TASK-2

MOVIE RATING PREDICTION WITH PYTHON

* Build a model that predicts the rating of a movie based on

features like genre, director, and actors. You can use regression

techniques to tackle this problem.

* The goal is to analyze historical movie data and develop a model

that accurately estimates the rating given to a movie by users or

critics.

* Movie Rating Prediction project enables you to explore data

analysis, preprocessing, feature engineering, and machine

learning modeling techniques. It provides insights into the factors

that influence movie ratings and allows you to build a model that

can estimate the ratings of movies accurately.

**Data Acquisition**

In [1]:

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC, LinearSVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.linear\_model import Perceptron

from sklearn.linear\_model import SGDClassifier

from sklearn.tree import DecisionTreeClassifier

import warnings

warnings.filterwarnings('ignore')

*#Input movies dataset*

movies = pd.read\_csv(r"/kaggle/input/movielens/movies.dat", sep='::', engine='python', encoding='latin1')

movies.columns =['MovieID', 'Title', 'Genres']

movies.dropna(inplace=True)

movies.head()

/opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5

warnings.warn(f"A NumPy version >={np\_minversion} and <{np\_maxversion}"

|  | MovieID | Title | Genres |
| --- | --- | --- | --- |
| 0 | 2 | Jumanji (1995) | Adventure|Children's|Fantasy |
| 1 | 3 | Grumpier Old Men (1995) | Comedy|Romance |
| 2 | 4 | Waiting to Exhale (1995) | Comedy|Drama |
| 3 | 5 | Father of the Bride Part II (1995) | Comedy |
| 4 | 6 | Heat (1995) | Action|Crime|Thriller |

*#Input ratings dataset*

ratings = pd.read\_csv(r"/kaggle/input/movielens/ratings.dat",sep='::', engine='python')

ratings.columns =['UserID', 'MovieID', 'Rating', 'Timestamp']

ratings.dropna(inplace=True)

*#Read the sample ratings dataset*

ratings.head()

Out[2]:

|  | UserID | MovieID | Rating | Timestamp |
| --- | --- | --- | --- | --- |
| 0 | 1 | 661 | 3 | 978302109 |
| 1 | 1 | 914 | 3 | 978301968 |
| 2 | 1 | 3408 | 4 | 978300275 |
| 3 | 1 | 2355 | 5 | 978824291 |
| 4 | 1 | 1197 | 3 | 978302268 |

In [3]:

*#Input users dataset*

users = pd.read\_csv(r"/kaggle/input/movielens/users.dat",sep='::',engine='python')

users.columns =['UserID', 'Gender', 'Age', 'Occupation', 'Zip-code']

users.dropna(inplace=True)

*#Read the sample users dataset*

users.head()

Out[3]:

|  | UserID | Gender | Age | Occupation | Zip-code |
| --- | --- | --- | --- | --- | --- |
| 0 | 2 | M | 56 | 16 | 70072 |
| 1 | 3 | M | 25 | 15 | 55117 |
| 2 | 4 | M | 45 | 7 | 02460 |
| 3 | 5 | M | 25 | 20 | 55455 |
| 4 | 6 | F | 50 | 9 | 55117 |

In [4]:

*#Merge the ratings and users with movieID and UserID*

ratings\_user = pd.merge(ratings,users, on=['UserID'])

ratings\_movie = pd.merge(ratings,movies, on=['MovieID'])

master\_data = pd.merge(ratings\_user,ratings\_movie,

on=['UserID', 'MovieID', 'Rating'])[['MovieID', 'Title', 'UserID', 'Age', 'Gender', 'Occupation', "Rating"]]

master\_data.head()

Out[4]:

|  | MovieID | Title | UserID | Age | Gender | Occupation | Rating |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1357 | Shine (1996) | 2 | 56 | M | 16 | 5 |
| 1 | 3068 | Verdict, The (1982) | 2 | 56 | M | 16 | 4 |
| 2 | 1537 | Shall We Dance? (Shall We Dansu?) (1996) | 2 | 56 | M | 16 | 4 |
| 3 | 647 | Courage Under Fire (1996) | 2 | 56 | M | 16 | 3 |
| 4 | 2194 | Untouchables, The (1987) | 2 | 56 | M | 16 | 4 |

In [5]:

*# all 5 rating movies list count = 225473*

master\_data[master\_data['Rating'] == 5]

Out[5]:

|  | MovieID | Title | UserID | Age | Gender | Occupation | Rating |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1357 | Shine (1996) | 2 | 56 | M | 16 | 5 |
| 6 | 2268 | Few Good Men, A (1992) | 2 | 56 | M | 16 | 5 |
| 10 | 3468 | Hustler, The (1961) | 2 | 56 | M | 16 | 5 |
| 15 | 3578 | Gladiator (2000) | 2 | 56 | M | 16 | 5 |
| 26 | 1610 | Hunt for Red October, The (1990) | 2 | 56 | M | 16 | 5 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 998065 | 1077 | Sleeper (1973) | 6040 | 25 | M | 6 | 5 |
| 998070 | 2022 | Last Temptation of Christ, The (1988) | 6040 | 25 | M | 6 | 5 |
| 998071 | 2028 | Saving Private Ryan (1998) | 6040 | 25 | M | 6 | 5 |
| 998076 | 1094 | Crying Game, The (1992) | 6040 | 25 | M | 6 | 5 |
| 998077 | 562 | Welcome to the Dollhouse (1995) | 6040 | 25 | M | 6 | 5 |

225473 rows × 7 columns

*# all 5 rating movies list and Age Lass Then 25 count = 47163*

master\_data[(master\_data['Rating'] == 5) & (master\_data['Age'] < 25 ) ]

|  | MovieID | Title | UserID | Age | Gender | Occupation | Rating |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1883 | 2987 | Who Framed Roger Rabbit? (1988) | 18 | 18 | F | 3 | 5 |
| 1884 | 2989 | For Your Eyes Only (1981) | 18 | 18 | F | 3 | 5 |
| 1885 | 2622 | Midsummer Night's Dream, A (1999) | 18 | 18 | F | 3 | 5 |
| 1889 | 1683 | Wings of the Dove, The (1997) | 18 | 18 | F | 3 | 5 |
| 1893 | 3793 | X-Men (2000) | 18 | 18 | F | 3 | 5 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 996033 | 150 | Apollo 13 (1995) | 6031 | 18 | F | 0 | 5 |
| 996036 | 1010 | Love Bug, The (1969) | 6031 | 18 | F | 0 | 5 |
| 996038 | 1036 | Die Hard (1988) | 6031 | 18 | F | 0 | 5 |
| 996039 | 2001 | Lethal Weapon 2 (1989) | 6031 | 18 | F | 0 | 5 |
| 996043 | 1097 | E.T. the Extra-Terrestrial (1982) | 6031 | 18 | F | 0 | 5 |

47163 rows × 7 columns

*# all 5 rating movies list and Age Lass Then 25 count = 47163*

master\_data[(master\_data['Rating'] < 3) & (master\_data['Age'] < 25 )]

Out[7]:

|  |
| --- |
|  | MovieID | Title | UserID | Age | Gender | Occupation | Rating |
| 1898 | 1186 | Sex, Lies, and Videotape (1989) | 18 | 18 | F | 3 | 1 |
| 1902 | 3438 | Teenage Mutant Ninja Turtles (1990) | 18 | 18 | F | 3 | 2 |
| 1905 | 3439 | Teenage Mutant Ninja Turtles II: The Secret of... | 18 | 18 | F | 3 | 1 |
| 1907 | 1690 | Alien: Resurrection (1997) | 18 | 18 | F | 3 | 1 |
| 1909 | 2 | Jumanji (1995) | 18 | 18 | F | 3 | 2 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 996023 | 785 | Kingpin (1996) | 6031 | 18 | F | 0 | 2 |
| 996025 | 1648 | House of Yes, The (1997) | 6031 | 18 | F | 0 | 2 |
| 996030 | 1394 | Raising Arizona (1987) | 6031 | 18 | F | 0 | 2 |
| 996034 | 151 | Rob Roy (1995) | 6031 | 18 | F | 0 | 1 |
| 996041 | 553 | Tombstone (1993) | 6031 | 18 | F | 0 | 1 |

40329 rows × 7 columns

**Data Visualization**

master\_data['Age'].value\_counts().plot(kind='bar', color= ['cyan', 'blue'],alpha=0.5,figsize=(15,7))

plt.show()

A graph of a bar

Description automatically generated with medium confidence

master\_data['Rating'].value\_counts().plot(kind='bar', color=['green', 'yellow'],alpha=0.5,figsize=(15,7))

plt.show()

A graph of a bar chart

Description automatically generated with medium confidence

res = master\_data[master\_data.Title == "Only You (1994)"]

plt.plot(res.groupby("Age")["MovieID"].count(),'--bo')

res.groupby("Age")["MovieID"].count()

Out[10]:

Age

18 34

25 72

35 29

45 12

50 4

56 7

Name: MovieID, dtype: int64

A graph with blue dots and lines

Description automatically generated

*#Find the ratings for all the movies reviewed by for a particular user of user id = 700*

res = master\_data[master\_data.UserID == 700]

plt.scatter(y=res.Title, x=res.Rating , color = 'aqua')

plt.show()

A graph with blue dots

Description automatically generated

res = master\_data.groupby("Title").size().sort\_values(ascending=False)[:25]

plt.ylabel("Title")

plt.xlabel("Viewership Count")

res.plot(kind="barh", color = ['lightseagreen', 'turquoise', 'deepskyblue'])

plt.show()

A graph with blue and green lines

Description automatically generated

res = master\_data.groupby("Gender").size().sort\_values(ascending=False)[:25]

plt.ylabel("Gender")

plt.xlabel("Viewership Count")

res.plot(kind="kde")

plt.show()

A graph with a blue line

Description automatically generated

In [14]:

res = master\_data.groupby("Rating").size().sort\_values(ascending=False)[:25]

plt.ylabel("Rating")

plt.xlabel("Viewership Count")

res.plot(kind='bar', color= ['red', 'darkorange'])

Out[14]:

<Axes: xlabel='Rating', ylabel='Rating'>

A graph of a bar chart

Description automatically generated with medium confidence

# A diagram of machine learning Description automatically generated**Machine Learning**

*#First 500 extracted records*

first\_500 = master\_data[500:]

first\_500.dropna(inplace=True)

In [16]:

linkcode

*#Use the following features:movie id,age,occupation*

features = first\_500[['MovieID','Age','Occupation']].values

*#Use rating as label*

labels = first\_500[['Rating']].values

*#Create train and test data set*

train, test, train\_labels, test\_labels = train\_test\_split(features,labels,test\_size=0.33,random\_state=42)

linkcode

A diagram of modeling process

Description automatically generated

# Machine Learning Models

### Logistic Regression

logreg = LogisticRegression()

logreg.fit(train, train\_labels)

Y\_pred = logreg.predict(test)

acc\_log = round(logreg.score(train, train\_labels) \* 100, 2)

acc\_log

Out[18]:

34.86

### K Nearest Neighbors Classifier

In [19]:

linkcode

knn = KNeighborsClassifier(n\_neighbors = 3)

knn.fit(train, train\_labels)

Y\_pred = knn.predict(test)

acc\_knn = round(knn.score(train, train\_labels) \* 100, 2)

acc\_knn

Out[19]:

44.79

### Gaussian Naive Bayes

In [20]:

linkcode

gaussian = GaussianNB()

gaussian.fit(train, train\_labels)

Y\_pred = gaussian.predict(test)

acc\_gaussian = round(gaussian.score(train, train\_labels) \* 100, 2)

acc\_gaussian

34.88

### Perceptron

In [21]:

perceptron = Perceptron()

perceptron.fit(train, train\_labels)

Y\_pred = perceptron.predict(test)

acc\_perceptron = round(perceptron.score(train, train\_labels) \* 100, 2)

acc\_perceptron

Out[21]:

33.05

linkcode

### Decision Tree

decision\_tree = DecisionTreeClassifier()

decision\_tree.fit(train, train\_labels)

Y\_pred = decision\_tree.predict(test)

acc\_decision\_tree = round(decision\_tree.score(train, train\_labels) \* 100, 2)

acc\_decision\_tree

Out[22]:

56.54

linkcode

# Thank You