```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
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
```

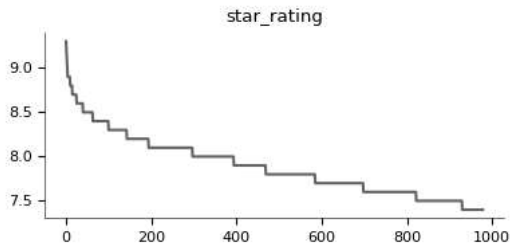
```
movies=pd.read_csv( '/content/imdbratings (1).csv')
```

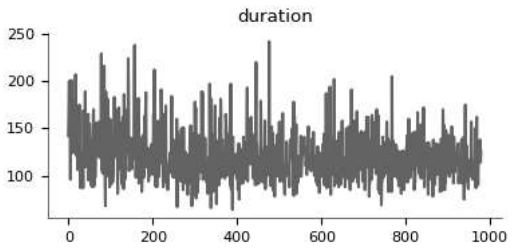
```
movies
```

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	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	9.1	The Godfather: Part II	R	Crime	200	[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	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....
...	...	...	...	...	...	...
974	7.4	Tootsie	PG	Comedy	116	[u'Dustin Hoffman', u'Jessica Lange', u'Teri G...
975	7.4	Back to the Future Part III	PG	Adventure	118	[u'Michael J. Fox', u'Christopher Lloyd', u'Ma...
976	7.4	Master and Commander: The Far Side of the World	PG-13	Action	138	[u'Russell Crowe', u'Paul Bettany', u'Billy Bo...
977	7.4	Poltergeist	PG	Horror	114	[u'JoBeth Williams', u"Heather O'Rourke", u'Cr...
978	7.4	Wall Street	R	Crime	126	[u'Charlie Sheen', u'Michael Douglas', u'Tamar...

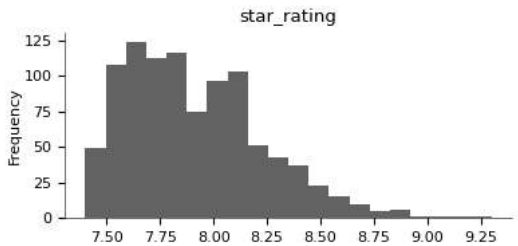
979 rows × 6 columns

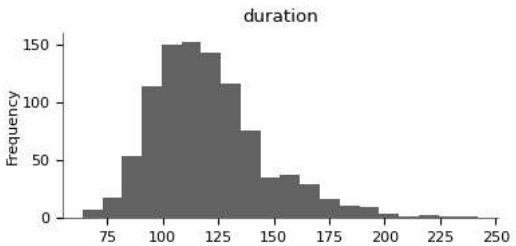
Values



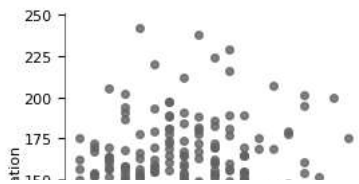


Distributions





2-d distributions



```
movies. head()
```

	star_rating	title	content_rating	genre	duration	actors_list
0	9.3	The Shawshank Redemption	R	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	9.1	The Godfather: Part II	R	Crime	200	[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	8.9	Pulp Fiction	R	Crime	154	[u'John Travolta', u'Uma Thurman', u'Samuel L....

```
movies.columns
```

```
Index(['star_rating', 'title', 'content_rating', 'genre', 'duration',
       'actors_list'],
      dtype='object')
```

```
movies.isnull().sum()
```

```
star_rating       0
title             0
content_rating    3
genre             0
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-29-305901486a81>:1: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this w
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  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)
```

```
train_df.head()
```

	Action	Adventure	Animation	Biography	Comedy	Crime	Drama	Family	Fantasy	Film-Noir	History	Horror	Mystery	Sci-Fi	Thriller	W
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

```
train_df=pd.concat([train_df,movies[ 'star_rating']],axis=1)
```

```
train_df.shape
```

```
(979, 18)
```

```
x=train_df.drop(['star_rating'],axis=1)
y=train_df['star_rating']
```

```
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2, random_state=42)
```

```
LR=LinearRegression()
```

```
LR.fit(x_train,y_train)
```

```
LinearRegression
LinearRegression()
```

```
y_pred=LR.predict(x_test)
```

```
print('RMSE using Linear regression is', metrics.mean_squared_error(y_test,y_pred,sample_weight=None))
```

```
RMSE using Linear regression is 0.0963980880321459
```

```
sv=SVR()
```

```
sv.fit(x_train,y_train)
```

```
SVR
SVR()
```

```
sv_pred=sv.predict(x_test)
```

```
print('RMSE using SVR is', metrics.mean_squared_error(y_test,sv_pred,sample_weight=None))
```

```
RMSE using SVR is 0.09749560506058148
```

```
clf=tree.DecisionTreeRegressor()
```

```
clf.fit(x_train,y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor()
```

```
DT_pred=clf.predict(x_test)
```

```
print('RMSE using DT is', metrics.mean_squared_error(y_test,DT_pred,sample_weight=None))
```

```
RMSE using DT is 0.18168370181405893
```