

MACHINE LEARNING

ASSIGNMENT 4

Name: Anjali Erra

Student ID:700740323

Video link:

https://drive.google.com/file/d/1mTTBpySCeifcWpVB8dcwyAMwTpGinKiv/view?usp=share_link

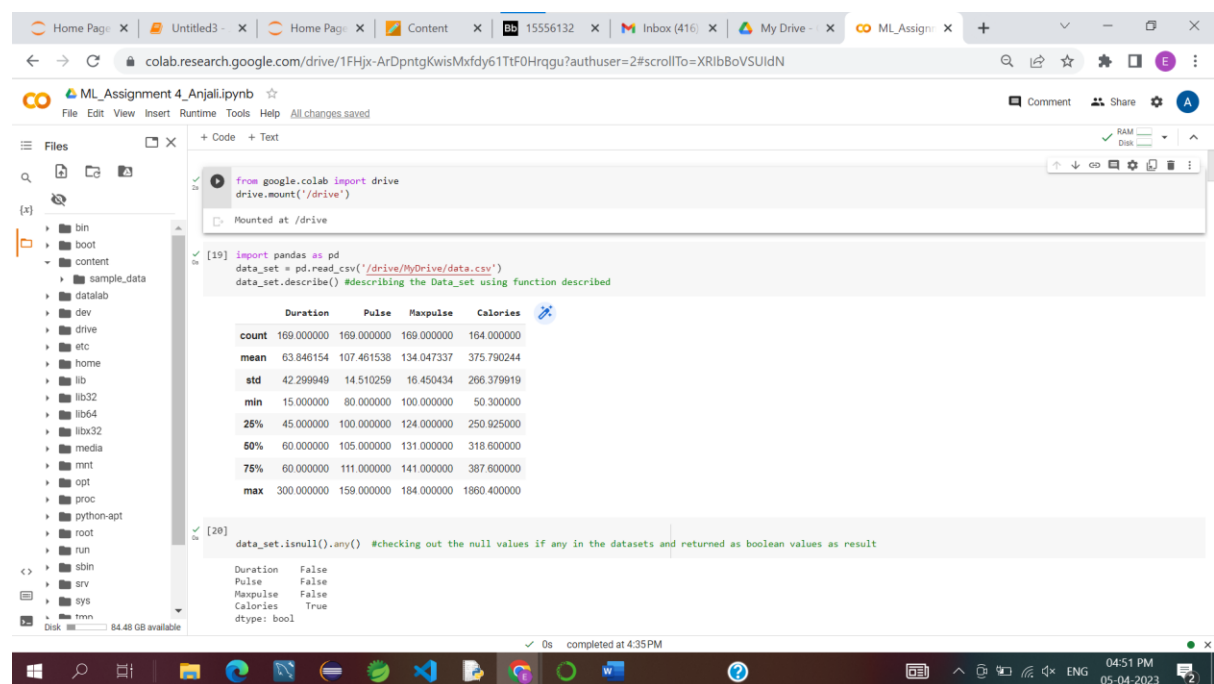
Github link: <https://github.com/Anjali555-erra/ML-Assignment4.git>

1.Pandas

1. Read the provided CSV file 'data.csv'.

<https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing>

2. Show the basic statistical description about the data.



The screenshot shows a Google Colab notebook titled 'ML_Assignment 4_Anjali.ipynb'. The notebook has two cells. The first cell contains the code to mount Google Drive and load a CSV file. The second cell contains the code to describe the data set. The output of the second cell is a statistical summary of the data set.

```
from google.colab import drive
drive.mount('/drive')

import pandas as pd
data_set = pd.read_csv('/drive/MyDrive/data.csv')
data_set.describe() #describing the Data_set using function described
```

	Duration	Pulse	Maxpulse	Calories
count	169.000000	169.000000	169.000000	164.000000
mean	63.846154	107.461538	134.047337	375.790244
std	42.299949	14.510259	16.450434	266.379919
min	15.000000	80.000000	100.000000	50.300000
25%	45.000000	100.000000	124.000000	250.925000
50%	60.000000	105.000000	131.000000	318.600000
75%	60.000000	111.000000	141.000000	387.600000
max	300.000000	159.000000	184.000000	1860.400000

```
data_set.isnull().any() #checking out the null values if any in the datasets and returned as boolean values as result
```

	Duration	Pulse	Maxpulse	Calories
Duration	False			
Pulse	False			
Maxpulse	False			
Calories	True			
dtype: bool				

3. Check if the data has null values.

a. Replace the null values with the mean

The screenshot shows a Google Colab notebook titled "ML_Assignment 4_Anjali.ipynb". The left sidebar displays a file explorer with a directory structure including "bin", "boot", "content", "sample_data", "datalab", "dev", "drive", "etc", "home", "lib", "lib32", "lib64", "libx32", "media", "mnt", "opt", "proc", and "python-apt". The main code area contains three cells:

- Cell [19]:

```
data_set.isnull().any() #checking out the null values if any in the datasets and returned as boolean values as result
```

	Duration	Pulse	Maxpulse	Calories
dtype: bool	False	False	False	True
- Cell [21]:

```
#replacing the null values using mean values  
data_set.fillna(data_set.mean(), inplace=True) # used inplace to make the changes in data to be persistence across this execution.  
data_set.isnull().any()
```

	Duration	Pulse	Maxpulse	Calories
dtype: bool	False	False	False	False
- Cell [22]:

```
data_set.agg({'Duration':['min','max','count','mean'],'Maxpulse':['min','max','count','mean']}) # used the agg function to aggregate the data
```

The bottom status bar indicates "0s completed at 4:35PM".

4. Select at least two columns and aggregate the data using: min, max, count, mean

The screenshot shows a Google Colab notebook with a file explorer on the left. The main code area contains cell [22]:

```
data_set.agg({'Duration':['min','max','count','mean'],'Maxpulse':['min','max','count','mean']})
```

	Duration	Maxpulse
min	15.000000	100.000000
max	300.000000	184.000000
count	169.000000	169.000000
mean	63.846154	134.047337

5. Filter the dataframe to select the rows with calories values between 500 and 1000.

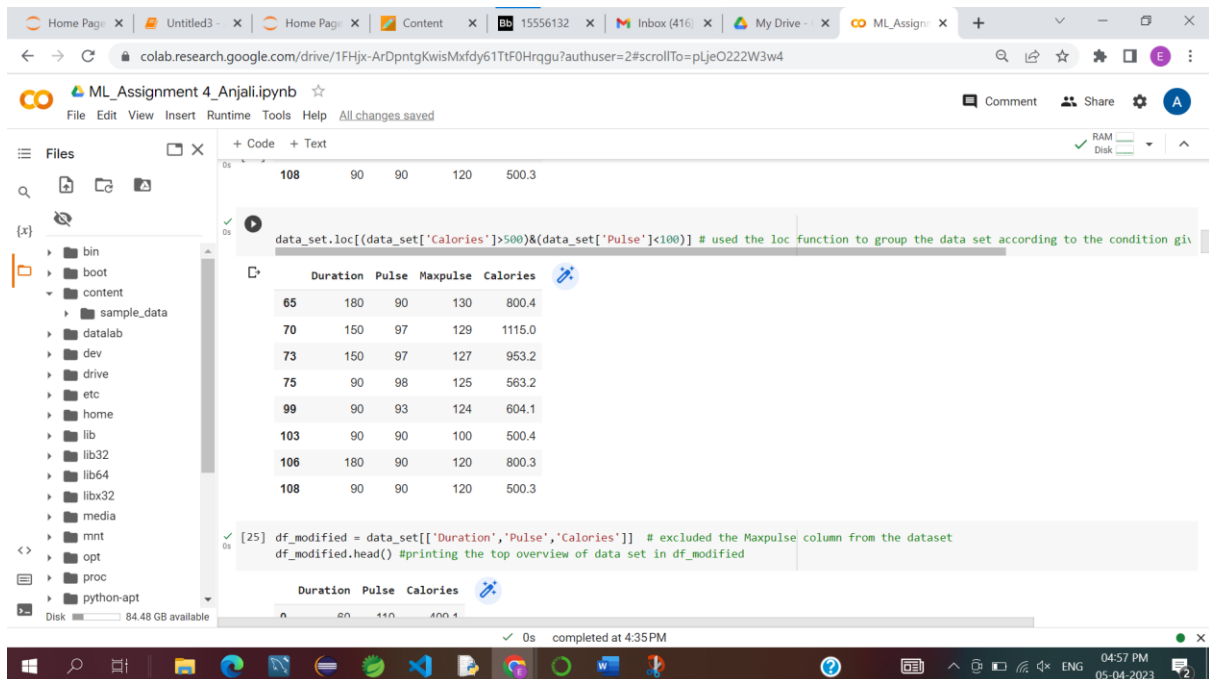
The screenshot shows a Google Colab notebook with a file explorer on the left. The main code area contains cell [23]:

```
data_set.loc[(data_set['Calories']>500)&(data_set['Calories']<1000)] # used the loc function to group the data set according to the condition
```

	Duration	Pulse	Maxpulse	Calories
51	80	123	146	643.1
62	160	109	135	853.0
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
72	90	100	127	700.0
73	150	97	127	953.2
75	90	98	125	563.2
78	120	100	130	500.4
90	180	101	127	600.1
99	90	93	124	604.1
103	90	90	100	500.4
106	180	90	120	800.3
108	90	90	120	500.3

The bottom status bar indicates "0s completed at 4:35PM".

6. Filter the dataframe to select the rows with calories values > 500 and pulse < 100 .



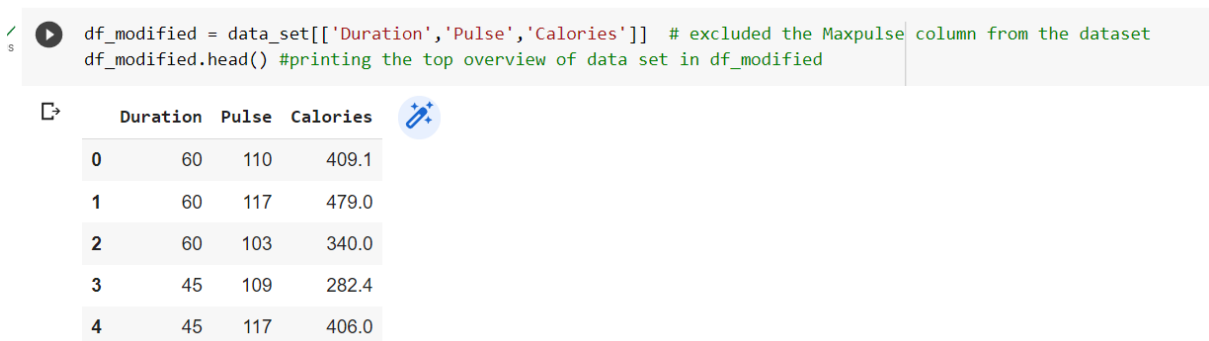
```
data_set.loc[(data_set['Calories']>500)&(data_set['Pulse']<100)] # used the loc function to group the data set according to the condition given
```

	Duration	Pulse	Maxpulse	Calories
65	180	90	130	800.4
70	150	97	129	1115.0
73	150	97	127	953.2
75	90	98	125	563.2
99	90	93	124	604.1
103	90	90	100	500.4
106	180	90	120	800.3
108	90	90	120	500.3

```
[25] df_modified = data_set[['Duration','Pulse','Calories']] # excluded the Maxpulse column from the dataset  
df_modified.head() #printing the top overview of data set in df_modified
```

	Duration	Pulse	Calories
0	60	110	409.1
1	60	117	479.0
2	60	103	340.0
3	45	109	282.4
4	45	117	406.0

7. Create a new “df_modified” dataframe that contains all the columns from df except for “Maxpulse”.



```
df_modified = data_set[['Duration','Pulse','Calories']] # excluded the Maxpulse column from the dataset  
df_modified.head() #printing the top overview of data set in df_modified
```

	Duration	Pulse	Calories
0	60	110	409.1
1	60	117	479.0
2	60	103	340.0
3	45	109	282.4
4	45	117	406.0

8. Delete the “Maxpulse” column from the main df dataframe

Home Page x Untitled3 - x Home Page x Content x 15556132 x Inbox (416) x My Drive - x ML_Assign x +

colab.research.google.com/drive/1FHjx-ArDpntgKwisMxfdy61TtF0Hrqu?authuser=2#scrollTo=pljeO222W3w4

ML_Assignment 4_Anjali.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

bin
boot
content
sample_data
datalab
dev
drive
etc
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
python-apt

RAM
Disk

0s

```
[26] del data_set['Maxpulse'] #deleting the label/column Maxpulse from the dataset
display (data_set)
```

	Duration	Pulse	Calories
0	60	110	409.1
1	60	117	479.0
2	60	103	340.0
3	45	109	282.4
4	45	117	406.0
...
164	60	105	290.8
165	60	110	300.0
166	60	115	310.2
167	75	120	320.4
168	75	125	330.4

169 rows x 3 columns

0s completed at 4:35 PM

05:01 PM
05-04-2023

9. Convert the datatype of Calories column to int datatype.

0s

```
[27] data_set.dtypes #before the conversion of data type for the value in data set
data_set['Calories'] = data_set['Calories'].astype(int) # applied conversion of data type for the
data_set.dtypes #display after conversion
```

```
Duration    int64
Pulse       int64
Calories    int64
dtype: object
```

10. Using pandas create a scatter plot for the two columns (Duration and Calories).

Home Page x Untitled3 - x Home Page x Content x 15556132 x Inbox (416) x My Drive - x ML_Assign x +

colab.research.google.com/drive/1FHjx-ArDpntgKwisMxfdy61TtF0Hrqu?authuser=2#scrollTo=2WK856DOYc3g

ML_Assignment 4_Anjali.ipynb

File Edit View Insert Runtime Tools Help All changes saved

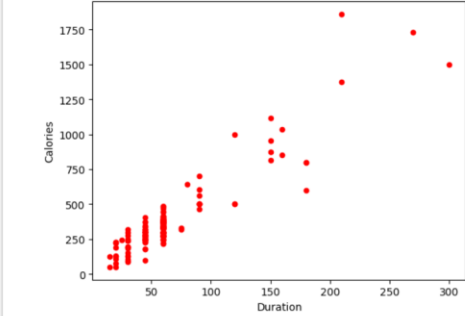
Files

bin
boot
content
sample_data
datalab
dev
drive
etc
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
python-apt

RAM
Disk

0s

```
data_set.plot.scatter(x='Duration',y='Calories',c='red') # displaying the pictorial plot for two column(Duration and calories)
scatter = ax.scatter(
    <Axes: xlabel='Duration', ylabel='Calories'>
```



0s

```
[29] # importing libraries all needed
```

0s completed at 4:35 PM

05:03 PM
05-04-2023

1. (Titanic Dataset)

1. Find the correlation between 'survived' (target column) and 'sex' column for the Titanic use case inclclass.

```
[29] # importing libraries all needed
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
import matplotlib.pyplot as plt
df=pd.read_csv("/drive/MyDrive/train.csv")
# Below we are Finding the correlation between 'survived' (target column) and 'sex' column for the Titanic use case

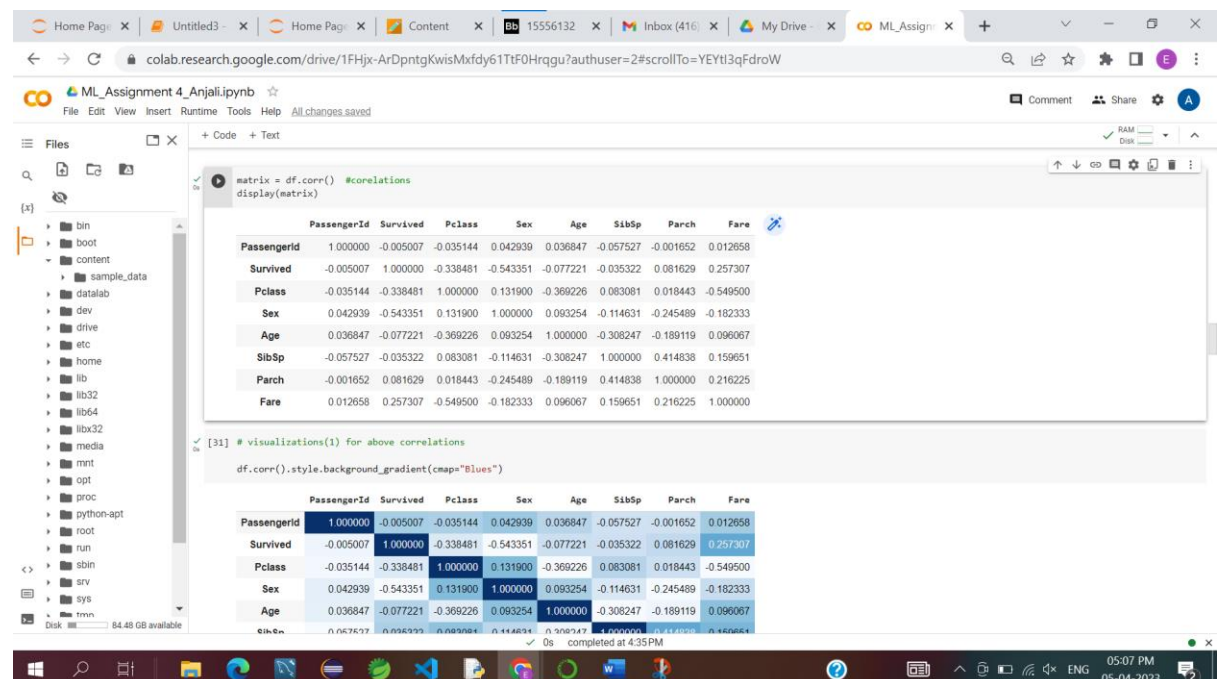
le = preprocessing.LabelEncoder()
df['Sex'] = le.fit_transform(df.Sex.values)
df['Survived'].corr(df['Sex'])

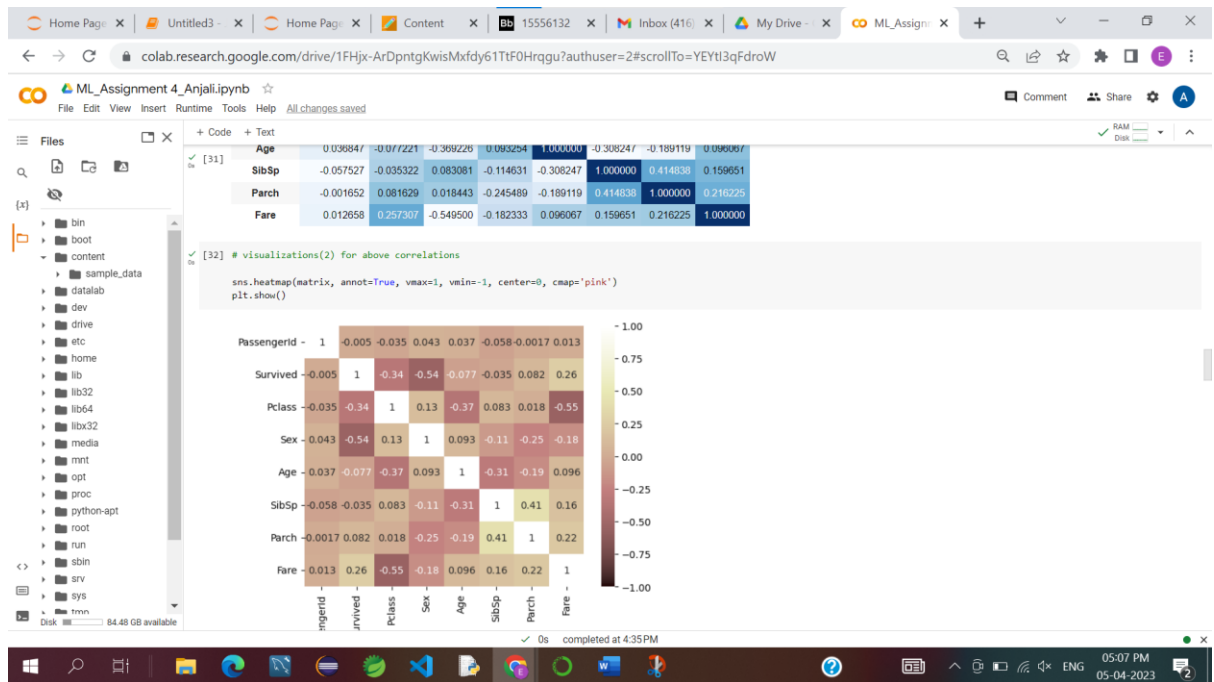
-0.543351380657755
```

a. Do you think we should keep this feature?

Yes, as By using correlation to demonstrate which variable has a high or low correlation with another variable, we can plot correlation matrices to show the strength of the relationship between the dependent and independent variables/columns.

2. Do at least two visualizations to describe or show correlations.





3. Implement Naïve Bayes method using scikit-learn library and report the accuracy.

The screenshot shows a Google Colab notebook titled "ML_Assignment 4_Anjali.ipynb". The code cell [42] shows the implementation of the Naïve Bayes method using scikit-learn. The code is as follows:

```

[42] # Implementing Naive Bayes methods ,,,1.3.1

train_raw = pd.read_csv('/drive/MyDrive/train.csv')
test_raw = pd.read_csv('/drive/MyDrive/test.csv')

# Join data to analyse and process the set as one.
train_raw['train'] = 1
test_raw['train'] = 0
df = train_raw.append(test_raw, sort=False)

features = ['Age', 'Embarked', 'Fare', 'Parch', 'Pclass', 'Sex', 'SibSp']
target = 'Survived'

df = df[features + [target] + ['train']]
# Categorical values need to be transformed into numeric.
df['Sex'] = df['Sex'].replace(['female', 'male'], [0, 1])
df['Embarked'] = df['Embarked'].replace(['S', 'C', 'Q'], [1, 2, 3])
train = df.query('train == 1')
test = df.query('train == 0')

<ipython-input-42-47218a524e76>:9: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
df = train_raw.append(test_raw, sort=False)

[43] train.dropna(axis=0, inplace=True)

```

Accuracy

```

#1.3.2
classifier = GaussianNB()
classifier.fit(X_train, Y_train)
y_pred = classifier.predict(X_val)

# Summary of the predictions made by the classifier
print(classification_report(Y_val, y_pred))
print(confusion_matrix(Y_val, y_pred))

# Accuracy score
from sklearn.metrics import accuracy_score
display('accuracy is', accuracy_score(Y_val, y_pred))

```

	precision	recall	f1-score	support
1	0.90	0.95	0.92	19
2	0.92	0.92	0.92	12
3	1.00	0.50	0.67	6
5	0.00	0.00	0.00	1
6	1.00	1.00	1.00	1
7	0.75	0.75	0.75	4
accuracy			0.84	43
macro avg	0.76	0.69	0.71	43
weighted avg	0.89	0.84	0.85	43

```

[[18 1 0 0 0 0]
 [ 1 11 0 0 0 0]
 [ 1 0 3 2 0 0]
 [ 0 0 0 0 1]
 [ 0 0 0 1 0]
 [ 0 0 1 0 3]]
"accuracy is"
0.8372893823255814

```

2. (Glass Dataset)

1. Implement Naïve Bayes method using scikit-learn library.

- Use the glass dataset available in Link also provided in your assignment.
- Use `train_test_split` to create training and testing part.

```

[49] #question 2 starts # Reading file from glass csv data file from current directory
glass=pd.read_csv("/drive/MyDrive/glass.csv")

```

```

#2.1.b
#2.2
features = ['R1', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']
target = 'Type'

X_train, X_val, Y_train, Y_val = train_test_split(glass[features], glass['Type'], test_size=0.2, random_state=1)

classifier = GaussianNB()
classifier.fit(X_train, Y_train)
y_pred = classifier.predict(X_val)

# Summarizing the predictions made by the classifier
print(classification_report(Y_val, y_pred))
print(confusion_matrix(Y_val, y_pred))

# Accuracy score
from sklearn.metrics import accuracy_score
display('accuracy is', accuracy_score(Y_val, y_pred))

```

	precision	recall	f1-score	support
1	0.90	0.95	0.92	19
2	0.92	0.92	0.92	12
3	1.00	0.50	0.67	6
5	0.00	0.00	0.00	1
6	1.00	1.00	1.00	1
7	0.75	0.75	0.75	4
accuracy			0.84	43
macro avg	0.76	0.69	0.71	43
weighted avg	0.89	0.84	0.85	43

```

[[18 1 0 0 0 0]
 [ 1 11 0 0 0 0]
 [ 1 0 3 2 0 0]
 [ 0 0 0 0 1]
 [ 0 0 1 0 3]]

```

2. Evaluate the model on testing part using `classification_report(y_true, y_pred)`

1. Implement linear SVM method using scikit library
 - a. Use the glass dataset available in Link also provided in your assignment.
 - b. Use train_test_split to create training and testing part
2. Evaluate the model on testing part using score and

The screenshot shows a Google Colab notebook titled 'ML_Assignment 4_Anjali.ipynb'. The code in cell [51] implements a Linear SVM classifier using the sklearn library. It loads the 'glass' dataset, splits it into training and testing sets, trains the classifier, and evaluates its performance. The output shows the accuracy score and a confusion matrix.

```
[51] #2nd part 2nd question
from sklearn.svm import SVC, LinearSVC

classifier = LinearSVC()

classifier.fit(X_train, Y_train)

y_pred = classifier.predict(X_val)

# Summary of the predictions made by the classifier
print(classification_report(Y_val, y_pred))
print(confusion_matrix(Y_val, y_pred))
# Accuracy score
from sklearn.metrics import accuracy_score
display('accuracy is', accuracy_score(Y_val, y_pred))
```

The output of the code is as follows:

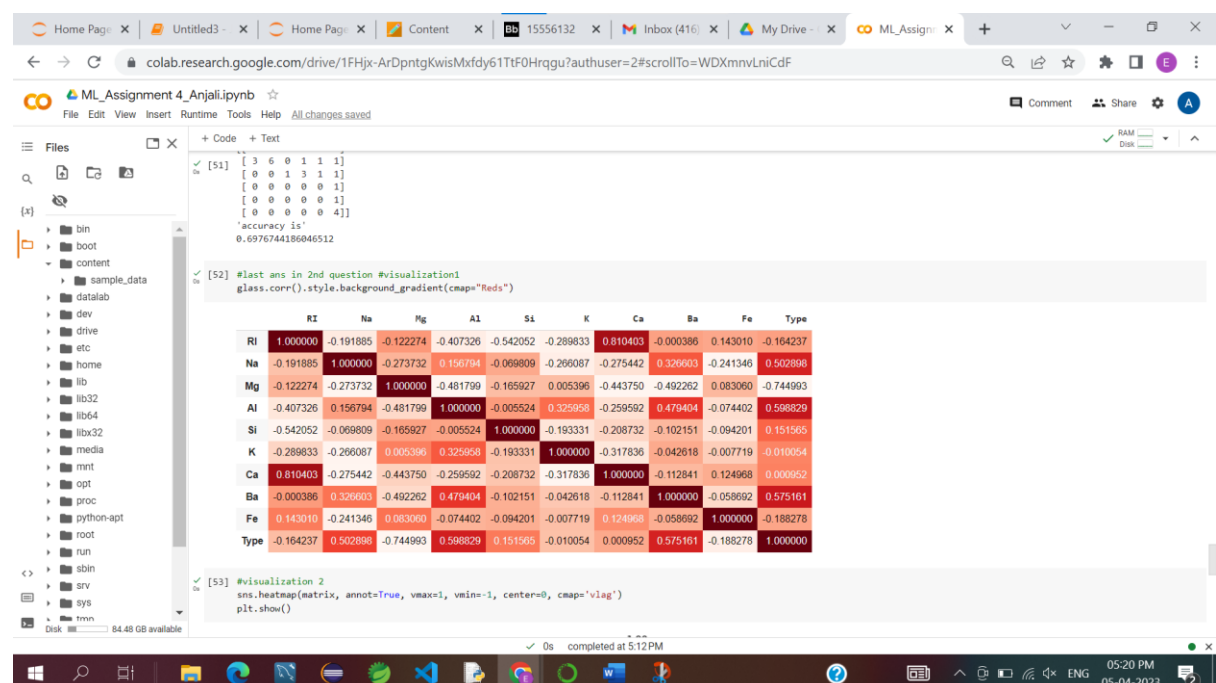
```
precision    recall  f1-score   support

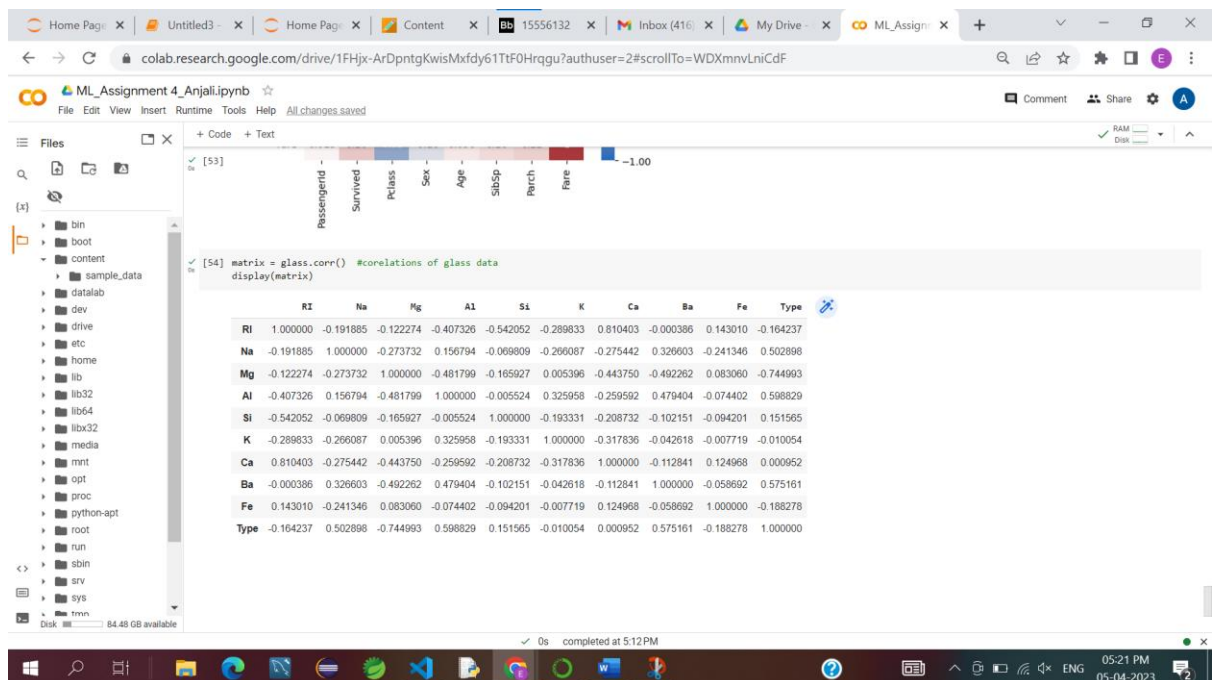
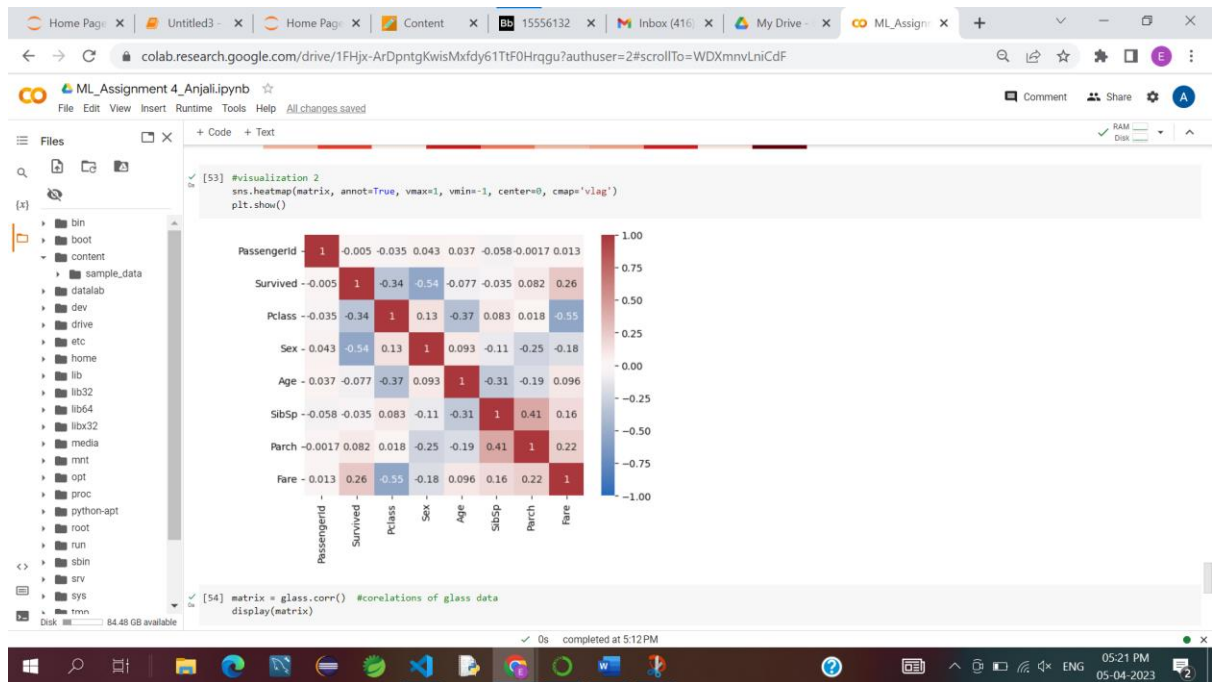
     1       0.86       1.00       0.93        19
     2       1.00       0.50       0.67        12
     3       1.00       0.17       0.29         6
     5       0.00       0.00       0.00         1
     6       0.00       0.00       0.00         1
     7       0.50       1.00       0.67         4

 accuracy          0.56          0.44          0.42          43
 macro avg          0.56          0.44          0.42          43
 weighted avg          0.56          0.44          0.42          43

[[19  0  0  0  0  0]
 [ 3  6  0  1  1  1]
 [ 0  0  1  3  1  1]
 [ 0  0  0  0  1]
 [ 0  0  0  0  1]
 [ 0  0  0  0  4]]
```

Do at least two visualizations to describe or show correlations in the Glass Dataset.





Which algorithm you got better accuracy? Can you justify why?

Ans: Naïve Bayes

Logistic regression and SVM are discriminative models, but the Multinomial "Naive Bayes" in this case has higher accuracy in comparison to the SVM algorithm from the executions above. While Naive Bayes is a generative model, logistic regression and SVM are discriminative models. It is well known that a generative model can outperform a discriminative model when you have very little data, taking into account the parameter (Dataset) sensitivity and implementation pattern.