In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.preprocessing import StandardScaler
from IPython.display import display, HTML
import re
```

In [2]:
```
data = pd.read_excel('moviadata.xlsx')
df = data.copy()
```

In [3]:
```
def getting_primary_info(df):
    print("-----------------------------------------------------------")
    print("Veri setinin şekli", df.shape)
    print("-----------------------------------------------------------")
    print("Veri seti değişken tipleri:\n", df.dtypes)
    print("-----------------------------------------------------------")
    # print("Veri setinin ilk 5 satırı")
    # display(HTML(df.head().to_html()))
    print("-----------------------------------------------------------")
    print("Veri setinin istatistiki verileri")
    description = df.describe()
    display(HTML(description.to_html()))
    print("-----------------------------------------------------------")
getting_primary_info(df)
```

```
-----------------------------------------------------------
Veri setinin şekli (148, 35)
-----------------------------------------------------------
Veri seti değişken tipleri:
rate            float64
red_hist         object
green_hist       object
blue_hist        object
std_pow         float64
max_pow         float64
min_pow         float64
mean_pow        float64
max_pow_freq      int64
max_pow_time    float64
band5           float64
band10          float64
band15          float64
band20          float64
band25          float64
band30          float64
band35          float64
band40          float64
band45          float64
band50          float64
band55          float64
band60          float64
band65          float64
band70          float64
band75          float64
band80          float64
band85          float64
band90          float64
band95          float64
band100         float64
band105         float64
band110         float64
band115         float64
band120         float64
band_last       float64
dtype: object
-----------------------------------------------------------
-----------------------------------------------------------
Veri setinin istatistiki verileri
```

	rate	std_pow	max_pow	min_pow	mean_pow	max_pow_freq	max_pow_time	band5	band10	band15	band20	band25	band30	band35	band40	
count	148.000000	1.480000e+02	1.480000e+02	148.0	1.480000e+02	1.480000e+02	1.480000e+02	1.480000e+02	1.480000e+02	1.480000e+02	1.480000e+02	1.480000e+02	1.480000e+02	1.480000e+02	1.480000e+02	1.48(
mean	5.331081	1.896213e+10	1.085332e+10	0.0	1.482124e+09	1.791562e+08	1.350805e+16	3.483230e+10	7.125239e+09	1.759782e+09	1.117690e+09	5.539606e+08	3.303728e+08	2.721451e+08	1.867846e+08	1.625
std	2.428203	3.863148e+10	5.935229e+10	0.0	3.361725e+09	1.257793e+08	1.366687e+16	4.975303e+10	8.534472e+09	4.114163e+09	1.745019e+09	7.698141e+08	6.071195e+08	4.752725e+08	3.661104e+08	3.02
min	1.200000	4.185129e+07	1.107235e-04	0.0	2.773301e+06	6.890625e+06	1.248073e-01	1.229730e+08	1.385863e+07	3.948682e+05	1.518932e+06	2.033913e+05	9.450773e+03	6.021471e+05	7.852623e+04	1.65
25%	3.200000	3.619859e+09	4.860344e-04	0.0	2.402728e+08	1.722656e+08	5.776036e+15	3.688364e+09	7.455356e+08	3.013516e+08	9.011356e+07	4.959274e+07	4.347031e+07	2.499896e+07	1.397051e+07	1.44
50%	5.000000	7.427953e+09	8.420181e-04	0.0	5.468599e+08	1.722656e+08	9.284354e+15	8.728091e+09	2.161204e+09	5.685484e+08	3.885267e+08	1.327766e+08	7.710597e+07	6.723851e+07	5.376671e+07	7.35
75%	7.400000	1.697167e+10	1.415636e-03	0.0	1.537877e+09	1.722656e+08	1.426302e+16	4.116775e+10	1.387036e+10	1.914108e+09	1.558572e+09	1.078215e+09	2.696110e+08	2.591653e+08	1.914128e+08	1.91
max	9.300000	3.588296e+11	5.126301e+11	0.0	3.224890e+10	8.613281e+08	5.984290e+16	2.135143e+11	3.323586e+10	3.379004e+10	1.417217e+10	4.386240e+09	3.285582e+09	2.217122e+09	2.026925e+09	2.16

```
-----------------------------------------------------------
```

In [ ]:
```
red_hist_values = df["red_hist"].apply(lambda x: x.strip("[]").split())

red_hist_values = red_hist_values.apply(lambda x: [float(value) for value in x])

for i in range(1, 256):
    column_name = f"red_{i}"
    df[column_name] = red_hist_values.apply(lambda x: x[i-1])

green_hist_values = df["green_hist"].apply(lambda x: x.strip("[]").split())

green_hist_values = green_hist_values.apply(lambda x: [float(value) for value in x])

for i in range(1, 256):
    column_name = f"green{i}"
    df[column_name] = green_hist_values.apply(lambda x: x[i-1])
```

In [5]:
```
df.head()
```

Out[5]:

	rate	red_hist	green_hist	blue_hist	std_pow	max_pow	min_pow	mean_pow	max_pow_freq	max_pow_time	...	green246	green247	green248	green249	green250	green251	green252	green
0	4.7	[2.47115556e+04 1.61302778e+03 7.60157407e+02 ...	[2.38848611e+04 1.16400000e+03 6.37972222e+02 ...	[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. ...	3.667384e+10	0.000816	0.0	2.388563e+08	34453125	4.856163e+16	...	12.287037	14.129630	13.277778	15.629630	19.694444	22.555556	25.888889	39.925
1	7.2	[2.78458435e+04 3.61925217e+03 2.93840000e+03 ...	[2.85503565e+04 1.57850435e+03 1.58126087e+03 ...	[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. ...	7.852526e+09	0.002450	0.0	4.155288e+09	172265625	1.552834e-01	...	11.791304	12.704348	15.139130	14.217391	11.600000	10.121739	11.173913	11.486
2	7.8	[2.33913418e+04 1.36652532e+03 6.38291139e+02 ...	[2.29676013e+04 4.22291139e+02 2.18778481e+02 ...	[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. ...	4.878064e+09	0.000683	0.0	5.179080e+08	172265625	9.699846e+15	...	26.272152	22.278481	21.677215	27.955696	127.848101	188.829114	63.487342	43.753
3	7.1	[20288.05479452 1980.80821918 552.99315068 ...	[2.05630137e+04 1.79503425e+03 4.71335616e+02 ...	[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. ...	3.202989e+10	0.000501	0.0	2.966353e+09	172265625	7.513687e+15	...	32.253425	22.006849	18.609589	25.205479	26.410959	35.527397	39.082192	25.643
4	7.5	[2.29062034e+04 1.23541808e+03 8.31406780e+02 ...	[2.50710847e+04 7.03593220e+02 5.47485876e+02 ...	[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. ...	3.715484e+10	0.001757	0.0	2.786652e+08	172265625	1.173261e+16	...	9.011299	5.672316	4.830508	5.016949	4.819209	4.553672	5.005650	4.073

5 rows × 545 columns

In [6]:
```
df = df.drop(columns = ["red_hist"      ,"green_hist"   ,"blue_hist"])
df.to_csv("movie_prep.csv",index = False)
```

In [8]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

df = df.dropna()
X, y = df.drop(columns=["rate"]), df["rate"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

# Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("Linear Regression Test R^2 Score:", r2)

y_pred_train = model.predict(X_train)
r2_train = r2_score(y_train, y_pred_train)
print("Linear Regression Train R^2 Score:", r2_train)

print("-----------------------")

# Random Forest
model_random_forest = RandomForestRegressor(n_estimators=100, min_samples_split=2)
model_random_forest.fit(X_train, y_train)

y_pred_rf = model_random_forest.predict(X_test)
r2_rf = r2_score(y_test, y_pred_rf)
print("Random Forest Test R^2 Score:", r2_rf)

y_pred_rf_train = model_random_forest.predict(X_train)
r2_rf_train = r2_score(y_train, y_pred_rf_train)
print("Random Forest Train R^2 Score:", r2_rf_train)

print("-----------------------")

# Gradient Boosting
model_gradient_boosting = GradientBoostingRegressor(n_estimators=80, min_samples_split=2)
model_gradient_boosting.fit(X_train, y_train)

y_pred_gb = model_gradient_boosting.predict(X_test)
r2_gb = r2_score(y_test, y_pred_gb)
print("Gradient Boosting Test R^2 Score:", r2_gb)

y_pred_gb_train = model_gradient_boosting.predict(X_train)
r2_gb_train = r2_score(y_train, y_pred_gb_train)
print("Gradient Boosting Train R^2 Score:", r2_gb_train)
```

```
Linear Regression Test R^2 Score: -3.5533542253133694
Linear Regression Train R^2 Score: 0.8303544931415981
-----------------------
Random Forest Test R^2 Score: 0.17121764692513597
Random Forest Train R^2 Score: 0.843318162555873 8
-----------------------
Gradient Boosting Test R^2 Score: 0.0179295133068732
Gradient Boosting Train R^2 Score: 0.9938510020181357
```

In [9]:
```
from catboost import CatBoostRegressor
from sklearn.metrics import r2_score

model = CatBoostRegressor(iterations=2000,
                          learning_rate=0.1,
                          depth=6)
model.fit(X_train, y_train, eval_set=(X_test, y_test), early_stopping_rounds=50, verbose=100)

y_pred_train = model.predict(X_train)
r2_train = r2_score(y_train, y_pred_train)
print("CatBoost Train R-squared:", r2_train)

y_pred_test = model.predict(X_test)
r2_test = r2_score(y_test, y_pred_test)
print("CatBoost Test R-squared:", r2_test)
```

```
0:      learn: 2.3250746        test: 2.5279296 best: 2.5279296 (0)     total: 177ms    remaining: 5m 54s
100:    learn: 0.2758103        test: 2.4130934 best: 2.3876818 (64)    total: 2.08s    remaining: 39.1s
Stopped by overfitting detector  (50 iterations wait)

bestTest = 2.387681804
bestIteration = 64

Shrink model to first 65 iterations.
```

```
CatBoost Train R-squared: 0.9044454068306678
CatBoost Test R-squared: 0.10299979751197219
```

In [ ]: