#import required libraries

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from itertools import product

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

#column name lists assignment

movies\_col = ['MovieID','Title','Genres']

ratings\_col = ['UserID','MovieID','Rating','Timestamp']

users\_col = ['UserID','Gender','Age','Occupation','Zip-code']

#data frame creation for movies, users and ratings files

movies\_df = pd.read\_csv('movies.dat', header=None, delimiter='::',

engine='python', names=movies\_col)

movies\_df.dropna(inplace=True)

print(movies\_df.head())

ratings\_df = pd.read\_csv('ratings.dat', header=None, delimiter='::',

engine='python', names=ratings\_col)

ratings\_df.dropna(inplace=True)

print(ratings\_df.head())

users\_df = pd.read\_csv('users.dat', header=None, delimiter='::',

engine='python', names=users\_col)

users\_df.dropna(inplace=True)

print(users\_df.head())

#merging above dataframes into one dataframe to create master\_data dataframe

master\_data = ratings\_df.merge(movies\_df, on = ['MovieID'], how = 'outer')

master\_data = master\_data.merge(users\_df, on = ['UserID'], how = 'outer')

master\_data.dropna(inplace=True)

master\_data.head()

#User Age distribution using histogram

master\_data['Age'].value\_counts().plot(kind='bar',figsize=(10,5))

plt.show()

master\_data.Age.plot.hist(bins=25)

plt.title("Distribution of users' ages")

plt.ylabel('number of users')

plt.xlabel('Age')

'''

Graph shows normally distributed data for age of users, most of the user fall in

25-34 age of bracket.

'''

#Visualize overall rating by users

master\_data['Rating'].value\_counts().plot(kind='bar',figsize=(10,5))

plt.show()

master\_data.loc[master\_data['Age'] ==1, 'Age Group'] = 'Under 18'

master\_data.loc[master\_data['Age'] ==18, 'Age Group'] = '18-24'

master\_data.loc[master\_data['Age'] ==25, 'Age Group'] = '25-34'

master\_data.loc[master\_data['Age']==35, 'Age Group'] = '35-44'

master\_data.loc[master\_data['Age'] ==45, 'Age Group'] = '45-49'

master\_data.loc[master\_data['Age'] ==50, 'Age Group'] = '50-55'

master\_data.loc[master\_data['Age']==56, 'Age Group'] = '56+'

#User rating of the movie 'Toy story'

user\_rating\_toy\_story = master\_data[master\_data.Title.str.contains('Toy Story')][['UserID','Title',

'Rating']].groupby(['Title'])['Rating'].agg(['sum','count']).reset\_index()

print(user\_rating\_toy\_story)

user\_rating\_toy\_story['Overall\_Rating'] = (user\_rating\_toy\_story['sum']/user\_rating\_toy\_story['count']).round()

user\_rating\_toy\_story.drop(columns='sum',axis=1,inplace=True)

user\_rating\_toy\_story.rename(columns = {'count':'Number\_of\_votes'}, inplace=True)

print(user\_rating\_toy\_story)

#Top 25 movies by viewership rating

group\_rating = master\_data[['Title','Rating']].groupby('Title')['Rating'].agg(['sum','count']).reset\_index()

group\_rating.rename(columns = {'count':'Number\_of\_votes','sum':'Total\_rating'}, inplace = True)

group\_rating['Overall\_rating'] = (group\_rating['Total\_rating']/group\_rating['Number\_of\_votes']).round()

print(group\_rating.head())

top\_25\_movies = group\_rating.sort\_values(by=['Overall\_rating','Number\_of\_votes'],

ascending = False)[['Title','Overall\_rating']].head(25).reset\_index()

top\_25\_movies.drop(columns='index',inplace=True,axis=1)

print(top\_25\_movies)

#ratings for all the movies reviewed by for a particular user of user id = 2696

movie\_list = master\_data[master\_data['UserID']==2696]['MovieID'].tolist()

movies\_for\_2696 = master\_data[master\_data['MovieID'].isin(movie\_list)][['Title','Rating']].groupby('Title')['Rating'].agg(['sum','count']).reset\_index()

movies\_for\_2696['Overall\_rating'] = (movies\_for\_2696['sum']/movies\_for\_2696['count']).round()

movies\_for\_2696.drop(columns=['sum','count'],inplace=True)

print(movies\_for\_2696)

#Feature Engineering

movie\_Genres\_list = master\_data.Genres.tolist()

movie\_genre\_list = []

i = 0

while(i<len(movie\_Genres\_list)):

movie\_genre\_list+=movie\_Genres\_list[i].split('|')

i+=1

print(movie\_genre\_list)

unique\_genre = list(set(movie\_genre\_list))

print(unique\_genre)

print(len(unique\_genre))

new\_data = pd.concat([master\_data,master\_data.Genres.str.get\_dummies()], axis=1)

print(new\_data.columns)

#df = new\_data[['MovieID','Rating','Gender','Age Group','Occupation']]

df = new\_data.drop(columns=['Title','Zip-code','Timestamp','Genres'])

print(df.head())

df.Occupation.value\_counts()

df.set\_index('MovieID', inplace = True)

X = df.drop(columns=['Rating'])

Y = df['Rating']

x1 = pd.get\_dummies(data=X)

x2 = pd.get\_dummies(X['Occupation'], prefix = 'Occupation')

X = pd.concat([x1,x2], axis=1)

X.columns

X.drop(columns = ['Occupation','Gender\_F','Age Group\_56+','Occupation\_20.0'],

axis = 1, inplace=True)

X.head()

XY = pd.concat([X,Y], axis=1)

XY.head()

XY.corr()

sns.heatmap(XY.corr())

sns.pairplot(XY.corr())

X = X.values

Y = Y.values

train, test, train\_labels, test\_labels = train\_test\_split(X,Y,test\_size=0.33,random\_state=42)

#applying decision tree classifier

decision\_tree = DecisionTreeClassifier()

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

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

acc\_decision\_tree = accuracy\_score(test\_labels, Y\_pred)\*100

print(acc\_decision\_tree) #35 accuracy score

#applying random forest classifier

random\_forest = RandomForestClassifier(n\_estimators=100)

random\_forest.fit(train, train\_labels)

Y\_pred = random\_forest.predict(test)

acc\_random\_forest = accuracy\_score(test\_labels, Y\_pred)\*100

print(acc\_random\_forest)

#36 accuracy score