_test = train_test_split(X,y, test_si
ze=0.2)
```

In [75]:

```
lasso = LassoCV()
lasso.fit(X, y)
coef = pd.Series(lasso.coef_, index = X.columns)

imp_coef = coef.sort_values()
import matplotlib
imp_coef.plot(kind = "barh")
```

Out[75]:

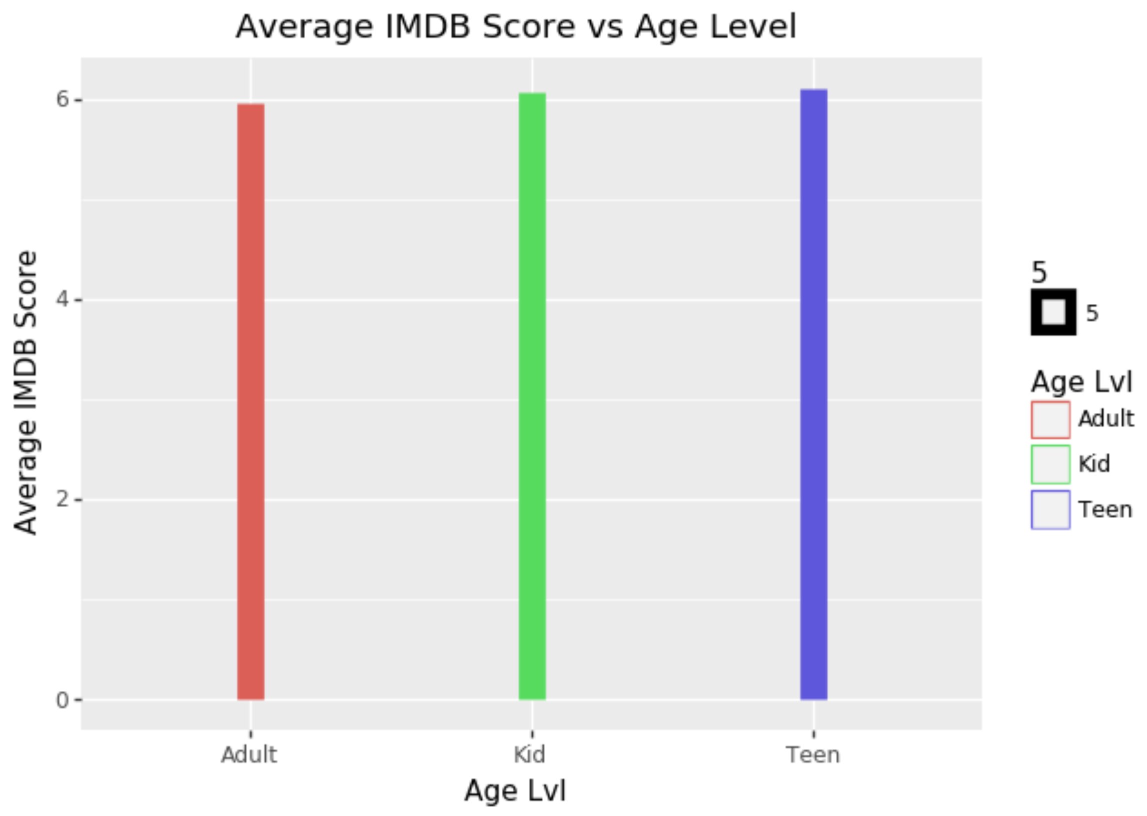
```
<matplotlib.axes._subplots.AxesSubplot at 0x1214f591
0>
```



Data Visualization

In [35]:

```
(ggplot(age_mean_votes, aes("Age Lvl", "Mean",color = "Age Lvl")
)
 + geom_density(aes(size = 5))
 + ylab("Average IMDB Score")
 + ggtitle("Average IMDB Score vs Age Level"))
```

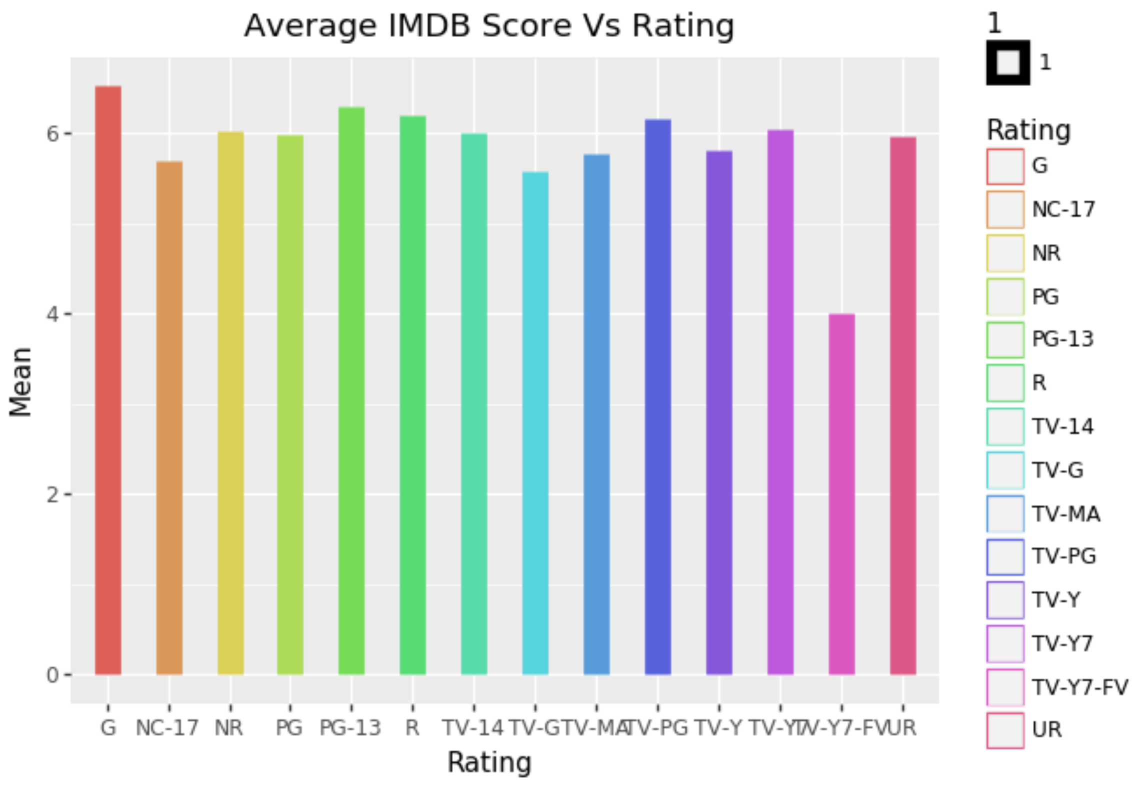


Out[35]:

```
<ggplot: (304195853)>
```

In [34]:

```
(ggplot(ratings_mean_votes, aes("Rating", "Mean", color = "Ratin
g"))
 + geom_density(aes(size = 1))
 + ggtitle("Average IMDB Score Vs Rating"))
```



Out[34]:

```
<ggplot: (309198941)>
```

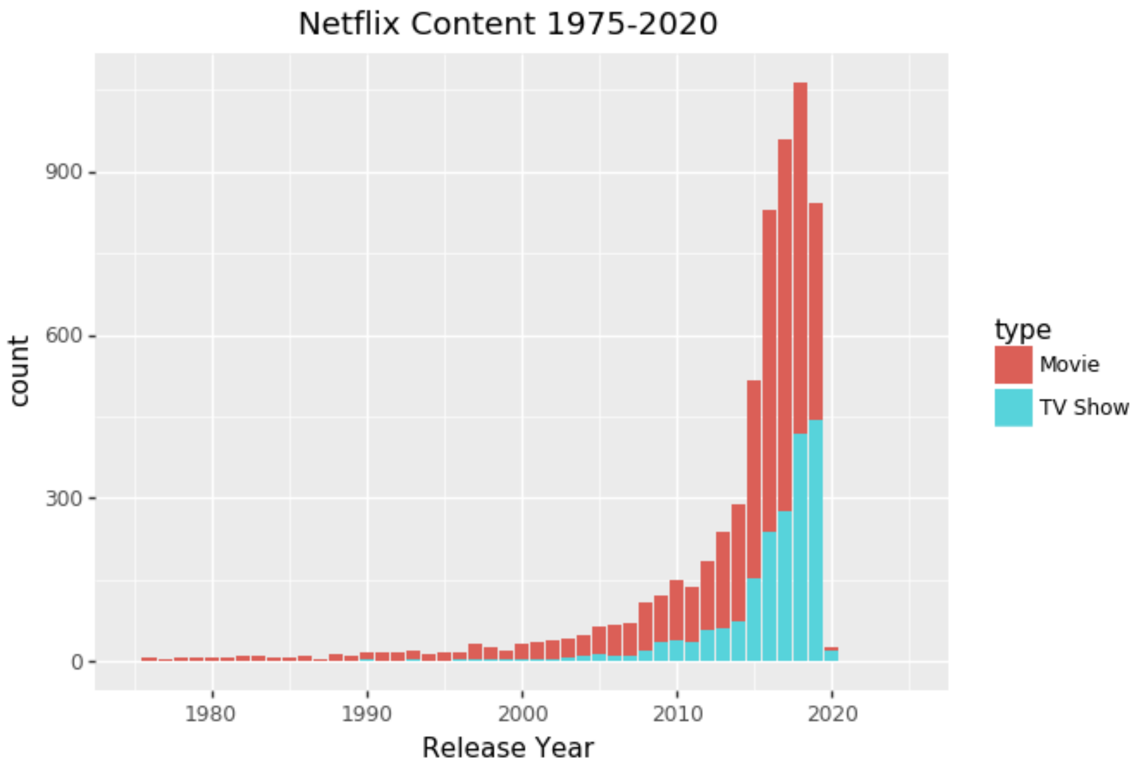
2. Is Netflix increasingly focusing on TV rather than movies in recent years?

Data Visualization

No, they aren't increasingly focusing on TV recently. In total there are a lot more movies than TV shows on Netflix, and we don't see an increase in the amount of TV shows added relative to the amount of movies added. The split seems fairly even. The difference in total amounts seen in the second plot can be attributed to the fact that TV shows were invented more recently, and there were 50+ years without any TV shows, which gave the movies category a head start.

In [42]:

```
(ggplot(TVandMovie, aes("release_year"))
 + geom_bar(aes(fill = "type"))
 + xlim(1975,2025)
 + ggtitle("Netflix Content 1975-2020")
 + xlab("Release Year"))
```

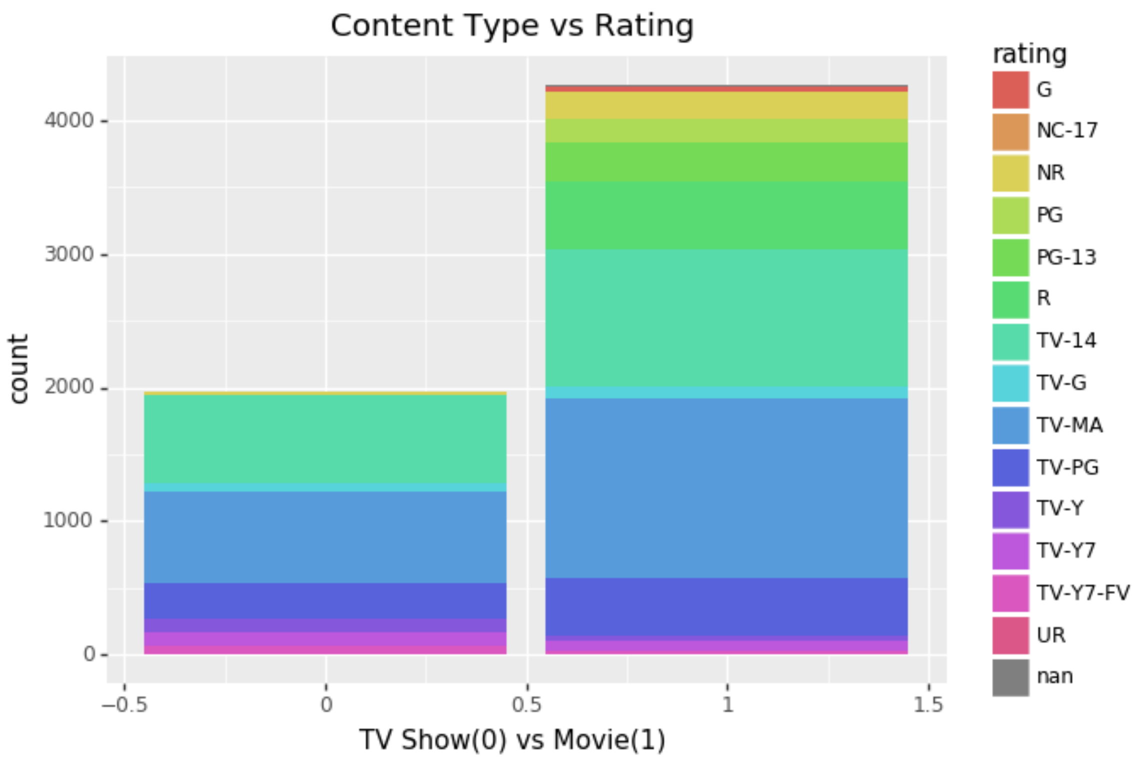


Out[42]:

```
<ggplot: (306979033)>
```

In [36]:

```
(ggplot(TVandMovie, aes("typeMovie", fill = "rating"))
 + geom_bar()
 + xlab("TV Show(0) vs Movie(1)")
 + ggtitle("Content Type vs Rating"))
```



Out[36]:

```
<ggplot: (306382401)>
```

3. How well can we predict the century a film was made?

Logistic Regression with Ridge Penalty and K Fold

Model Performance

This model did pretty well! We can do a better job than random chance when predicting the century a film was made. Although our accuracy was only 62% on average, our best k fold model had an accuracy of about 70%. This was our best prediction model by far.

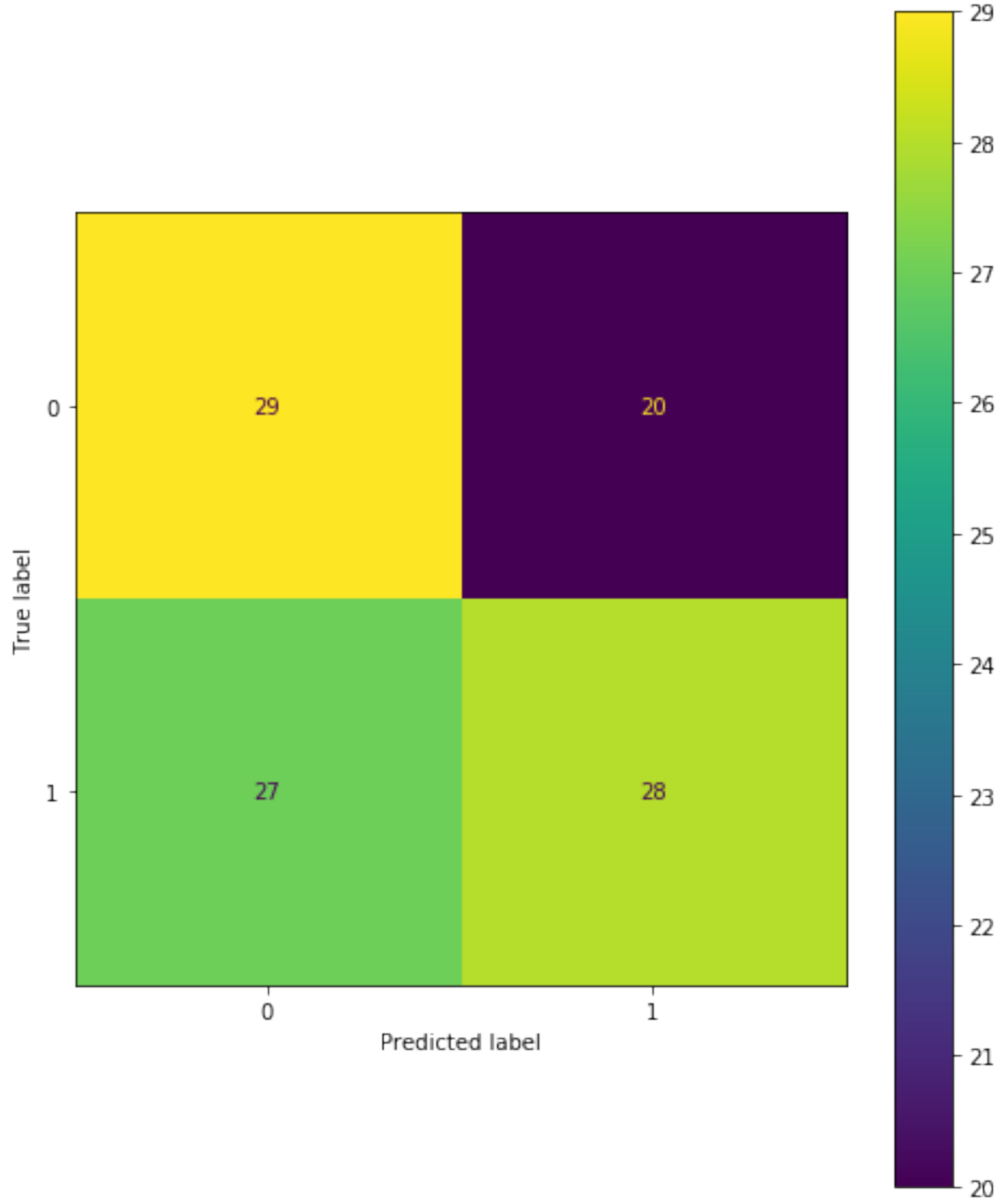
In [367]:

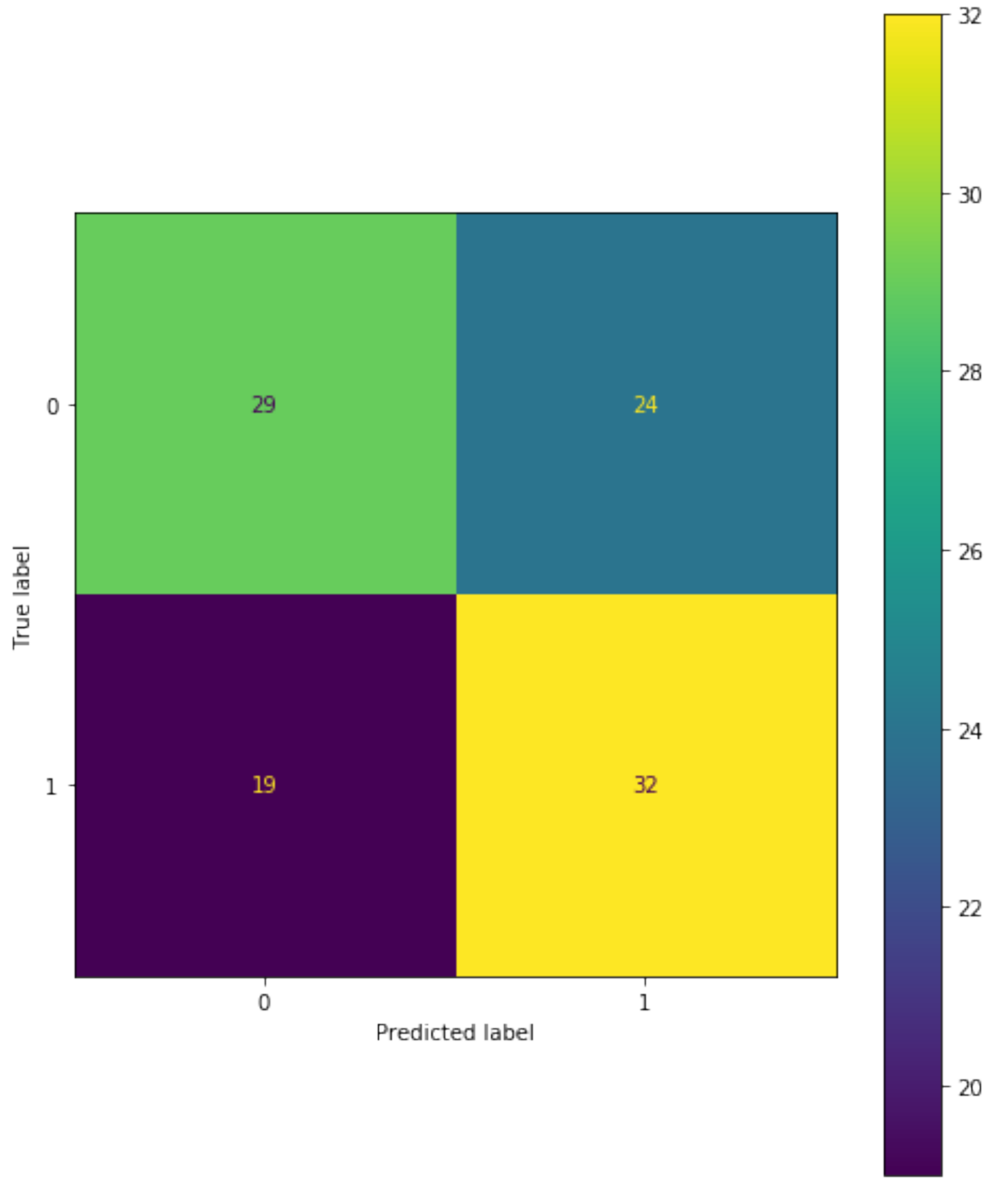
```
feat = ["genre", "director", "country", "release_year", "rating"
, "duration", "weighted_average_vote", "age_level"]
cont_feat = ["duration","weighted_average_vote"]
X = net[feat]
y = net["century_21"]

century_21 = np.where(y == 1)[0]
notcentury_21 = np.where(y == 0)[0]

century_21DownSample = np.random.choice(century_21, size = len(n
otcentury_21),replace = False)

downSampledData = np.concatenate([notcentury_21, century_21DownS
ample])
downSampledData

X_d = net.iloc[downSampledData,][feat]
y_d = net.iloc[downSampledData,]["century_21"]
```

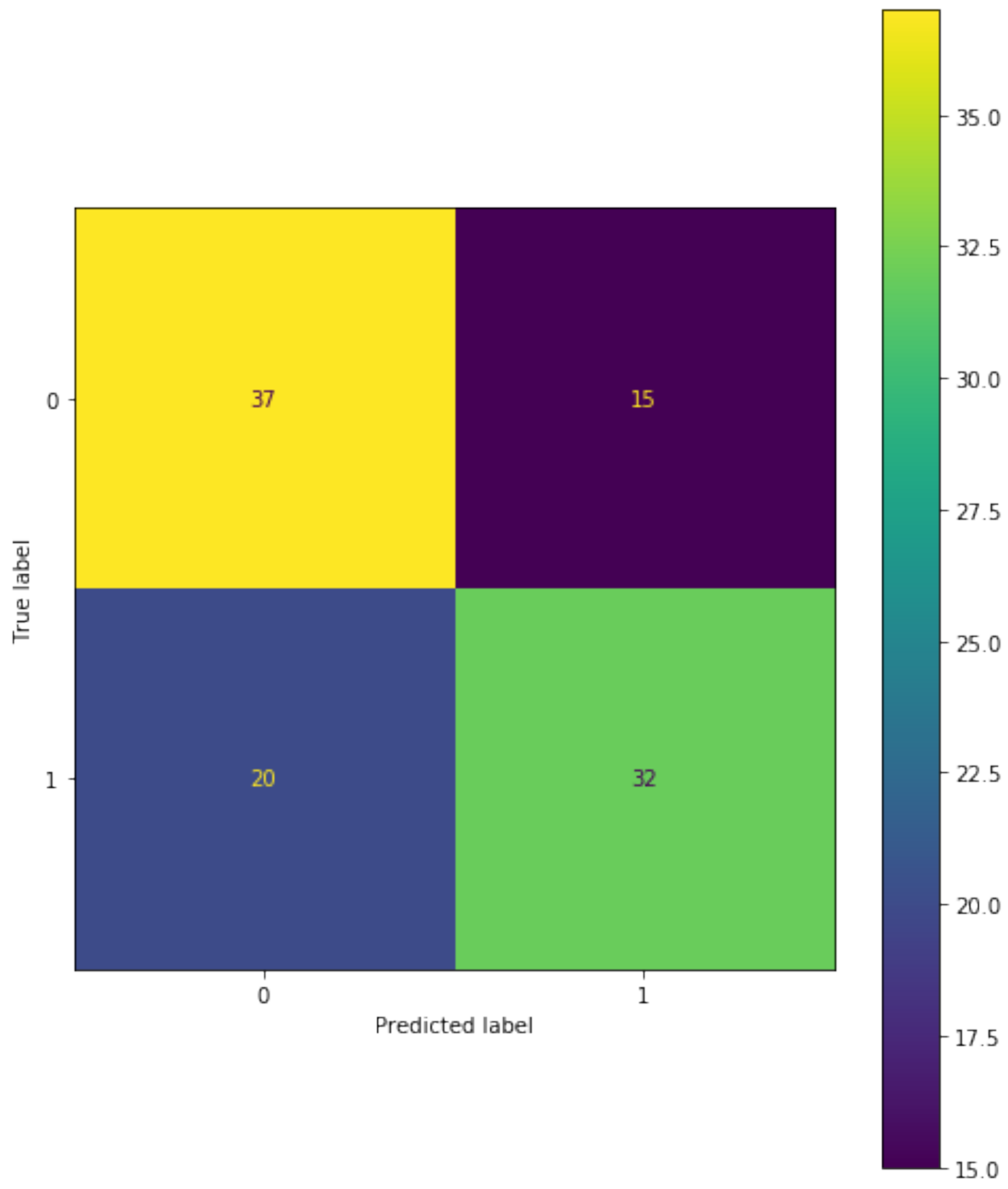
In [21]:

```
logistic = LogisticRegression(penalty='l2')
n_folds = 5
acc = []

for i in range(5):
    X_train, X_test, y_train, y_test = train_test_split(X_d, y_d
, test_size = 0.2)
    logistic.fit(X_train,y_train)
    acc.append(logistic.score(X_test,y_test))
    plot_confusion_matrix(logistic,X_test,y_test)

print(np.mean(acc))
print(acc)
```

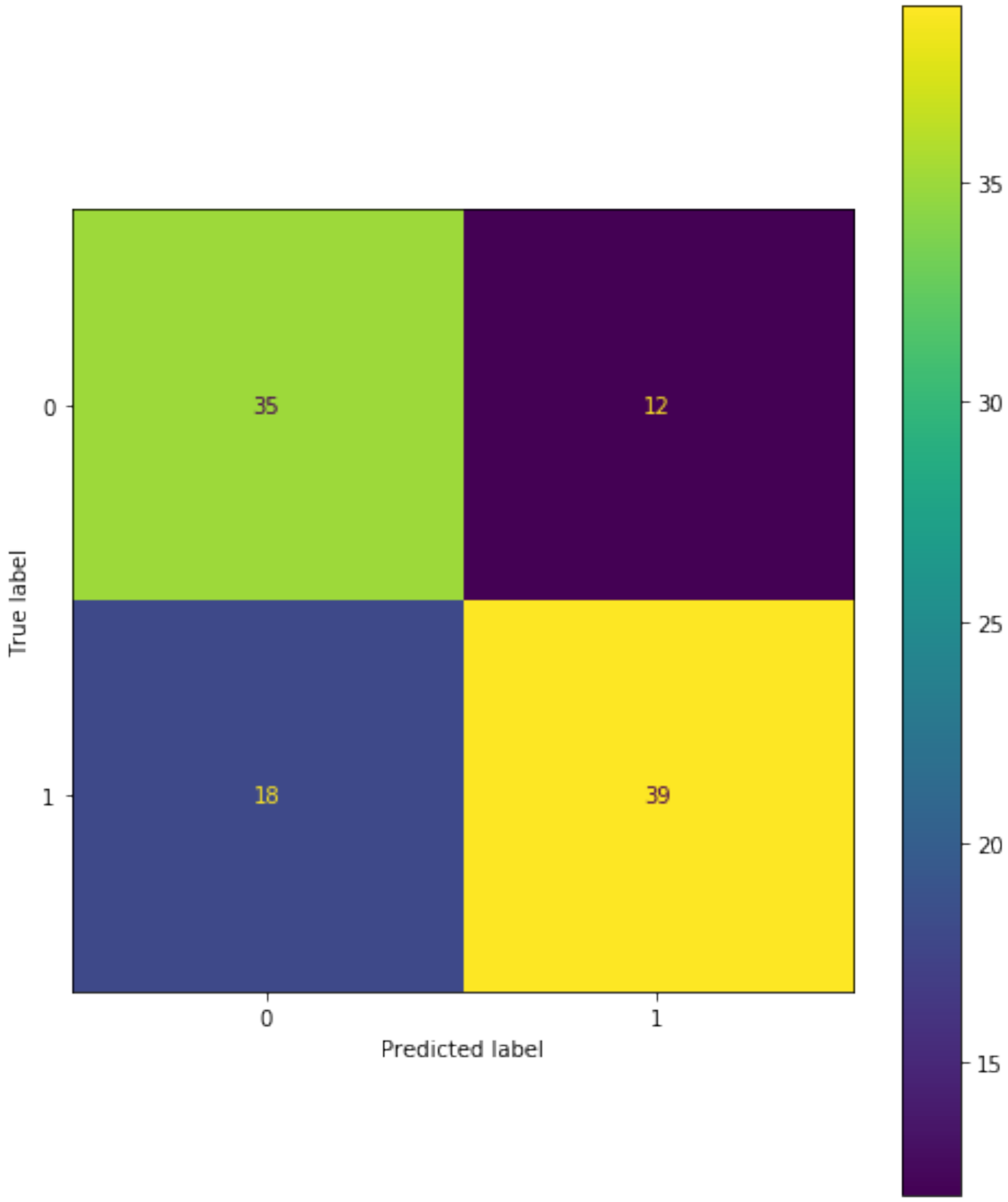
```
0.6230769230769231
[0.5480769230769231, 0.5865384615384616, 0.663461538
4615384, 0.7115384615384616, 0.6057692307692307]
```

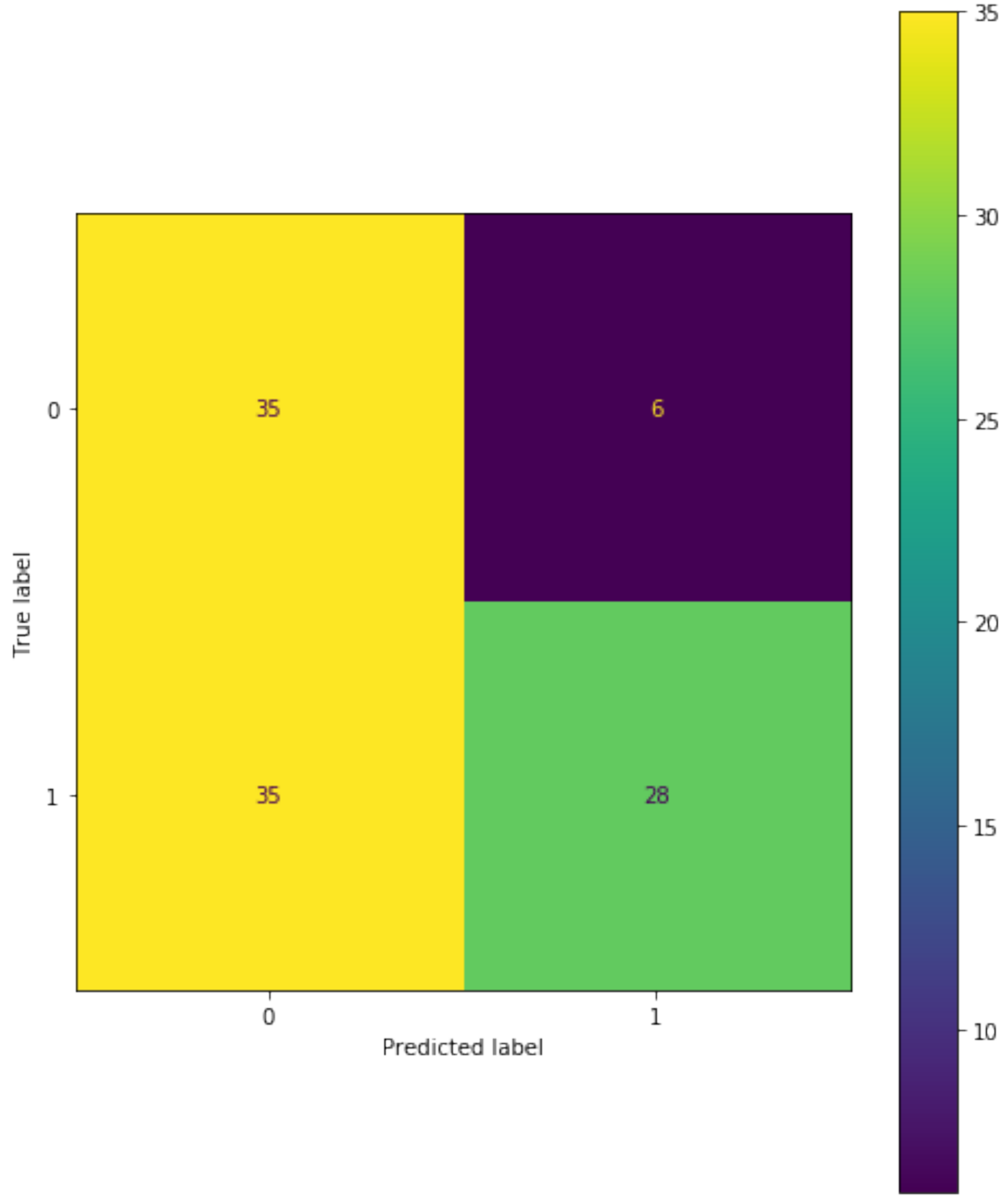

29
20
27
28
0
1
True label
0
1
Predicted label
29
28
27
26
25
24
23
22
21
20


29
24
19
32
0
1
True label
0
1
Predicted label
32
30
28
26
24
22
20



True label
Predicted label
0
1
0
1
37
15
20
32
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
15.0

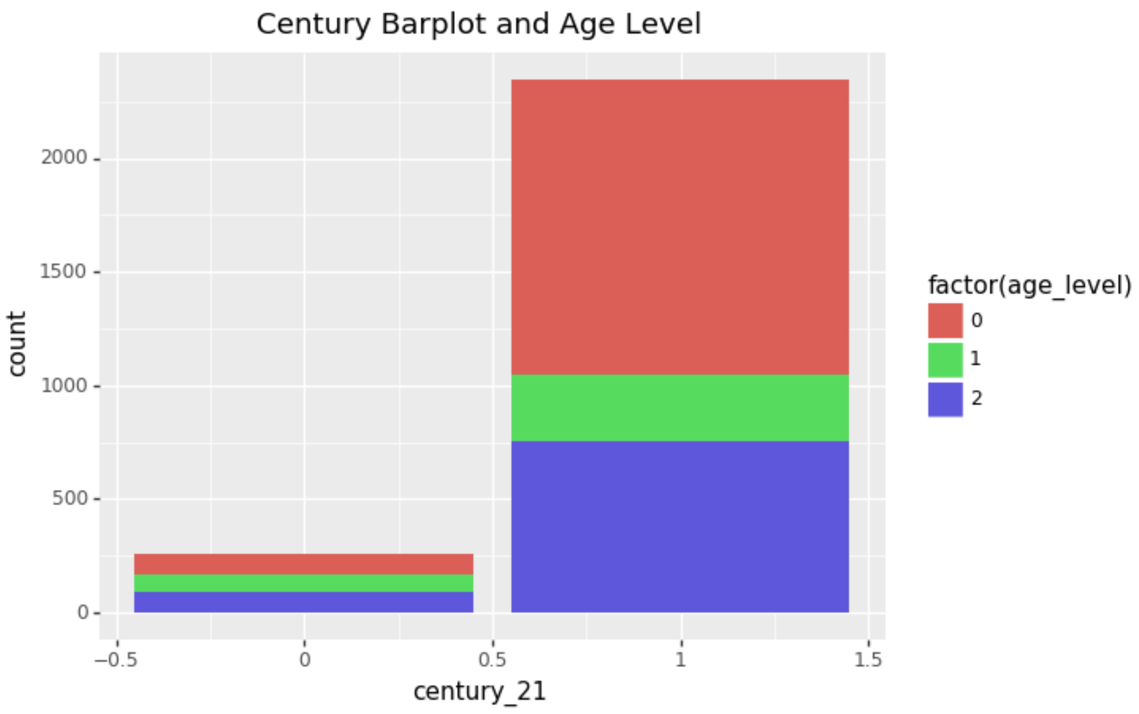

35
12
18
39
0
1
0
1
True label
Predicted label
35
30
25
20
15



Data Visualization

In [11]:

```
(ggplot(net, aes("century_21", fill = "factor(age_level)"))
 + geom_bar() +
labs(title = "Century Barplot and Age Level"))
```

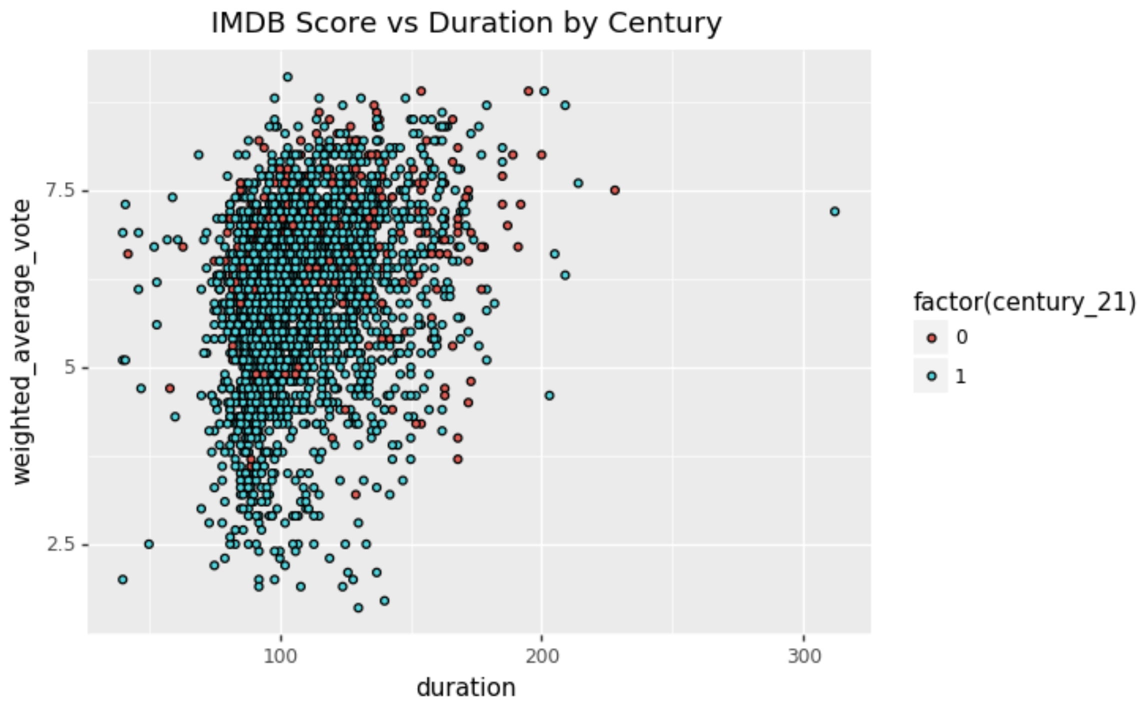


Out[11]:

```
<ggplot: (303856457)>
```

In [12]:

```
(ggplot(net, aes(x = "duration", y = "weighted_average_vote", fi
ll = "factor(century_21)"))
 + geom_point() +
labs(title = "IMDB Score vs Duration by Century"))
```



Out[12]:

```
<ggplot: (303865997)>
```

4. How well can we predict the IMDB score of a film?

Linear & Ridge Regression Model with K Fold

In [76]:

```
feat = ["genre", "director", "country", "release_year", "rating"
, "duration", "century_21", "age_level"]
cont_feat = ["rating","duration"]
X = net[feat]
y = net["weighted_average_vote"]

zscore = StandardScaler()
zscore.fit(X[cont_feat])
X[cont_feat] = zscore.transform(X[cont_feat])
```

Model Performance

Clearly, a linear model isn't the best way to predict IMDB score, especially with this many categorical variables. We can't do a great job of predicting IMDB score. Our model performed poorly with an R2 of only 0.13 and a mean error of 0.93. The best alpha was 10, so a high penalization works best for this model, but even that can't improve our accuracy much since this really just isn't the right method to predict IMDB score.

In [77]:

```
rr_tune_list = RidgeCV(cv = 5, alphas = [0.001,0.01,0.1,1,10]).f
it(X_train,y_train)

print("TRAIN: ", mean_absolute_error(y_train, rr_tune_list.predi
ct(X_train)))
print("TEST: ", mean_absolute_error(y_test, rr_tune_list.predict
(X_test)))
print("R2:", rr_tune_list.score(X_test,y_test))
print("BEST ALPHA:", rr_tune_list.alpha_)
```

```
TRAIN:  0.8948886536915989
TEST:  0.9259754056705626
R2: 0.12511319825919076
BEST ALPHA: 10.0
```

Predicted vs True

Our model was best at predicting IMDB scores between 5.5 and 6.5, as you can see in the predicted vs true plot that's the place where we're closest to the accuracy line. Our model was also very tentative, especially with predicting low scores. The minimum IMDB score prediction was 4.7, and the actual minimum value was 1.6. It was still a bit tentative with predicting high values, the max prediction was 8.8 and the actual max was 9.1, but this wasn't as pronounced.

In [78]:

```
kf = KFold(5)
y_pred = cross_val_predict(rr_tune_list, X, y, cv = kf)
true_vs_pred = pd.DataFrame({"predict": y_pred,"trueV": y})
true_vs_pred.head()
```

Out[78]:

	predict	trueV
0	5.525877	3.8
1	5.793415	5.2
2	5.474540	4.8
3	5.688059	6.3
4	5.692803	5.8

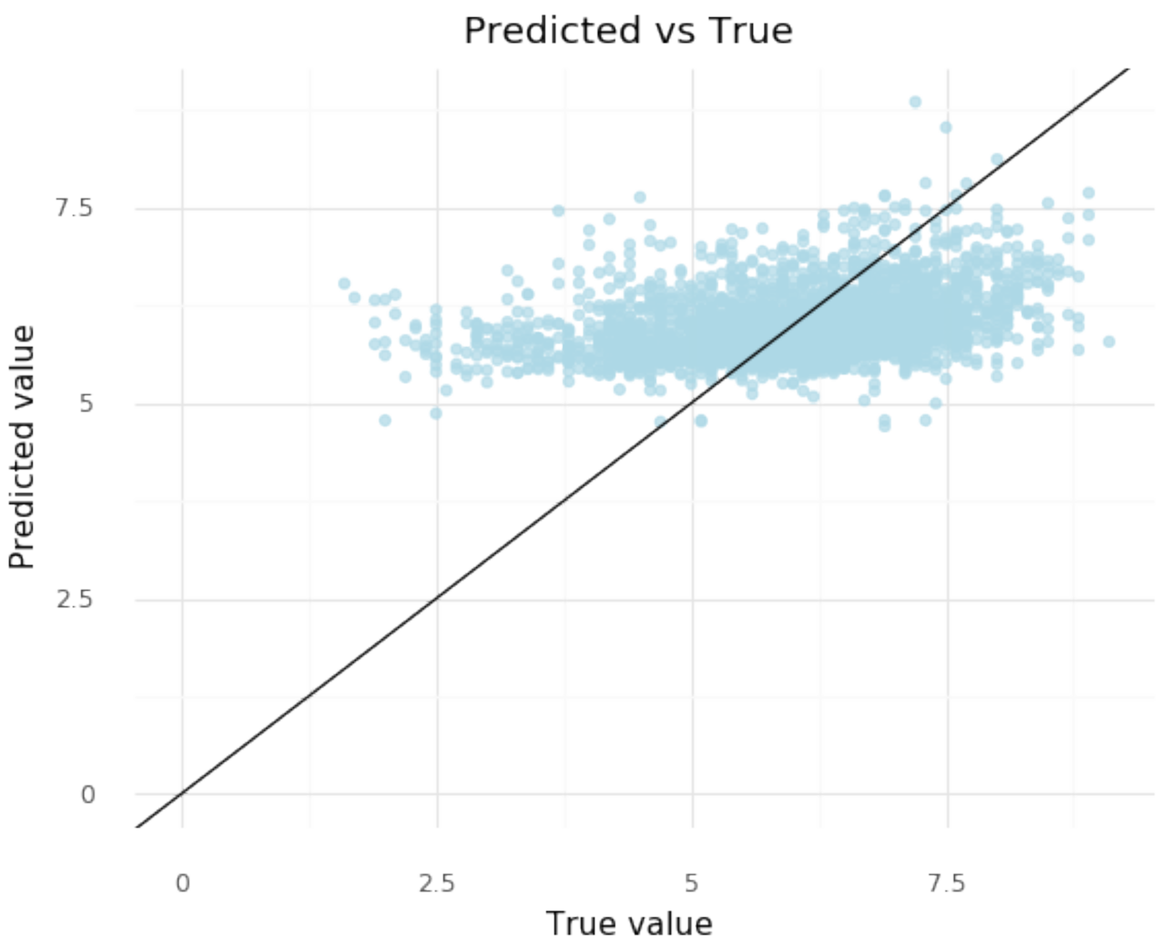
In [79]:

```
print(true_vs_pred.predict.min())
print(true_vs_pred.predict.max())
print(true_vs_pred.trueV.min())
print(true_vs_pred.trueV.max())
```

```
4.704789679831627
8.857508324666838
1.6
9.1
```

In [80]:

```
predicted_plot = (ggplot(true_vs_pred, aes(x = "trueV", y = "pre
dict")) +
 geom_point(color = "lightblue",alpha = 0.7) +
 geom_abline() +
    labs(title = "Predicted vs True",
    x = "True value",
    y = "Predicted value") +
theme_minimal())
predicted_plot = predicted_plot + expand_limits(x = 0, y = 0)
predicted_plot
```



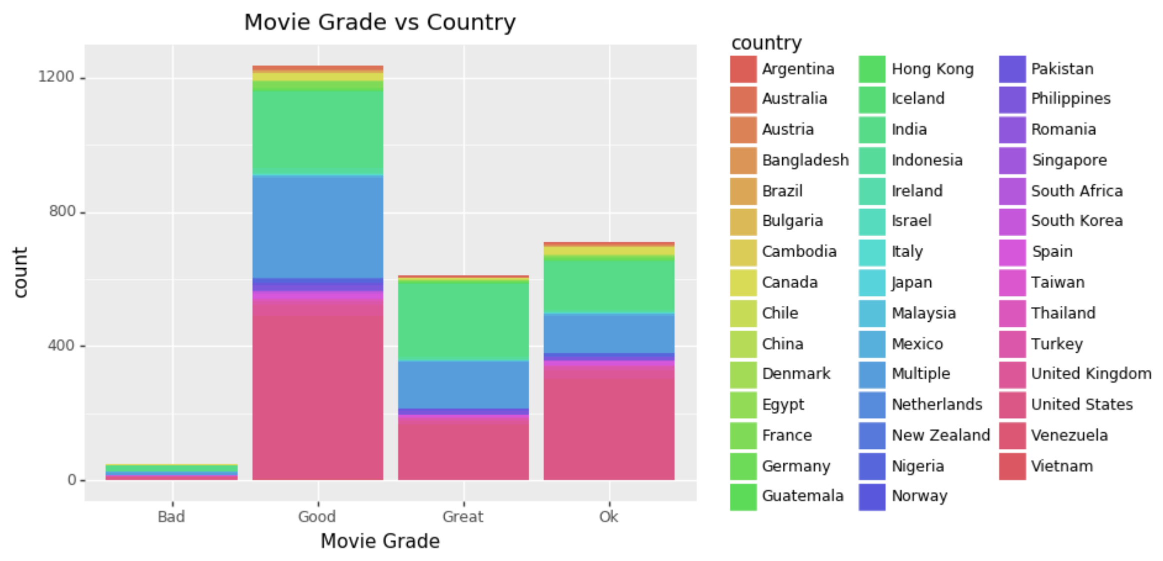
Out[80]:

```
<ggplot: (306517881)>
```

Data Visualization

In [54]:

```
(ggplot(netflix, aes("Score_group", fill = "country" ))
 + geom_bar()
 + xlab("Movie Grade")
 + ggtitle("Movie Grade vs Country"))
```

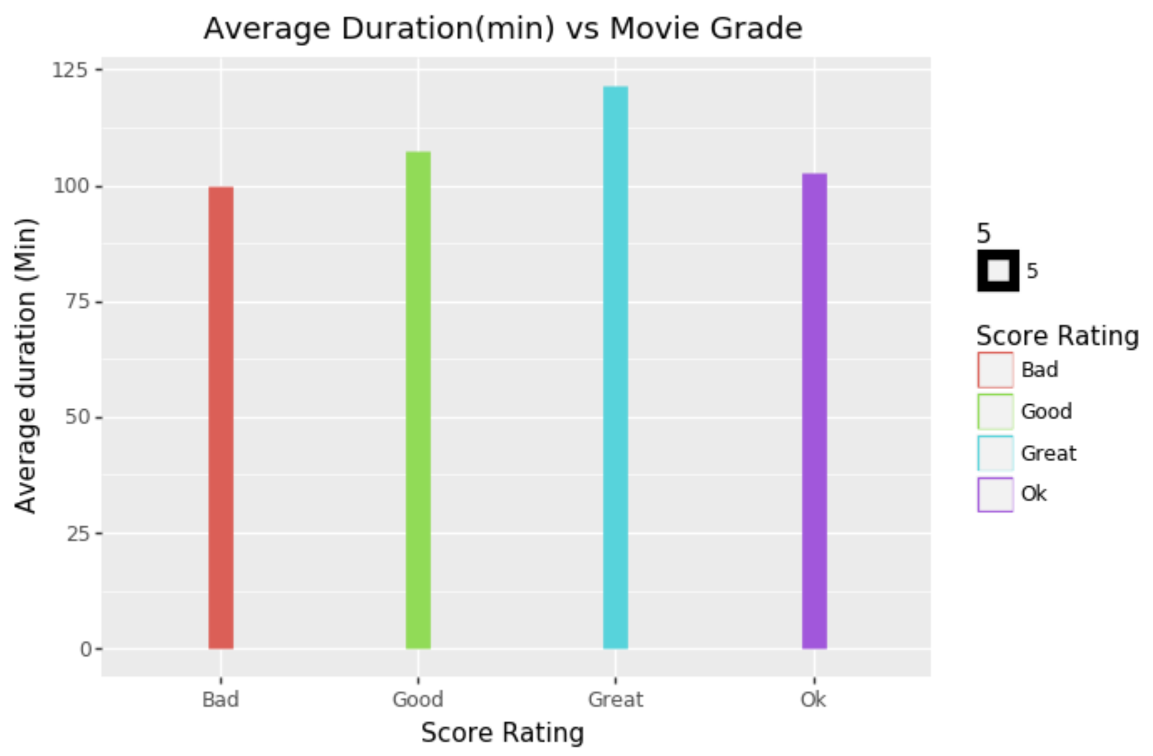


Out[54]:

```
<ggplot: (310571301)>
```

In [55]:

```
(ggplot(scoreRating_mean_votes, aes("Score Rating", "Duration",c
olor = "Score Rating"))
 + geom_density(aes(size = 5))
 + ylab("Average duration (Min)")
 + ggtitle("Average Duration(min) vs Movie Grade"))
```

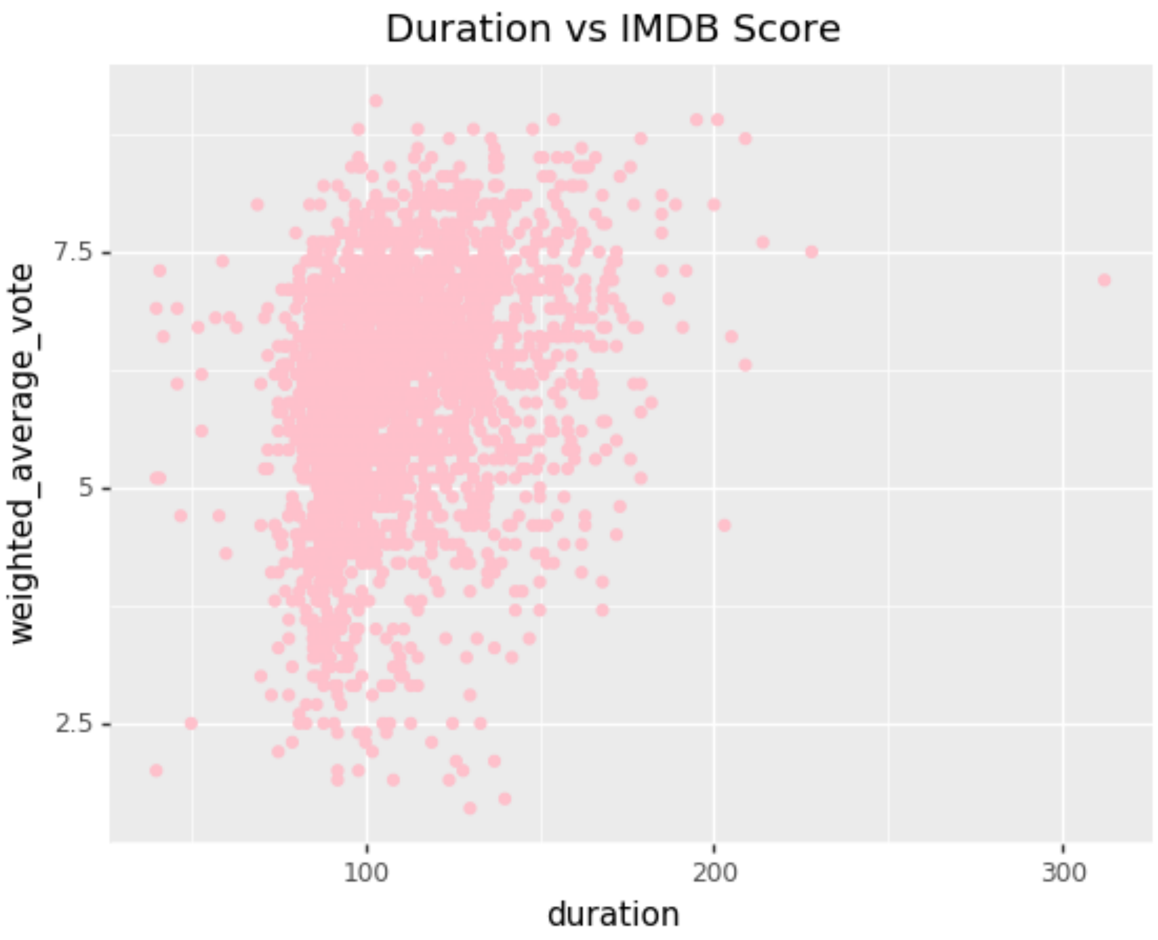


Out[55]:

```
<ggplot: (309265005)>
```

In [28]:

```
(ggplot(net, aes(x = "duration", y = "weighted_average_vote")) +
geom_point(color = "pink") +
labs(title = "Duration vs IMDB Score"))
```



Out[28]:

```
<ggplot: (306253533)>
```

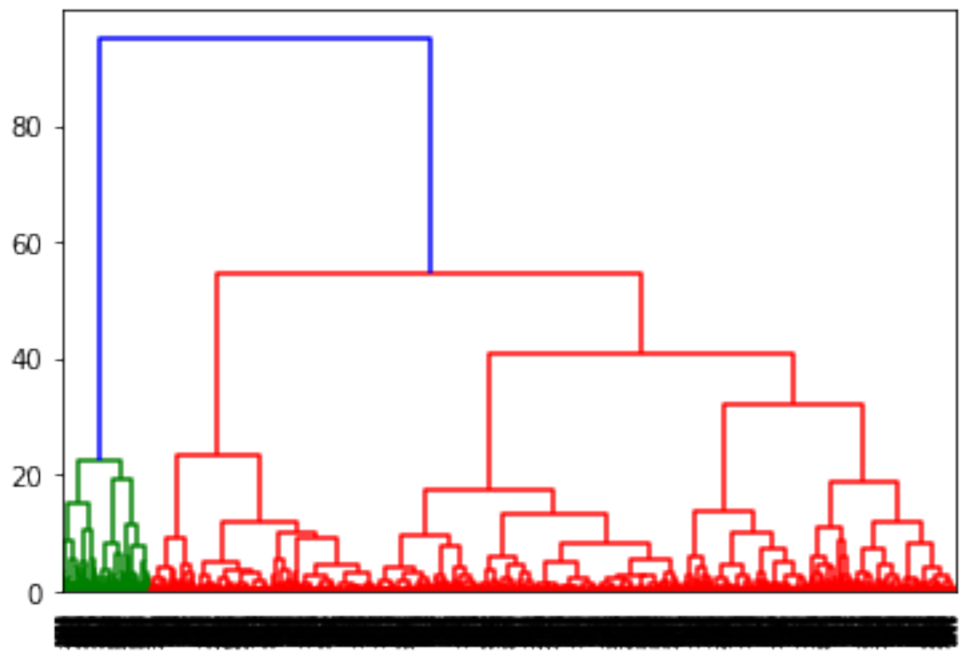
5. What content is similar?

Heirarchical Agglomerative Clustering

Our model clustered by year: release years before 2000 and release years after. Our data visualizations below show that the model didn't really consider any other variables when clustering.

In [16]:

```
features = ["weighted_average_vote", "release_year", "duration",
"century_21"]
X = net[features]
z = StandardScaler()
X[features] = z.fit_transform(X)

hac = AgglomerativeClustering(affinity = "euclidean",
                              linkage = "ward")
hac.fit(X)
dendro = sch.dendrogram(sch.linkage(X, method='ward'))
```



In [17]:

```
n_components = [2,3,4,5,6]
sils = []
for n in n_components:
    hac = AgglomerativeClustering(n_clusters = n,
                                  affinity = "euclidean",
                                 linkage = "ward")
    hac.fit(X)
    m = hac.labels_
    sils.append(silhouette_score(X, m))

print(sils)
```

```
[0.6202046171846157, 0.29942123792795217, 0.27229266
563387117, 0.2903019598523148З, 0.28027353674736477]
```

In [18]:

```
hac = AgglomerativeClustering(n_clusters =2,
                              affinity = "euclidean",
                             linkage = "ward")
hac.fit(X)
```

Out[18]:

```
AgglomerativeClustering(affinity='euclidean', comput
e_full_tree='auto',
                        connectivity=None, distance_
threshold=None,
                        linkage='ward', memory=None,
n_clusters=2)
```

In [19]:

```
membership = hac.labels_
```

In [20]:

```
silhouette_score(X,membership)
```

Out[20]:

```
0.6202046171846157
```

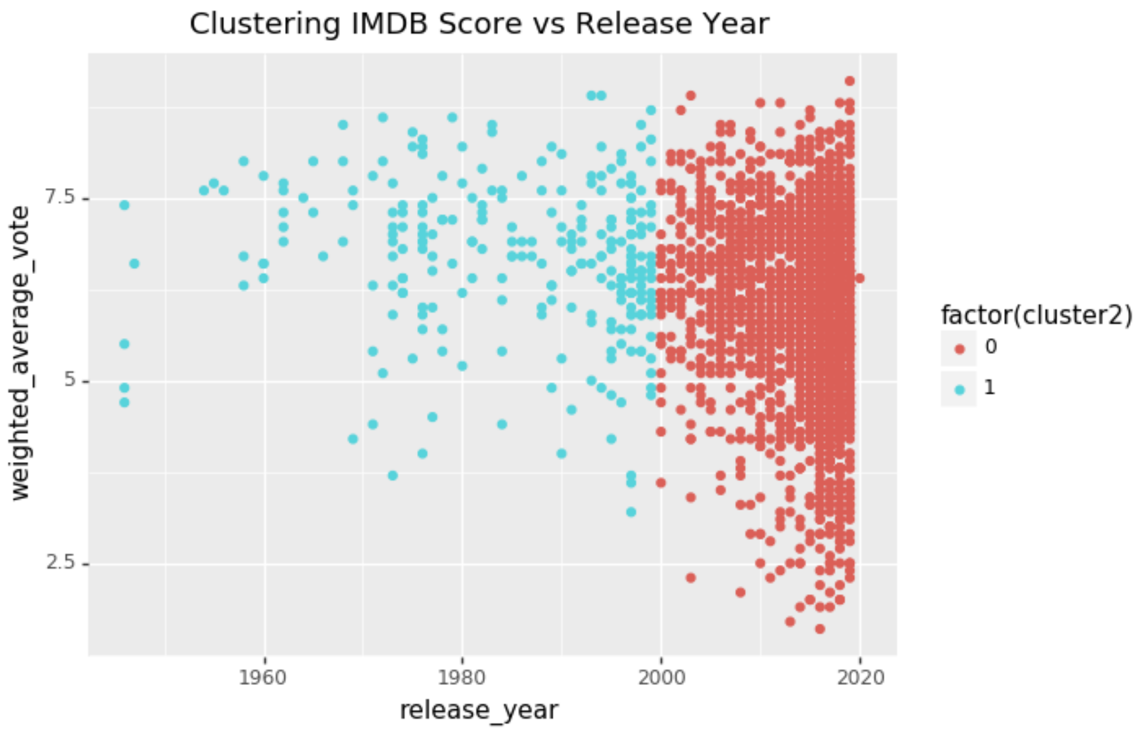
In [21]:

```
net["cluster2"] = membership
```

Data Visualization

In [22]:

```
(ggplot(net, aes(x = "release_year", y = "weighted_average_vote"
)) +
 geom_point(aes(color = "factor(cluster2)")) +
labs(title = "Clustering IMDB Score vs Release Year"))
```

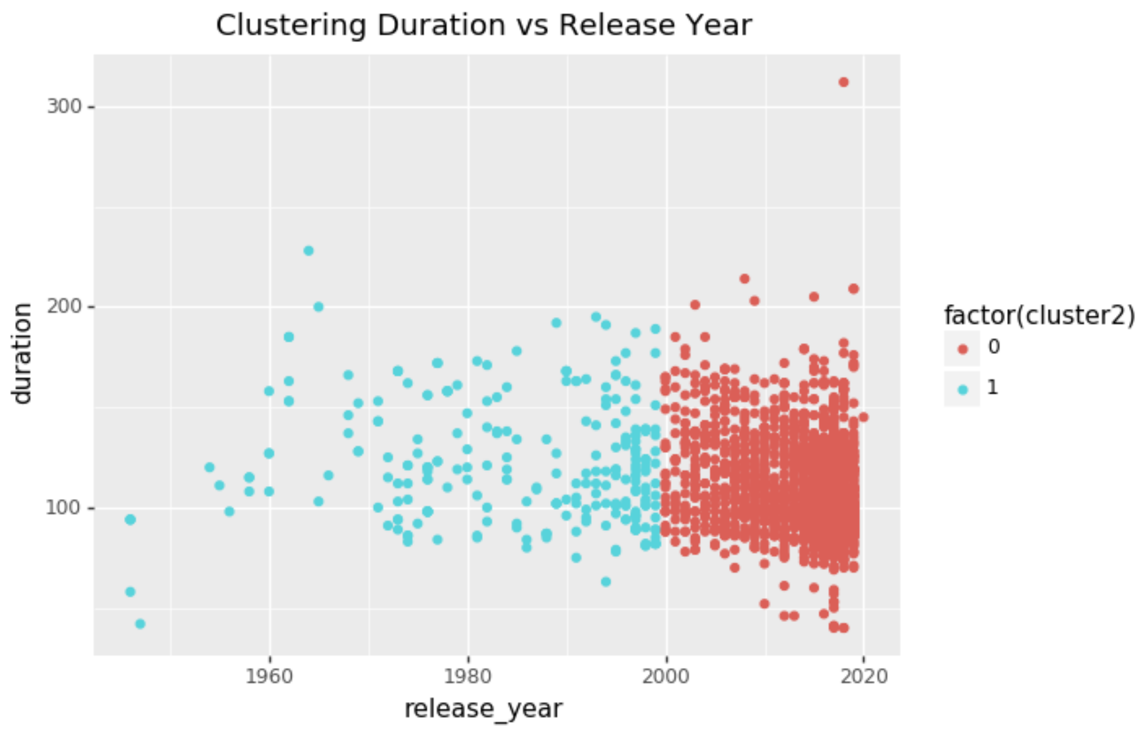


Out[22]:

```
<ggplot: (305935745)>
```

In [24]:

```
(ggplot(net, aes(x = "release_year", y = "duration")) + geom_poi
nt(aes(color = "factor(cluster2)")) +
labs(title = "Clustering Duration vs Release Year"))
```



Out[24]:

```
<ggplot: (303811265)>
```

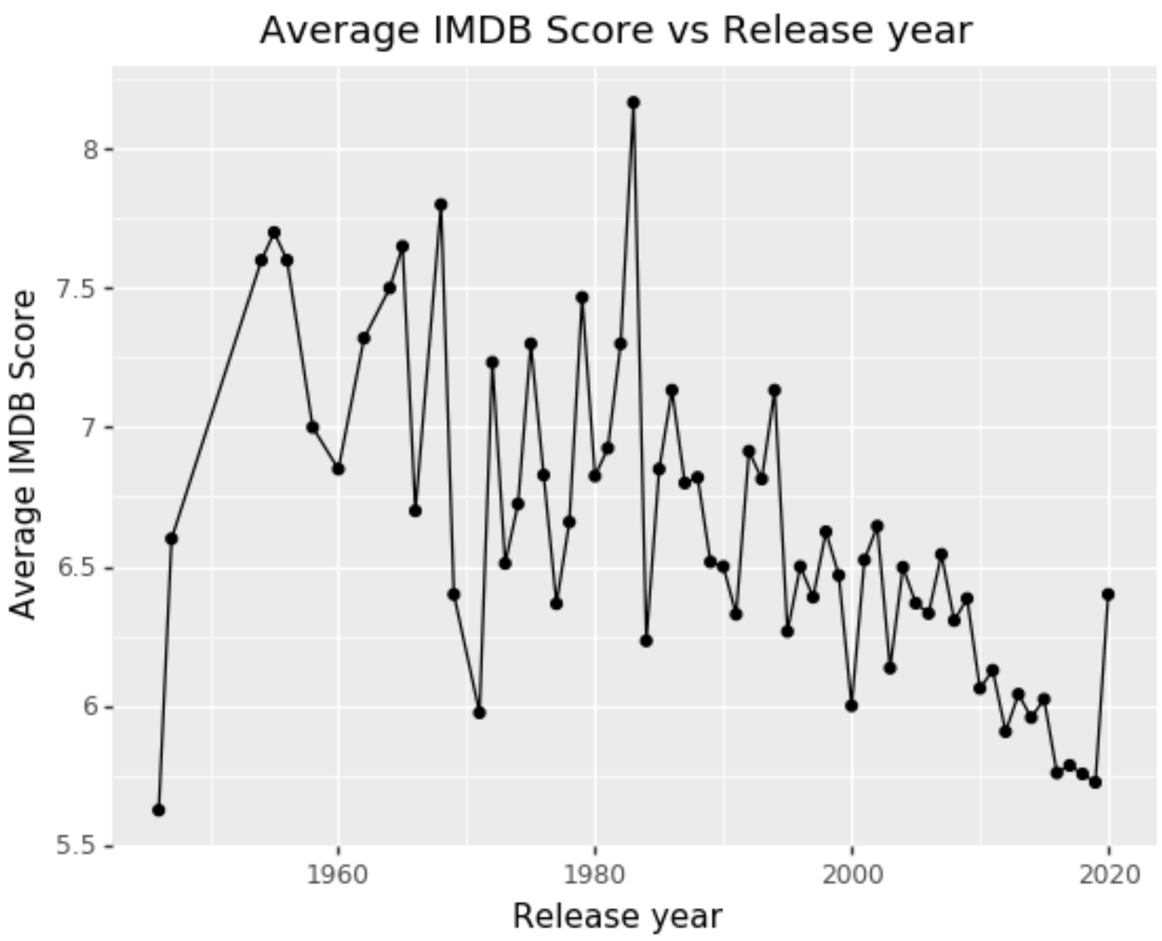
6. What release years had the highest average IMDB rating?

Data Visualization

Interestingly, IMDB score has decreased over time for the most part, especially in recent years. Either movies are getting worse, or critics are being tougher when giving out ratings. We see a random huge spike of IMDB score in the early 80s, and another spike in 2020. This is good news-- maybe movies are getting better moving forward and the average score will start increasing.

In [40]:

```
(ggplot(year_mean_votes, aes("Release year","Mean"))
 + geom_point(aes())
 + geom_line()
 + ylab("Average IMDB Score")
 + ggtitle("Average IMDB Score vs Release year"))
```

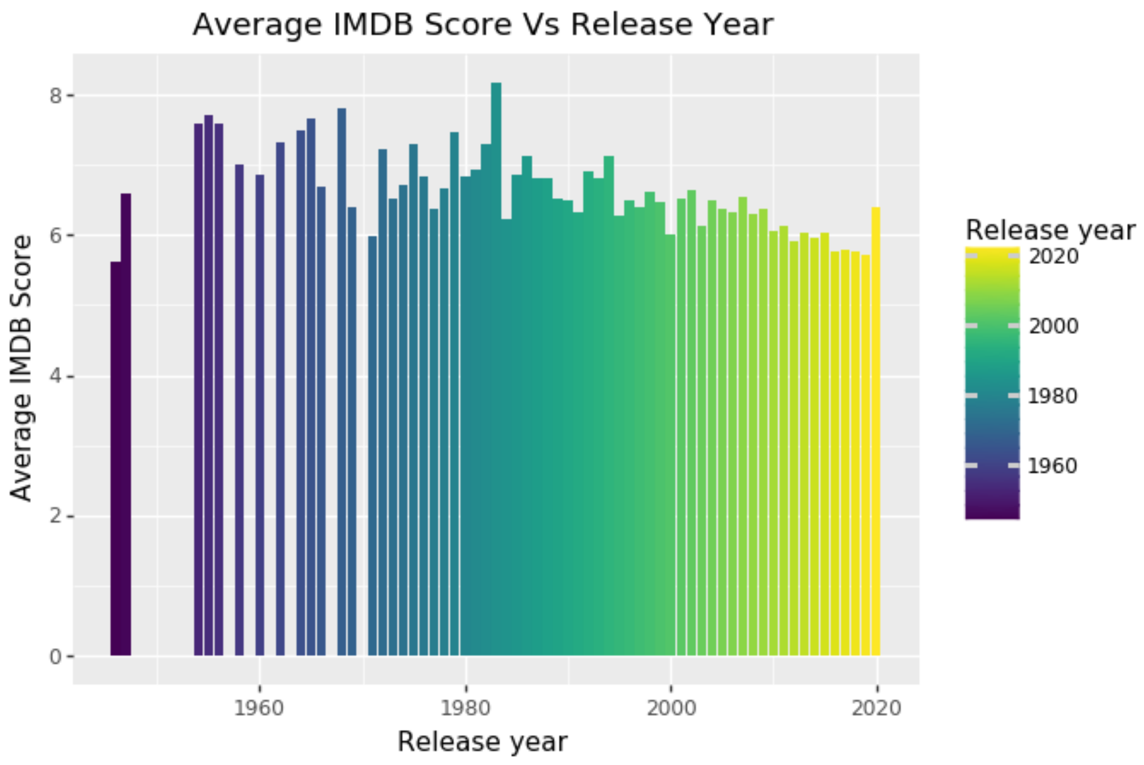


Out[40]:

```
<ggplot: (306987581)>
```

In [41]:

```
(ggplot(year_mean_votes, aes("Release year", "Mean", fill = "Rel
ease year"))
 + geom_bar(stat = "identity")
 + ggtitle("Average IMDB Score Vs Release Year")
 + ylab("Average IMDB Score"))
```



Out[41]:

```
<ggplot: (307717969)>
```