CS5590 APS -Python Programming

LAB1

Dukkipati, Sri Sai Nithin Chowdary – 4

Inakollu, Sri Naga Bhuvaneswari - 9

Kolluri, Nikhitha – 12

INTRODUCTION:

In this lab we will go through some basics of python, use BeautifulSoup package for web scrapping, plotting patterns using some classification algorithms, applying tokenization, lemmatization and trigrams on some file, creating some multiple regression and evaluating the model using RMSE and R2 techniques. Cleaning all the dataset before its usage.

OBJECTIVES:

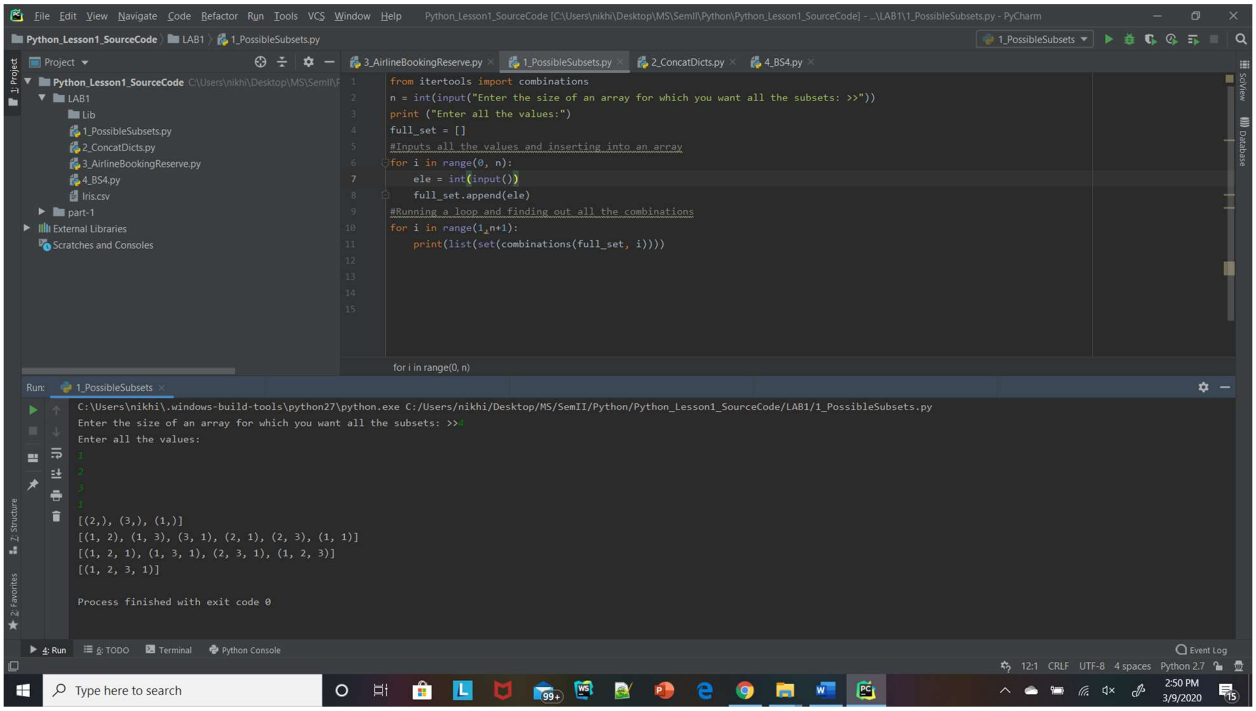
Some objectives of this lab are:

- Basics of python such as performing some operations on subsets, concatenating two dictionaries and some operations on it.
- Airline Booking Reservation System which takes in the details of customer, his preferences and then display the available airlines from which the customer can select source, destination, the type of airlines he wish to fly in, the class (business or economy) and number of bags he wish to check-in . The system should display the fare for the ticket and prints the ticket. The system should display all the details of the employee that is booking the ticket.
- Use the BeautifulSoup package and from the school catalogue site get all the course names and their description.
- Take a dataset which will have both numeric and non-numeric data, perform data analysis on the chosen dataset and plot patterns. Before plotting patterns remove all the null values from the data set, remove all the features which are unrelated to the target class and all the categorical features should be encoded. Report the classification of the data set based on some classification algorithms such as Naïve Bayes, SVM and KNN.
- Choose any dataset you wish and them perform K-means on that and then visualize the data using matplotlib or seaborn. Select the best K by applying elbow method, evaluate the silhouette score. Before evaluating the score, clean the data set with EDA.
- Take an input file which has few sentences and apply tokenization, lemmatization, trigrams, most repeated trigrams, extract few sentences and concatenate them and print the result.
- Choose a data set of your choice, create multiple regression. Use RMSE and R2 to evaluate the model, display the improved results after EDA.

APPROACHES/METHODS:

1. Return all possible subsets for a given collection of integers without including the null subset.

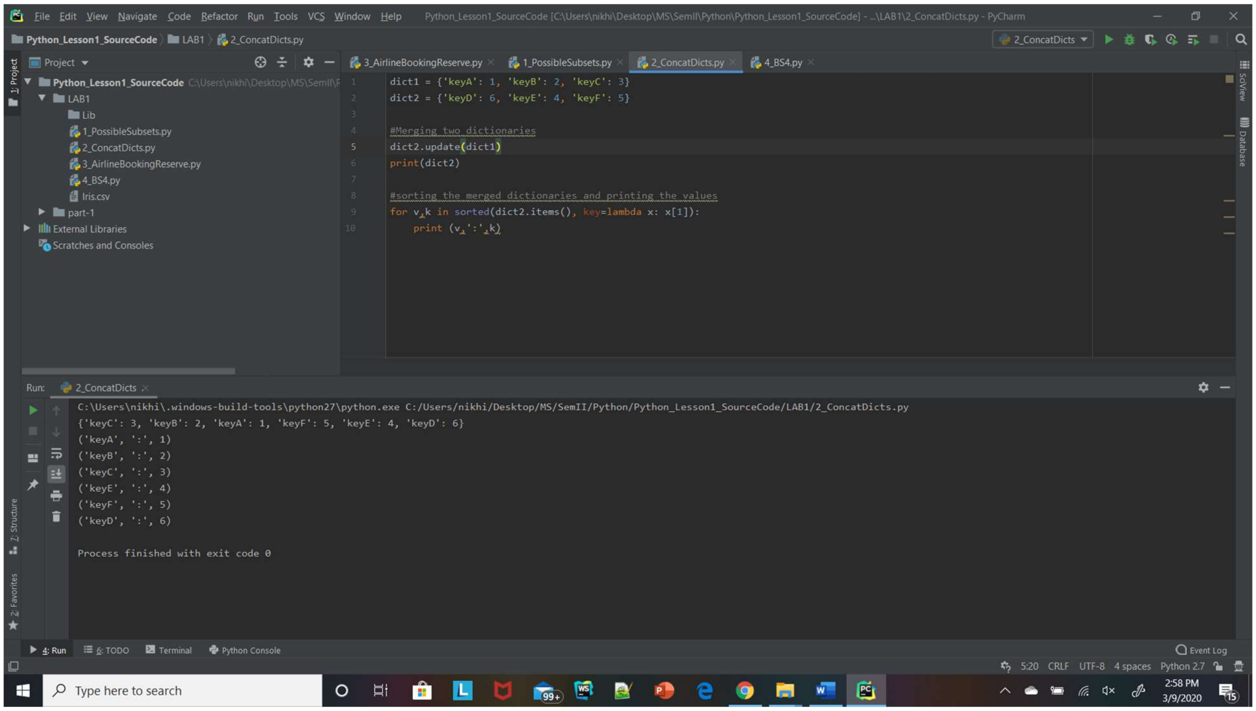
a. First we will be accepting the length of the integers from the user.
b. Then we will ask them to input the integers for the set using a for loop and append all these integers to a list.
c. Then using itertools.combinations we will get all the subsets.
d. If you apply set on this combinations we can eliminate the duplicates.
e. We use a loop to get all the subsets of a given set.



```
from itertools import combinations
n = int(input("Enter the size of an array for which you want all the subsets: >>"))
print ("Enter all the values:")
full_set = []
#Inputs all the values and inserting into an array
for i in range(0, n):
    ele = int(input())
    full_set.append(ele)
#Running a loop and finding out all the combinations
for i in range(1,n+1):
    print(list(set(combinations(full_set, i))))
```

```
C:\Users\nikhi\.windows-build-tools\python27\python.exe C:/Users/nikhi/Desktop/MS/SemII/Python/Python_Lesson1_SourceCode/LAB1/1_PossibleSubsets.py
Enter the size of an array for which you want all the subsets: >>4
Enter all the values:
1
2
3
1
[(2,), (3,), (1,)]
[(1, 2), (1, 3), (3, 1), (2, 1), (2, 3), (1, 1)]
[(1, 2, 1), (1, 3, 1), (2, 3, 1), (1, 2, 3)]
[(1, 2, 3, 1)]

Process finished with exit code 0
```

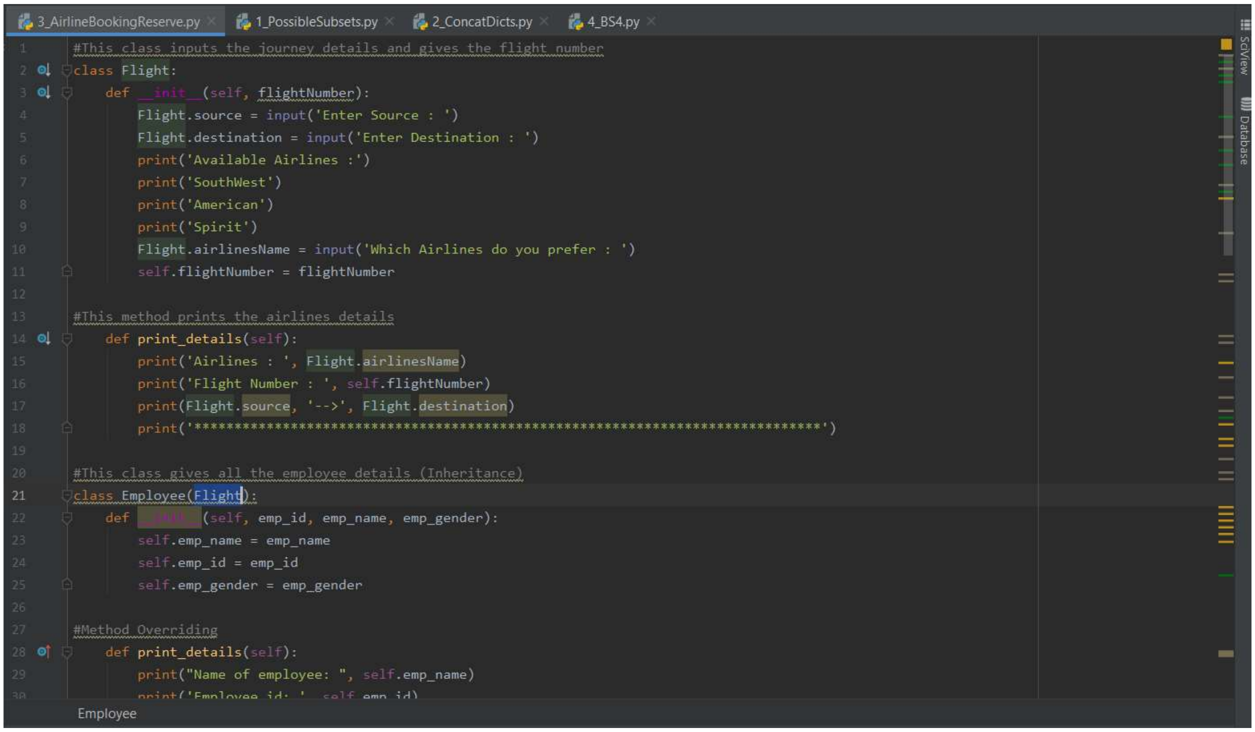
2. Concatenate two dictionaries and sort them based on the values.
    a. First declare and define two dictionaries.
    b. Merge the dictionaries using the update method.
    c. Sorting the merged dictionary by using sorted method which takes lambda function and sorts based on the value field.
    d. We use a for loop to print all the key value pairs in sorted order.



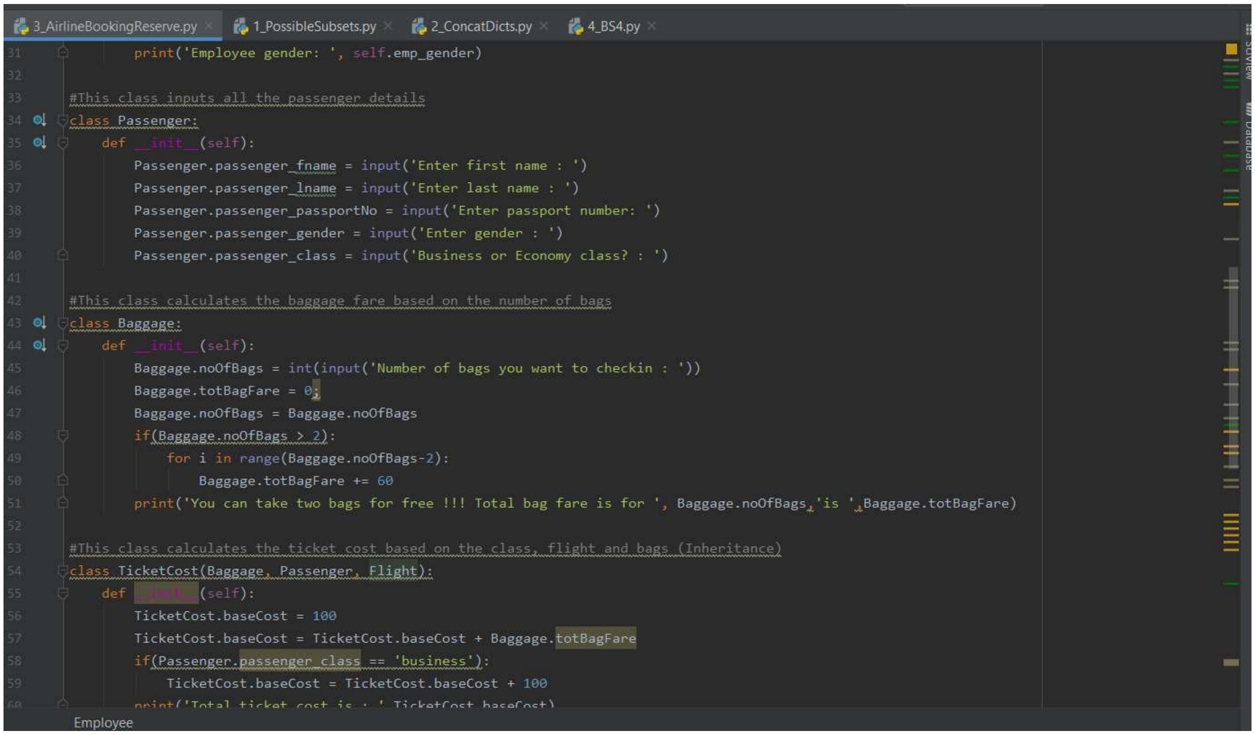
```
dict1 = {'keyA': 1, 'keyB': 2, 'keyC': 3}
dict2 = {'keyD': 6, 'keyE': 4, 'keyF': 5}

#Merging two dictionaries
dict2.update(dict1)
print(dict2)

#sorting the merged dictionaries and printing the values
for v,k in sorted(dict2.items(), key=lambda x: x[1]):
    print (v,':',k)
```

```
C:\Users\nikhi\.windows-build-tools\python27\python.exe C:/Users/nikhi/Desktop/MS/SemII/Python/Python_Lesson1_SourceCode/LAB1/2_ConcatDicts.py
{'keyC': 3, 'keyB': 2, 'keyA': 1, 'keyF': 5, 'keyE': 4, 'keyD': 6}
('keyA', ':', 1)
('keyB', ':', 2)
('keyC', ':', 3)
('keyE', ':', 4)
('keyF', ':', 5)
('keyD', ':', 6)

Process finished with exit code 0
```

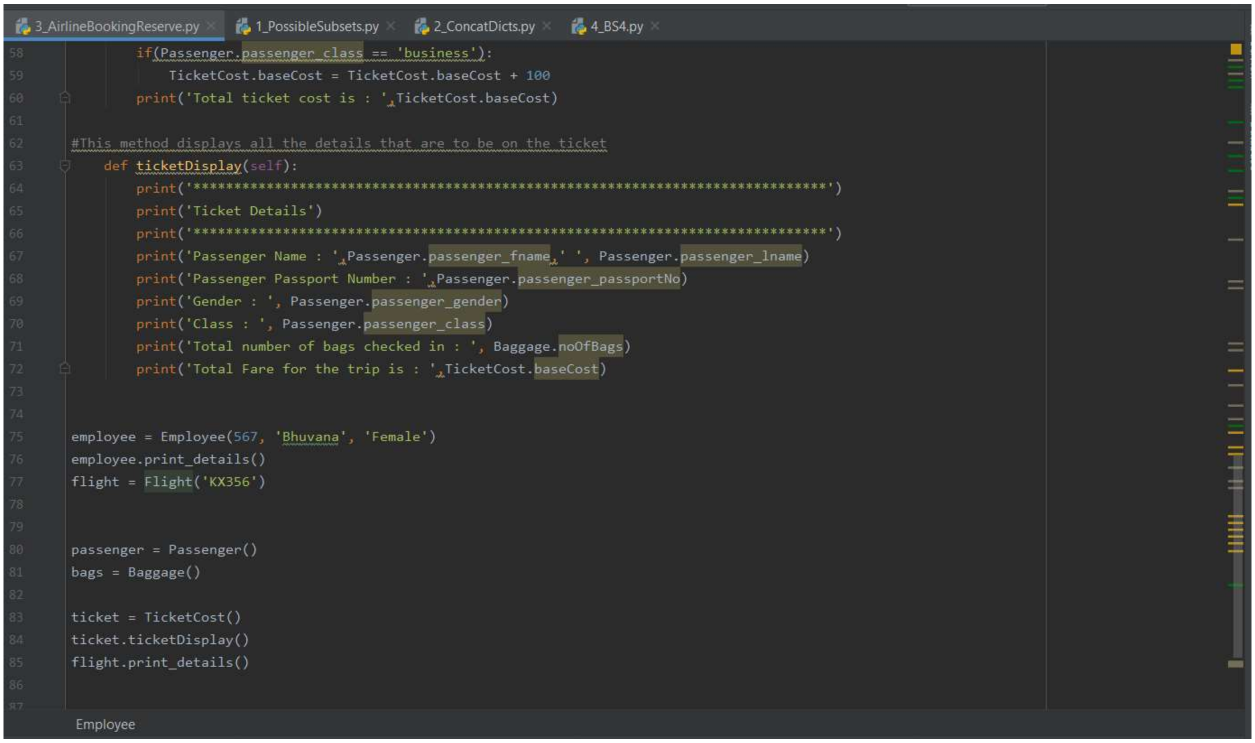
3. Airline Booking Reservation System
    a. First we have taken a flight class which will take some inputs from user, like source, destination, airlines, class and will generate the flight number. It also has a method that displays the details.
    b. Employee class will display all the details of the employee. It uses the concept of inheritance and overrides the printing method of the Flight class.
    c. Passenger class will take all the details of the passenger.
    d. Baggage class would take the number of check-in bags from the passenger and calculate the baggage fare.
    e. TicketCost class takes the inputs from all the classes and would calculate the fare of the ticket. A method in TicketCost would display all the ticket details.



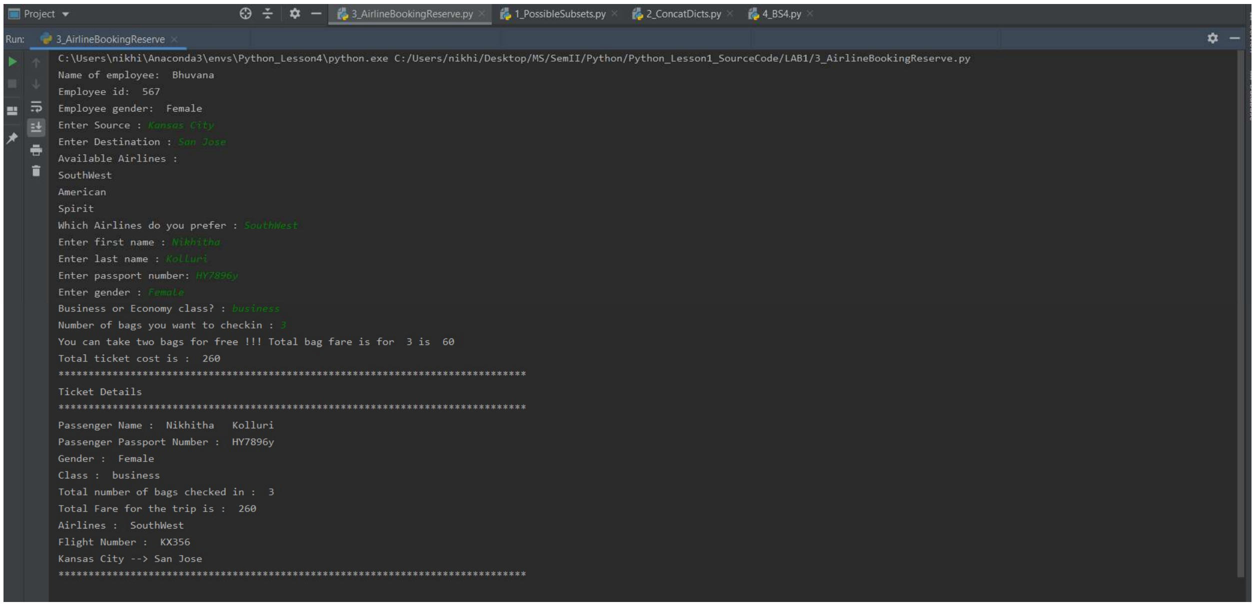
```python
#This class inputs the journey details and gives the flight number
class Flight:
    def __init__(self, flightNumber):
        Flight.source = input('Enter Source : ')
        Flight.destination = input('Enter Destination : ')
        print('Available Airlines :')
        print('SouthWest')
        print('American')
        print('Spirit')
        Flight.airlinesName = input('Which Airlines do you prefer : ')
        self.flightNumber = flightNumber

#This method prints the airlines details
    def print_details(self):
        print('Airlines : ', Flight.airlinesName)
        print('Flight Number : ', self.flightNumber)
        print(Flight.source, '-->', Flight.destination)
        print('********************************************************************************')

#This class gives all the employee details (Inheritance)
class Employee(Flight):
    def __init__(self, emp_id, emp_name, emp_gender):
        self.emp_name = emp_name
        self.emp_id = emp_id
        self.emp_gender = emp_gender

#Method Overriding
    def print_details(self):
        print("Name of employee: ", self.emp_name)
        print('Employee id: ', self.emp_id)
```



```python
        print('Employee gender: ', self.emp_gender)

#This class inputs all the passenger details
class Passenger:
    def __init__(self):
        Passenger.passenger_fname = input('Enter first name : ')
        Passenger.passenger_lname = input('Enter last name : ')
        Passenger.passenger_passportNo = input('Enter passport number: ')
        Passenger.passenger_gender = input('Enter gender : ')
        Passenger.passenger_class = input('Business or Economy class? : ')

#This class calculates the baggage fare based on the number of bags
class Baggage:
    def __init__(self):
        Baggage.noOfBags = int(input('Number of bags you want to checkin : '))
        Baggage.totBagFare = 0;
        Baggage.noOfBags = Baggage.noOfBags
        if(Baggage.noOfBags > 2):
            for i in range(Baggage.noOfBags-2):
                Baggage.totBagFare += 60
        print('You can take two bags for free !!! Total bag fare is for ', Baggage.noOfBags,'is ',Baggage.totBagFare)

#This class calculates the ticket cost based on the class, flight and bags (Inheritance)
class TicketCost(Baggage, Passenger, Flight):
    def __init__(self):
        TicketCost.baseCost = 100
        TicketCost.baseCost = TicketCost.baseCost + Baggage.totBagFare
        if(Passenger.passenger_class == 'business'):
            TicketCost.baseCost = TicketCost.baseCost + 100
        print('Total ticket cost is : ', TicketCost.baseCost)
```

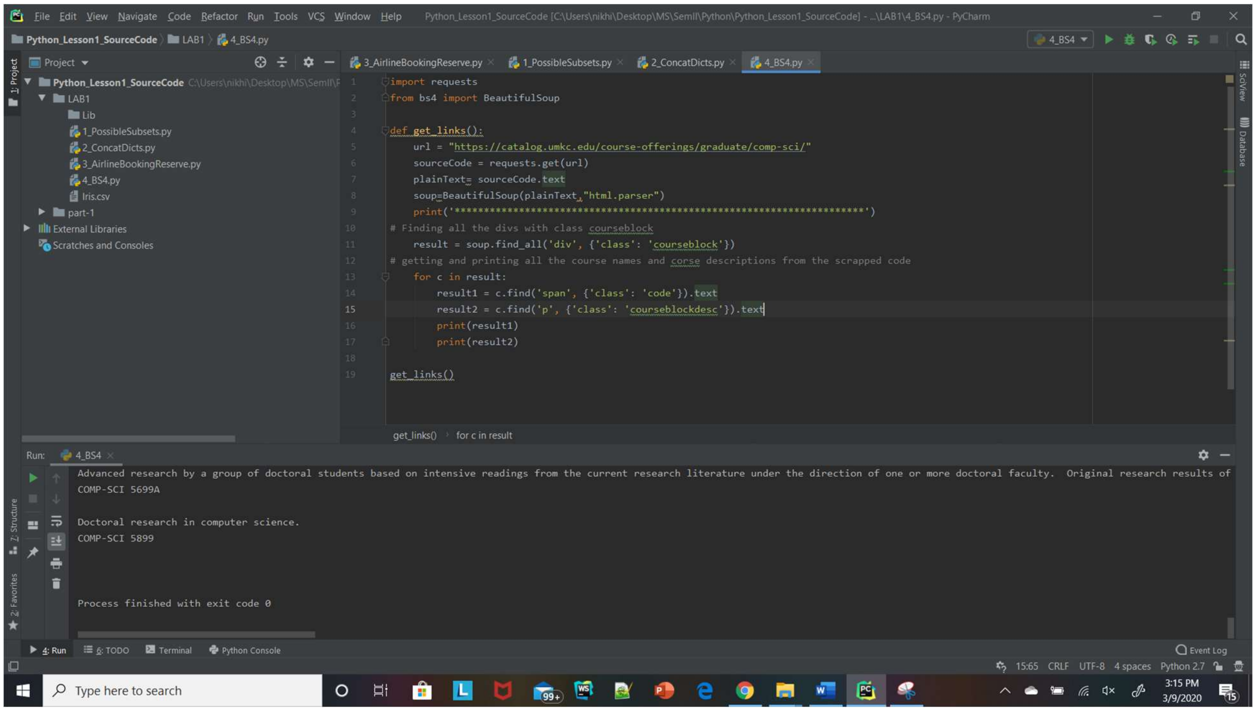


```python
        if(Passenger.passenger_class == 'business'):
            TicketCost.baseCost = TicketCost.baseCost + 100
        print('Total ticket cost is : ',TicketCost.baseCost)

#This method displays all the details that are to be on the ticket
    def ticketDisplay(self):
        print('*******************************************************************************')
        print('Ticket Details')
        print('*******************************************************************************')
        print('Passenger Name : ',Passenger.passenger_fname,' ', Passenger.passenger_lname)
        print('Passenger Passport Number : ',Passenger.passenger_passportNo)
        print('Gender : ', Passenger.passenger_gender)
        print('Class : ', Passenger.passenger_class)
        print('Total number of bags checked in : ', Baggage.noOfBags)
        print('Total Fare for the trip is : ',TicketCost.baseCost)


employee = Employee(567, 'Bhuvana', 'Female')
employee.print_details()
flight = Flight('KX356')


passenger = Passenger()
bags = Baggage()

ticket = TicketCost()
ticket.ticketDisplay()
flight.print_details()
```



```
C:\Users\nikhi\Anaconda3\envs\Python_Lesson4\python.exe C:/Users/nikhi/Desktop/MS/SemII/Python/Python_Lesson1_SourceCode/LAB1/3_AirlineBookingReserve.py
Name of employee:  Bhuvana
Employee id:  567
Employee gender:  Female
Enter Source : Kansas City
Enter Destination : San Jose
Available Airlines :
SouthWest
American
Spirit
Which Airlines do you prefer : SouthWest
Enter first name : Nikhitha
Enter last name : Kolluri
Enter passport number: HY7896y
Enter gender : Female
Business or Economy class? : business
Number of bags you want to checkin : 3
You can take two bags for free !!! Total bag fare is for  3 is  60
Total ticket cost is :  260
******************************************************************************
Ticket Details
******************************************************************************
Passenger Name :  Nikhitha   Kolluri
Passenger Passport Number :  HY7896y
Gender :  Female
Class :  business
Total number of bags checked in :  3
Total Fare for the trip is :  260
Airlines :  SouthWest
Flight Number :  KX356
Kansas City --> San Jose
******************************************************************************
```

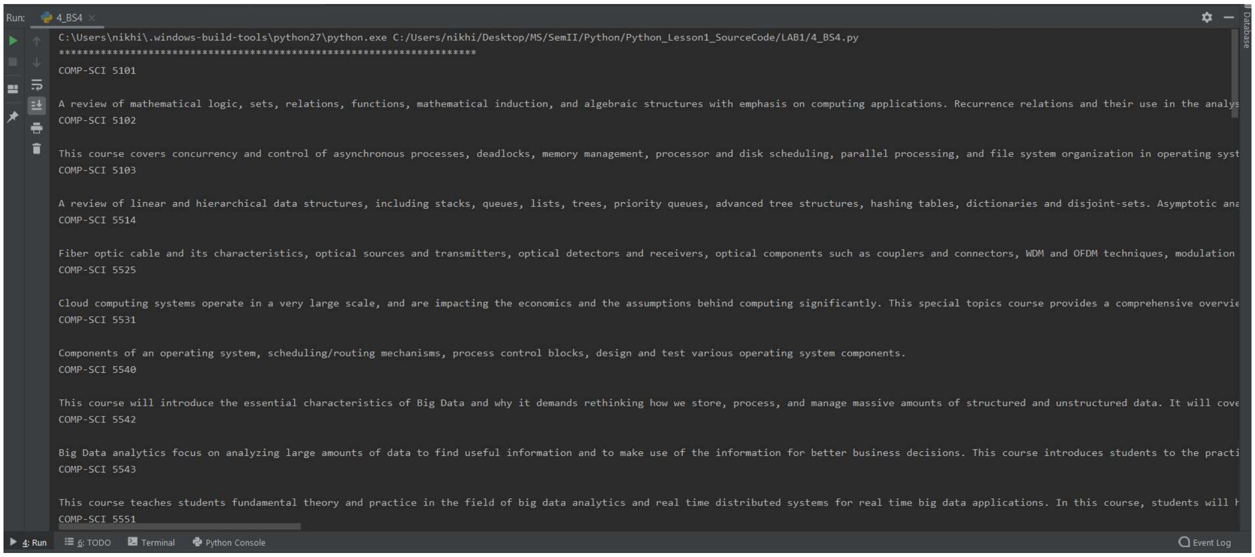
4. WebScrapping using BeautifulSoup package
    a. First we would get the html of the given URL using BS4 package
    b. Then convert all the html into plain text.
    c. We find all the div tags with class='courseblock'.
    d. We then run a loop to get all the course names and course descriptions.
    e. We then get the text only without the tags using the text option.
    f. Finally we display the result.



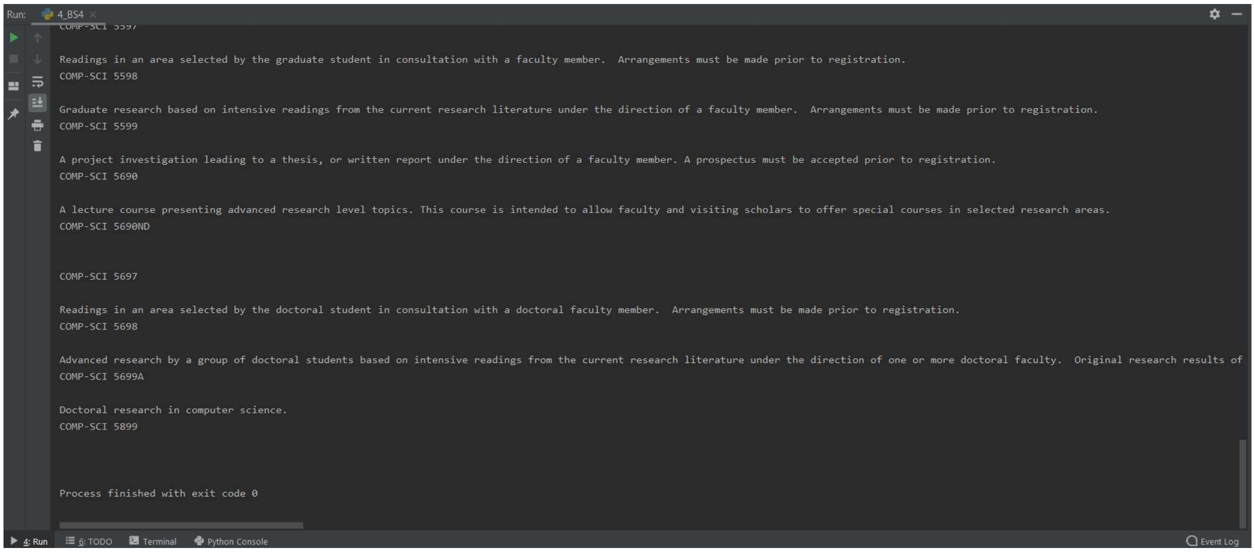
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Python_Lesson1_SourceCode [C:\Users\nikhi\Desktop\MS\SemII\Python\Python_Lesson1_SourceCode] - ...\LAB1\4_BS4.py - PyCharm
Python_Lesson1_SourceCode
LAB1
4_BS4.py
Project
Lib
1_PossibleSubsets.py
2_ConcatDicts.py
3_AirlineBookingReserve.py
4_BS4.py
Iris.csv
part-1
External Libraries
Scratches and Consoles
get_links()
for c in result
Run:
4_BS4
4: Run
6: TODO
Terminal
Python Console
Event Log
15:65 CRLF UTF-8 4 spaces Python 2.7
Type here to search
3:15 PM
3/9/2020

```python
import requests
from bs4 import BeautifulSoup

def get_links():
    url = "https://catalog.umkc.edu/course-offerings/graduate/comp-sci/"
    sourceCode = requests.get(url)
    plainText= sourceCode.text
    soup=BeautifulSoup(plainText,"html.parser")
    print('**********************************************************************')
# Finding all the divs with class courseblock
    result = soup.find_all('div', {'class': 'courseblock'})
# getting and printing all the course names and corse descriptions from the scrapped code
    for c in result:
        result1 = c.find('span', {'class': 'code'}).text
        result2 = c.find('p', {'class': 'courseblockdesc'}).text
        print(result1)
        print(result2)

get_links()
```

```
Advanced research by a group of doctoral students based on intensive readings from the current research literature under the direction of one or more doctoral faculty.  Original research results of
COMP-SCI 5699A

Doctoral research in computer science.
COMP-SCI 5899



Process finished with exit code 0
```



```
C:\Users\nikhi\.windows-build-tools\python27\python.exe C:/Users/nikhi/Desktop/MS/SemII/Python/Python_Lesson1_SourceCode/LAB1/4_BS4.py
**********************************************************************
COMP-SCI 5101

A review of mathematical logic, sets, relations, functions, mathematical induction, and algebraic structures with emphasis on computing applications. Recurrence relations and their use in the analys
COMP-SCI 5102

This course covers concurrency and control of asynchronous processes, deadlocks, memory management, processor and disk scheduling, parallel processing, and file system organization in operating syst
COMP-SCI 5103

A review of linear and hierarchical data structures, including stacks, queues, lists, trees, priority queues, advanced tree structures, hashing tables, dictionaries and disjoint-sets. Asymptotic ana
COMP-SCI 5514

Fiber optic cable and its characteristics, optical sources and transmitters, optical detectors and receivers, optical components such as couplers and connectors, WDM and OFDM techniques, modulation
COMP-SCI 5525

Cloud computing systems operate in a very large scale, and are impacting the economics and the assumptions behind computing significantly. This special topics course provides a comprehensive overvie
COMP-SCI 5531

Components of an operating system, scheduling/routing mechanisms, process control blocks, design and test various operating system components.
COMP-SCI 5540

This course will introduce the essential characteristics of Big Data and why it demands rethinking how we store, process, and manage massive amounts of structured and unstructured data. It will cove
COMP-SCI 5542

Big Data analytics focus on analyzing large amounts of data to find useful information and to make use of the information for better business decisions. This course introduces students to the practi
COMP-SCI 5543

This course teaches students fundamental theory and practice in the field of big data analytics and real time distributed systems for real time big data applications. In this course, students will h
COMP-SCI 5551
```



```
COMP-SCI 5597

Readings in an area selected by the graduate student in consultation with a faculty member.  Arrangements must be made prior to registration.
COMP-SCI 5598

Graduate research based on intensive readings from the current research literature under the direction of a faculty member.  Arrangements must be made prior to registration.
COMP-SCI 5599

A project investigation leading to a thesis, or written report under the direction of a faculty member. A prospectus must be accepted prior to registration.
COMP-SCI 5690

A lecture course presenting advanced research level topics. This course is intended to allow faculty and visiting scholars to offer special courses in selected research areas.
COMP-SCI 5690ND


COMP-SCI 5697

Readings in an area selected by the doctoral student in consultation with a doctoral faculty member.  Arrangements must be made prior to registration.
COMP-SCI 5698

Advanced research by a group of doctoral students based on intensive readings from the current research literature under the direction of one or more doctoral faculty.  Original research results of
COMP-SCI 5699A

Doctoral research in computer science.
COMP-SCI 5899



Process finished with exit code 0
```

5. SVM, KNN, Naïve Bayes classifier

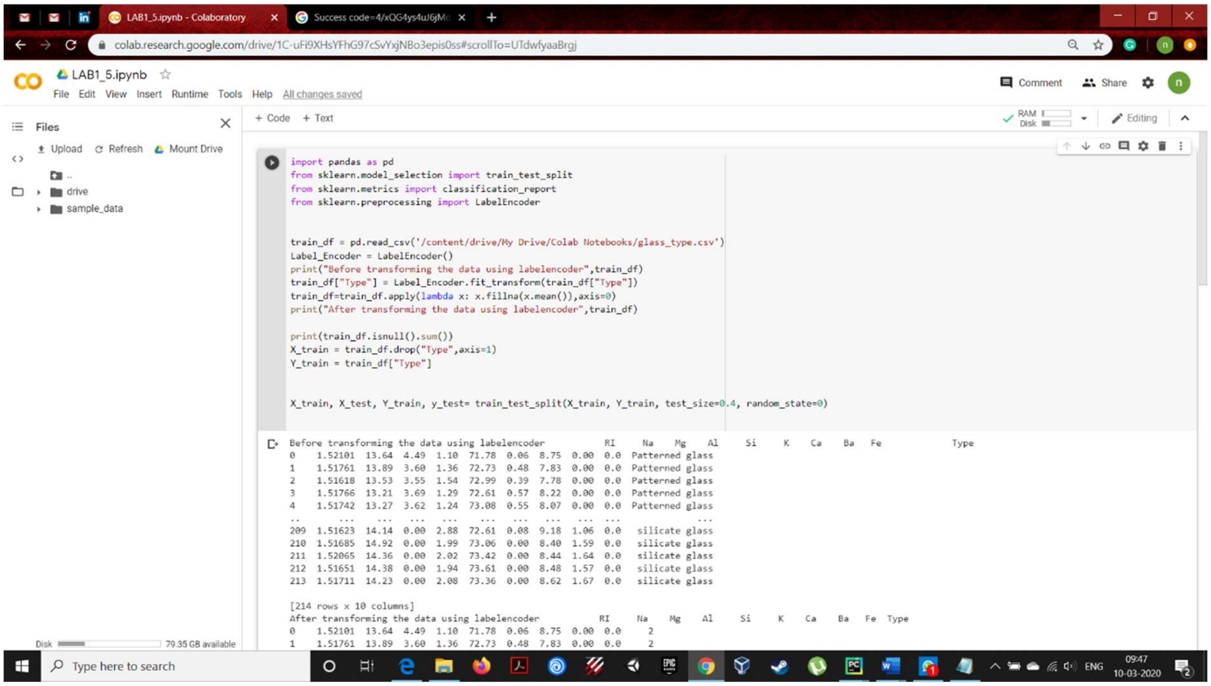
5a.

a. Took the data glass_type.cvs file which consists of both numeric and non-numberic data

b. Used label label encoder to change non numeric data to numeric one

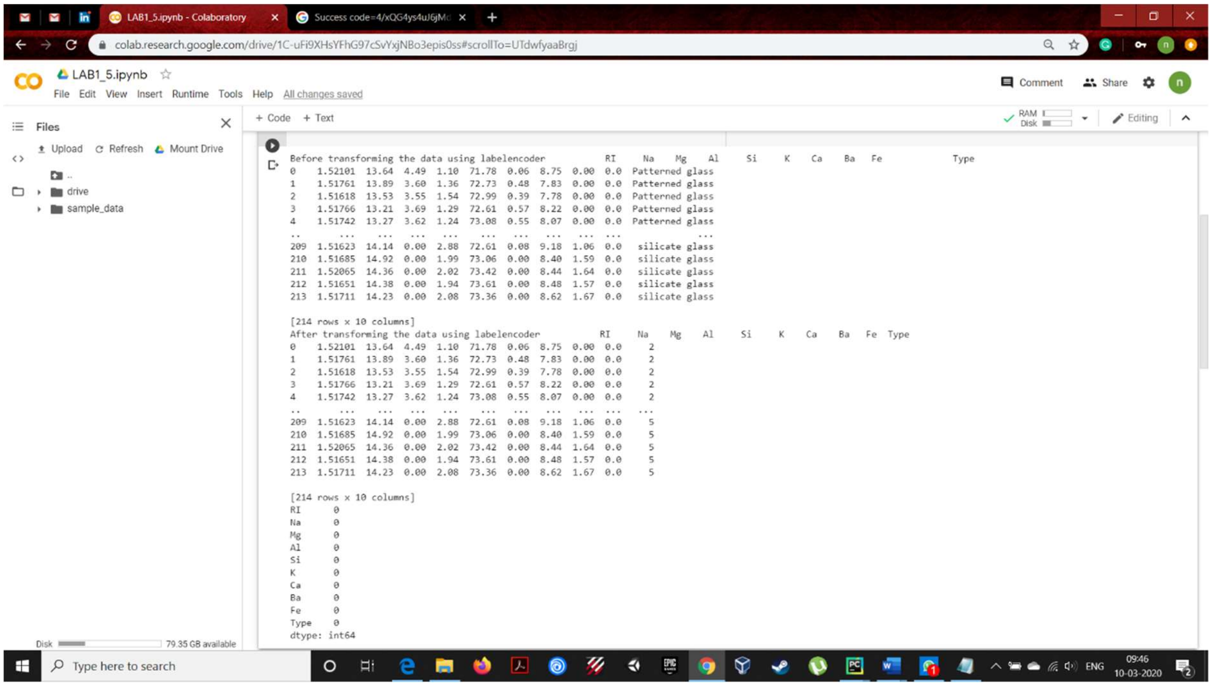
c. Used fillna to d=fill all the null values with mean data(adding noise)

d. Used spilt function for splitting the data into test and training data


LAB1_5.ipynb - Colaboratory
Success code=4/xQG4ys4uJ6jM
colab.research.google.com/drive/1C-uFi9XHsYFhG97cSvYxjNBo3epis0ss#scrollTo=UTdwfyaaBrgj
LAB1_5.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
Upload
Refresh
Mount Drive
drive
sample_data
+ Code
+ Text
RAM
Disk
Editing
Disk 79.35 GB available
Type here to search
ENG
09:47
10-03-2020

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder


train_df = pd.read_csv('/content/drive/My Drive/Colab Notebooks/glass_type.csv')
Label_Encoder = LabelEncoder()
print("Before transforming the data using labelencoder",train_df)
train_df["Type"] = Label_Encoder.fit_transform(train_df["Type"])
train_df=train_df.apply(lambda x: x.fillna(x.mean()),axis=0)
print("After transforming the data using labelencoder",train_df)

print(train_df.isnull().sum())
X_train = train_df.drop("Type",axis=1)
Y_train = train_df["Type"]


X_train, X_test, Y_train, y_test= train_test_split(X_train, Y_train, test_size=0.4, random_state=0)
```

```
Before transforming the data using labelencoder          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  0.00  0.0  Patterned glass
1    1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0  Patterned glass
2    1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0  Patterned glass
3    1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0  Patterned glass
4    1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0  Patterned glass
..       ...    ...   ...   ...    ...   ...   ...   ...  ...              ...
209  1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0   silicate glass
210  1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0   silicate glass
211  1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0   silicate glass
212  1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0   silicate glass
213  1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0   silicate glass

[214 rows x 10 columns]
After transforming the data using labelencoder          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  0.00  0.0     2
1    1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0     2
```


LAB1_5.ipynb - Colaboratory
Success code=4/xQG4ys4uJ6jM
colab.research.google.com/drive/1C-uFi9XHsYFhG97cSvYxjNBo3epis0ss#scrollTo=UTdwfyaaBrgj
LAB1_5.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
Upload
Refresh
Mount Drive
drive
sample_data
+ Code
+ Text
RAM
Disk
Editing
Disk 79.35 GB available
Type here to search
ENG
09:46
10-03-2020

```
Before transforming the data using labelencoder         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  0.00  0.0  Patterned glass
1    1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0  Patterned glass
2    1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0  Patterned glass
3    1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0  Patterned glass
4    1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0  Patterned glass
..       ...    ...   ...   ...    ...   ...   ...   ...  ...              ...
209  1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0   silicate glass
210  1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0   silicate glass
211  1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0   silicate glass
212  1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0   silicate glass
213  1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0   silicate glass

[214 rows x 10 columns]
After transforming the data using labelencoder        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  0.00  0.0     2
1    1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0     2
2    1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0     2
3    1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0     2
4    1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0     2
..       ...    ...   ...   ...    ...   ...   ...   ...  ...   ...
209  1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0     5
210  1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0     5
211  1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0     5
212  1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0     5
213  1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0     5

[214 rows x 10 columns]
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

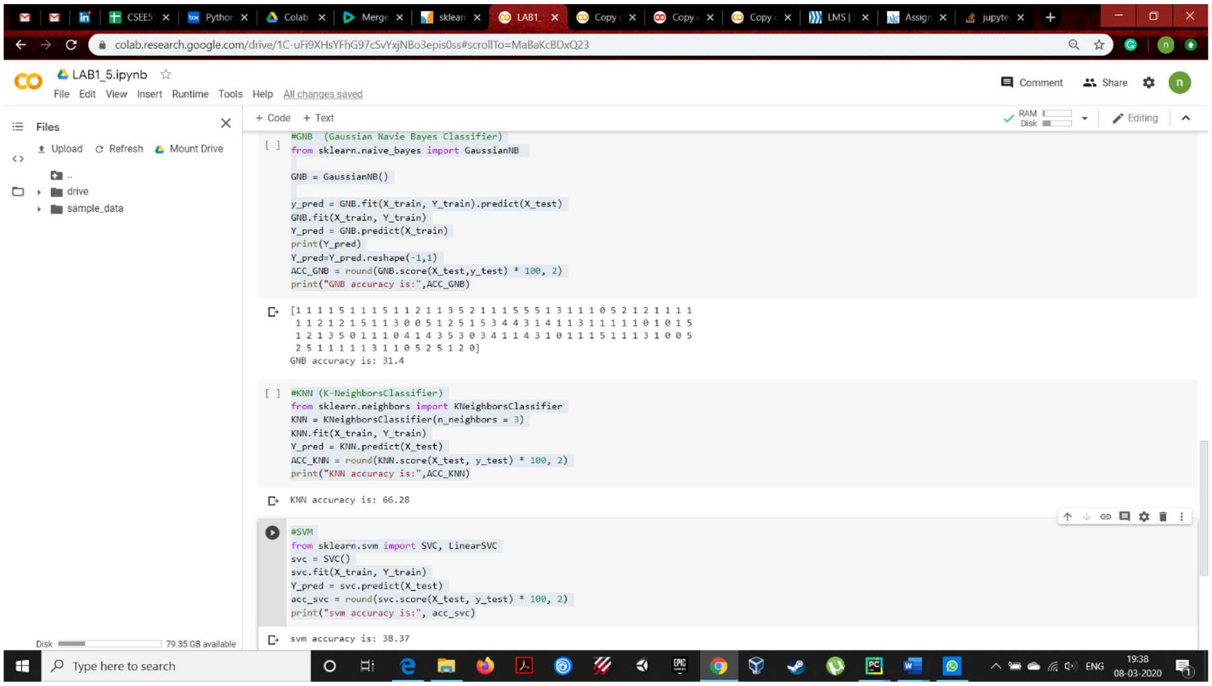
5b.

e. Called the class svm, gaussianNb, KNeihborsClassifier.

f. Fitted the model with training data

g. Predicted the accuracy with score function with test data

We got more accuracy with KNN where k=3



```
#GNB  (Gaussian Navie Bayes Classifier)
from sklearn.naive_bayes import GaussianNB

GNB = GaussianNB()

y_pred = GNB.fit(X_train, Y_train).predict(X_test)
GNB.fit(X_train, Y_train)
Y_pred = GNB.predict(X_train)
print(Y_pred)
Y_pred=Y_pred.reshape(-1,1)
ACC_GNB = round(GNB.score(X_test,y_test) * 100, 2)
print("GNB accuracy is:",ACC_GNB)
```

```
[1 1 1 1 5 1 1 1 5 1 1 2 1 1 3 5 2 1 1 1 5 5 5 1 3 1 1 1 0 5 2 1 2 1 1 1 1
 1 1 2 1 2 1 5 1 1 3 0 0 5 1 2 5 1 5 3 4 4 3 1 4 1 1 3 1 1 1 1 1 0 1 0 1 5
 1 2 1 3 5 0 1 1 1 0 4 1 4 3 5 3 0 3 4 1 1 4 3 1 0 1 1 1 5 1 1 1 3 1 0 0 5
 2 5 1 1 1 1 1 3 1 1 0 5 2 5 1 2 0]
GNB accuracy is: 31.4
```

```
#KNN (K-NeighborsClassifier)
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier(n_neighbors = 3)
KNN.fit(X_train, Y_train)
Y_pred = KNN.predict(X_test)
ACC_KNN = round(KNN.score(X_test, y_test) * 100, 2)
print("KNN accuracy is:",ACC_KNN)
```

```
KNN accuracy is: 66.28
```

```
#SVM
from sklearn.svm import SVC, LinearSVC
svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_test, y_test) * 100, 2)
print("svm accuracy is:", acc_svc)
```

```
svm accuracy is: 38.37
```

6. K-means

6a.

a. Used cc.csv data set

b. Visualized data using bar graph with seaborn library

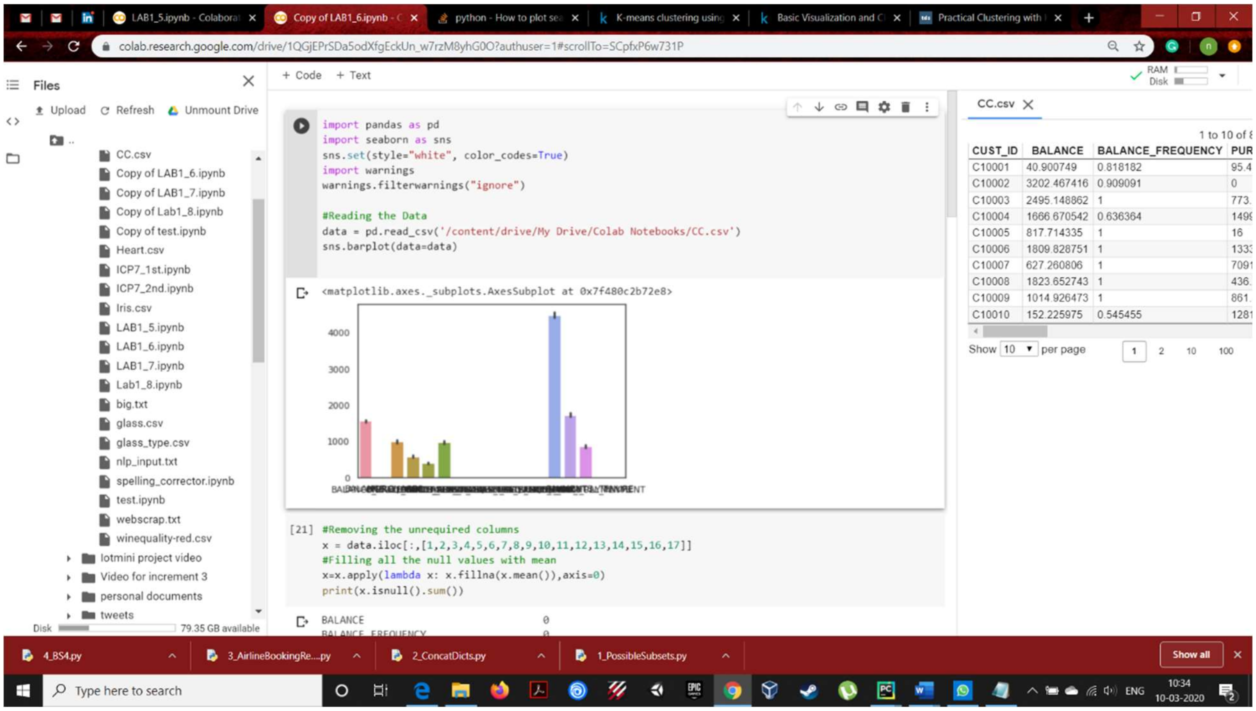
c. Extracted columns that are required

d. Cleaned the data, used fillna to fill all null values(fitting noise data)

e. Used StandardSCaler function for converting the data into numeric form

f. Used datafram method to convert the data again back to dataframe

g. Used KMeans clustering function to get best k value



```
import pandas as pd
import seaborn as sns
sns.set(style="white", color_codes=True)
import warnings
warnings.filterwarnings("ignore")

#Reading the Data
data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/CC.csv')
sns.barplot(data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f480c2b72e8>
```

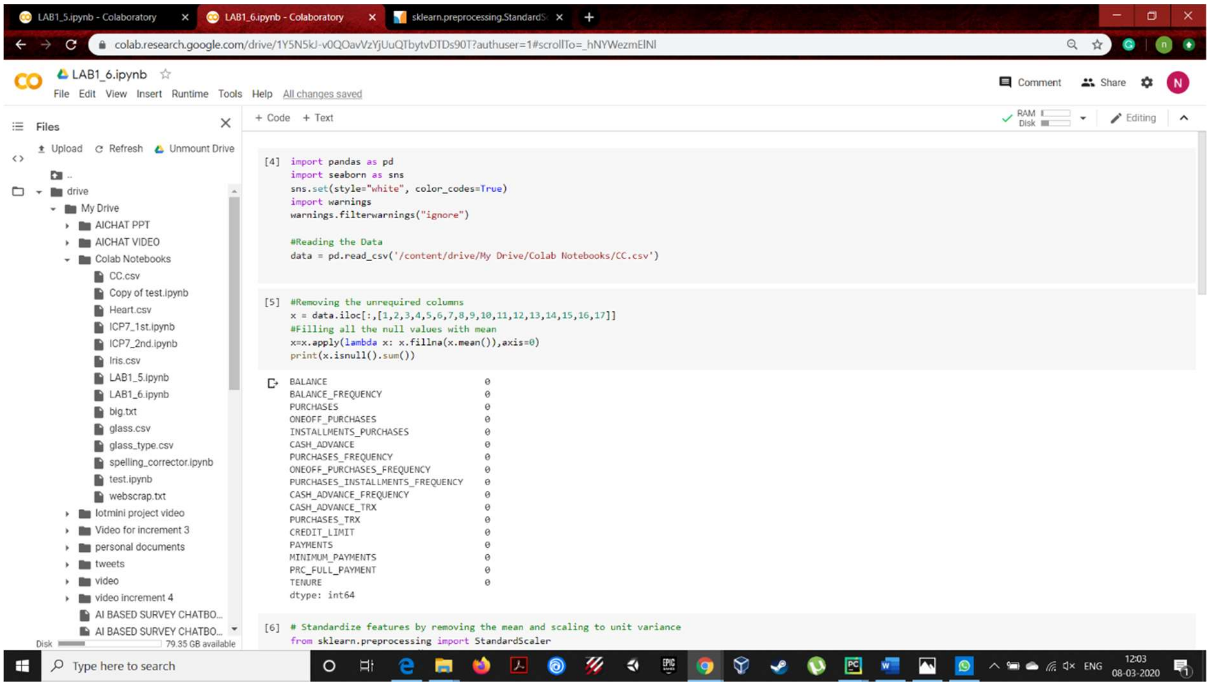
1 to 10 of 8

CUST_ID	BALANCE	BALANCE_FREQUENCY	PUR
C10001	40.900749	0.818182	95.4
C10002	3202.467416	0.909091	0
C10003	2495.148862	1	773.
C10004	1666.670542	0.636364	1499
C10005	817.714335	1	16
C10006	1809.828751	1	1333
C10007	627.260806	1	7091
C10008	1823.652743	1	436.
C10009	1014.926473	1	861.
C10010	152.225975	0.545455	1281

Show 10 ▾ per page   1   2   10   100

```
[21] #Removing the unrequired columns
     x = data.iloc[:,[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]]
     #Filling all the null values with mean
     x=x.apply(lambda x: x.fillna(x.mean()),axis=0)
     print(x.isnull().sum())
```

```
BALANCE                             0
BALANCE_FREQUENCY                   0
```

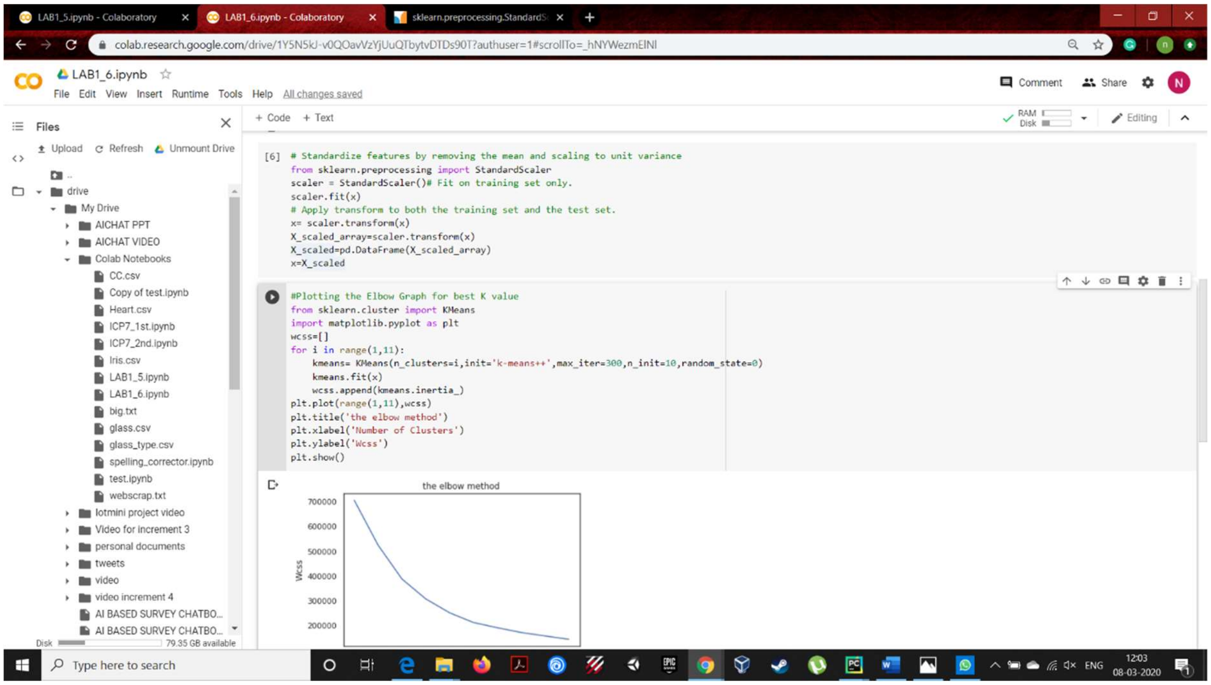


```
[4]  import pandas as pd
     import seaborn as sns
     sns.set(style="white", color_codes=True)
     import warnings
     warnings.filterwarnings("ignore")

     #Reading the Data
     data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/CC.csv')
```

```
[5]  #Removing the unrequired columns
     x = data.iloc[:,[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]]
     #Filling all the null values with mean
     x=x.apply(lambda x: x.fillna(x.mean()),axis=0)
     print(x.isnull().sum())
```

```
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

```
[6]  # Standardize features by removing the mean and scaling to unit variance
     from sklearn.preprocessing import StandardScaler
```



LAB1_6.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
Upload Refresh Unmount Drive
the elbow method
700000
600000
500000
400000
300000
200000
Wcss

```
[6] # Standardize features by removing the mean and scaling to unit variance
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()# Fit on training set only.
    scaler.fit(x)
    # Apply transform to both the training set and the test set.
    x= scaler.transform(x)
    X_scaled_array=scaler.transform(x)
    X_scaled=pd.DataFrame(X_scaled_array)
    x=X_scaled
```

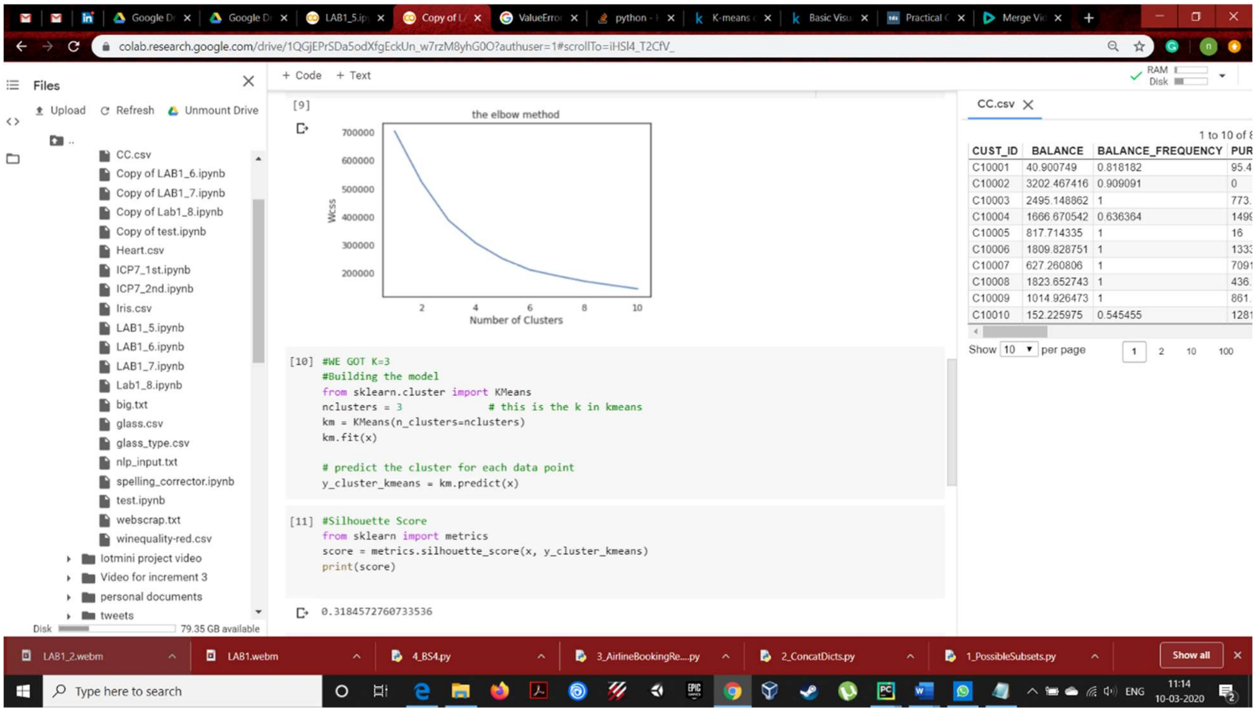
```
#Plotting the Elbow Graph for best K value
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
wcss=[]
for i in range(1,11):
    kmeans= KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('the elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```

6b.

h. plotted an elbow graph for visualizing the elbow where k lies

i. Fitted a Knn model with the best k value (which is n=3)

j. finally calculated the silhouette score



1 to 10 of 8

CUST_ID	BALANCE	BALANCE_FREQUENCY	PUR
C10001	40.900749	0.818182	95.4
C10002	3202.467416	0.909091	0
C10003	2495.148862	1	773.
C10004	1666.670542	0.636364	1499
C10005	817.714335	1	16
C10006	1809.828751	1	1333
C10007	627.260806	1	7091
C10008	1823.652743	1	436.
C10009	1014.926473	1	861.
C10010	152.225975	0.545455	1281

Show 10 per page 1 2 10 100

```
[10] #WE GOT K=3
     #Building the model
     from sklearn.cluster import KMeans
     nclusters = 3               # this is the k in kmeans
     km = KMeans(n_clusters=nclusters)
     km.fit(x)

     # predict the cluster for each data point
     y_cluster_kmeans = km.predict(x)
```

```
[11] #Silhouette Score
     from sklearn import metrics
     score = metrics.silhouette_score(x, y_cluster_kmeans)
     print(score)
```

```
0.3184572760733536
```

7Ans.

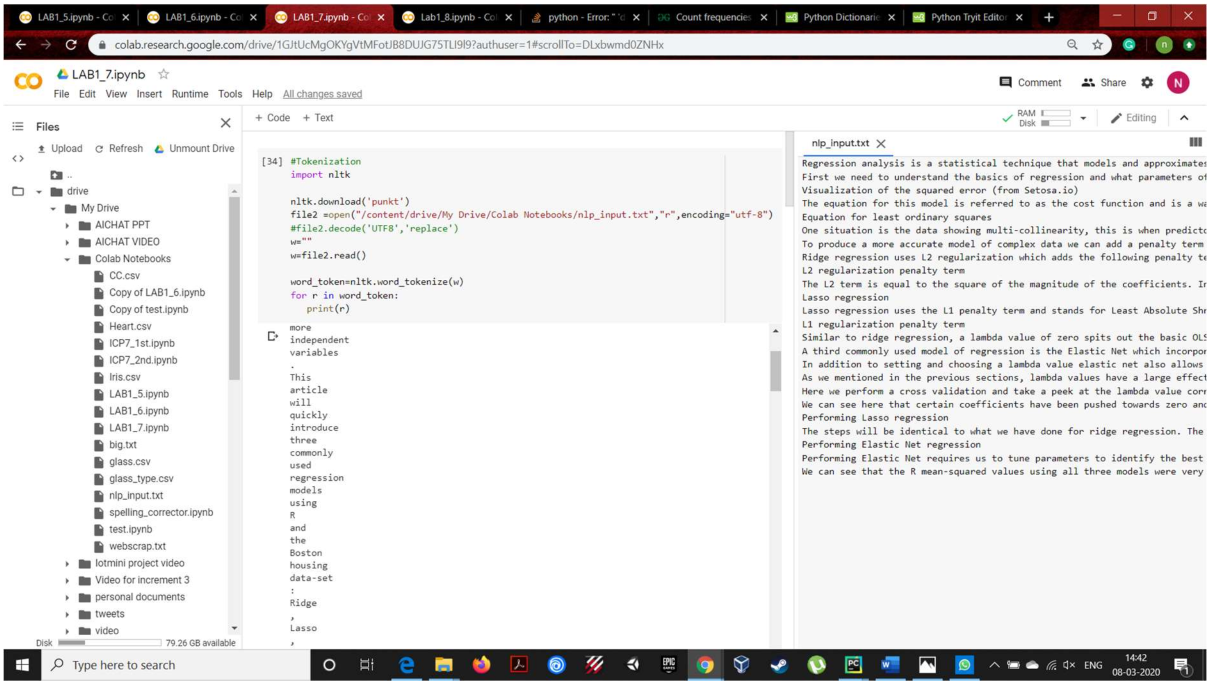
7a,b

a. Read the given file nlp_input.txt with encoding UTF-8

b. Used nltk library

c. Used word_tokenize function for tokenizing each word

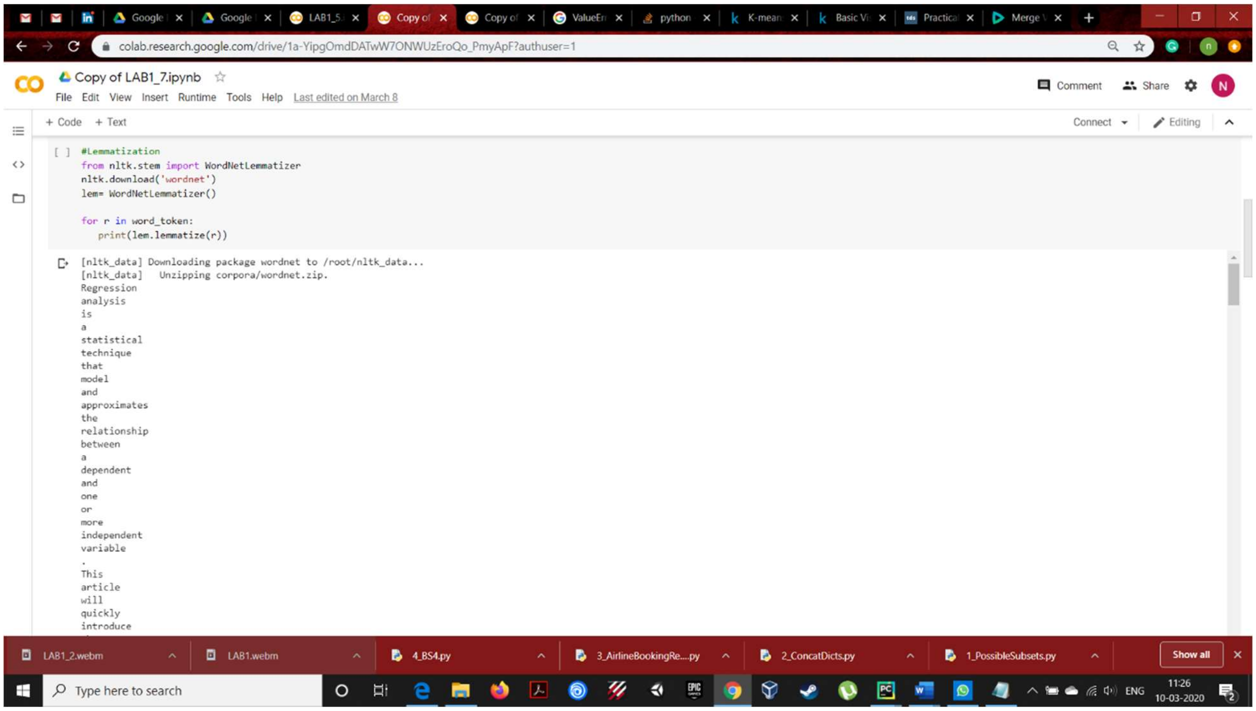
d. Used lemmatize function to apply lemmatization on each word



```
[34] #Tokenization
     import nltk

     nltk.download('punkt')
     file2 =open("/content/drive/My Drive/Colab Notebooks/nlp_input.txt","r",encoding="utf-8")
     #file2.decode('UTF8','replace')
     w=""
     w=file2.read()

     word_token=nltk.word_tokenize(w)
     for r in word_token:
        print(r)
```

```
more
independent
variables
.
This
article
will
quickly
introduce
three
commonly
used
regression
models
using
R
and
the
Boston
housing
data-set
:
Ridge
,
Lasso
,
```

```
Regression analysis is a statistical technique that models and approximates
First we need to understand the basics of regression and what parameters of
Visualization of the squared error (from Setosa.io)
The equation for this model is referred to as the cost function and is a wa
Equation for least ordinary squares
One situation is the data showing multi-collinearity, this is when predicto
To produce a more accurate model of complex data we can add a penalty term
Ridge regression uses L2 regularization which adds the following penalty te
L2 regularization penalty term
The L2 term is equal to the square of the magnitude of the coefficients. Ir
Lasso regression
Lasso regression uses the L1 penalty term and stands for Least Absolute Shr
L1 regularization penalty term
Similar to ridge regression, a lambda value of zero spits out the basic OLS
A third commonly used model of regression is the Elastic Net which incorpor
In addition to setting and choosing a lambda value elastic net also allows
As we mentioned in the previous sections, lambda values have a large effect
Here we perform a cross validation and take a peek at the lambda value corr
We can see here that certain coefficients have been pushed towards zero and
Performing Lasso regression
The steps will be identical to what we have done for ridge regression. The
Performing Elastic Net regression
Performing Elastic Net requires us to tune parameters to identify the best
We can see that the R mean-squared values using all three models were very
```



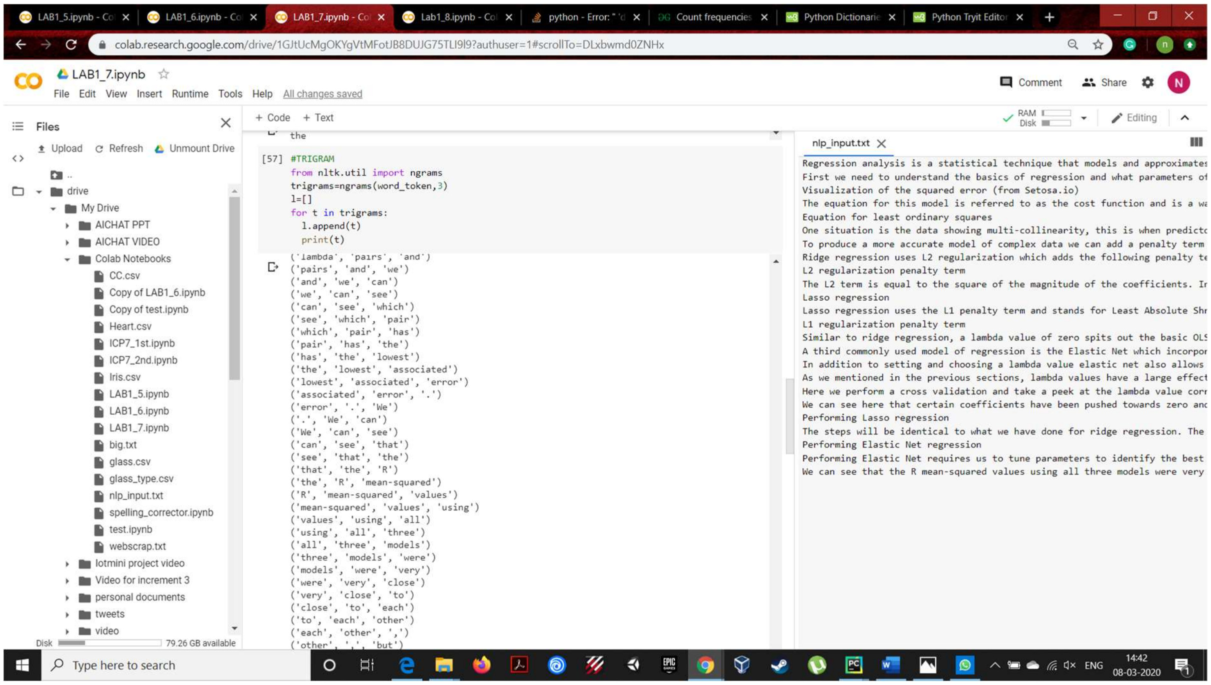
```
[ ] #Lemmatization
    from nltk.stem import WordNetLemmatizer
    nltk.download('wordnet')
    lem= WordNetLemmatizer()

    for r in word_token:
      print(lem.lemmatize(r))
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
Regression
analysis
is
a
statistical
technique
that
model
and
approximates
the
relationship
between
a
dependent
and
one
or
more
independent
variable
.
This
article
will
quickly
introduce
```

7c.

e. Used ngram fuction for n=3 to get all trigrams

f. Used FreqDist() class to find the all trigrams frequency



```
the
```

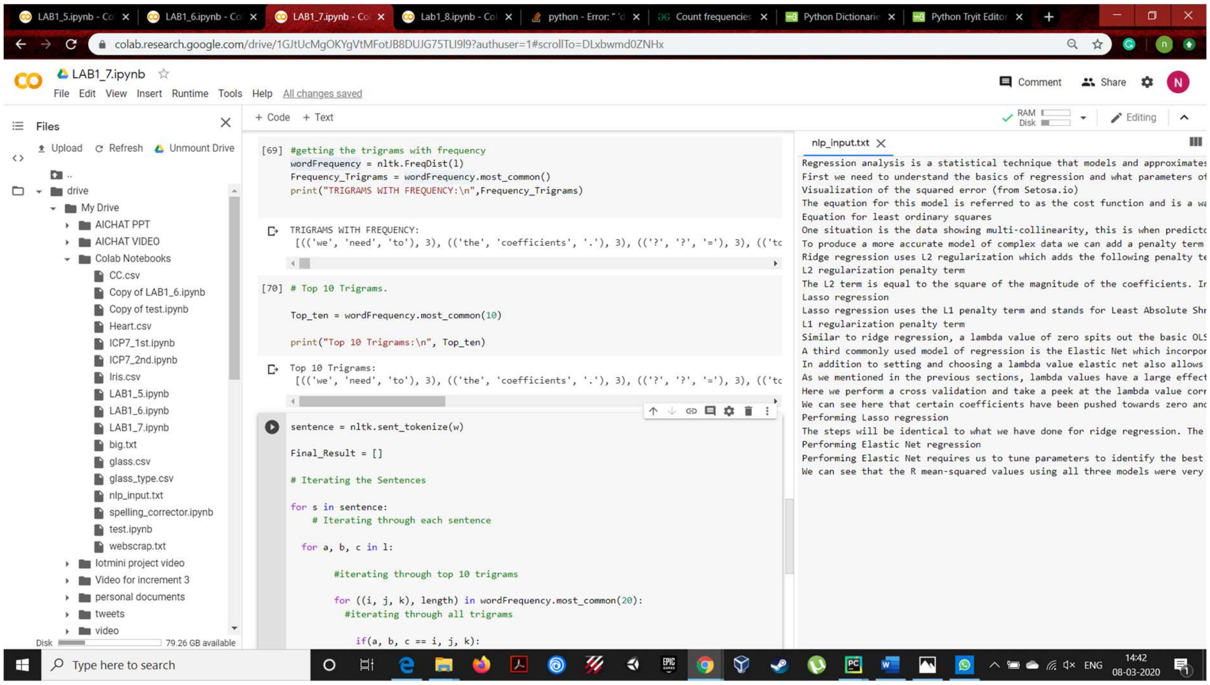
```
[57] #TRIGRAM
     from nltk.util import ngrams
     trigrams=ngrams(word_token,3)
     l=[]
     for t in trigrams:
       l.append(t)
       print(t)
```

```
('lambda', 'pairs', 'and')
('pairs', 'and', 'we')
('and', 'we', 'can')
('we', 'can', 'see')
('can', 'see', 'which')
('see', 'which', 'pair')
('which', 'pair', 'has')
('pair', 'has', 'the')
('has', 'the', 'lowest')
('the', 'lowest', 'associated')
('lowest', 'associated', 'error')
('associated', 'error', '.')
('error', '.', 'We')
('.', 'We', 'can')
('We', 'can', 'see')
('can', 'see', 'that')
('see', 'that', 'the')
('that', 'the', 'R')
('the', 'R', 'mean-squared')
('R', 'mean-squared', 'values')
('mean-squared', 'values', 'using')
('values', 'using', 'all')
('using', 'all', 'three')
('all', 'three', 'models')
('three', 'models', 'were')
('models', 'were', 'very')
('were', 'very', 'close')
('very', 'close', 'to')
('close', 'to', 'each')
('to', 'each', 'other')
('each', 'other', ',')
('other', ',', 'but')
```

```
Regression analysis is a statistical technique that models and approximates
First we need to understand the basics of regression and what parameters of
Visualization of the squared error (from Setosa.io)
The equation for this model is referred to as the cost function and is a wa
Equation for least ordinary squares
One situation is the data showing multi-collinearity, this is when predicto
To produce a more accurate model of complex data we can add a penalty term
Ridge regression uses L2 regularization which adds the following penalty te
L2 regularization penalty term
The L2 term is equal to the square of the magnitude of the coefficients. Ir
Lasso regression
Lasso regression uses the L1 penalty term and stands for Least Absolute Shr
L1 regularization penalty term
Similar to ridge regression, a lambda value of zero spits out the basic OLS
A third commonly used model of regression is the Elastic Net which incorpor
In addition to setting and choosing a lambda value elastic net also allows
As we mentioned in the previous sections, lambda values have a large effect
Here we perform a cross validation and take a peek at the lambda value corr
We can see here that certain coefficients have been pushed towards zero anc
Performing Lasso regression
The steps will be identical to what we have done for ridge regression. The
Performing Elastic Net regression
Performing Elastic Net requires us to tune parameters to identify the best
We can see that the R mean-squared values using all three models were very
```

7d.

g. Used most_common(10) function for extracting the top 10 trigrams based on their frequency



```
[69] #getting the trigrams with frequency
     wordFrequency = nltk.FreqDist(l)
     Frequency_Trigrams = wordFrequency.most_common()
     print("TRIGRAMS WITH FREQUENCY:\n",Frequency_Trigrams)
```

```
TRIGRAMS WITH FREQUENCY:
 [(('we', 'need', 'to'), 3), (('the', 'coefficients', '.'), 3), (('?', '?', '='), 3), (('tc
```

```
[70] # Top 10 Trigrams.

     Top_ten = wordFrequency.most_common(10)

     print("Top 10 Trigrams:\n", Top_ten)
```

```
Top 10 Trigrams:
 [(('we', 'need', 'to'), 3), (('the', 'coefficients', '.'), 3), (('?', '?', '='), 3), (('tc
```

```
sentence = nltk.sent_tokenize(w)

Final_Result = []

# Iterating the Sentences

for s in sentence:
    # Iterating through each sentence

  for a, b, c in l:

        #iterating through top 10 trigrams

        for ((i, j, k), length) in wordFrequency.most_common(20):
          #iterating through all trigrams

            if(a, b, c == i, j, k):
```

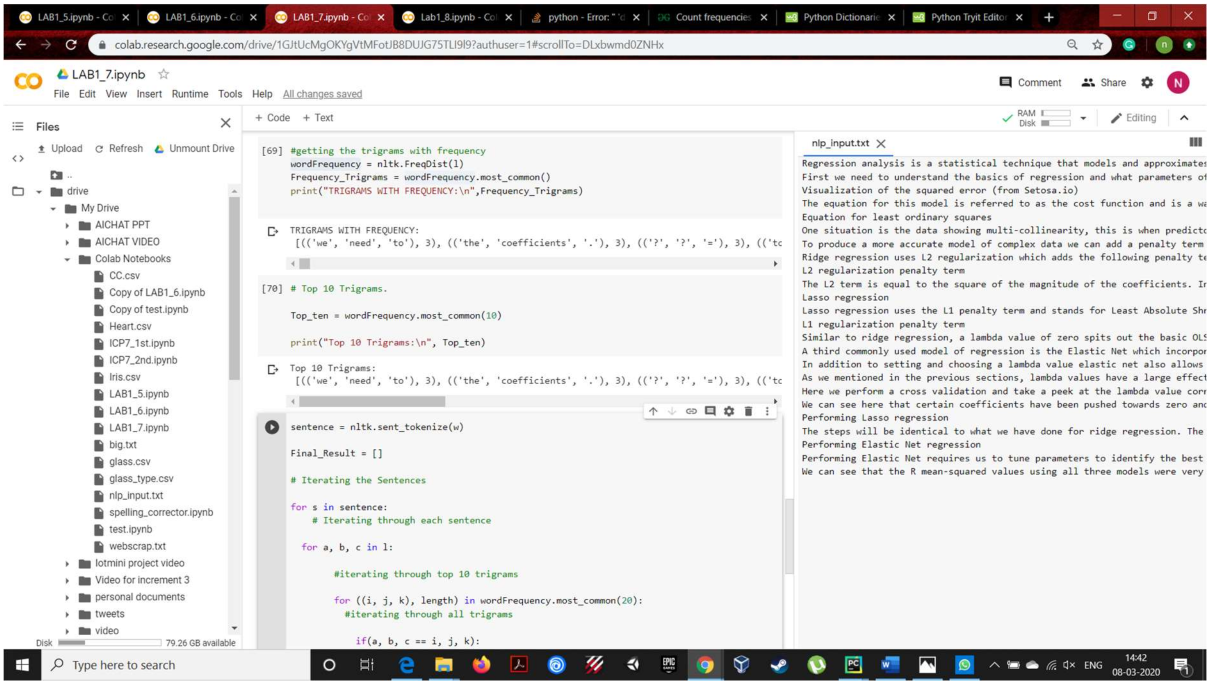
nlp_input.txt

```
Regression analysis is a statistical technique that models and approximates
First we need to understand the basics of regression and what parameters of
Visualization of the squared error (from Setosa.io)
The equation for this model is referred to as the cost function and is a wa
Equation for least ordinary squares
One situation is the data showing multi-collinearity, this is when predicto
To produce a more accurate model of complex data we can add a penalty term
Ridge regression uses L2 regularization which adds the following penalty te
L2 regularization penalty term
The L2 term is equal to the square of the magnitude of the coefficients. In
Lasso regression
Lasso regression uses the L1 penalty term and stands for Least Absolute Shr
L1 regularization penalty term
Similar to ridge regression, a lambda value of zero spits out the basic OLS
A third commonly used model of regression is the Elastic Net which incorpor
In addition to setting and choosing a lambda value elastic net also allows
As we mentioned in the previous sections, lambda values have a large effect
Here we perform a cross validation and take a peek at the lambda value corr
We can see here that certain coefficients have been pushed towards zero anc
Performing Lasso regression
The steps will be identical to what we have done for ridge regression. The
Performing Elastic Net regression
Performing Elastic Net requires us to tune parameters to identify the best
We can see that the R mean-squared values using all three models were very
```

7e,f,g,h .

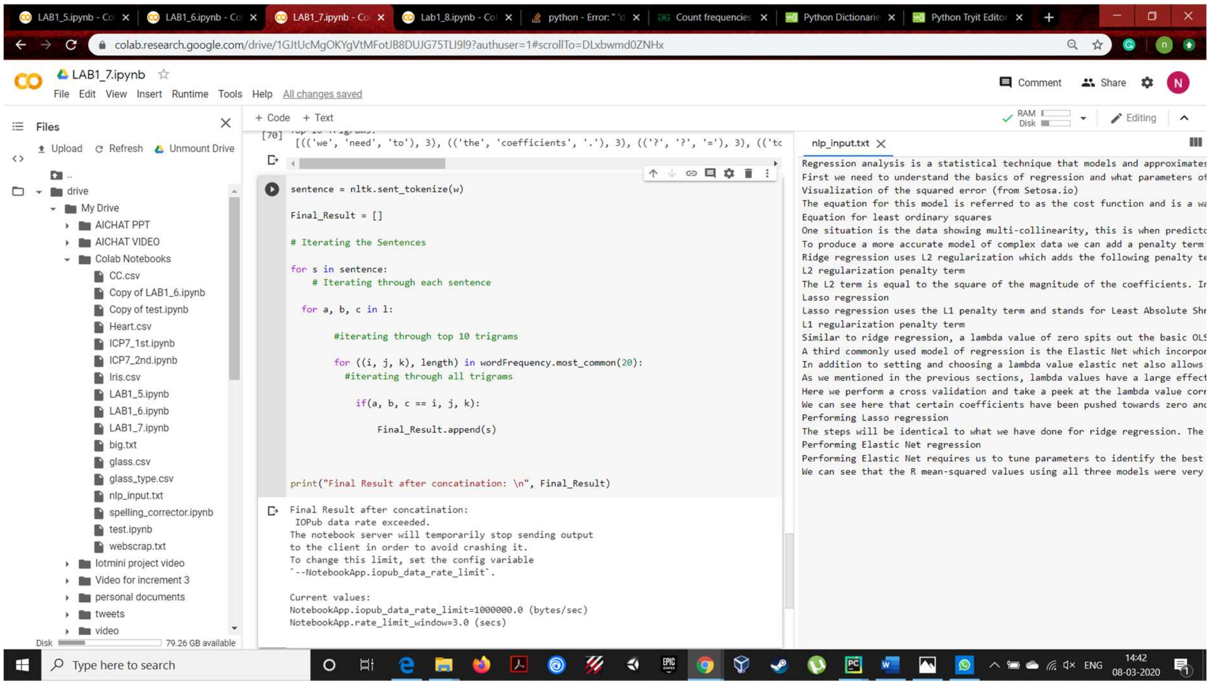
h. Used sent_tokenize to get each sentence

i. Iterated sentence with trigrams to get sentence with most trigrams with most frequency


LAB1_7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
Upload Refresh Unmount Drive
nlp_input.txt

```
[69] #getting the trigrams with frequency
     wordFrequency = nltk.FreqDist(l)
     Frequency_Trigrams = wordFrequency.most_common()
     print("TRIGRAMS WITH FREQUENCY:\n",Frequency_Trigrams)

     TRIGRAMS WITH FREQUENCY:
      [(('we', 'need', 'to'), 3), (('the', 'coefficients', '.'), 3), (('?', '?', '='), 3), (('tc

[70] # Top 10 Trigrams.

     Top_ten = wordFrequency.most_common(10)

     print("Top 10 Trigrams:\n", Top_ten)

     Top 10 Trigrams:
      [(('we', 'need', 'to'), 3), (('the', 'coefficients', '.'), 3), (('?', '?', '='), 3), (('tc

     sentence = nltk.sent_tokenize(w)

     Final_Result = []

     # Iterating the Sentences

     for s in sentence:
         # Iterating through each sentence

       for a, b, c in l:

             #iterating through top 10 trigrams

             for ((i, j, k), length) in wordFrequency.most_common(20):
               #iterating through all trigrams

                 if(a, b, c == i, j, k):
```

```
Regression analysis is a statistical technique that models and approximates
First we need to understand the basics of regression and what parameters of
Visualization of the squared error (from Setosa.io)
The equation for this model is referred to as the cost function and is a wa
Equation for least ordinary squares
One situation is the data showing multi-collinearity, this is when predicto
To produce a more accurate model of complex data we can add a penalty term
Ridge regression uses L2 regularization which adds the following penalty te
L2 regularization penalty term
The L2 term is equal to the square of the magnitude of the coefficients. In
Lasso regression
Lasso regression uses the L1 penalty term and stands for Least Absolute Shr
L1 regularization penalty term
Similar to ridge regression, a lambda value of zero spits out the basic OLS
A third commonly used model of regression is the Elastic Net which incorpor
In addition to setting and choosing a lambda value elastic net also allows
As we mentioned in the previous sections, lambda values have a large effect
Here we perform a cross validation and take a peek at the lambda value corr
We can see here that certain coefficients have been pushed towards zero and
Performing Lasso regression
The steps will be identical to what we have done for ridge regression. The
Performing Elastic Net regression
Performing Elastic Net requires us to tune parameters to identify the best
We can see that the R mean-squared values using all three models were very
```



```
[70]  [(('we', 'need', 'to'), 3), (('the', 'coefficients', '.'), 3), (('?', '?', '='), 3), (('tc

     sentence = nltk.sent_tokenize(w)

     Final_Result = []

     # Iterating the Sentences

     for s in sentence:
         # Iterating through each sentence

       for a, b, c in l:

             #iterating through top 10 trigrams

             for ((i, j, k), length) in wordFrequency.most_common(20):
               #iterating through all trigrams

                 if(a, b, c == i, j, k):

                     Final_Result.append(s)



     print("Final Result after concatination: \n", Final_Result)

     Final Result after concatination:
      IOPub data rate exceeded.
     The notebook server will temporarily stop sending output
     to the client in order to avoid crashing it.
     To change this limit, set the config variable
     `--NotebookApp.iopub_data_rate_limit`.

     Current values:
     NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
     NotebookApp.rate_limit_window=3.0 (secs)
```

8. Multiple Regression (Calculating RMSE, and R2)

a. Used winequality-red.csv file

b. Dropped predicting quality from training data

c. Used LinearRegression class to for fitting the model

d. Calculated the RMSE score and R^2 score (before cleaning the data)

e. Cleaned the data and filled all the null values with value

f. Used fillna to fill all null values with mean value(adding noise)

g. Fitted the model again with cleaned data

h. calculated the RMSE and R^2 score (After cleaning data)

Note:

Before cleaning the data

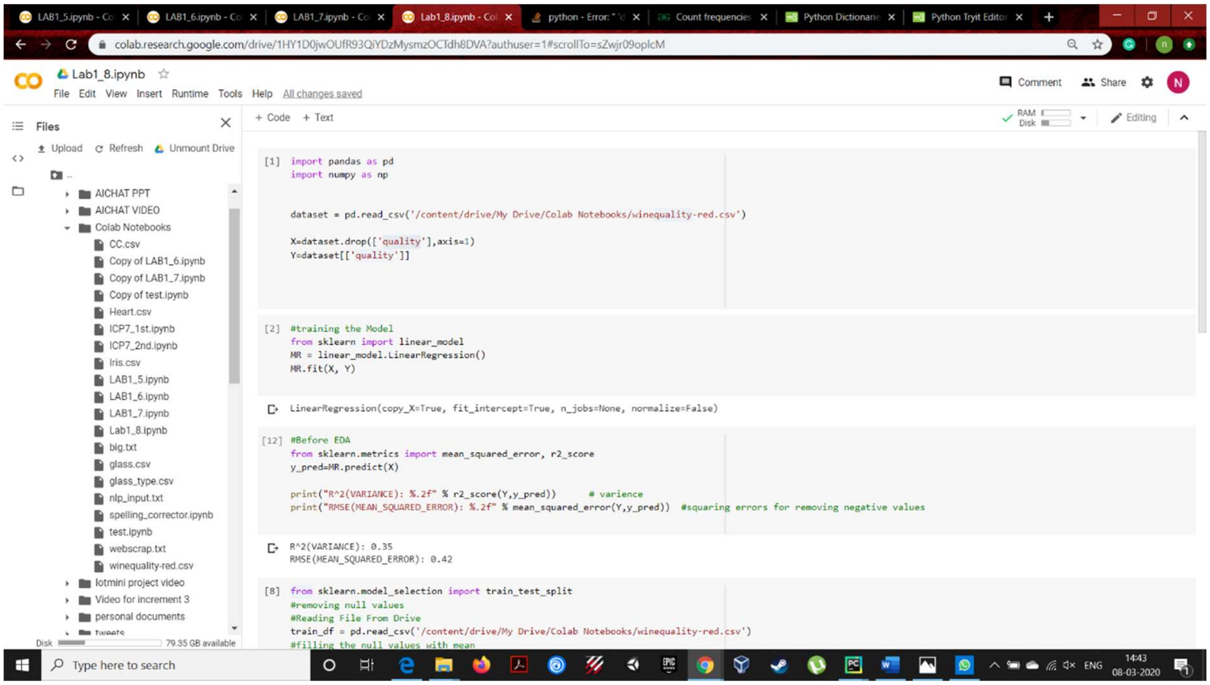
RMSE : 0.42

R^2 score: 0.35

After Cleaning the data

RMSE : 0.42

R^2 score: 0.36



colab.research.google.com/drive/1HY1D0jwOUfR93QiYDzMysmzOCTdh8DVA?authuser=1#scrollTo=sZwjr09oplcM

Lab1_8.ipynb

All changes saved

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


    dataset = pd.read_csv('/content/drive/My Drive/Colab Notebooks/winequality-red.csv')

    X=dataset.drop(['quality'],axis=1)
    Y=dataset[['quality']]
```

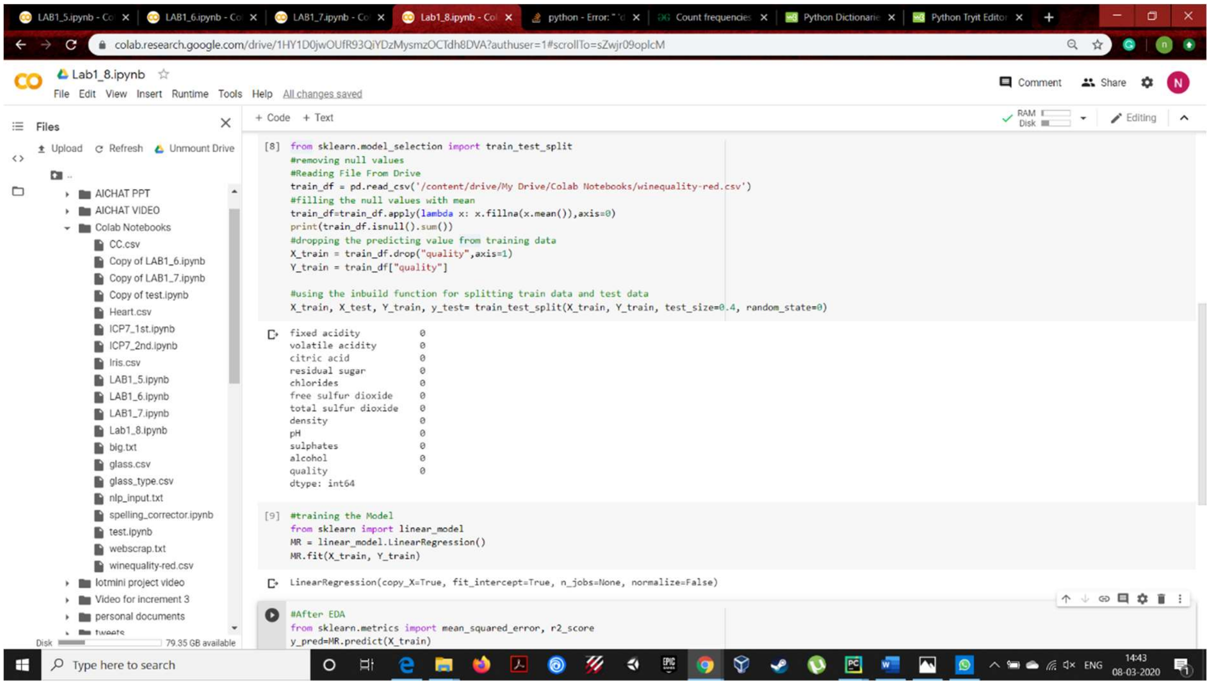
```
[2] #training the Model
    from sklearn import linear_model
    MR = linear_model.LinearRegression()
    MR.fit(X, Y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
[12] #Before EDA
     from sklearn.metrics import mean_squared_error, r2_score
     y_pred=MR.predict(X)

     print("R^2(VARIANCE): %.2f" % r2_score(Y,y_pred))       # varience
     print("RMSE(MEAN_SQUARED_ERROR): %.2f" % mean_squared_error(Y,y_pred))  #squaring errors for removing negative values
```

```
R^2(VARIANCE): 0.35
RMSE(MEAN_SQUARED_ERROR): 0.42
```

```
[8] from sklearn.model_selection import train_test_split
    #removing null values
    #Reading File From Drive
    train_df = pd.read_csv('/content/drive/My Drive/Colab Notebooks/winequality-red.csv')
    #filling the null values with mean
```



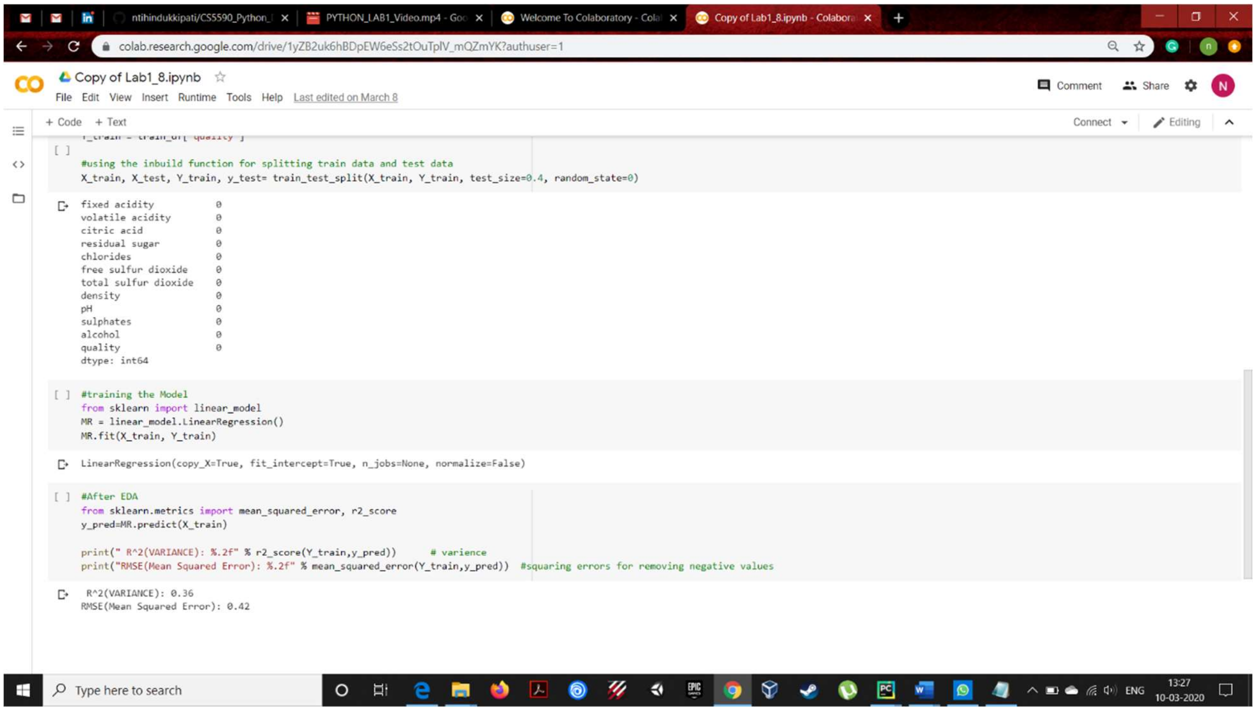
```
[8] from sklearn.model_selection import train_test_split
    #removing null values
    #Reading File From Drive
    train_df = pd.read_csv('/content/drive/My Drive/Colab Notebooks/winequality-red.csv')
    #filling the null values with mean
    train_df=train_df.apply(lambda x: x.fillna(x.mean()),axis=0)
    print(train_df.isnull().sum())
    #dropping the predicting value from training data
    X_train = train_df.drop("quality",axis=1)
    Y_train = train_df["quality"]

    #using the inbuild function for splitting train data and test data
    X_train, X_test, Y_train, y_test= train_test_split(X_train, Y_train, test_size=0.4, random_state=0)
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

```
[9] #training the Model
    from sklearn import linear_model
    MR = linear_model.LinearRegression()
    MR.fit(X_train, Y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
#After EDA
from sklearn.metrics import mean_squared_error, r2_score
y_pred=MR.predict(X_train)
```


colab.research.google.com/drive/1yZB2uk6hBDpEW6eSs2tOuTplV_mQZmYK?authuser=1
Copy of Lab1_8.ipynb
File Edit View Insert Runtime Tools Help Last edited on March 8
Type here to search
13:27
10-03-2020

```
[ ] #using the inbuild function for splitting train data and test data
    X_train, X_test, Y_train, y_test= train_test_split(X_train, Y_train, test_size=0.4, random_state=0)
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

```
[ ] #training the Model
    from sklearn import linear_model
    MR = linear_model.LinearRegression()
    MR.fit(X_train, Y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
[ ] #After EDA
    from sklearn.metrics import mean_squared_error, r2_score
    y_pred=MR.predict(X_train)

    print(" R^2(VARIANCE): %.2f" % r2_score(Y_train,y_pred))       # varience
    print("RMSE(Mean Squared Error): %.2f" % mean_squared_error(Y_train,y_pred))  #squaring errors for removing negative values
```

```
 R^2(VARIANCE): 0.36
RMSE(Mean Squared Error): 0.42
```

DATASETS USED:

- Glass_type.csv
- Cc.csv

- Winequality-red.csv

EVALUATION & DISCUSSION:

1. A set of integers were taken and all the possible subsets without the empty set were printed.
2. Two dictionaries were taken and concatenation and sorting were applied to the merged dictionary.
3. Airline Booking Reservation System was build using the concepts of class, inheritance and method overriding.
4. By using the BeautifulSoup, webscrapping was performed, all the course names and their descriptions were displayed.
5. Take random data set, clean data, split the data, fit and evaluate the accuracy for each of three classifiers (KNN, Naive Bayes, SVM).
6. Take random data set, clean the data, use kmeans function for iterating through all values and get best k value, plot a graph to find elbow, fit model with best k value, and calculate the silhouette score
7. Take the given text file, use the in-build functions to tokenize the data, lemmatize, Find all trigrams using ngrams function where n=3, freqDest() to get frequency, most_common(10) to get top ten trigrams and sentence tokenize to tokenize each sentence and get sentence with words with more frequency of trigrams.
8. Take the random dataset clean the data, remove predicting column from training data set, fit the model, calculate the RMSE, R^2 score, and find the score without cleaning the data.

CONSLUSION:

All the programs were successfully executed according to the objectives specified and have met the evaluation criteria.