```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as ssn
from sklearn.model_selection import train_test_split
from sklearn.linear model import LinearRegression
from sklearn import metrics
from sklearn.svm import SVR
from sklearn import tree
movies=pd.read_csv('http://bit.ly/imdbratings')
movies.head()
         star_rating
                                          title content_rating genre duration
                                                                                                                   actors_list
                  9.3 The Shawshank Redemption
                                                                  Crime
                                                                               142 [u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...
      1
                  9.2
                                   The Godfather
                                                               R Crime
                                                                               175
                                                                                       [u'Marlon Brando', u'Al Pacino', u'James Caan']
      2
                             The Godfather: Part II
                                                                               200
                  9.1
                                                               R Crime
                                                                                       [u'Al Pacino', u'Robert De Niro', u'Robert Duv...
      3
                  9.0
                                 The Dark Knight
                                                           PG-13 Action
                                                                               152
                                                                                       [u'Christian Bale', u'Heath Ledger', u'Aaron E...
      4
                                     Pulp Fiction
                                                               R Crime
                                                                                     [u'John Travolta', u'Uma Thurman', u'Samuel L....
                  89
                                                                               154
movies.columns
     dtype='object')
movies.isnull().sum()
     star_rating
     title
                        0
     content_rating
     genre
     duration
                        0
     actors_list
                        0
     dtype: int64
content_rating_null_values=list(movies.content_rating.isnull())
for i in range(len(content_rating_null_values)):
  if content_rating_null_values[i]==True:
    print(i)
     187
     649
     936
movies.iloc[187,2]='PG13'
movies.iloc[649,2]='PG'
movies.iloc[936,2]='PG13'
movies.drop(['title'],axis=1,inplace=True)
movies.drop(['actors_list'],axis=1, inplace=True)
categorical_features=[i for i in movies.select_dtypes(include=np.object)]
     <ipython-input-9-6cbec47f27d9>:1: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this wa
     Deprecated in NumPy 1.20; for more details and guidance: <a href="https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations">https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations</a>
       categorical\_features = [i \ for \ i \ in \ movies.select\_dtypes(include = np.object)]
dummy_df=pd.DataFrame()
dummy_df['duration']=movies.duration
for feature in categorical_features:
  df=pd.get_dummies(movies[feature])
train_df=pd.concat([df,dummy_df],axis=1)
```

Action Adventure Animation Biography Comedy Crime Drama Family Fantasy Film-Noir History Horror Mystery Sci-Fi Thrill

train_df.head()

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0	A	ction Auve	ilitale AllIII	at1011 b1	.ogi apiiy	conleady	CITILE	Di allia	гаштту	rantasy	LITIII-MOTI	IIIS COI y	1101 1 01	nyster y	301-11	1111 11
2 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
3 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
### ### ##############################	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
rain_df-pd.concat([train_df,movies['star_rating']],axis=1) rain_df.shape (379, 18) =train_df.drop(['star_rating'],axis=1) =train_df.drop(['star_rating']) =train_xtest.y_train.y_test-train_test_split(x,y,test_size=0.2,random_state=42) RetinearRegression() =tinearRegression() =tinearRegression()	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
rain_df.shape (979, 18) tetain_df.dron(['star_rating'],axis=1) tetain_df.dron(['star_rating'],axis=1) _train_X_test,y_train_y_test=train_test_split(x,y,test_size=0.2,random_state=42) R-tincarRegression() R-fit(X_train_y_train)	4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
<pre>ctrain_df.dron(['star_rating'],axis=1)</pre>	rain_df=p	d.concat([t	crain_df,mo	/ies['sta	r_rating']],axis	=1)									
-train_df.drop(['star_rating'],axis=1) -train_df.'star_rating'] _train_X_test,y_train_y_test-train_test_split(x,y,test_size=0.2,random_state=42) RetinearRegression() Refit(X_train_y_train) - tinearRegression tinearRegression()	rain_df.s	hape														
<pre>_train_df['star_rating'] _train_x_test_y_train_y_test_train_test_split(x,y,test_size=0.2,random_state=42) RelinearRegression() R.fit(X_train,y_train)</pre>	(979,	18)														
ReLinearRegression() R. Fit(X_train,y_train) * LinearRegression LinearRegression() pred=LR.predict(X_test) rint('RMSE using Linear regression is',metrics.mean_squared_error(y_test,y_pred,sample_weight=None)) RMSE using Linear regression is 0.0963980880321459 V=SVR() v. Fit(X_train,y_train) * SVR SVR() v_pred=sv.predict(X_test) rint('RMSE using SVR is',metrics.mean_squared_log_error(y_test,sv_pred,sample_weight=None)) RMSE using SVR is 0.001221107353436723 1f=tree.DecisionTreeRegressor() 1f.fit(X_train,y_train) * DecisionTreeRegressor()				,axis=1)												
R.fit(X_train,y_train) * LinearRegression LinearRegression LinearRegression LinearRegression LinearRegression LinearRegression LinearRegression LinearRegression LinearRegression Linear regression is ',metrics.mean_squared_error(y_test,y_pred,sample_weight=None)) RMSE using Linear regression is 0.09639808808321459 v=SVR() v.fit(X_train,y_train) * SVR SVR() V_pred=sv.predict(X_test) rint('RMSE using SVR is',metrics.mean_squared_log_error(y_test,sv_pred,sample_weight=None)) RMSE using SVR is 0.001221107353436723 1f=tree.DecisionTreeRegressor() 1f.fit(X_train,y_train) * DecisionTreeRegressor DecisionTreeRegressor)	_train,X_	test,y_trai	in,y_test=t	rain_test	_split(x,	y,test_s	size=0.	2,rando	om_state:	=42)						
LinearRegression LinearRegression() _pred=LR.predict(X_test) rint('RMSE using Linear regression is',metrics.mean_squared_error(y_test,y_pred,sample_weight=None)) RMSE using Linear regression is 0.0963980880321459 v=SVR() v-SVR() v.fit(X_train,y_train)	R=LinearR	egression())													
LinearRegression() _pred=LR.predict(X_test) rint('RMSE using Linear regression is',metrics.mean_squared_error(y_test,y_pred,sample_weight=None)) RMSE using Linear regression is 0.0963980880321459 v=SVR() v.fit(X_train,y_train)	R.fit(X_t	rain,y_trai	in)													
rint('RMSE using Linear regression is',metrics.mean_squared_error(y_test,y_pred,sample_weight=None)) RMSE using Linear regression is 0.0963980880321459 v=SVR() v.fit(X_train,y_train) v_SVR SVR() v_pred=sv.predict(X_test) rint('RMSE using SVR is',metrics.mean_squared_log_error(y_test,sv_pred,sample_weight=None)) RMSE using SVR is 0.001221107353436723 If=tree.DecisionTreeRegressor() 1f.fit(X_train,y_train) v_DecisionTreeRegressor()																
RMSE using Linear regression is 0.0963980880321459 v-SVR() v.fit(X_train,y_train) * SVR SVR() v_pred=sv.predict(X_test) rint('RMSE using SVR is',metrics.mean_squared_log_error(y_test,sv_pred,sample_weight=None)) RMSE using SVR is 0.001221107353436723 If=tree.DecisionTreeRegressor() 1f.fit(X_train,y_train) * DecisionTreeRegressor()	_pred=LR.	predict(X_t	test)													
RMSE using Linear regression is 0.0963980880321459 v-SVR() v.fit(X_train,y_train) * SVR SVR() v_pred=sv.predict(X_test) rint('RMSE using SVR is',metrics.mean_squared_log_error(y_test,sv_pred,sample_weight=None)) RMSE using SVR is 0.001221107353436723 If=tree.DecisionTreeRegressor() 1f.fit(X_train,y_train) * DecisionTreeRegressor()	rint('RMS	E using Lir	near regress	sion is',	metrics.m	ean_squa	ared_er	ror(y_t	test,y_p	red,sampl	e_weight=No	one))				
<pre>v.fit(X_train,y_train) * SVR SVR() v_pred=sv.predict(X_test) rint('RMSE using SVR is',metrics.mean_squared_log_error(y_test,sv_pred,sample_weight=None)) RMSE using SVR is 0.001221107353436723 If=tree.DecisionTreeRegressor() 1f.fit(X_train,y_train) * DecisionTreeRegressor DecisionTreeRegressor()</pre>																
<pre>v SVR SVR() v_pred=sv.predict(X_test) rint('RMSE using SVR is',metrics.mean_squared_log_error(y_test,sv_pred,sample_weight=None)) RMSE using SVR is 0.001221107353436723 If=tree.DecisionTreeRegressor() If.fit(X_train,y_train)</pre>	v=SVR()															
<pre>v_pred=sv.predict(X_test) v_pred=sv.predict(X_test) rint('RMSE using SVR is',metrics.mean_squared_log_error(y_test,sv_pred,sample_weight=None)) RMSE using SVR is 0.001221107353436723 If=tree.DecisionTreeRegressor() If.fit(X_train,y_train)</pre>	v.fit(X_t	rain,y_trai	in)													
rint('RMSE using SVR is',metrics.mean_squared_log_error(y_test,sv_pred,sample_weight=None)) RMSE using SVR is 0.001221107353436723 If=tree.DecisionTreeRegressor() If.fit(X_train,y_train) * DecisionTreeRegressor DecisionTreeRegressor()																
RMSE using SVR is 0.001221107353436723 If=tree.DecisionTreeRegressor() If.fit(X_train,y_train) * DecisionTreeRegressor DecisionTreeRegressor()	v_pred=sv	.predict(X_	_test)													
<pre>lf=tree.DecisionTreeRegressor() lf.fit(X_train,y_train)</pre>	rint('RMS	E using SVF	R is',metri	cs.mean_s	quared_lo	g_error	(y_test	,sv_pre	ed,samplo	e_weight=	None))					
<pre>lf.fit(X_train,y_train) * DecisionTreeRegressor DecisionTreeRegressor()</pre>	RMSE	using SVR i	is 0.001221	107353436	723											
<pre>* DecisionTreeRegressor DecisionTreeRegressor()</pre>	lf=tree.D	ecisionTree	eRegressor()												
DecisionTreeRegressor()	lf.fit(X_	train,y_tra	ain)													
T_pred=clf.predict(X_test)																
	T_pred=cl	f.predict(>	(_test)													

RMSE using DT is 0.1824832199546485

 $\verb|print('RMSE using DT is', metrics.mean_squared_error(y_test, DT_pred, sample_weight=None))| \\$