```
#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()

#Exploratory Data Analysis

#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)

'''
We have used columns like Gender, occupation, age and Genre to get the rating
predictions here.
'''

#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
```