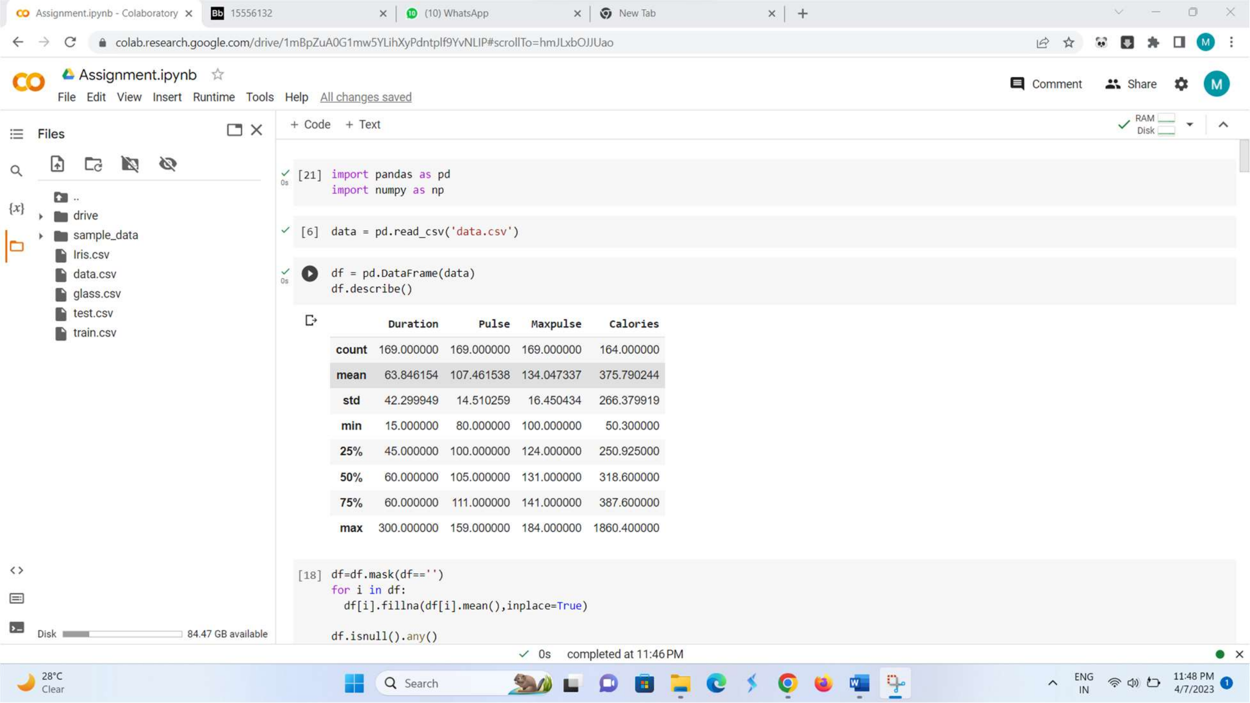
Name: Mythresh Maddina

700: 700741162

Github Link: https://github.com/MythreshM/CS5710_Assignment4

Question1:



```
[21] import pandas as pd
     import numpy as np
```

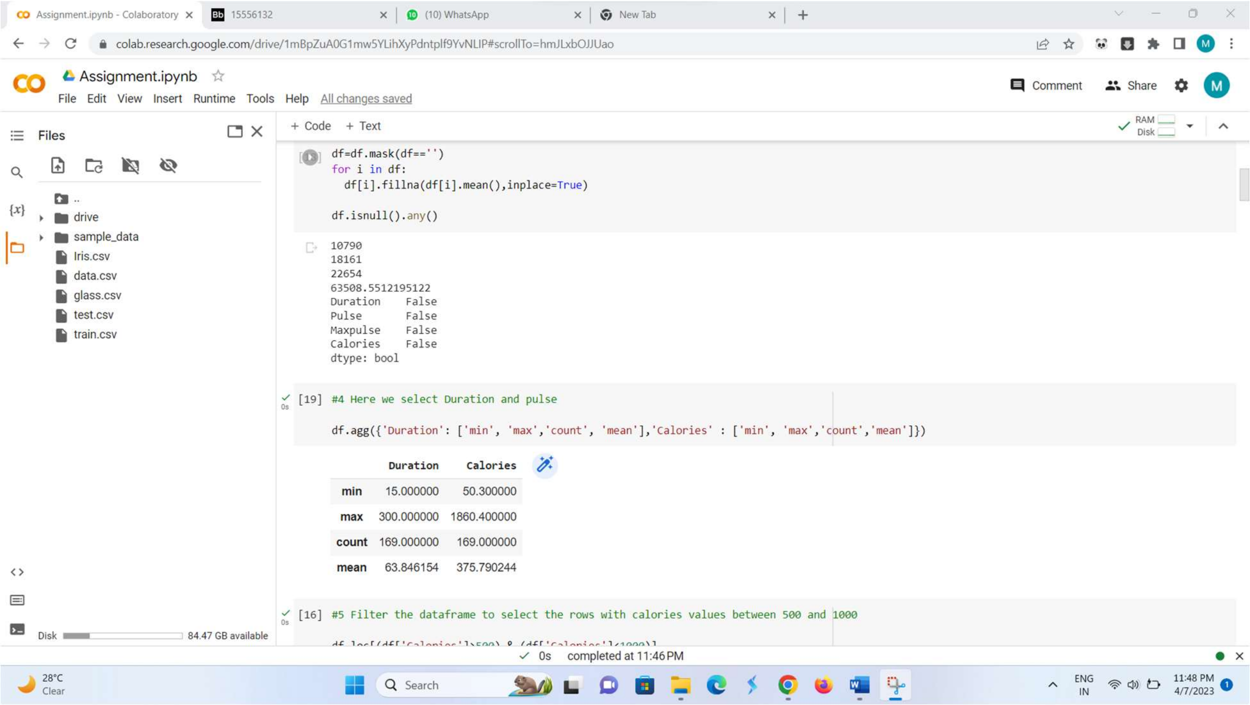
```
[6]  data = pd.read_csv('data.csv')
```

```
     df = pd.DataFrame(data)
     df.describe()
```

```
         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
```

```
[18] df=df.mask(df=='')
     for i in df:
       df[i].fillna(df[i].mean(),inplace=True)

     df.isnull().any()
```

✓ 0s completed at 11:46 PM


Assignment.ipynb - Colaboratory
15556132
(10) WhatsApp
New Tab
colab.research.google.com/drive/1mBpZuA0G1mw5YLihXyPdntplf9YvNLIP#scrollTo=hmJLxbOJJUao
Assignment.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
..
drive
sample_data
Iris.csv
data.csv
glass.csv
test.csv
train.csv
Disk 84.47 GB available
✓ 0s completed at 11:46 PM
28°C
Clear
Search
ENG
IN
11:48 PM
4/7/2023

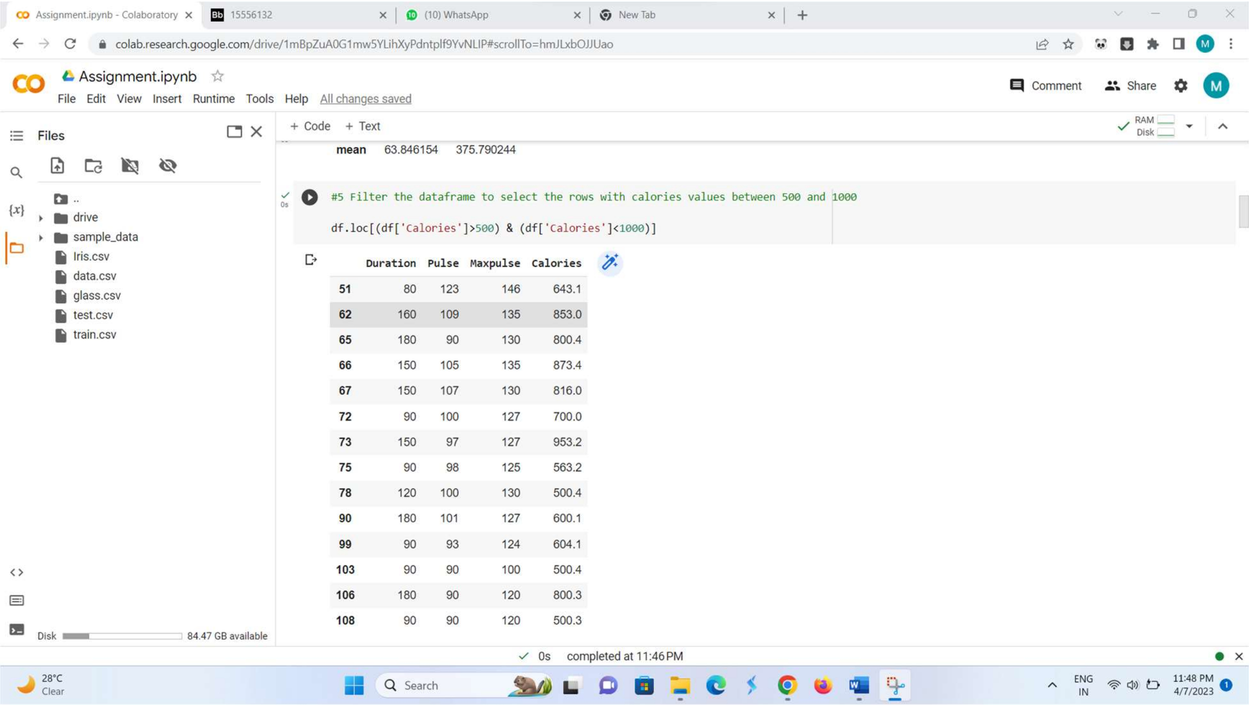
```
df=df.mask(df=='')
for i in df:
  df[i].fillna(df[i].mean(),inplace=True)

df.isnull().any()
```

```
10790
18161
22654
63508.5512195122
Duration    False
Pulse       False
Maxpulse    False
Calories    False
dtype: bool
```

```
[19] #4 Here we select Duration and pulse

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

	Duration	Calories
min	15.000000	50.300000
max	300.000000	1860.400000
count	169.000000	169.000000
mean	63.846154	375.790244

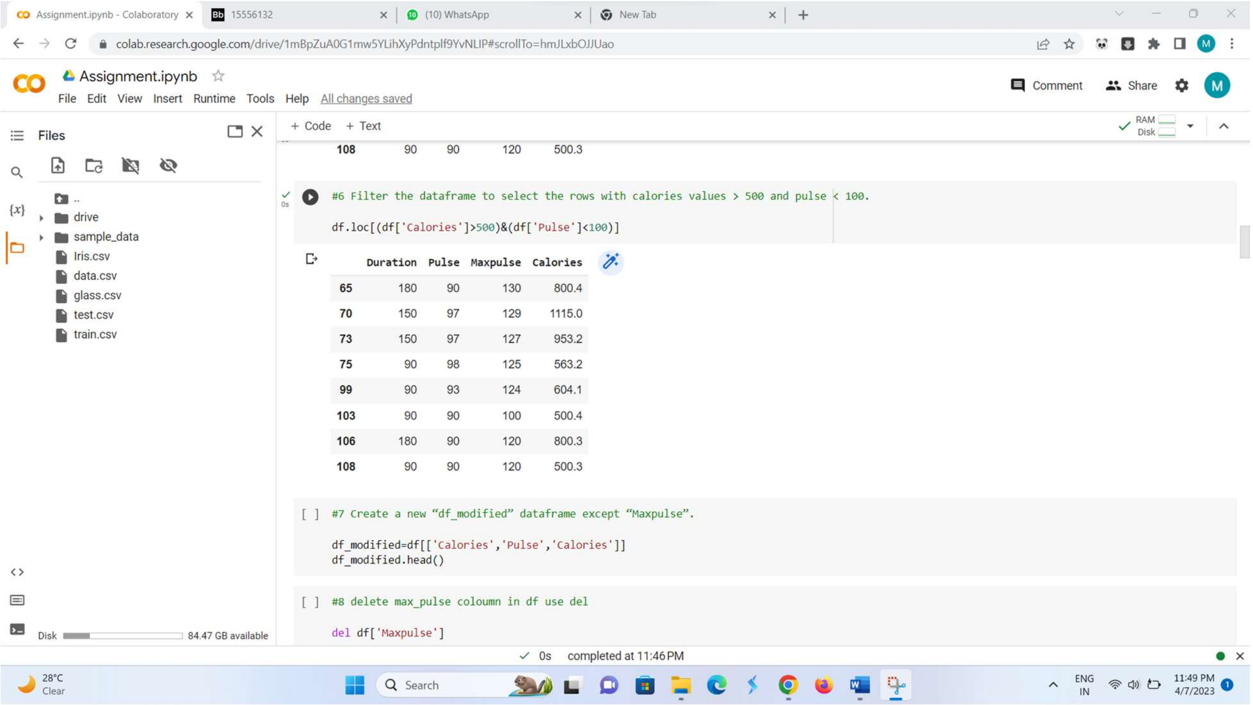
```
[16] #5 Filter the dataframe to select the rows with calories values between 500 and 1000

df.loc[(df['Calories']>500) & (df['Calories']<1000)]
```


Assignment.ipynb - Colaboratory
15556132
(10) WhatsApp
New Tab
colab.research.google.com/drive/1mBpZuA0G1mw5YLihXyPdntplf9YvNLIP#scrollTo=hmJLxbOJJUao
Assignment.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
..
drive
sample_data
Iris.csv
data.csv
glass.csv
test.csv
train.csv
Disk 84.47 GB available
✓ 0s completed at 11:46 PM
28°C
Clear
Search
ENG
IN
11:48 PM
4/7/2023

mean	63.846154	375.790244

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

df.loc[(df['Calories']>500) & (df['Calories']<1000)]
```

	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


Assignment.ipynb - Colaboratory
15556132
(10) WhatsApp
New Tab
colab.research.google.com/drive/1mBpZuA0G1mw5YLihXyPdntplf9YvNLIP#scrollTo=hmJLxbOJJUao
Assignment.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
drive
sample_data
Iris.csv
data.csv
glass.csv
test.csv
train.csv
Disk 84.47 GB available
✓ 0s completed at 11:46 PM
28°C
Clear
Search
ENG
IN
11:49 PM
4/7/2023

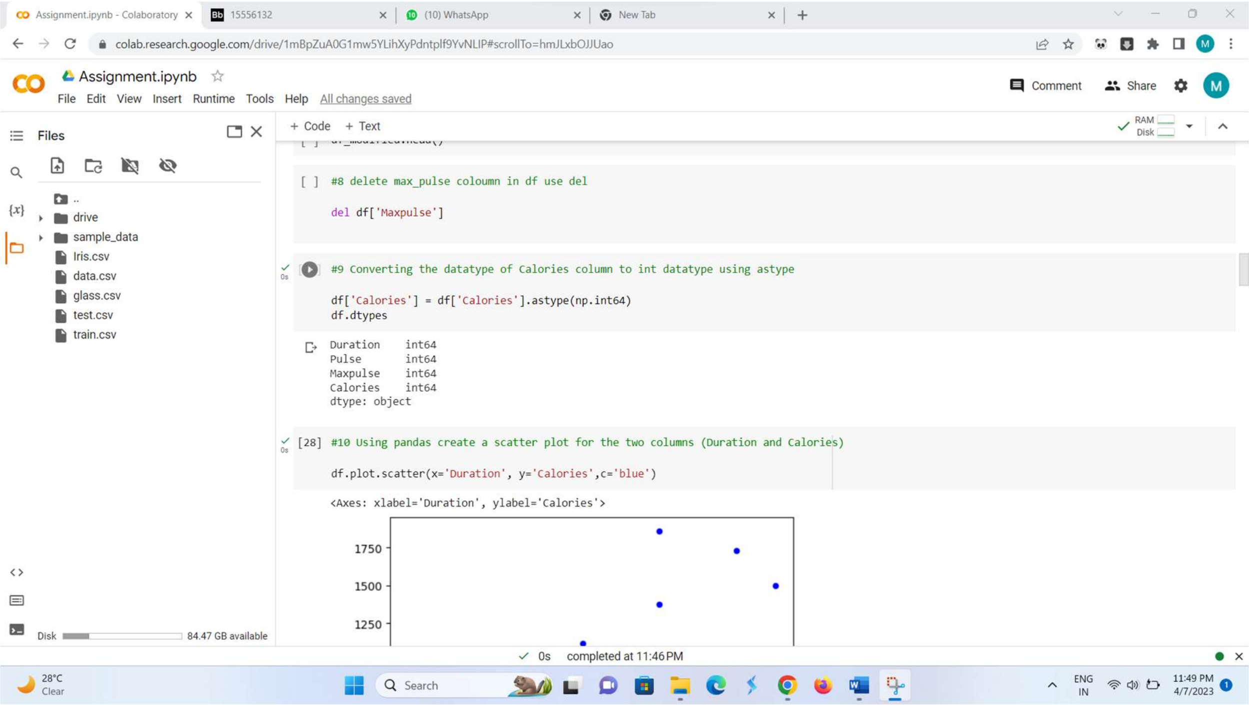
```
108     90     90     120     500.3
```

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

df.loc[(df['Calories']>500)&(df['Pulse']<100)]
```

	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

```
[ ] #7 Create a new “df_modified” dataframe except “Maxpulse”.

    df_modified=df[['Calories','Pulse','Calories']]
    df_modified.head()
```

```
[ ] #8 delete max_pulse coloumn in df use del

    del df['Maxpulse']
```


Assignment.ipynb - Colaboratory
15556132
(10) WhatsApp
New Tab
colab.research.google.com/drive/1mBpZuA0G1mw5YLihXyPdntplf9YvNLIP#scrollTo=hmJLxbOJJUao
Assignment.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
drive
sample_data
Iris.csv
data.csv
glass.csv
test.csv
train.csv
Disk 84.47 GB available
✓ 0s completed at 11:46 PM
28°C
Clear
Search
ENG
IN
11:49 PM
4/7/2023

```
[ ] #8 delete max_pulse coloumn in df use del

    del df['Maxpulse']
```

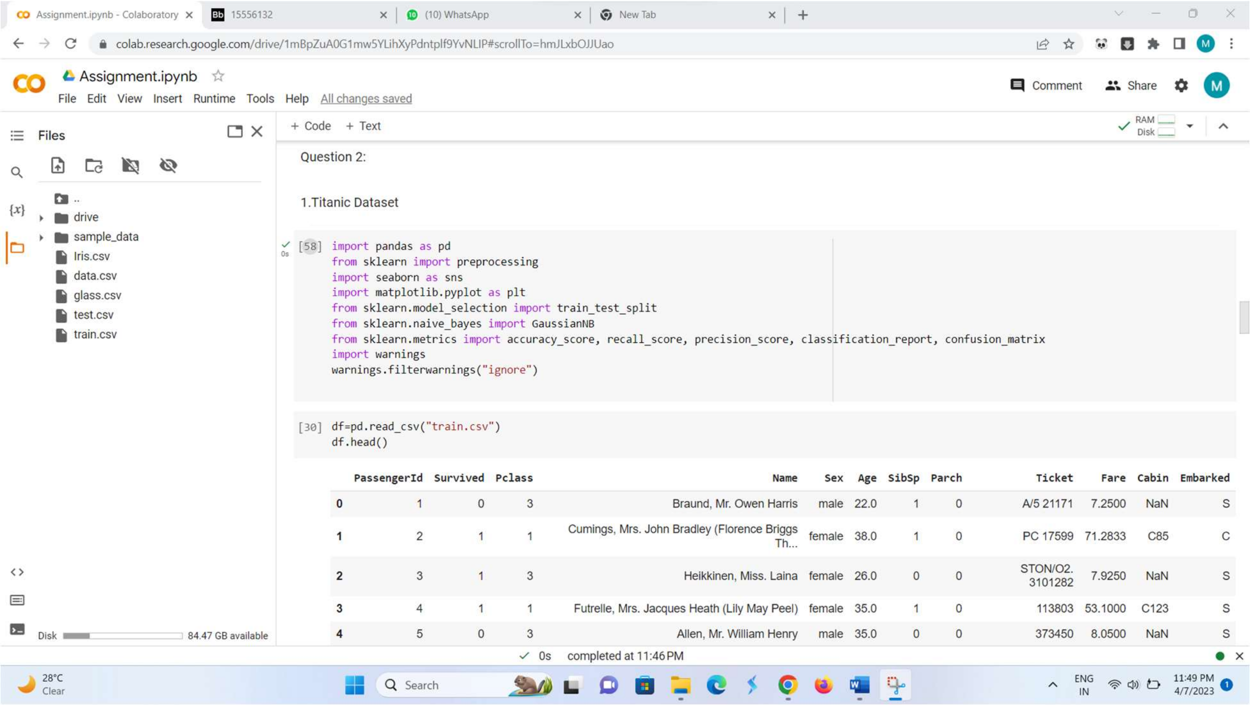
```
#9 Converting the datatype of Calories column to int datatype using astype

df['Calories'] = df['Calories'].astype(np.int64)
df.dtypes
```

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

```
[28] #10 Using pandas create a scatter plot for the two columns (Duration and Calories)

     df.plot.scatter(x='Duration', y='Calories',c='blue')
```

```
<Axes: xlabel='Duration', ylabel='Calories'>
```

Question2:



Question 2:

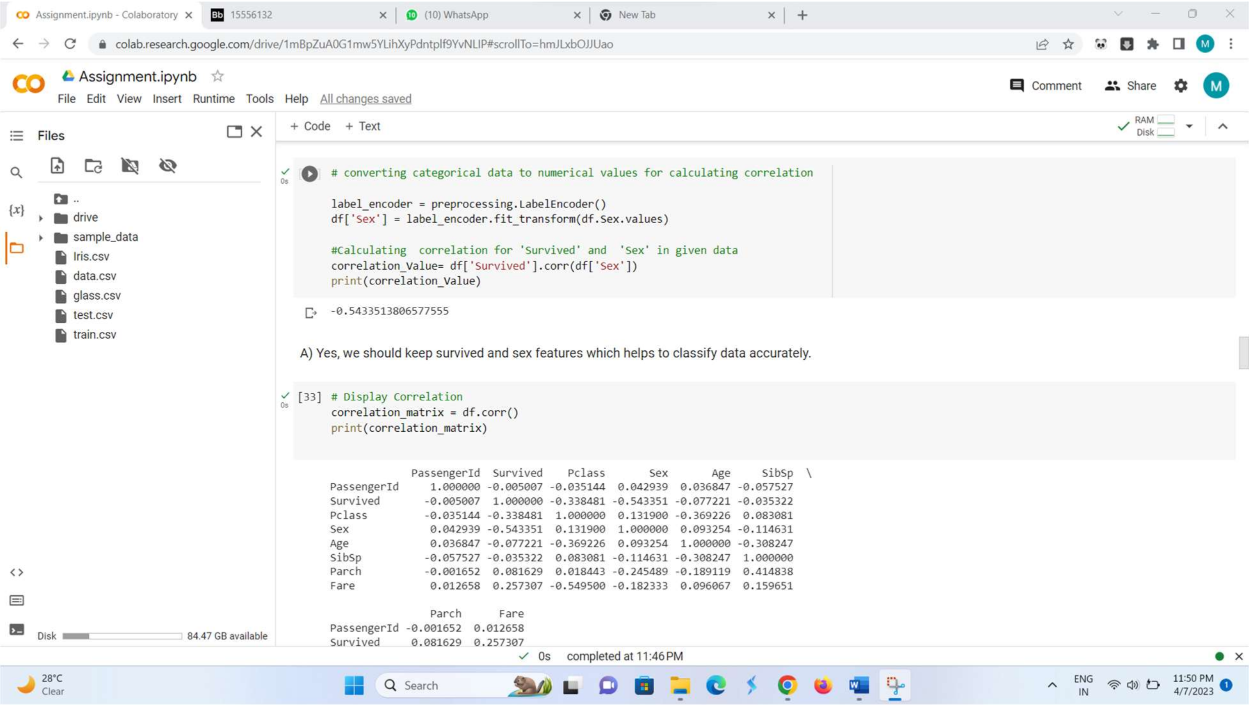
1.Titanic Dataset

```
[58] import pandas as pd
     from sklearn import preprocessing
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.naive_bayes import GaussianNB
     from sklearn.metrics import accuracy_score, recall_score, precision_score, classification_report, confusion_matrix
     import warnings
     warnings.filterwarnings("ignore")
```

```
[30] df=pd.read_csv("train.csv")
     df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

✓ 0s completed at 11:46 PM



```
# converting categorical data to numerical values for calculating correlation

label_encoder = preprocessing.LabelEncoder()
df['Sex'] = label_encoder.fit_transform(df.Sex.values)

#Calculating  correlation for 'Survived' and  'Sex' in given data
correlation_Value= df['Survived'].corr(df['Sex'])
print(correlation_Value)
```

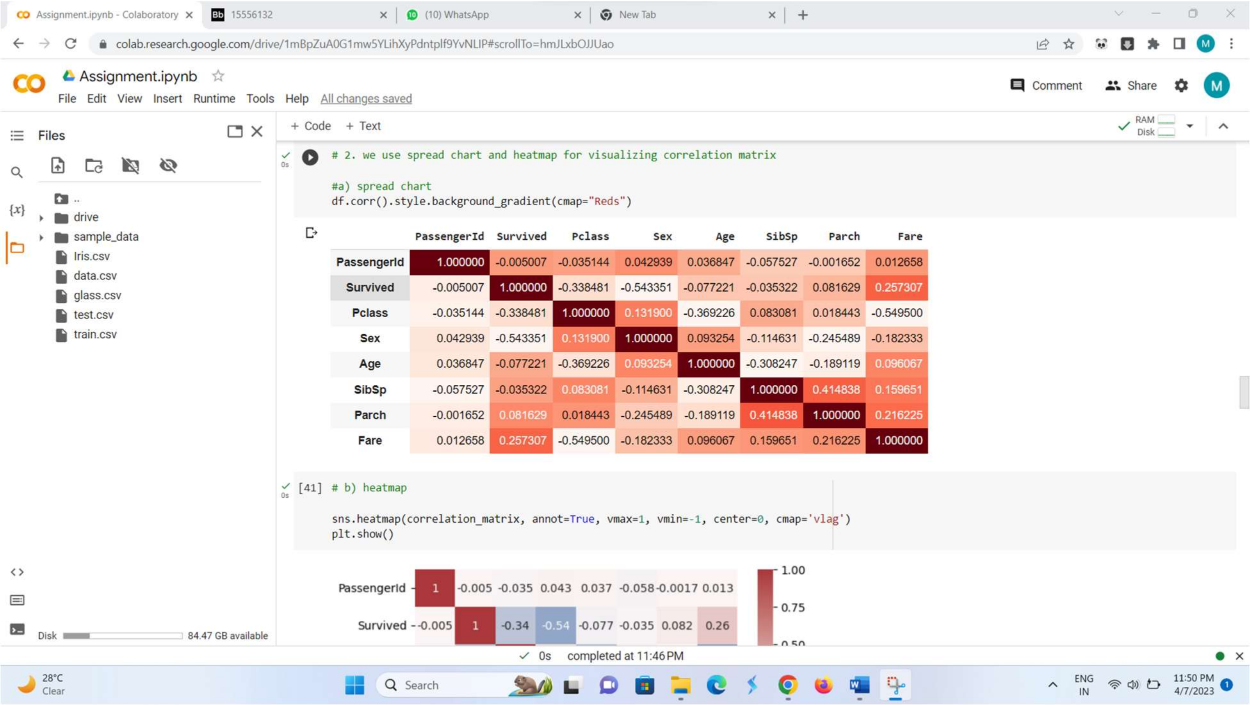
```
-0.5433513806577555
```

A) Yes, we should keep survived and sex features which helps to classify data accurately.

```
[33] # Display Correlation
     correlation_matrix = df.corr()
     print(correlation_matrix)
```

```
             PassengerId  Survived    Pclass       Sex       Age     SibSp  \
PassengerId     1.000000 -0.005007 -0.035144  0.042939  0.036847 -0.057527
Survived       -0.005007  1.000000 -0.338481 -0.543351 -0.077221 -0.035322
Pclass         -0.035144 -0.338481  1.000000  0.131900 -0.369226  0.083081
Sex             0.042939 -0.543351  0.131900  1.000000  0.093254 -0.114631
Age             0.036847 -0.077221 -0.369226  0.093254  1.000000 -0.308247
SibSp          -0.057527 -0.035322  0.083081 -0.114631 -0.308247  1.000000
Parch          -0.001652  0.081629  0.018443 -0.245489 -0.189119  0.414838
Fare            0.012658  0.257307 -0.549500 -0.182333  0.096067  0.159651

                Parch      Fare
PassengerId -0.001652  0.012658
Survived     0.081629  0.257307
```

✓ 0s completed at 11:46 PM

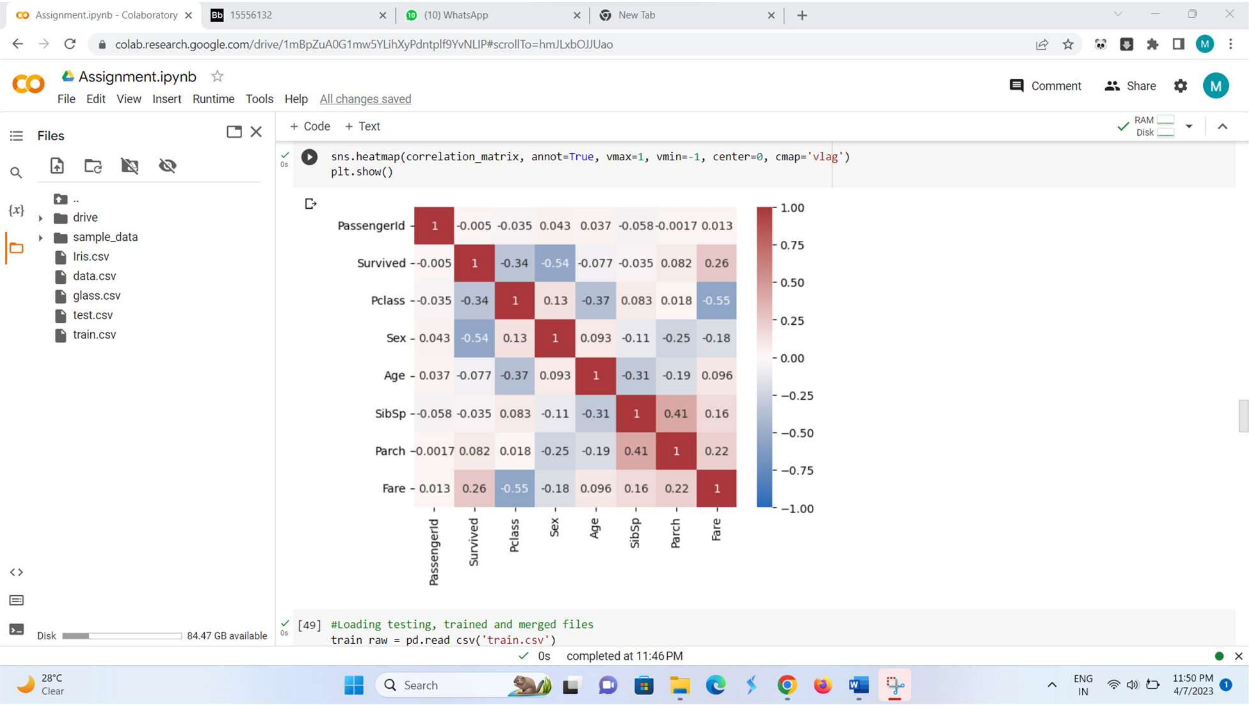


```
# 2. we use spread chart and heatmap for visualizing correlation matrix

#a) spread chart
df.corr().style.background_gradient(cmap="Reds")
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.042939	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.543351	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	0.131900	-0.369226	0.083081	0.018443	-0.549500
Sex	0.042939	-0.543351	0.131900	1.000000	0.093254	-0.114631	-0.245489	-0.182333
Age	0.036847	-0.077221	-0.369226	0.093254	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.114631	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.245489	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	-0.182333	0.096067	0.159651	0.216225	1.000000

```
[41] # b) heatmap

sns.heatmap(correlation_matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')
plt.show()
```

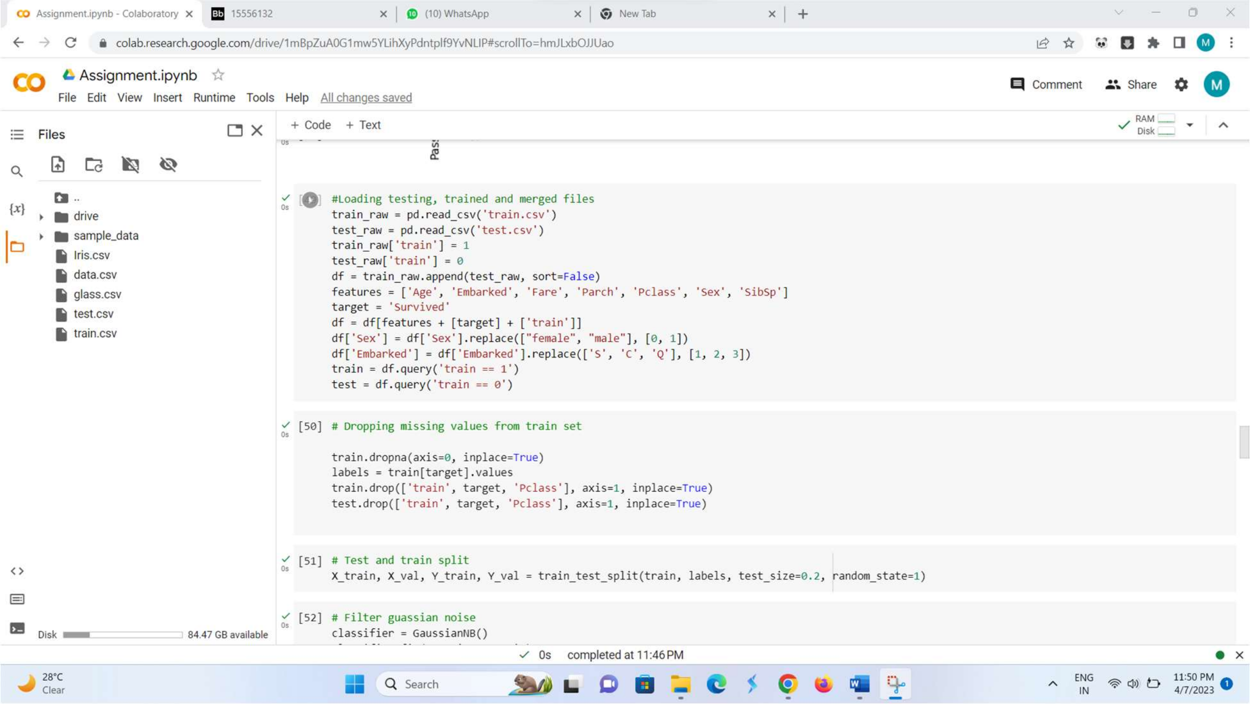
PassengerId	1	-0.005	-0.035	0.043	0.037	-0.058	-0.0017	0.013
Survived	-0.005	1	-0.34	-0.54	-0.077	-0.035	0.082	0.26



```
sns.heatmap(correlation_matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')
plt.show()
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
PassengerId	1	-0.005	-0.035	0.043	0.037	-0.058	-0.0017	0.013
Survived	-0.005	1	-0.34	-0.54	-0.077	-0.035	0.082	0.26
Pclass	-0.035	-0.34	1	0.13	-0.37	0.083	0.018	-0.55
Sex	0.043	-0.54	0.13	1	0.093	-0.11	-0.25	-0.18
Age	0.037	-0.077	-0.37	0.093	1	-0.31	-0.19	0.096
SibSp	-0.058	-0.035	0.083	-0.11	-0.31	1	0.41	0.16
Parch	-0.0017	0.082	0.018	-0.25	-0.19	0.41	1	0.22
Fare	0.013	0.26	-0.55	-0.18	0.096	0.16	0.22	1

```
[49] #Loading testing, trained and merged files
train raw = pd.read csv('train.csv')
```


Assignment.ipynb - Colaboratory
15556132
(10) WhatsApp
New Tab
colab.research.google.com/drive/1mBpZuA0G1mw5YLihXyPdntplf9YvNLIP#scrollTo=hmJLxbOJJUao
Assignment.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
..
drive
sample_data
Iris.csv
data.csv
glass.csv
test.csv
train.csv
Disk 84.47 GB available
Pas

```
#Loading testing, trained and merged files
train_raw = pd.read_csv('train.csv')
test_raw = pd.read_csv('test.csv')
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']]
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')
```

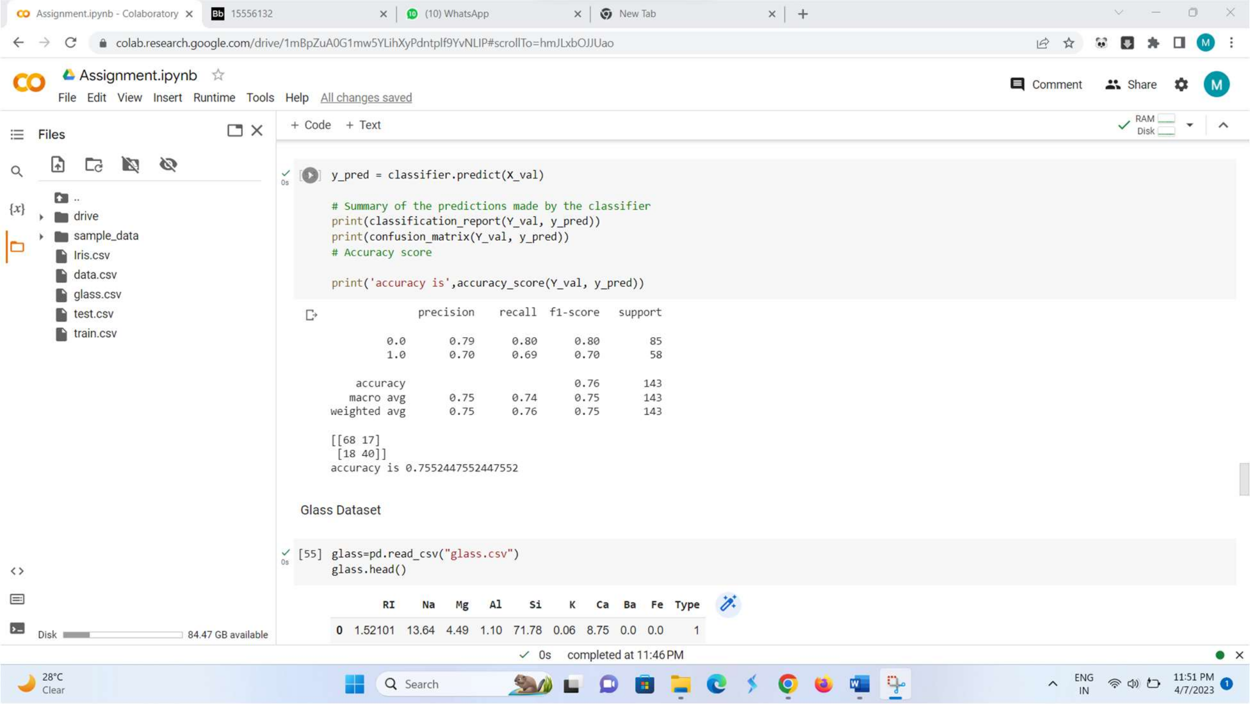
```
[50] # Dropping missing values from train set

     train.dropna(axis=0, inplace=True)
     labels = train[target].values
     train.drop(['train', target, 'Pclass'], axis=1, inplace=True)
     test.drop(['train', target, 'Pclass'], axis=1, inplace=True)
```

```
[51] # Test and train split
     X_train, X_val, Y_train, Y_val = train_test_split(train, labels, test_size=0.2, random_state=1)
```

```
[52] # Filter guassian noise
     classifier = GaussianNB()
```

✓ 0s completed at 11:46 PM

Question 3:



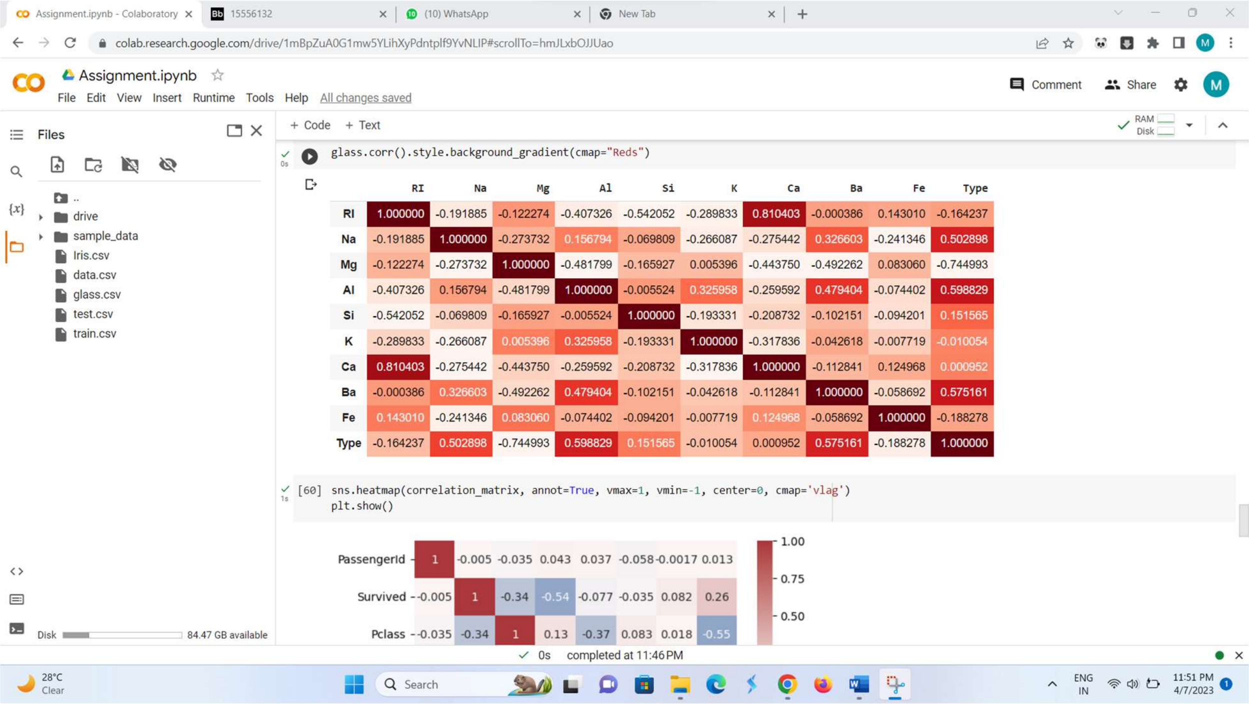
```
y_pred = classifier.predict(X_val)

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

print('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.79      0.80      0.80        85
         1.0       0.70      0.69      0.70        58

    accuracy                           0.76       143
   macro avg       0.75      0.74      0.75       143
weighted avg       0.75      0.76      0.75       143

[[68 17]
 [18 40]]
accuracy is 0.7552447552447552
```

Glass Dataset

```
[55] glass=pd.read_csv("glass.csv")
     glass.head()
```

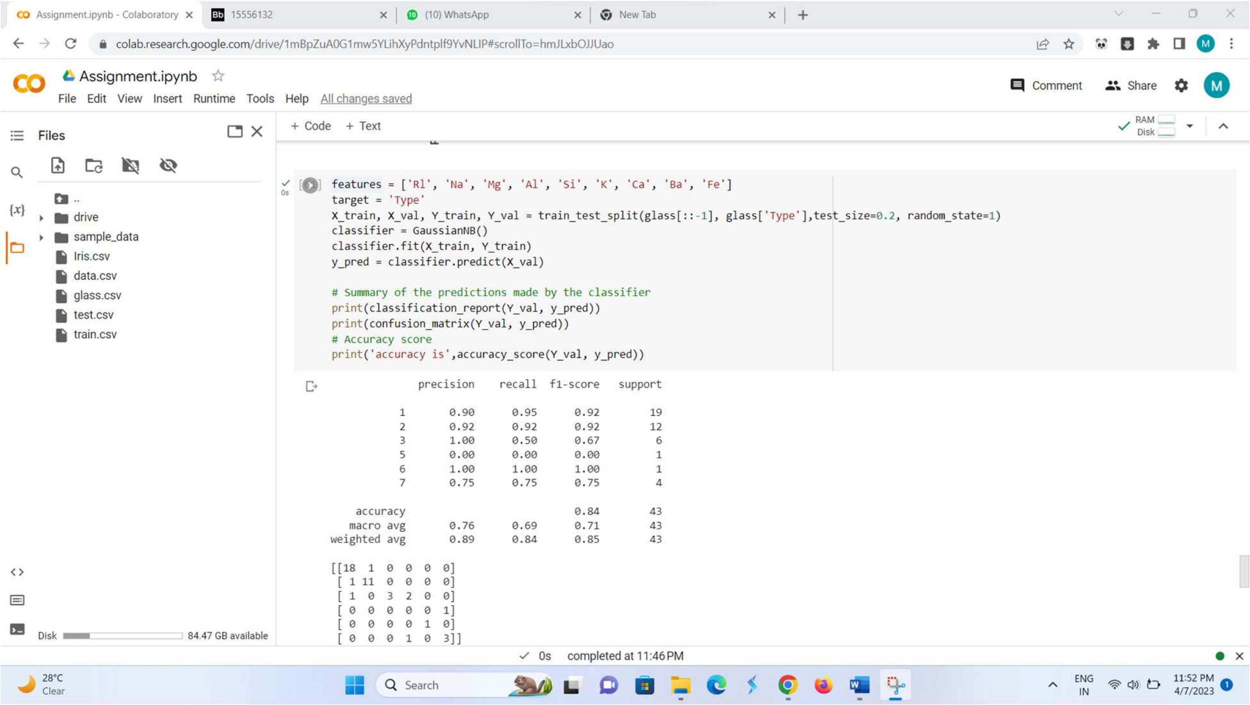
	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1



```
glass.corr().style.background_gradient(cmap="Reds")
```

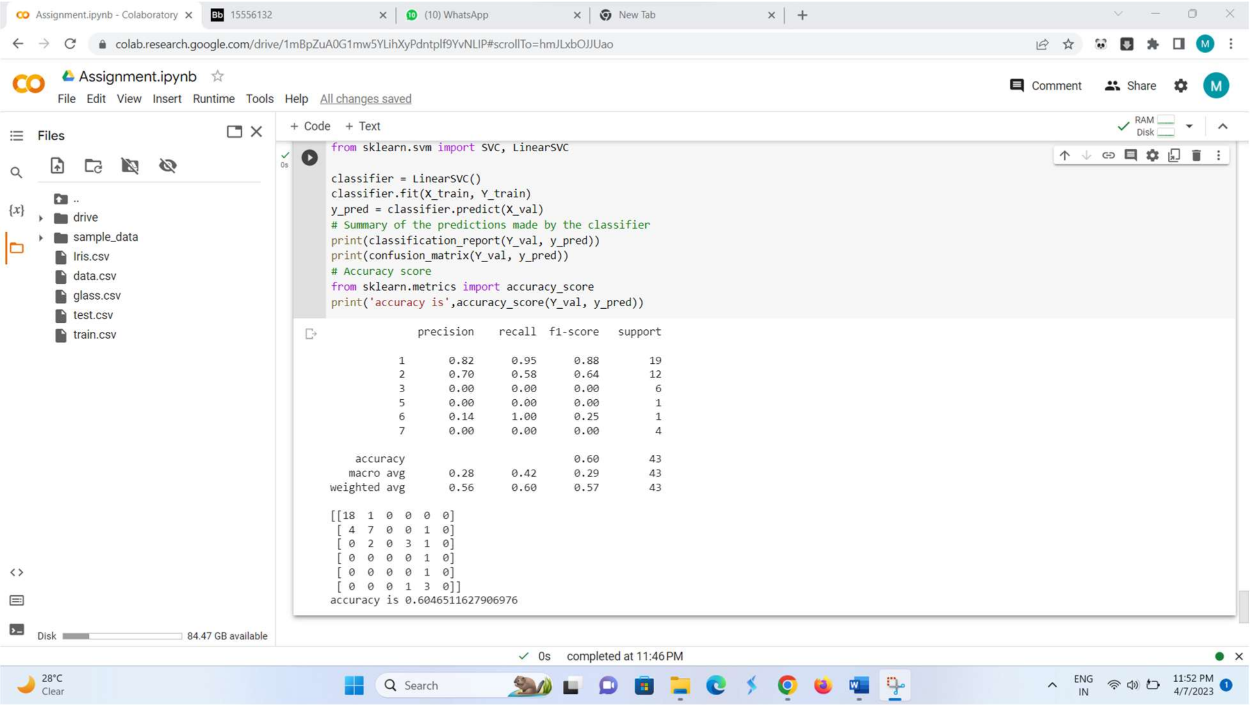
	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
RI	1.000000	-0.191885	-0.122274	-0.407326	-0.542052	-0.289833	0.810403	-0.000386	0.143010	-0.164237
Na	-0.191885	1.000000	-0.273732	0.156794	-0.069809	-0.266087	-0.275442	0.326603	-0.241346	0.502898
Mg	-0.122274	-0.273732	1.000000	-0.481799	-0.165927	0.005396	-0.443750	-0.492262	0.083060	-0.744993
Al	-0.407326	0.156794	-0.481799	1.000000	-0.005524	0.325958	-0.259592	0.479404	-0.074402	0.598829
Si	-0.542052	-0.069809	-0.165927	-0.005524	1.000000	-0.193331	-0.208732	-0.102151	-0.094201	0.151565
K	-0.289833	-0.266087	0.005396	0.325958	-0.193331	1.000000	-0.317836	-0.042618	-0.007719	-0.010054
Ca	0.810403	-0.275442	-0.443750	-0.259592	-0.208732	-0.317836	1.000000	-0.112841	0.124968	0.000952
Ba	-0.000386	0.326603	-0.492262	0.479404	-0.102151	-0.042618	-0.112841	1.000000	-0.058692	0.575161
Fe	0.143010	-0.241346	0.083060	-0.074402	-0.094201	-0.007719	0.124968	-0.058692	1.000000	-0.188278
Type	-0.164237	0.502898	-0.744993	0.598829	0.151565	-0.010054	0.000952	0.575161	-0.188278	1.000000

```
[60] sns.heatmap(correlation_matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')
     plt.show()
```



```python
features = ['Rl', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']
target = 'Type'
X_train, X_val, Y_train, Y_val = train_test_split(glass[::-1], glass['Type'],test_size=0.2, random_state=1)
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
print('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  0  1]
 [ 0  0  0  0  1  0]
 [ 0  0  0  1  0  3]]
```



```python
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
print('accuracy is',accuracy_score(Y_val, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.82      0.95      0.88        19
           2       0.70      0.58      0.64        12
           3       0.00      0.00      0.00         6
           5       0.00      0.00      0.00         1
           6       0.14      1.00      0.25         1
           7       0.00      0.00      0.00         4

    accuracy                           0.60        43
   macro avg       0.28      0.42      0.29        43
weighted avg       0.56      0.60      0.57        43

[[18  1  0  0  0  0]
 [ 4  7  0  0  1  0]
 [ 0  2  0  3  1  0]
 [ 0  0  0  0  1  0]
 [ 0  0  0  0  1  0]
 [ 0  0  0  1  3  0]]
accuracy is 0.6046511627906976
```