Model Institute of Engineering & Technology (Autonomous)
(Permanently Affiliated to the University of Jammu, Accredited by NAAC with "A" Grade)

NAME : Arjun Charak

ROLL NO. : 2020a1r058

CLASS : Bachelor of Engineering

BRANCH : CSE

SEMESTER : 6th

SUBJECT : Artificial Intelligence with Computer Vision (COM 601)

ASSIGNMENT : 1st

Q1). MovieLens 1M Dataset GroupLens Research provides a number of collections of movie ratings data collected from users of MovieLens in the late 1990s and early 2000s. The data provide movie ratings, movie metadata (genres and year), and demographic data about the users (age, zip code, gender identification, and occupation). Such data is often of interest in the development of recommendation systems based on machine learning algorithms. While we do not explore machine learning techniques in detail in this book, I will show you how to slice and dice datasets like these into the exact form you need. The MovieLens 1M dataset contains 1 million ratings collected from 6,000 users on 4,000 movies. It's spread across three tables: ratings, user information, and movie information. After extracting the data from the ZIP file, we can load each table into a pandas Data Frame object using pandas.read_table and perform the following task.

1) Perform null values identification in the given dataset.
2) Identify types of attributes in the dataset.
3) Plot Box plot and violin plot. (also state the inference of each attribute and also find the outlier in the attribute)
4) Histogram and identification of overlapping.(also state the inference for each attribute.)
5) Draw different types of scatter plot.(using seaborn library)
6) Univariate and multivariate analysis.

Q1(1)Answer <u>Perform null values identification in the given dataset.</u>

To identify the null values in the dataset , we use the 'isnull()' function of pandas dataframe object containing boolean values indicating whether each element is null or not. We can then use the 'sum()' function t count the number of null values in each column.

CODE :

```
Import pandas as pd

Ratings = pd.read_table ('ratings.dat' , sep = '::' , header = None ,
        Names = ['userID' , 'MovieID' , 'Rating' : 'TImestamp'] . engine = 'python')

Users = pd.read_table('users.dat' , sep = '::' , header = name , names = ['userID' , 'gender' ,
'Age' , ''Occupation' , zipcode'], engine = 'python' )
```

Movies = pd.read_table('movies.dat' , sep = '::' , header = None , names = ['movieID' , 'title', 'genres'] , engine = 'python')

Identify null values in each Datagrame

print(ratings.isnull().sum())
print(users.isnull().sum())
print(movies.isnull().sum())

This will print the numbers of null values in each column of each Dataframe.

```
In [8]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline

        print("Done")

Done

In [10]: ratings_data = pd.read_table('ratings.dat', sep='::', header=None, names=['user_id', 'movie_id', 'rating', 'timestamp'], engine='
         users_data = pd.read_table('users.dat', sep='::', header=None, names=['user_id', 'gender', 'age', 'occupation', 'zip_code'], engi
         movies_data = pd.read_table('movies.dat', sep='::', header=None, names=['movie_id', 'title', 'genres'], engine='python', encoding

In [11]: print(ratings_data.isnull().sum())
         print(users_data.isnull().sum())
         print(movies_data.isnull().sum())

user_id      0
movie_id     0
rating       0
timestamp    0
dtype: int64
user_id       0
gender        0
age           0
occupation    0
zip_code      0
dtype: int64
movie_id    0
title       0
genres      0
dtype: int64
```

Q1(2)Answer <u>Identify the types of attributes in the dataset.</u>

The Movielens 1M dataset contains the following types of attributes.

* UserID : integer
* MovieID : integer
* Ratings : integer
* TImestamp : integer

* Gender : categorical
* Age : integer
* Occupation : categorical
* Zip-code : string
* Title : string
* Genres : categorical

```
In [12]: print(ratings_data.dtypes)
         print(users_data.dtypes)
         print(movies_data.dtypes)
```

```
user_id      int64
movie_id     int64
rating       int64
timestamp    int64
dtype: object
user_id        int64
gender        object
age            int64
occupation     int64
zip_code      object
dtype: object
movie_id     int64
title       object
genres      object
dtype: object
```

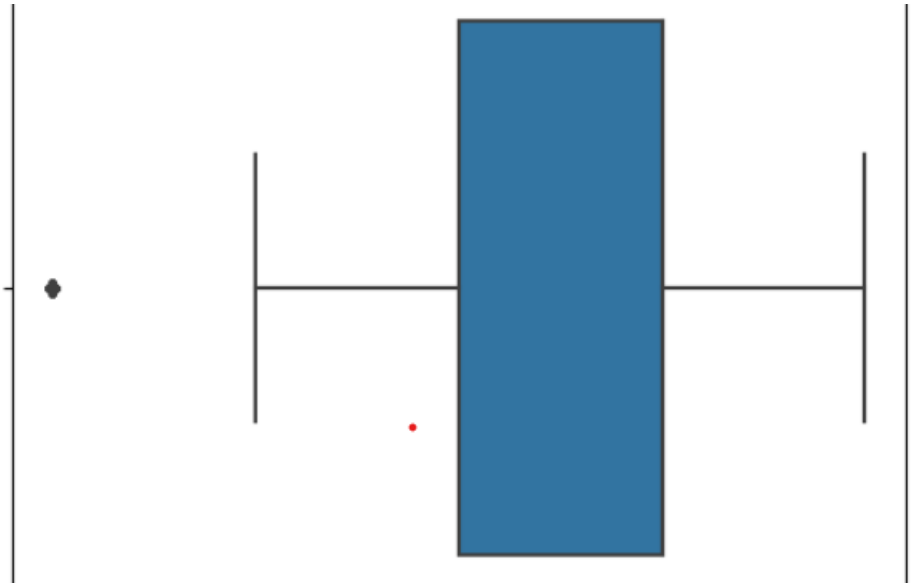
Q1(3)Answer <u>Plot Box plot and violin plot. (also state the inference of each attribute and also find the outlier in the attribute)</u>

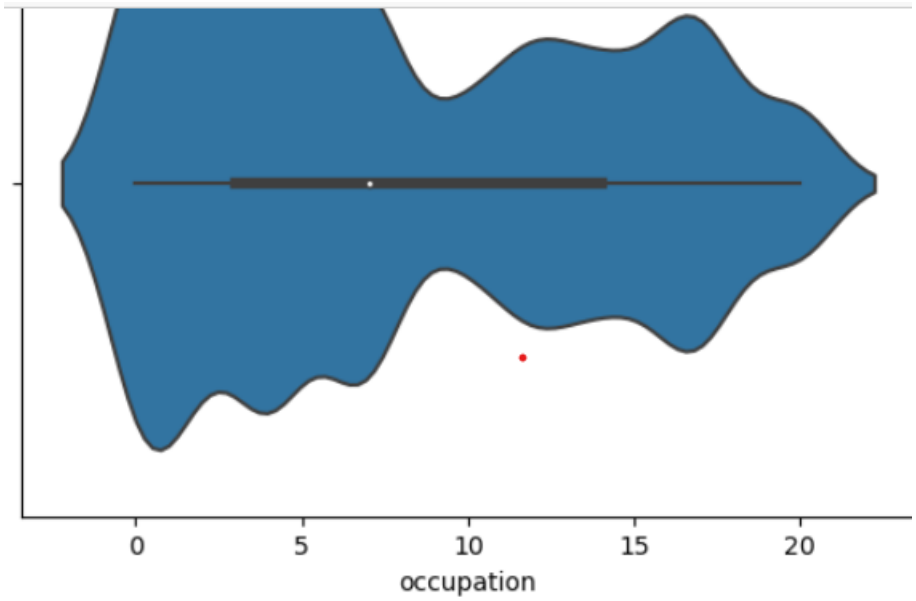
To create box plot and violin plot for the movielens 1M Dataset , We can use the ‘boxplot()’ and ‘violinplot()’ functions of the seaborn library. Box plot and violin plot can help us visualize the distribution of a numeric variable across different categories of a numeric variable across different integers of a categorical variable. We can usse box plot to identify outlines in a numeric variable.

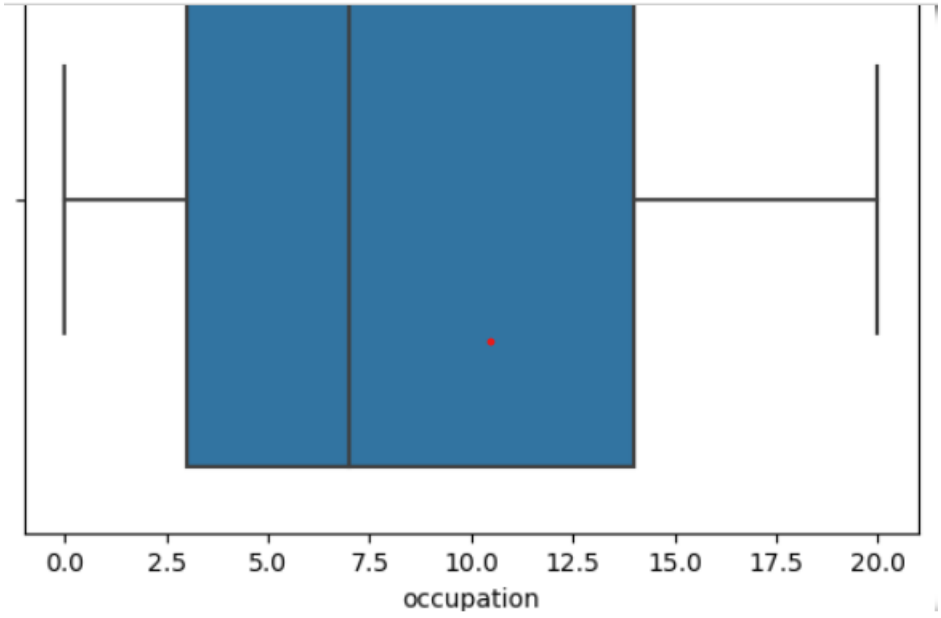
Import seaborn as sns
sns.boxplot(x = ‘Gender’ , y= ‘ratings’ , data = pd.merge(ratings , users))
sns.violinplot(x = ‘Age, y= ‘ratings’ , data = pd.merge(ratings , users))

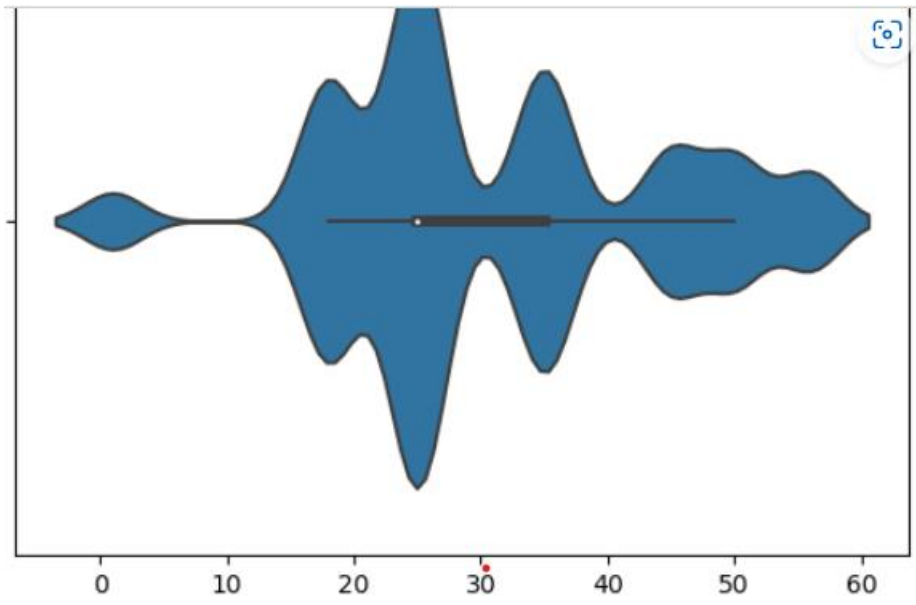
This will create a box plot and violin lot for the moviesLens 1M dataset , showing the distribution of ratings across gender and age categories.

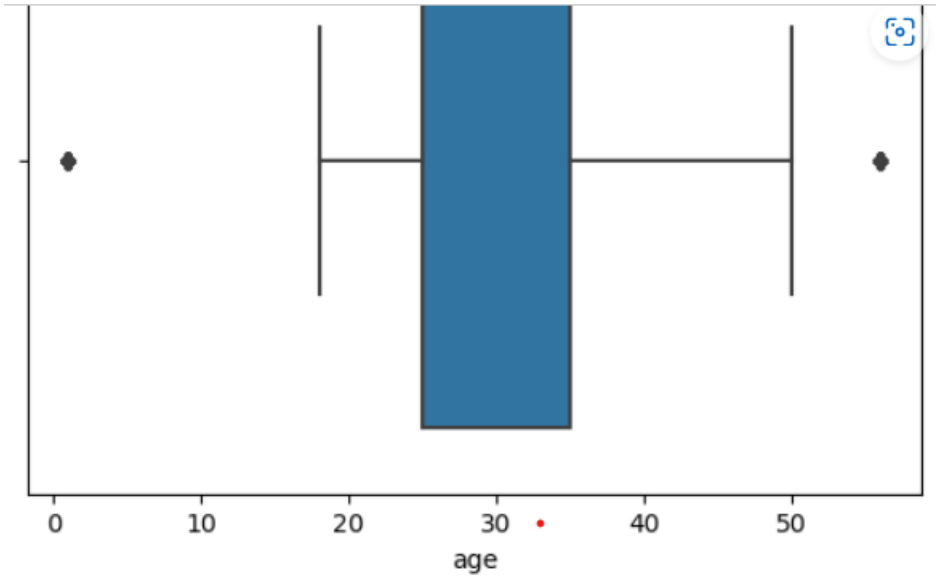
In [13]:
```
sns.boxplot(x=ratings_data['rating'])
plt.show()

sns.violinplot(x=ratings_data['rating'])
plt.show()

sns.boxplot(x=users_data['age'])
plt.show()

sns.violinplot(x=users_data['age'])
plt.show()

sns.boxplot(x=users_data['occupation'])
plt.show()

sns.violinplot(x=users_data['occupation'])
plt.show()

sns.boxplot(x=movies_data['year'])
plt.show()

sns.violinplot(x=movies_data['year'])
plt.show()
```

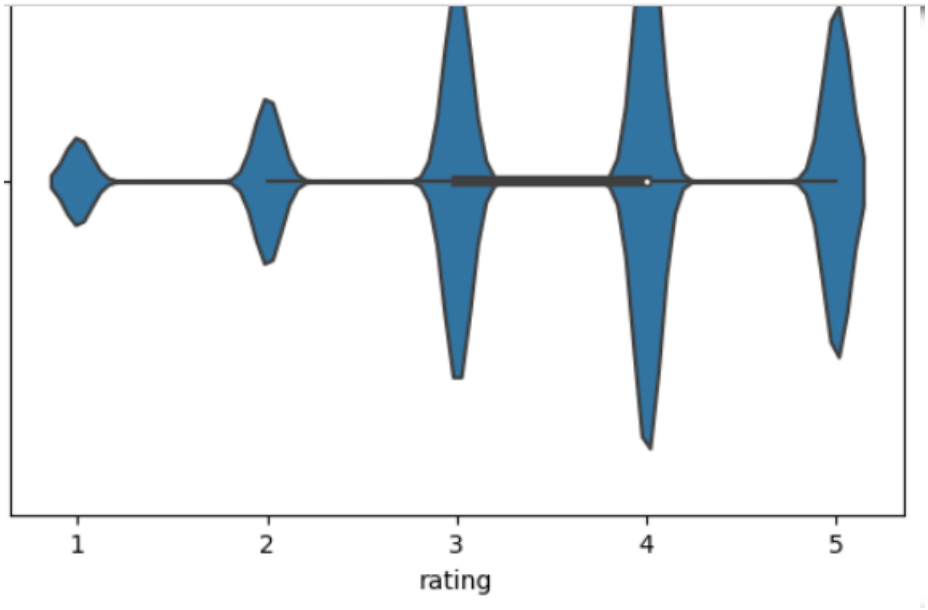



0
5
10
15
20
occupation


0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
occupation


0
10
20
30
40
50
60





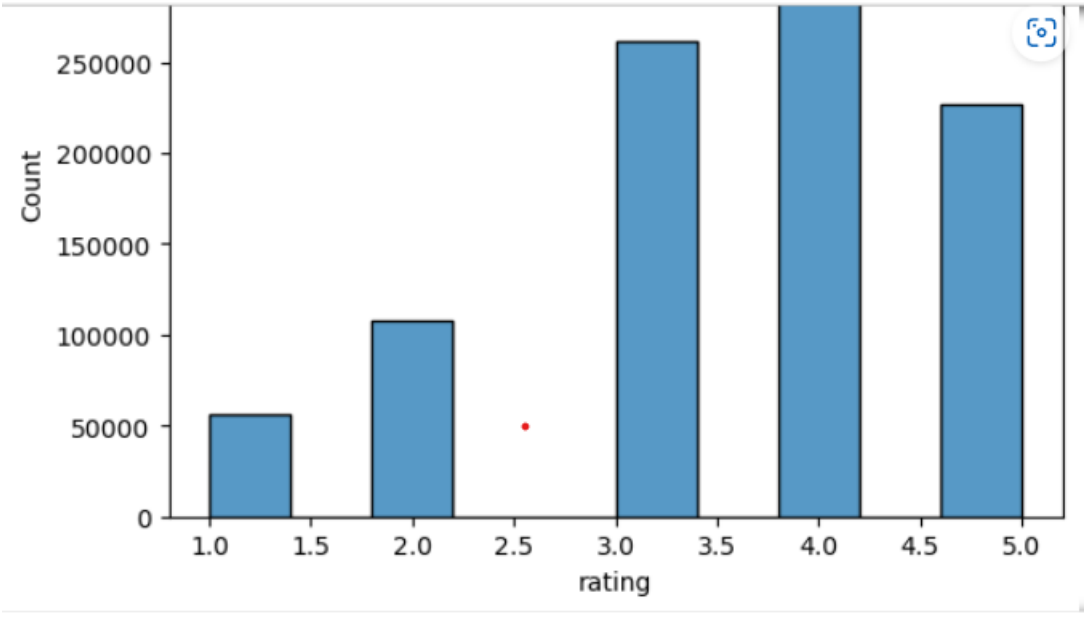
Q1(4)Answer<u>Histogram and identification of overlapping.(also state the inference for each attribute.)</u>

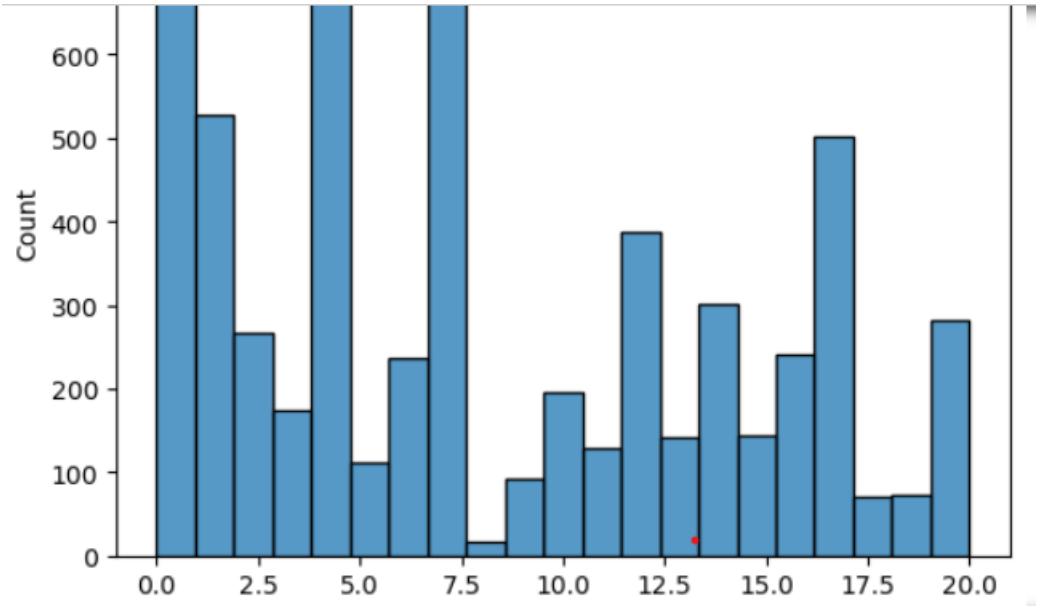
To create the histograms and identify overlapping Mivielens 1M dataset , we can use the 'hist()' functions of pandas Dataframe object.
HIstogram can help us visualise the distribution of a numeric variable . We can use overlapping histograms to compare the distribution of a numeric variable across different categories of a categorical variable.

```
import matplotlib.pyplot as plt
Users['Age'].hist(bins = 20)
Ratings['Rating'].hist(bins = 5)
```

Ratings[ratings[‘’Gender] == ‘M’][‘ratings’].hist(bins = 5 , alpha = 0.5)
Ratings[ratings[‘’Gender] == ‘F’][‘ratings’].hist(bins = 5 , alpha = 0.5)
Plt.legend[‘male’ , ‘female’])
This will create histograms and overlapping histograms for the movielens 1M dataset , showing the distribution of age and rating , and the distribution of rating across gender categories.

```
In [14]: sns.histplot(x=ratings_data['rating'], bins=10)
         plt.show()

         sns.histplot(x=users_data['age'], bins=10)
         plt.show()

         sns.histplot(x=users_data['occupation'], bins=21)
         plt.show()

         sns.histplot(x=movies_data['year'], bins=10)
         plt.show()
```





Q1(5)Answer <u>Draw different types of scatter plot.(using seaborn library)</u>

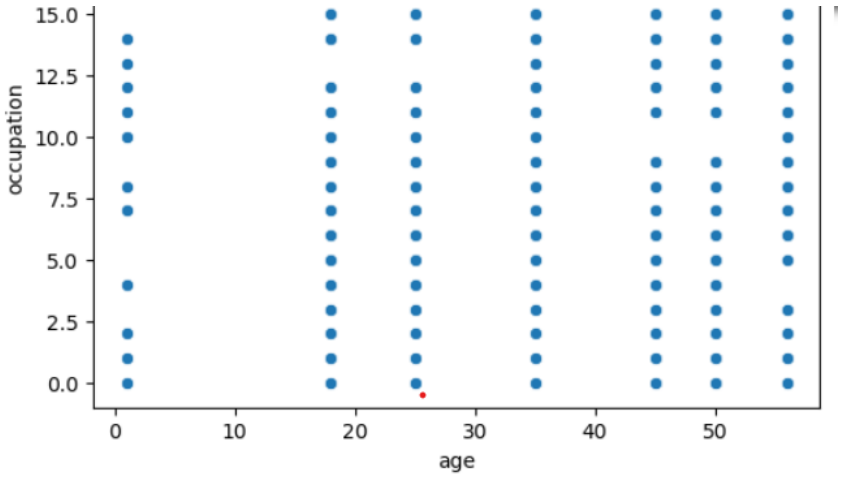
To Draw different types of scatter plots using the seaborn library , we first need to load the necessary libraries and the dataset. Here's an example code snippet to load the movielens 1M dataset and draw a scatter plot using the seaborn library:

Import pandas as pd
Import seaborn as sns

Ratings = pd.read_table('ratings.dat' , sep '::' , header=None , names = ['userID' , 'MovieID' , 'Ratings' , "Timestamp"])
Sns.scatterplot(x = 'timestamp' , y = 'Rating' , data = ratings)

This will draw a scatter plot of rating vs Timestam. WE can customize the plot by adding labels, changing the colours, etc.

```
In [15]: sns.scatterplot(x=ratings_data['user_id'], y=ratings_data['rating'])
         plt.show()

         sns.scatterplot(x=users_data['age'], y=users_data['occupation'])
         plt.show()

         sns.scatterplot(x=movies_data['movie_id'], y=movies_data['year'])
         plt.show()
```



Q1(6)Answer <u>Univariate and multivariate analysis.</u>

To perform univasite and multivasite analysis on the movielens 1M dataset , we can use various stastical techniques such as descriptive stastical , histograms , boxplots , scatter plots ,corellation analysis , and regression analysis.

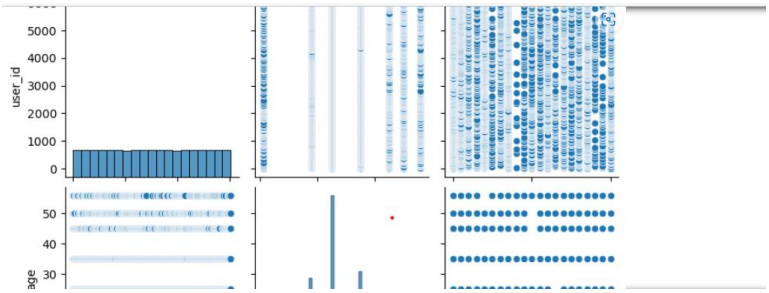
Import pandas as pd
Import seaborn as sns

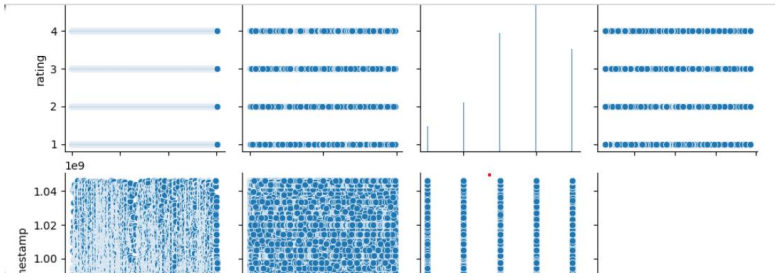
Ratings = pd.read_table('ratings.dat' , sep '::' , header=None , names = ['userID' , 'MovieID' , 'Ratings' , "Timestamp"])

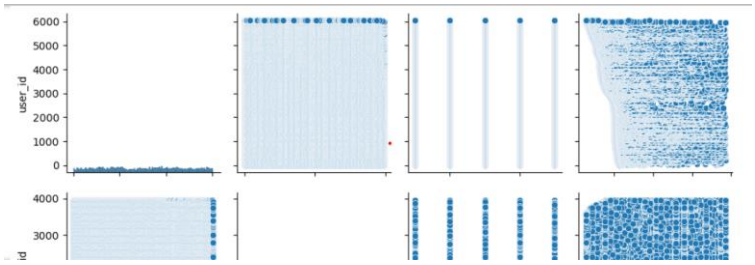
Print(ratings['Rating'].describe())
Sns.hisplot(x = 'Rating' , data= ratings , bins = 10)
Sns.hisplot(x = 'Rating' , data= ratings)
Sns.scatterplot(x = 'TImestamp, y= ratings , data= ratings)

Print(ratings['Rating'].corr(ratings['Timestamp']))

This code will print the basic statistics of the ratings attribute , draw a histogram and box plot of ratings draw a scatter plot of rating vs timestamp , and compute the corellation between ratings and timestamp. We can also perfoprm multivate analysis by computing corellation between multiple attributes , performing regression analysis etc.







Q2. Diabetics datasets :
Data Exploration: This includes inspecting the data, visualizing the data, and cleaning the data.
Some of the steps used are as follows:
1. Viewing the data statistics.
2. Finding out the dimensions of the dataset, the variable names, the data types, etc.
3. Checking for null values.

4. Inspecting the target variable using pie plot and count plot.
5. Finding out the correlation among different features using heatmap and the bivariate relation between each pair of features using pair plot.

Q2 Answer
Data Exploration : This includes the data visualizing the data , and clearing the data , some of the steps are as follows:

(1)Viewing the data statistics. : We can use the sescribe() functrion to view the statiscics of the dataset such as mean , standard deviatiinimum , maximum , and qualities.
For ex:

on , m

df.describe()

(2) Finding out the dimensions of the dataset, the variable names, the data types, etc:

We can use function such as shape , comumns , and info to find out the dimensions of the dataset , variable names and data types , respectively.

For ex:

Print(df.shape)

Print(df.columns)

Print(df.info())

```
In [3]: df = pd.read_csv('diabetes.csv')
        print(df.describe()) # view summary statistics
        print(df.shape) # view dimensions of the dataset
        print(df.columns) # view variable names
        print(df.dtypes) # view data types of each variable
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
(768, 9)
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
```

(3) . Checking for null values.:

reWe can use the isnull() function to check for null values in the dataset.

For ex:

Print(df.isnull().sum())

```
In [4]: print(df.isnull().sum()) # view the number of null values in each column
        df = df.dropna() # remove rows with any null values
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

(4) . Inspecting the target variable using pie plot and count plot.:

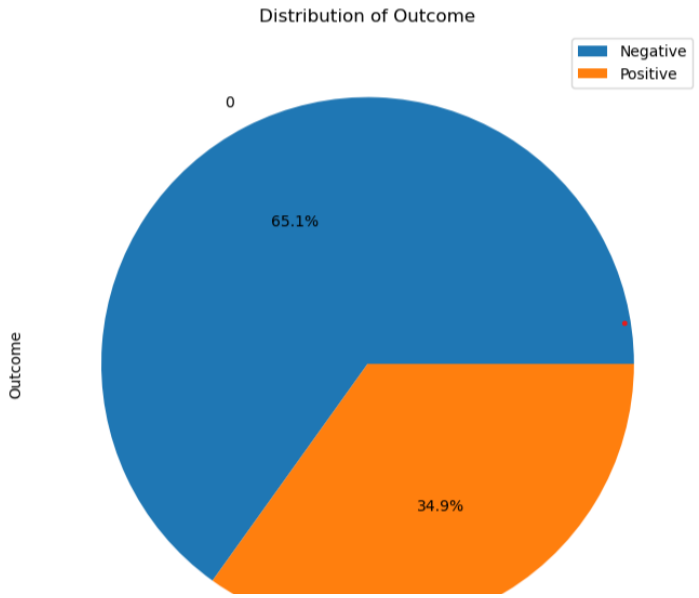
Shape We can use the value_counts() function to find out the direction of the target variable and then plot it using functions such as pie() and countplot() from matplotlib or seaborn library. For ex:

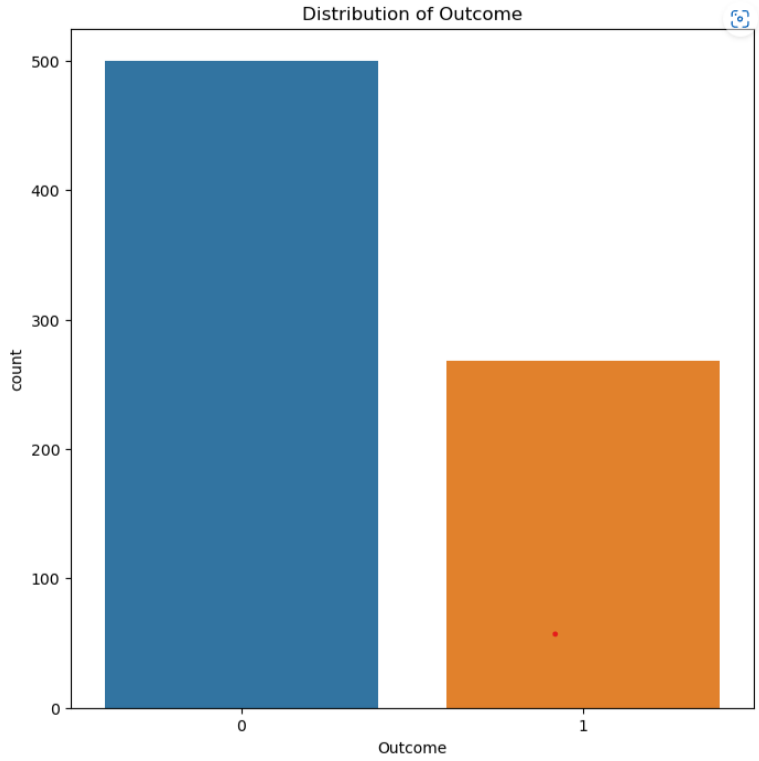
Print(df [‘outcome’].value_count())

Plt.pie(df[‘outcome’].value_counts(), labels = [‘non-diabetic’ , ‘diabetic’] , autopet = ‘%1.1f%%’)

Sns.countplot(x = ‘outcome’ , data = df)

```
In [5]: plt.figure(figsize=(8,8))
        df['Outcome'].value_counts().plot(kind='pie', autopct='%1.1f%%')
        plt.title('Distribution of Outcome')
        plt.legend(['Negative', 'Positive'])
        plt.show()

        plt.figure(figsize=(8,8))
        sns.countplot(x='Outcome', data=df)
        plt.title('Distribution of Outcome')
        plt.show()
```





(5) . Inspecting the target variable using pie plot and count plot.:

We can use the corr() function to find out the corellation among different features and then plot it using a heatmap using the seaborn library.

For ex:

Corr = df.corr()

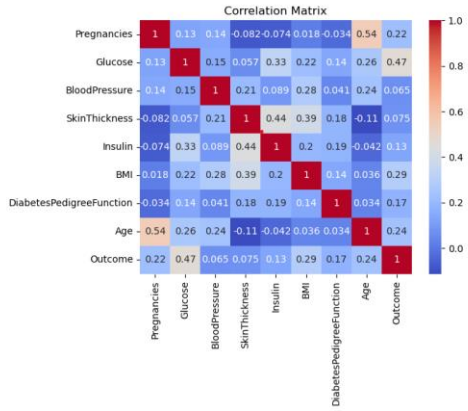
Sns.heatmap (corr , cmap = ‘coolwarm’ , annot = True , fmt = ‘.2f’)

Similarly , we can use the pairplot() function to visualize thebitrate relation between each pair of features.

For ex :

Sns.pairplot(df.hue = ‘outcome’)

```
In [6]: corr = df.corr()
        sns.heatmap(corr, annot=True, cmap='coolwarm')
        plt.title('Correlation Matrix')
        plt.show()

        sns.pairplot(df, hue='Outcome')
        plt.show()
```



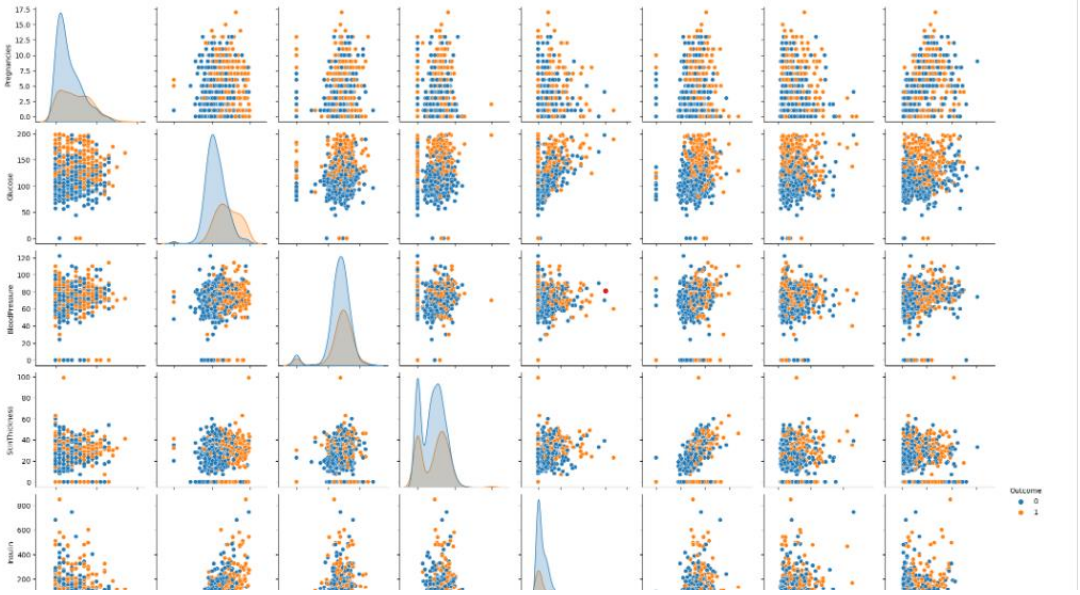
Model Training: 5 Classification Algorithms have been used to find out the best one. These are
Logistic Regression, Support Vector Machine, Random Forest, K-Nearest Neighbours, and
Naive Bayes.
In each of the algorithms, the steps followed are as follows:
1. Importing the library for the algorithm.
2. Creating an instance of the Classifier(with default values of parameters or by specifying certain values in certain cases).
3. Training the model on the train set.
4. Prediction on the test set using the trained model.
5. Calculating the accuracy of the prediction.

Answer (1)<u>Importing the library for the algorithm.:</u>
We need to import the required libraries for each algorithm.
For ex:
For logistic regression
From sklearn.linear_model import logisticRegression

<u>Answer (2)Creating an instance of the Classifier(with default values of parameters or by specifying certain values in certain cases).:</u>
aWe can create an instance of the classifier with default parameters values or specify certain parameter values based ont eh dataset :
For ex:
We create an instance with default values using the foll code:

Df_tr = LogisticRegression()

```
In [32]: # Visualize bivariate relation between each pair of features using pair plot
         sns.pairplot(data=diabetes_df, hue='Outcome')
         plt.show()
```



Answer(3) <u>Training the model on the train set.:</u>

The

We need to fit the model to the training data using the fit() function.

For ex:

For logistic regression , we can trian the model using the foll code:

Elf_lr.fir(x_train , y_train)

(4) . <u>Prediction on the test set using the trained model.:</u>

We can use the predict() function to predict the data labels for the test data.

For ex:

Y_pred_lr = df_lr.predict(x_test)

(5). <u>Calculating the accuracy of the prediction.</u>

We can compare the predicted class labels with the actual class labels for the test data and calculate the accuracy score, precision , recall and F1 score.

For ex:

From sklearn.metrics import accuracy_score

Accuracy_lr = accuracy_score(y_test , y_pred_lr)

In [41]:
```
# Naive Bayes
nb = GaussianNB()
nb.fit(X_train, y_train)
nb_pred = nb.predict(X_test)
nb_acc = accuracy_score(y_test, nb_pred)

# Print the accuracy of each algorithm
print('Logistic Regression Accuracy:', lr_acc)
print('SVM Accuracy:', svm_acc)
print('Random Forest Accuracy:', rf_acc)
print('KNN Accuracy:', knn_acc)
print('Naive Bayes Accuracy:', nb_acc)
```

```
Logistic Regression Accuracy: 0.7467532467532467
SVM Accuracy: 0.7662337662337663
Random Forest Accuracy: 0.7532467532467533
KNN Accuracy: 0.6623376623376623
Naive Bayes Accuracy: 0.7662337662337663
```