# This Python 3 environment comes with many helpful analytics libraries installed

# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python

# For example, here's several helpful packages to load in

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

# Input data files are available in the "../input/" directory.

# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input directory

import os

print(os.listdir("../input"))

# Any results you write to the current directory are saved as output.

# data analysis and wrangling

import pandas as pd

import numpy as np

import random as rnd

# visualization

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

# machine learning

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

#Data acquisition of the movies dataset

df\_movie=pd.read\_csv('../input/movies.dat', sep = '::', engine='python')

df\_movie.columns =['MovieIDs','MovieName','Category']

df\_movie.dropna(inplace=True)

df\_movie.head()

#Data acquisition of the rating dataset

df\_rating = pd.read\_csv("../input/ratings.dat",sep='::', engine='python')

df\_rating.columns =['ID','MovieID','Ratings','TimeStamp']

df\_rating.dropna(inplace=True)

df\_rating.head()

#Data acquisition of the users dataset

df\_user = pd.read\_csv("../input/users.dat",sep='::',engine='python')

df\_user.columns =['UserID','Gender','Age','Occupation','Zip-code']

df\_user.dropna(inplace=True)

df\_user.head()

df = pd.concat([df\_movie, df\_rating,df\_user], axis=1)

df.head()

#Visualize user age distribution

df['Age'].value\_counts().plot(kind='barh',alpha=0.7,figsize=(10,10))

plt.show()

df.Age.plot.hist(bins=25)

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

plt.ylabel('count of users')

plt.xlabel('Age')

labels = ['0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79']

df['age\_group'] = pd.cut(df.Age, range(0, 81, 10), right=False, labels=labels)

df[['Age', 'age\_group']].drop\_duplicates()[:10]

#Visualize overall rating by users

df['Ratings'].value\_counts().plot(kind='bar',alpha=0.7,figsize=(10,10))

plt.show()

groupedby\_movieName = df.groupby('MovieName')

groupedby\_rating = df.groupby('Ratings')

groupedby\_uid = df.groupby('UserID')

#groupedby\_age = df.loc[most\_50.index].groupby(['MovieName', 'age\_group'])

movies = df.groupby('MovieName').size().sort\_values(ascending=True)[:1000]

print(movies)

ToyStory\_data = groupedby\_movieName.get\_group('Toy Story 2 (1999)')

ToyStory\_data.shape

#Find and visualize the user rating of the movie “Toy Story”

plt.figure(figsize=(10,10))

plt.scatter(ToyStory\_data['MovieName'],ToyStory\_data['Ratings'])

plt.title('Plot showing the user rating of the movie “Toy Story”')

plt.show()

#Find and visualize the viewership of the movie “Toy Story” by age group

ToyStory\_data[['MovieName','age\_group']]

#Visualize the rating data by user of user id = 2696

userid\_2696 = groupedby\_uid.get\_group(2696)

userid\_2696[['UserID','Ratings']]

#First 500 extracted records

first\_500 = df[500:]

first\_500.dropna(inplace=True)

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

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

#Use rating as label

labels = first\_500[['Ratings']].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)

#Create a histogram for movie

df.Age.plot.hist(bins=25)

plt.title("Movie & Rating")

plt.ylabel('MovieID')

plt.xlabel('Ratings')

#Create a histogram for age

df.Age.plot.hist(bins=25)

plt.title("Age & Rating")

plt.ylabel('Age')

plt.xlabel('Ratings')

# 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

# Support Vector Machines

svc = SVC()

svc.fit(train, train\_labels)

Y\_pred = svc.predict(test)

acc\_svc = round(svc.score(train, train\_labels) \* 100, 2)

acc\_svc

# K Nearest Neighbors Classifier

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

# Gaussian Naive Bayes

gaussian = GaussianNB()

gaussian.fit(train, train\_labels)

Y\_pred = gaussian.predict(test)

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

acc\_gaussian

# Perceptron

perceptron = Perceptron()

perceptron.fit(train, train\_labels)

Y\_pred = perceptron.predict(test)

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

acc\_perceptron

# Linear SVC

linear\_svc = LinearSVC()

linear\_svc.fit(train, train\_labels)

Y\_pred = linear\_svc.predict(test)

acc\_linear\_svc = round(linear\_svc.score(train, train\_labels) \* 100, 2)

acc\_linear\_svc

# Stochastic Gradient Descent

sgd = SGDClassifier()

sgd.fit(train, train\_labels)

Y\_pred = sgd.predict(test)

acc\_sgd = round(sgd.score(train, train\_labels) \* 100, 2)

acc\_sgd

# 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

# Random Forest

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

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

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

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

acc\_random\_forest = round(random\_forest.score(train, train\_labels) \* 100, 2)

acc\_random\_forest

models = pd.DataFrame({

'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',

'Random Forest', 'Naive Bayes', 'Perceptron',

'Stochastic Gradient Decent', 'Linear SVC',

'Decision Tree'],

'Score': [acc\_svc, acc\_knn, acc\_log,

acc\_random\_forest, acc\_gaussian, acc\_perceptron,

acc\_sgd, acc\_linear\_svc, acc\_decision\_tree]})

models.sort\_values(by='Score', ascending=False)