Python Programming LAB1

Report

Submitted By:

Dhairya Chandra (6)
Tanvi Jain (22)

Course:

CSEE5590/490

INTRODUCTION

This lab assignment focused on how to make us familiar with python basic topicsThis lab assignment focused on how to make us familiar with the basics of python

The Dataset we used in completing the Lab Assignment was:

1. Heart Disease UCI - https://www.kaggle.com/ronitf/heart-disease-uci
2. Wine Quality - https://archive.ics.uci.edu/ml/datasets/Wine+Quality

OBJECTIVE

Main objective is to learn the basics of python and using multiple packages like beautifulsoup, nltk, etc. Also learned how to classification on data by splitting data into test and train.

Here is the list of Questions:

1. Given a collection of integers that might contain duplicates, nums, return all possible subsets
2. Concatenate two dictionaries and sort the concatenated dictionary by value.
3. Airline Booking Reservation System
4. fetch the course name and overview of course using beautiful soup
5. Perform 3 classification algorithms on dataset.
6. Apply K-means on the dataset and also find silhouette score, elbow score.
7. Perform Natural Language Processing on an input file
8. Do Multiple Regression on a dataset. Evaluate both RMSE and R2.

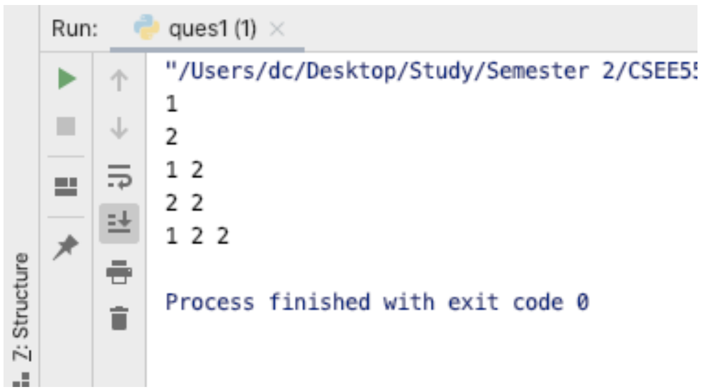
QUESTION 1

Given a collection of integers that might contain duplicates, nums, return all possible subsets. Do not include null subset.

SOLUTION

```python
def subset(arr, n):
    _list = []

    for i in range(2 ** n):
        subset = ""
        # consider each element in the set
        for j in range(n):
            if (i & (1 << j)) != 0:
                subset += str(arr[j]) + " "
        # if subset is encountered for the first time
        if subset not in _list and len(subset) > 0:
            _list.append(subset)
            # consider every subset
    for subset in _list:
        # split the subset and print its elements
        arr = subset.split(' ')
        for string in arr:
            print(string, end=" ")
        print()

if __name__ == '__main__':
    arr = [1, 2, 2]
    n = len(arr)
    subset(arr, n)
```

OUTPUT



```
"/Users/dc/Desktop/Study/Semester 2/CSEE5
1
2
1 2
2 2
1 2 2

Process finished with exit code 0
```

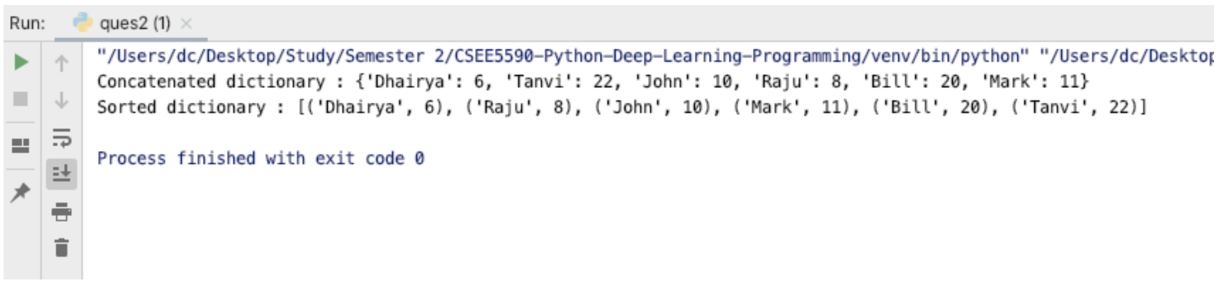
QUESTION 2

Concatenate two dictionaries and sort the concatenated dictionary by value.

SOLUTION

```python
# Create first dictionary
dict1 = {'Dhairya': 6, 'Tanvi': 22, 'John': 10}

# Create second dictionary
dict2 = {'Raju': 8, 'Bill': 20, 'Mark': 11}

# Concatenating two dictionaries
dict1.update(dict2)

# Printing the concatenated dictionary
print('Concatenated dictionary :', dict1)

# Sorting dictionaries by values
sortdict =  sorted(dict1.items(), key=lambda x: x[1])

# Printing the sorted dictionary
print('Sorted dictionary :', sortdict)
```

OUTPUT

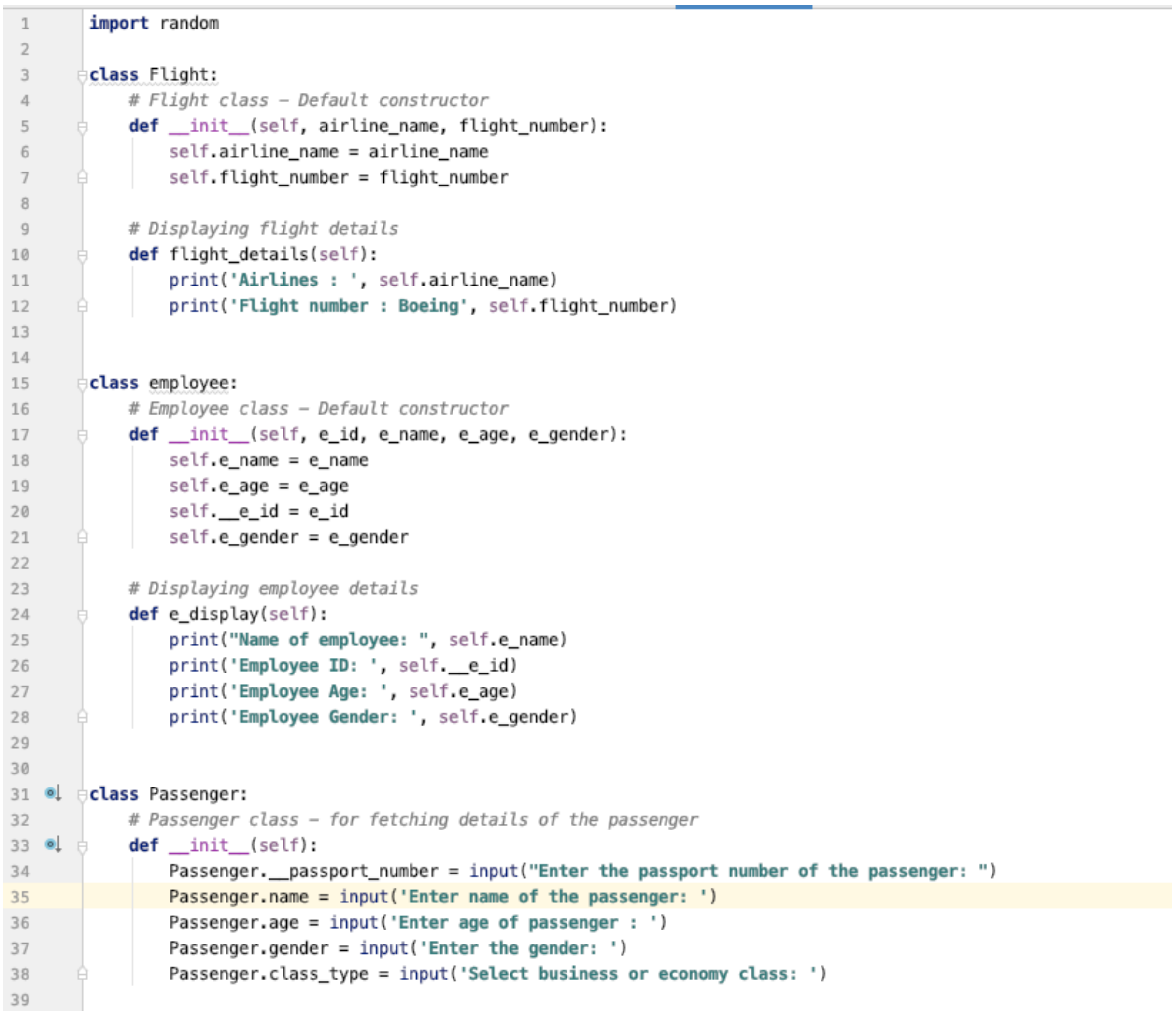


```
"/Users/dc/Desktop/Study/Semester 2/CSEE5590-Python-Deep-Learning-Programming/venv/bin/python" "/Users/dc/Desktop
Concatenated dictionary : {'Dhairya': 6, 'Tanvi': 22, 'John': 10, 'Raju': 8, 'Bill': 20, 'Mark': 11}
Sorted dictionary : [('Dhairya', 6), ('Raju', 8), ('John', 10), ('Mark', 11), ('Bill', 20), ('Tanvi', 22)]

Process finished with exit code 0
```

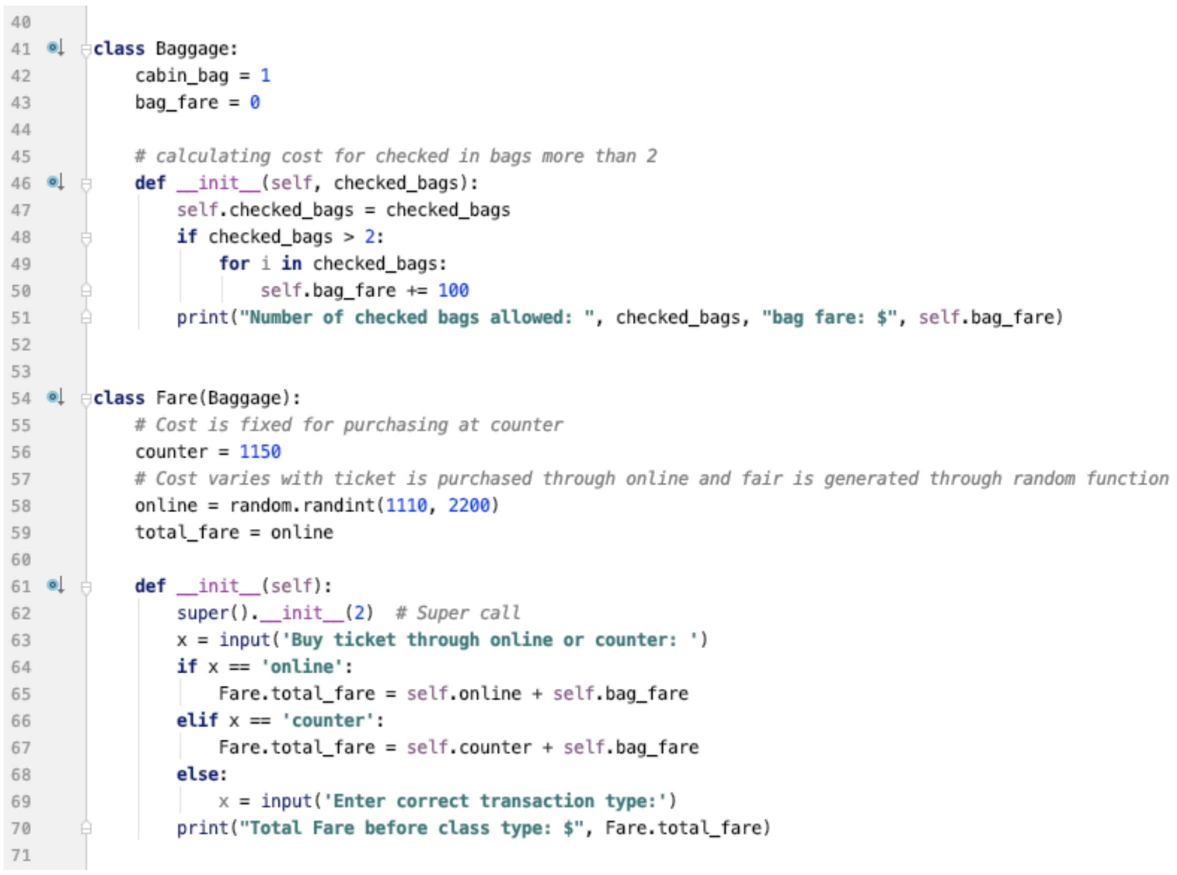
QUESTION 3

Airline booking reservation system

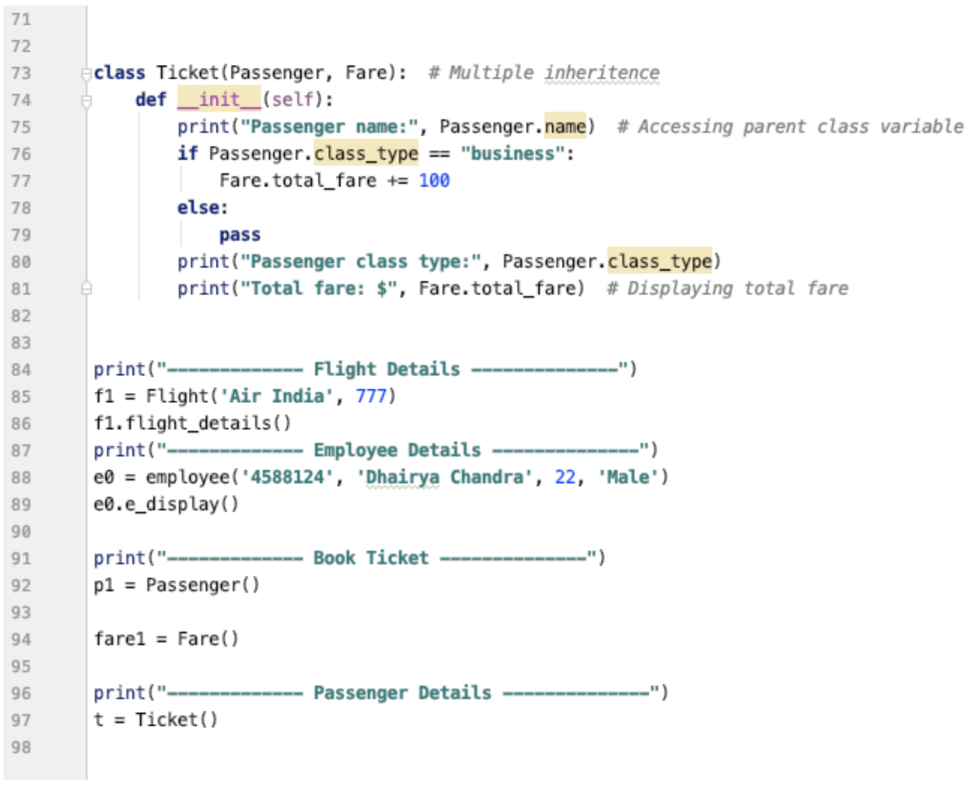
SOLUTION



```python
import random

class Flight:
    # Flight class - Default constructor
    def __init__(self, airline_name, flight_number):
        self.airline_name = airline_name
        self.flight_number = flight_number

    # Displaying flight details
    def flight_details(self):
        print('Airlines : ', self.airline_name)
        print('Flight number : Boeing', self.flight_number)


class employee:
    # Employee class - Default constructor
    def __init__(self, e_id, e_name, e_age, e_gender):
        self.e_name = e_name
        self.e_age = e_age
        self.__e_id = e_id
        self.e_gender = e_gender

    # Displaying employee details
    def e_display(self):
        print("Name of employee: ", self.e_name)
        print('Employee ID: ', self.__e_id)
        print('Employee Age: ', self.e_age)
        print('Employee Gender: ', self.e_gender)


class Passenger:
    # Passenger class - for fetching details of the passenger
    def __init__(self):
        Passenger.__passport_number = input("Enter the passport number of the passenger: ")
        Passenger.name = input('Enter name of the passenger: ')
        Passenger.age = input('Enter age of passenger : ')
        Passenger.gender = input('Enter the gender: ')
        Passenger.class_type = input('Select business or economy class: ')

```



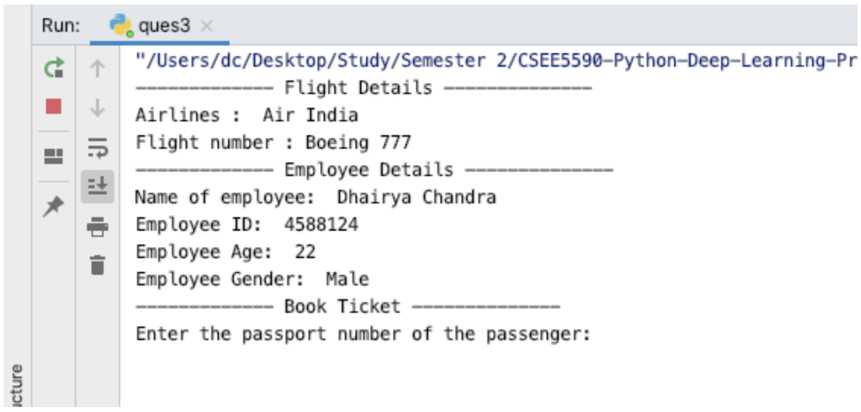
```python

class Baggage:
    cabin_bag = 1
    bag_fare = 0

    # calculating cost for checked in bags more than 2
    def __init__(self, checked_bags):
        self.checked_bags = checked_bags
        if checked_bags > 2:
            for i in checked_bags:
                self.bag_fare += 100
        print("Number of checked bags allowed: ", checked_bags, "bag fare: $", self.bag_fare)


class Fare(Baggage):
    # Cost is fixed for purchasing at counter
    counter = 1150
    # Cost varies with ticket is purchased through online and fair is generated through random function
    online = random.randint(1110, 2200)
    total_fare = online

    def __init__(self):
        super().__init__(2)  # Super call
        x = input('Buy ticket through online or counter: ')
        if x == 'online':
            Fare.total_fare = self.online + self.bag_fare
        elif x == 'counter':
            Fare.total_fare = self.counter + self.bag_fare
        else:
            x = input('Enter correct transaction type:')
        print("Total Fare before class type: $", Fare.total_fare)

```



```python


class Ticket(Passenger, Fare):  # Multiple inheritence
    def __init__(self):
        print("Passenger name:", Passenger.name)  # Accessing parent class variable
        if Passenger.class_type == "business":
            Fare.total_fare += 100
        else:
            pass
        print("Passenger class type:", Passenger.class_type)
        print("Total fare: $", Fare.total_fare)  # Displaying total fare


print("------------- Flight Details --------------")
f1 = Flight('Air India', 777)
f1.flight_details()
print("------------- Employee Details --------------")
e0 = employee('4588124', 'Dhairya Chandra', 22, 'Male')
e0.e_display()

print("------------- Book Ticket --------------")
p1 = Passenger()

fare1 = Fare()

print("------------- Passenger Details --------------")
t = Ticket()

```

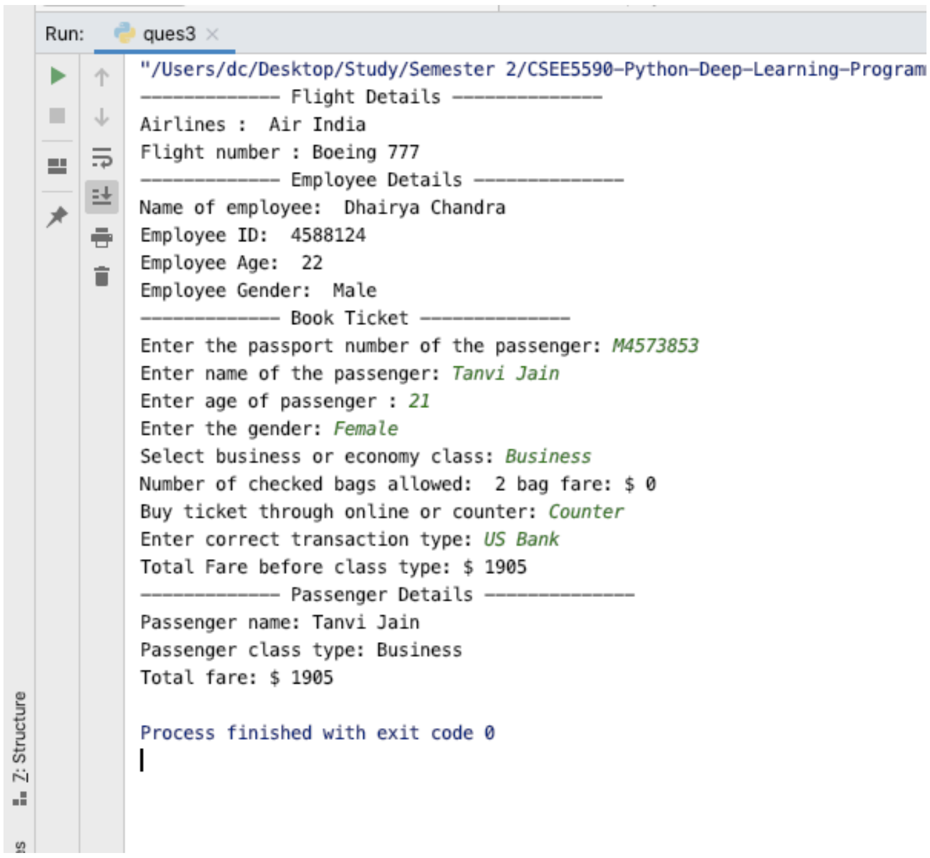
OUTPUT

This will be showed by default:



```
"/Users/dc/Desktop/Study/Semester 2/CSEE5590-Python-Deep-Learning-Pr
------------- Flight Details --------------
Airlines :  Air India
Flight number : Boeing 777
------------- Employee Details --------------
Name of employee:  Dhairya Chandra
Employee ID:  4588124
Employee Age:  22
Employee Gender:  Male
------------- Book Ticket --------------
Enter the passport number of the passenger:
```

Now employee has to enter information of passenger so that ticket get generated:



```
"/Users/dc/Desktop/Study/Semester 2/CSEE5590-Python-Deep-Learning-Program
------------- Flight Details --------------
Airlines :  Air India
Flight number : Boeing 777
------------- Employee Details --------------
Name of employee:  Dhairya Chandra
Employee ID:  4588124
Employee Age:  22
Employee Gender:  Male
------------- Book Ticket --------------
Enter the passport number of the passenger: M4573853
Enter name of the passenger: Tanvi Jain
Enter age of passenger : 21
Enter the gender: Female
Select business or economy class: Business
Number of checked bags allowed:  2 bag fare: $ 0
Buy ticket through online or counter: Counter
Enter correct transaction type: US Bank
Total Fare before class type: $ 1905
------------- Passenger Details --------------
Passenger name: Tanvi Jain
Passenger class type: Business
Total fare: $ 1905

Process finished with exit code 0
```

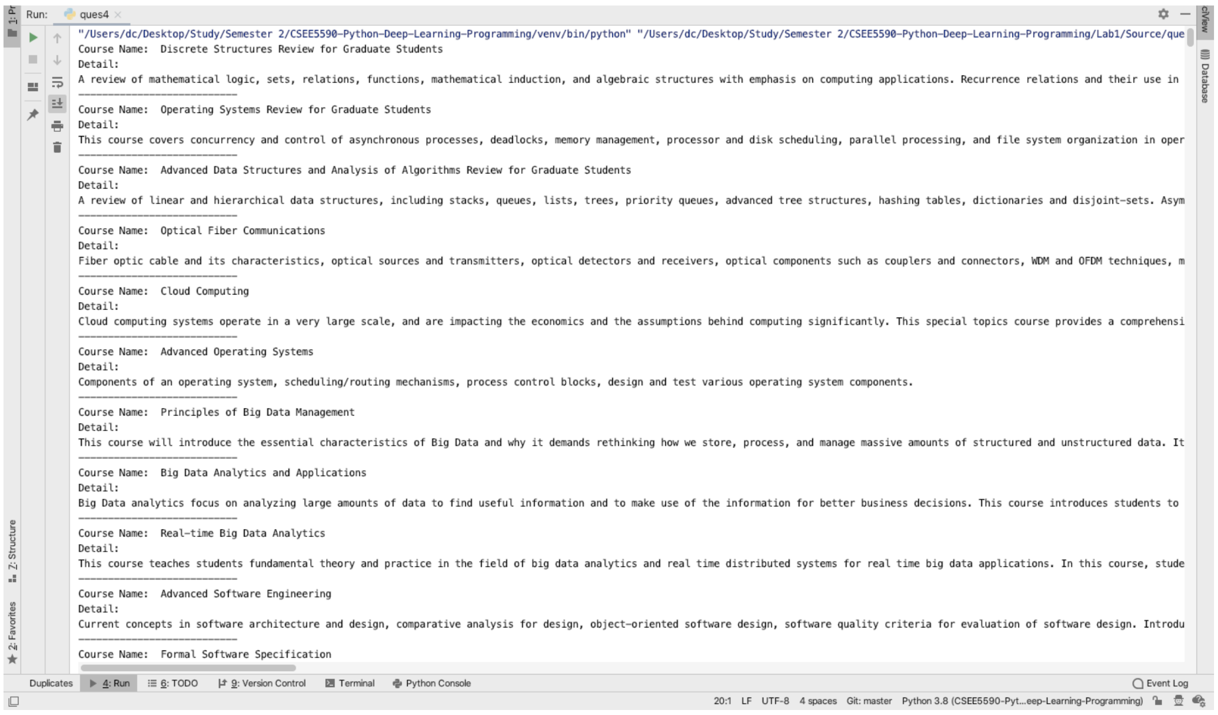
QUESTION 4

fetching the course name and overview of course using Beautiful Soup

SOLUTION

```python
from bs4 import BeautifulSoup
import requests

# Enter URL from where you have to fetch the data
url  = requests.get("https://catalog.umkc.edu/course-offerings/graduate/comp-sci/")
data = url.text
soup = BeautifulSoup(data, "html.parser")

# Using FindAll to find specific class

for p in soup.findAll('p',{'class':'courseblocktitle'}):
    for title, detail in zip(soup.findAll('span', {'class': 'title'}),
                             soup.findAll('p', {'class': 'courseblockdesc'})):

        # Print each course detail
        print ("Course Name: ", title.string)
        print("Detail: ", detail.string)
        print("---------------------------")
```

OUTPUT



```
"/Users/dc/Desktop/Study/Semester 2/CSEE5590-Python-Deep-Learning-Programming/venv/bin/python" "/Users/dc/Desktop/Study/Semester 2/CSEE5590-Python-Deep-Learning-Programming/Lab1/Source/que
Course Name:  Discrete Structures Review for Graduate Students
Detail:
A review of mathematical logic, sets, relations, functions, mathematical induction, and algebraic structures with emphasis on computing applications. Recurrence relations and their use in
---------------------------
Course Name:  Operating Systems Review for Graduate Students
Detail:
This course covers concurrency and control of asynchronous processes, deadlocks, memory management, processor and disk scheduling, parallel processing, and file system organization in oper
---------------------------
Course Name:  Advanced Data Structures and Analysis of Algorithms Review for Graduate Students
Detail:
A review of linear and hierarchical data structures, including stacks, queues, lists, trees, priority queues, advanced tree structures, hashing tables, dictionaries and disjoint-sets. Asym
---------------------------
Course Name:  Optical Fiber Communications
Detail:
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, m
---------------------------
Course Name:  Cloud Computing
Detail:
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 comprehensi
---------------------------
Course Name:  Advanced Operating Systems
Detail:
Components of an operating system, scheduling/routing mechanisms, process control blocks, design and test various operating system components.
---------------------------
Course Name:  Principles of Big Data Management
Detail:
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
---------------------------
Course Name:  Big Data Analytics and Applications
Detail:
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
---------------------------
Course Name:  Real-time Big Data Analytics
Detail:
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, stude
---------------------------
Course Name:  Advanced Software Engineering
Detail:
Current concepts in software architecture and design, comparative analysis for design, object-oriented software design, software quality criteria for evaluation of software design. Introdu
---------------------------
Course Name:  Formal Software Specification
```

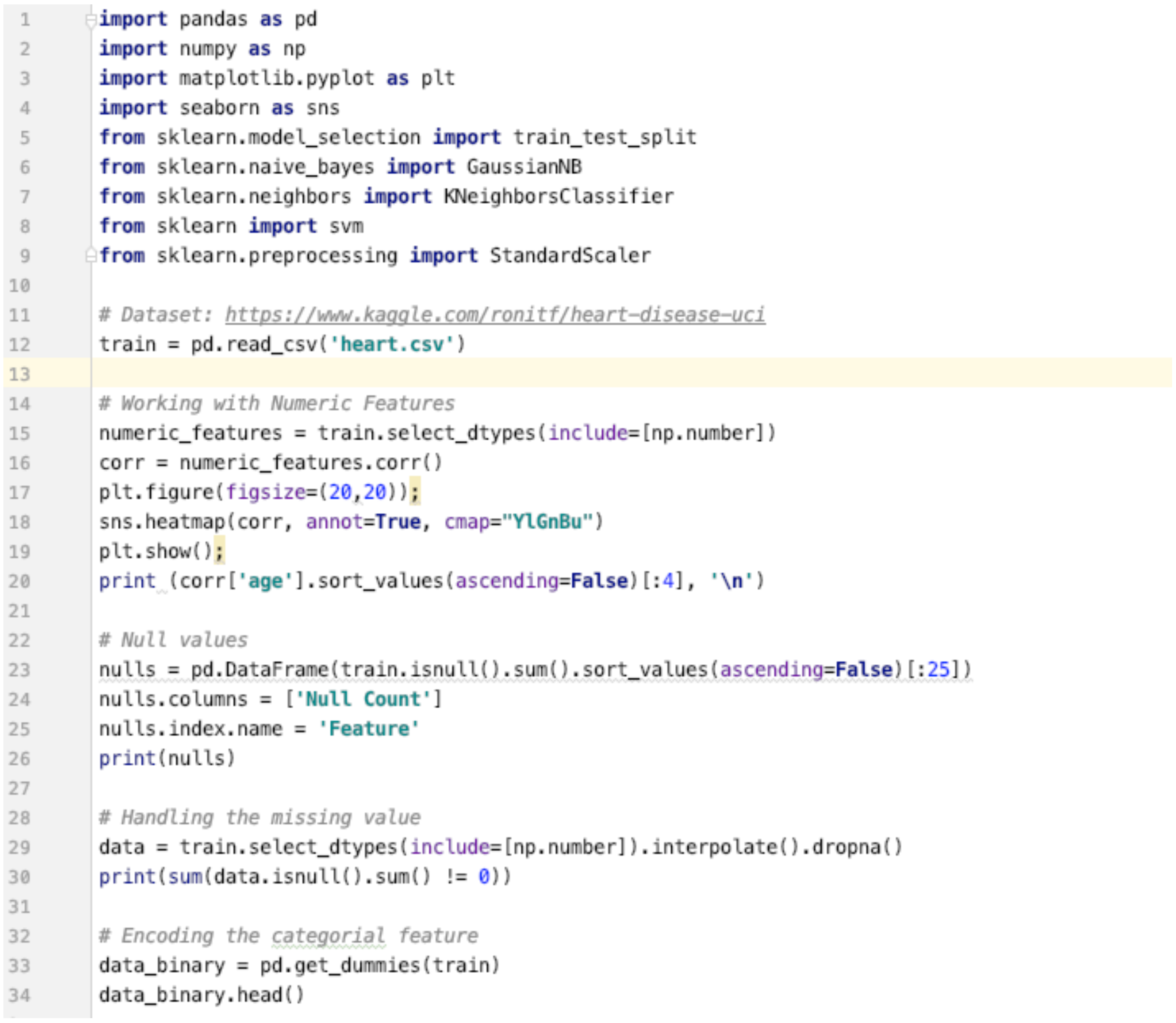
```
20:1  LF  UTF-8  4 spaces  Git: master  Python 3.8 (CSEE5590-Pyt...eep-Learning-Programming)
```

QUESTION 5

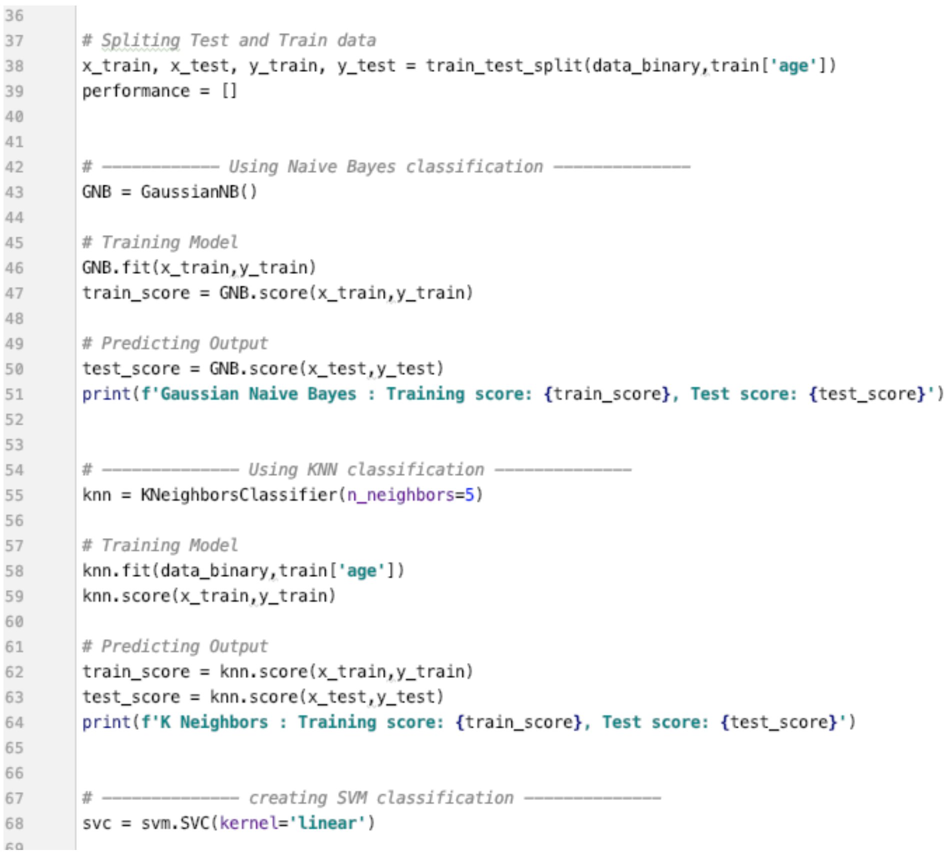
Perform three classification algorithms Naïve Bayes, SVM and KNN on any dataset.

SOLUTION

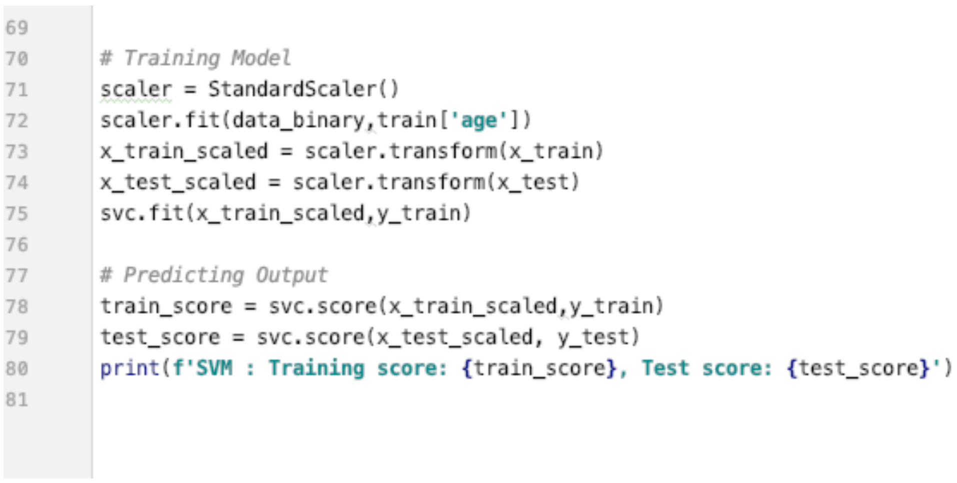
Dataset Used is **Heart Disease UCI** (https://www.kaggle.com/ronitf/heart-disease-uci)



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.preprocessing import StandardScaler

# Dataset: https://www.kaggle.com/ronitf/heart-disease-uci
train = pd.read_csv('heart.csv')

# Working with Numeric Features
numeric_features = train.select_dtypes(include=[np.number])
corr = numeric_features.corr()
plt.figure(figsize=(20,20));
sns.heatmap(corr, annot=True, cmap="YlGnBu")
plt.show();
print (corr['age'].sort_values(ascending=False)[:4], '\n')

# Null values
nulls = pd.DataFrame(train.isnull().sum().sort_values(ascending=False)[:25])
nulls.columns = ['Null Count']
nulls.index.name = 'Feature'
print(nulls)

# Handling the missing value
data = train.select_dtypes(include=[np.number]).interpolate().dropna()
print(sum(data.isnull().sum() != 0))

# Encoding the categorial feature
data_binary = pd.get_dummies(train)
data_binary.head()
```

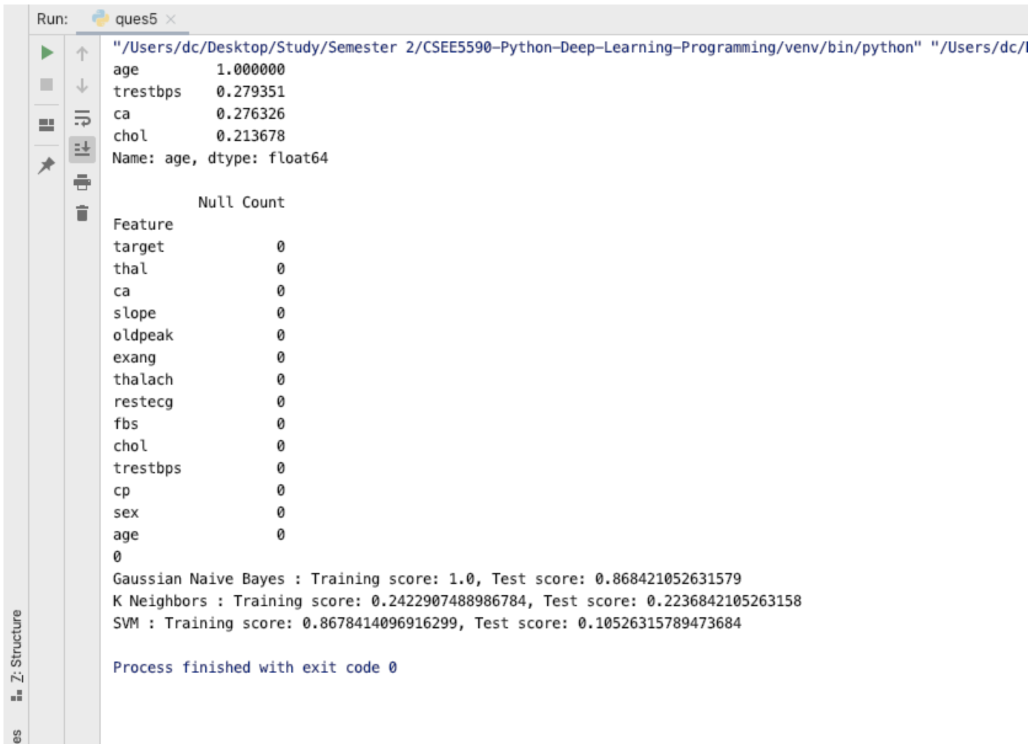


```python

# Spliting Test and Train data
x_train, x_test, y_train, y_test = train_test_split(data_binary,train['age'])
performance = []


# ------------ Using Naive Bayes classification --------------
GNB = GaussianNB()

# Training Model
GNB.fit(x_train,y_train)
train_score = GNB.score(x_train,y_train)

# Predicting Output
test_score = GNB.score(x_test,y_test)
print(f'Gaussian Naive Bayes : Training score: {train_score}, Test score: {test_score}')


# -------------- Using KNN classification --------------
knn = KNeighborsClassifier(n_neighbors=5)

# Training Model
knn.fit(data_binary,train['age'])
knn.score(x_train,y_train)

# Predicting Output
train_score = knn.score(x_train,y_train)
test_score = knn.score(x_test,y_test)
print(f'K Neighbors : Training score: {train_score}, Test score: {test_score}')


# -------------- creating SVM classification --------------
svc = svm.SVC(kernel='linear')
```

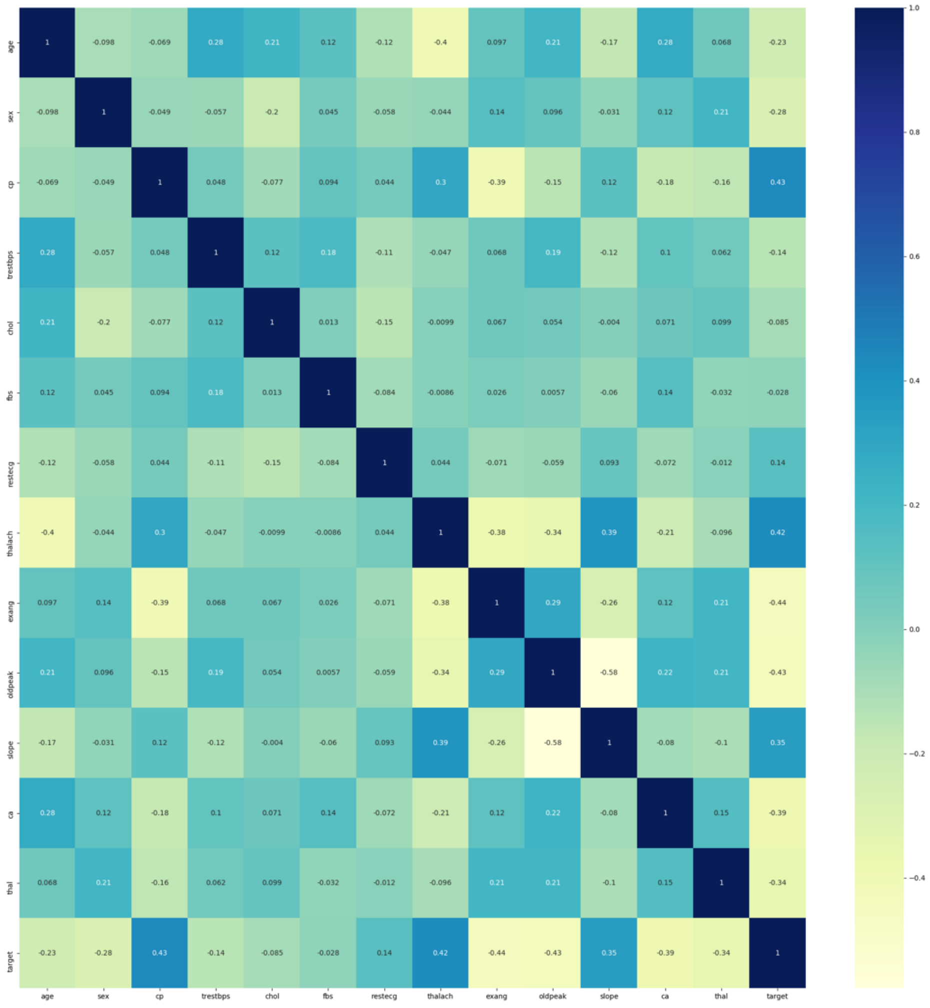


```python

# Training Model
scaler = StandardScaler()
scaler.fit(data_binary,train['age'])
x_train_scaled = scaler.transform(x_train)
x_test_scaled = scaler.transform(x_test)
svc.fit(x_train_scaled,y_train)

# Predicting Output
train_score = svc.score(x_train_scaled,y_train)
test_score = svc.score(x_test_scaled, y_test)
print(f'SVM : Training score: {train_score}, Test score: {test_score}')

```

OUTPUT



```
"/Users/dc/Desktop/Study/Semester 2/CSEE5590-Python-Deep-Learning-Programming/venv/bin/python" "/Users/dc/I
age         1.000000
trestbps    0.279351
ca          0.276326
chol        0.213678
Name: age, dtype: float64

          Null Count
Feature
target             0
thal               0
ca                 0
slope              0
oldpeak            0
exang              0
thalach            0
restecg            0
fbs                0
chol               0
trestbps           0
cp                 0
sex                0
age                0
0
Gaussian Naive Bayes : Training score: 1.0, Test score: 0.868421052631579
K Neighbors : Training score: 0.2422907488986784, Test score: 0.2236842105263158
SVM : Training score: 0.8678414096916299, Test score: 0.10526315789473684

Process finished with exit code 0
```



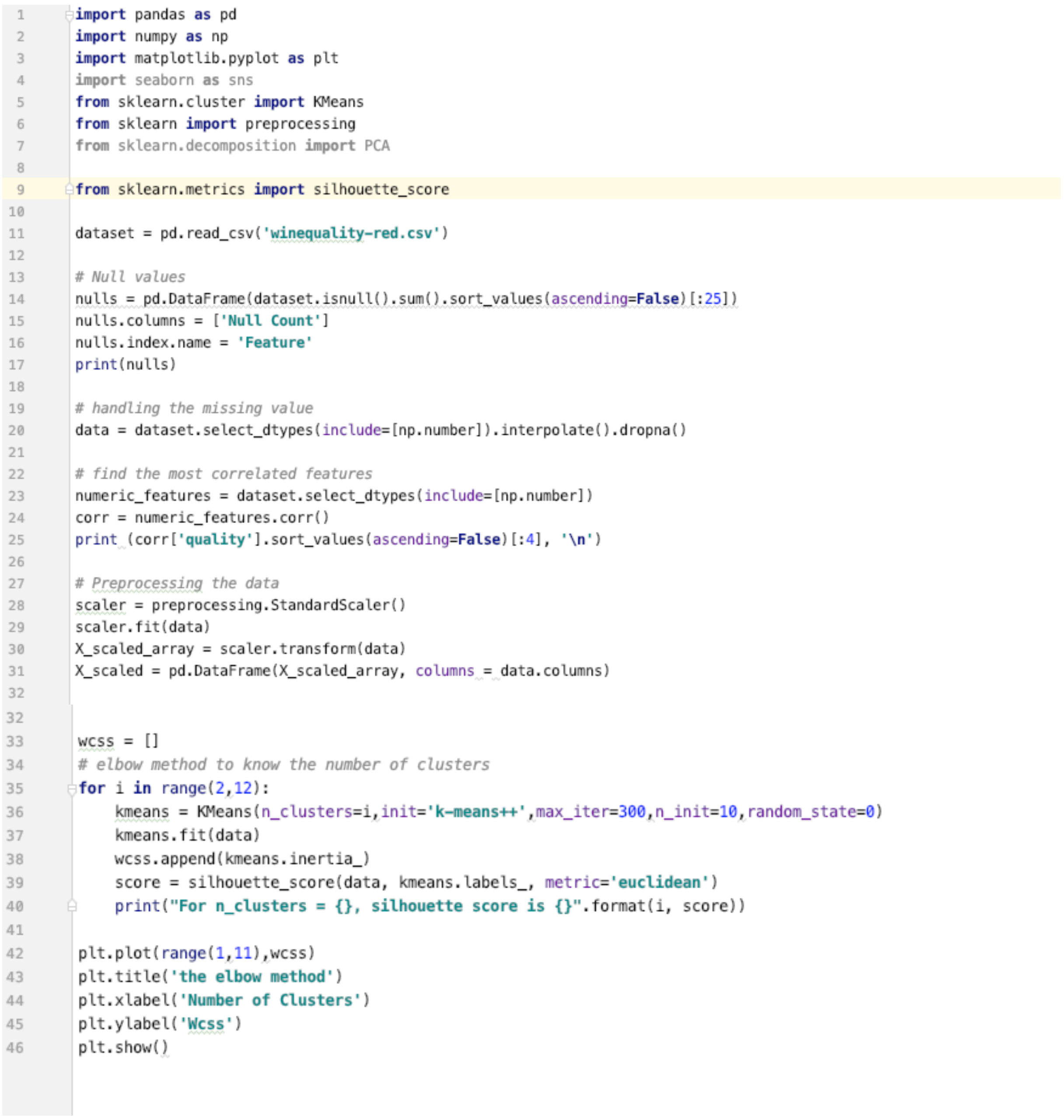
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1	-0.098	-0.069	0.28	0.21	0.12	-0.12	-0.4	0.097	0.21	-0.17	0.28	0.068	-0.23
sex	-0.098	1	-0.049	-0.057	-0.2	0.045	-0.058	-0.044	0.14	0.096	-0.031	0.12	0.21	-0.28
cp	-0.069	-0.049	1	0.048	-0.077	0.094	0.044	0.3	-0.39	-0.15	0.12	-0.18	-0.16	0.43
trestbps	0.28	-0.057	0.048	1	0.12	0.18	-0.11	-0.047	0.068	0.19	-0.12	0.1	0.062	-0.14
chol	0.21	-0.2	-0.077	0.12	1	0.013	-0.15	-0.0099	0.067	0.054	-0.004	0.071	0.099	-0.085
fbs	0.12	0.045	0.094	0.18	0.013	1	-0.084	-0.0086	0.026	0.0057	-0.06	0.14	-0.032	-0.028
restecg	-0.12	-0.058	0.044	-0.11	-0.15	-0.084	1	0.044	-0.071	-0.059	0.093	-0.072	-0.012	0.14
thalach	-0.4	-0.044	0.3	-0.047	-0.0099	-0.0086	0.044	1	-0.38	-0.34	0.39	-0.21	-0.096	0.42
exang	0.097	0.14	-0.39	0.068	0.067	0.026	-0.071	-0.38	1	0.29	-0.26	0.12	0.21	-0.44
oldpeak	0.21	0.096	-0.15	0.19	0.054	0.0057	-0.059	-0.34	0.29	1	-0.58	0.22	0.21	-0.43
slope	-0.17	-0.031	0.12	-0.12	-0.004	-0.06	0.093	0.39	-0.26	-0.58	1	-0.08	-0.1	0.35
ca	0.28	0.12	-0.18	0.1	0.071	0.14	-0.072	-0.21	0.12	0.22	-0.08	1	0.15	-0.39
thal	0.068	0.21	-0.16	0.062	0.099	-0.032	-0.012	-0.096	0.21	0.21	-0.1	0.15	1	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.085	-0.028	0.14	0.42	-0.44	-0.43	0.35	-0.39	-0.34	1

QUESTION 6

Apply K-means on any dataset and visualize the clusters using matplotlib or seaborn

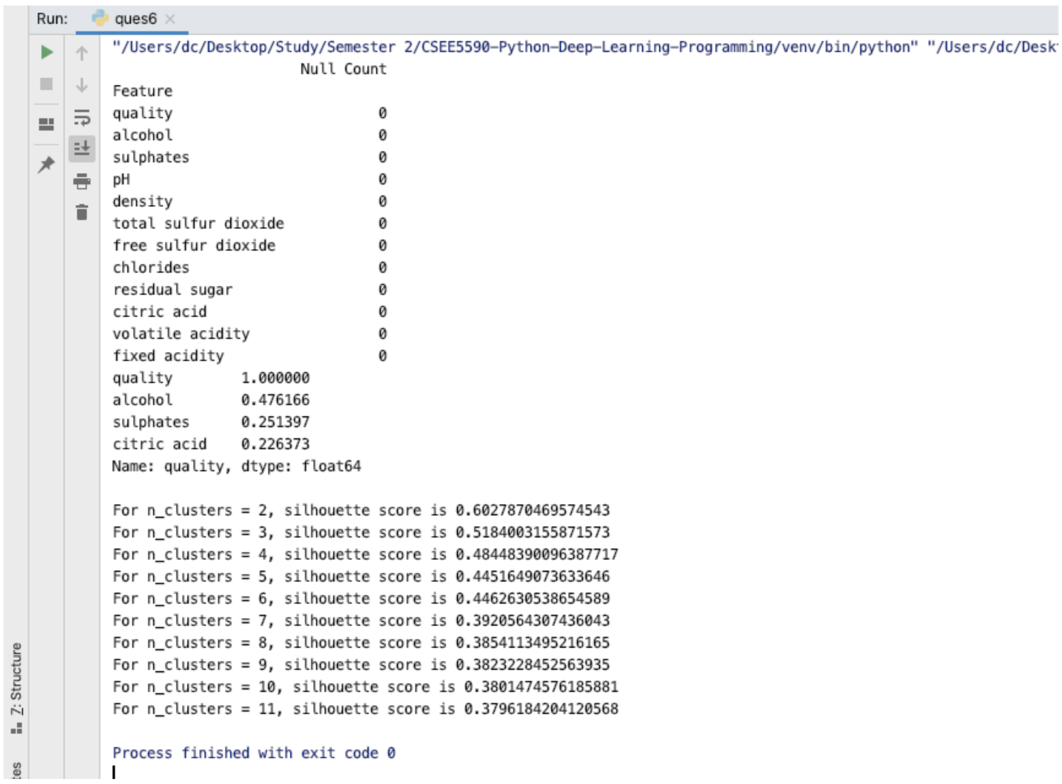
SOLUTION

Dataset Used is **Wine Quality Data Set** (https://archive.ics.uci.edu/ml/datasets/Wine+Quality)

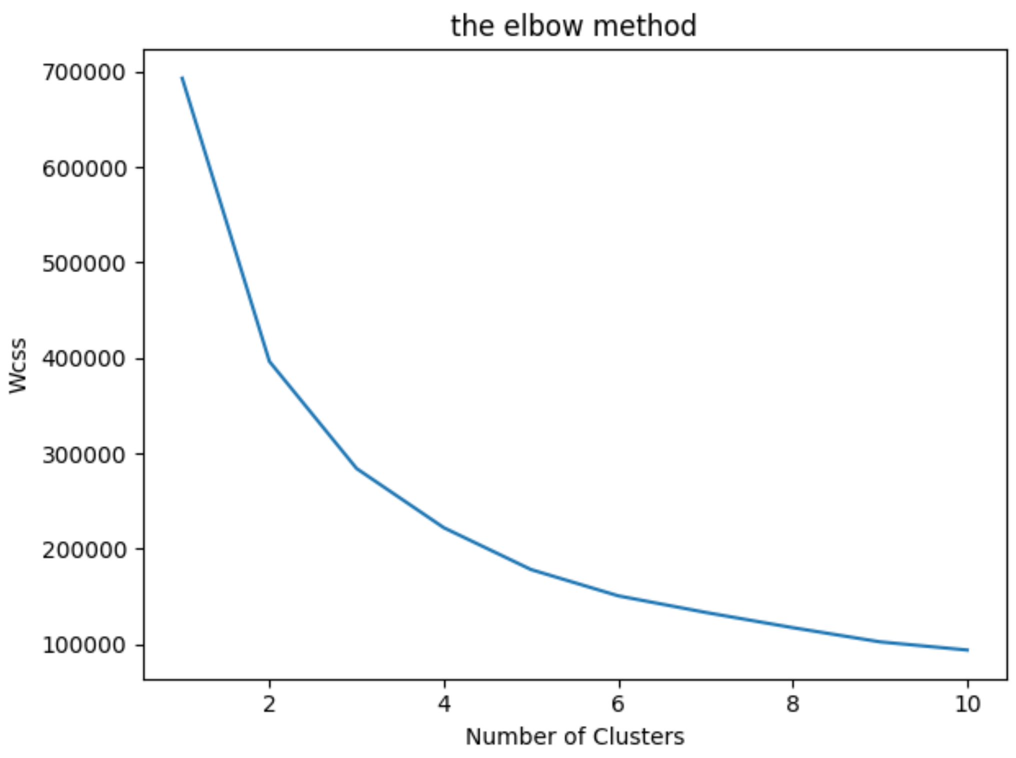


```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.decomposition import PCA

from sklearn.metrics import silhouette_score

dataset = pd.read_csv('winequality-red.csv')

# Null values
nulls = pd.DataFrame(dataset.isnull().sum().sort_values(ascending=False)[:25])
nulls.columns = ['Null Count']
nulls.index.name = 'Feature'
print(nulls)

# handling the missing value
data = dataset.select_dtypes(include=[np.number]).interpolate().dropna()

# find the most correlated features
numeric_features = dataset.select_dtypes(include=[np.number])
corr = numeric_features.corr()
print (corr['quality'].sort_values(ascending=False)[:4], '\n')

# Preprocessing the data
scaler = preprocessing.StandardScaler()
scaler.fit(data)
X_scaled_array = scaler.transform(data)
X_scaled = pd.DataFrame(X_scaled_array, columns = data.columns)


wcss = []
# elbow method to know the number of clusters
for i in range(2,12):
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(data)
    wcss.append(kmeans.inertia_)
    score = silhouette_score(data, kmeans.labels_, metric='euclidean')
    print("For n_clusters = {}, silhouette score is {}".format(i, score))

plt.plot(range(1,11),wcss)
plt.title('the elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```

OUTPUT



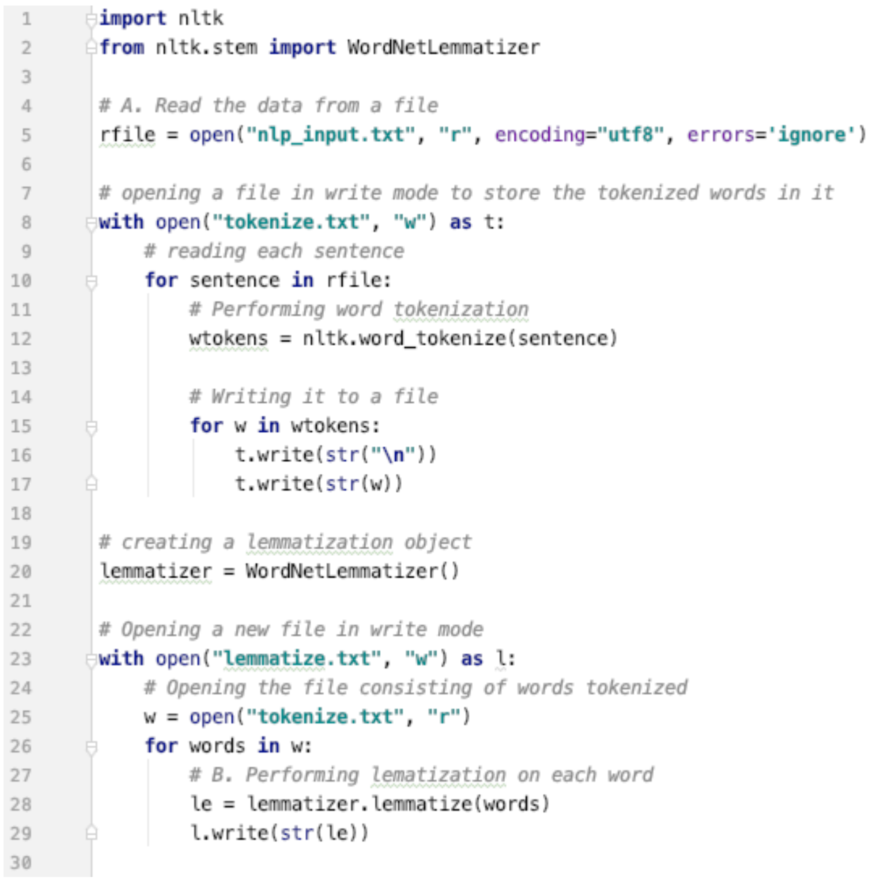
```
"/Users/dc/Desktop/Study/Semester 2/CSEE5590-Python-Deep-Learning-Programming/venv/bin/python" "/Users/dc/Desk
                      Null Count
Feature
quality                        0
alcohol                        0
sulphates                      0
pH                             0
density                        0
total sulfur dioxide           0
free sulfur dioxide            0
chlorides                      0
residual sugar                 0
citric acid                    0
volatile acidity               0
fixed acidity                  0
quality        1.000000
alcohol        0.476166
sulphates      0.251397
citric acid    0.226373
Name: quality, dtype: float64

For n_clusters = 2, silhouette score is 0.6027870469574543
For n_clusters = 3, silhouette score is 0.5184003155871573
For n_clusters = 4, silhouette score is 0.48448390096387717
For n_clusters = 5, silhouette score is 0.4451649073633646
For n_clusters = 6, silhouette score is 0.4462630538654589
For n_clusters = 7, silhouette score is 0.3920564307436043
For n_clusters = 8, silhouette score is 0.3854113495216165
For n_clusters = 9, silhouette score is 0.3823228452563935
For n_clusters = 10, silhouette score is 0.3801474576185881
For n_clusters = 11, silhouette score is 0.3796184204120568

Process finished with exit code 0
```



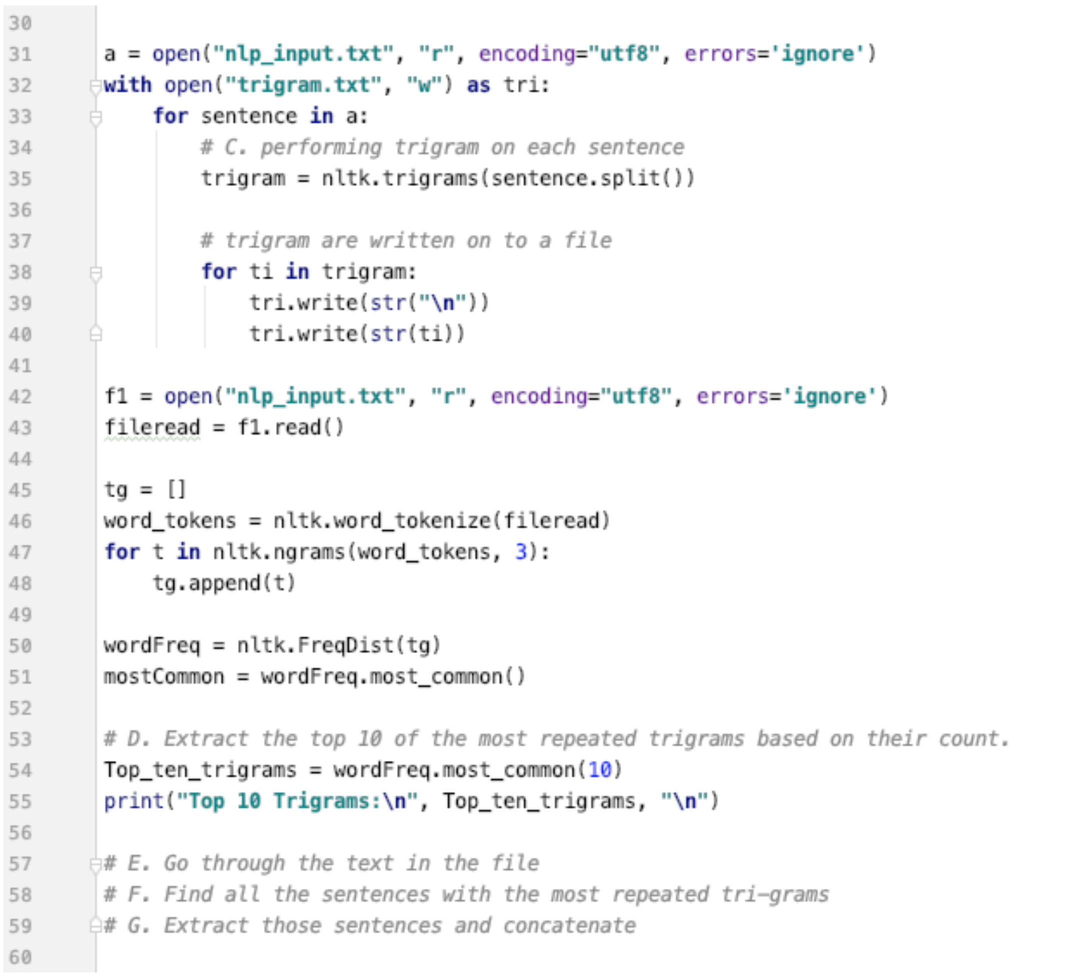
QUESTION 7

Perform Natural Language Processing on an input file

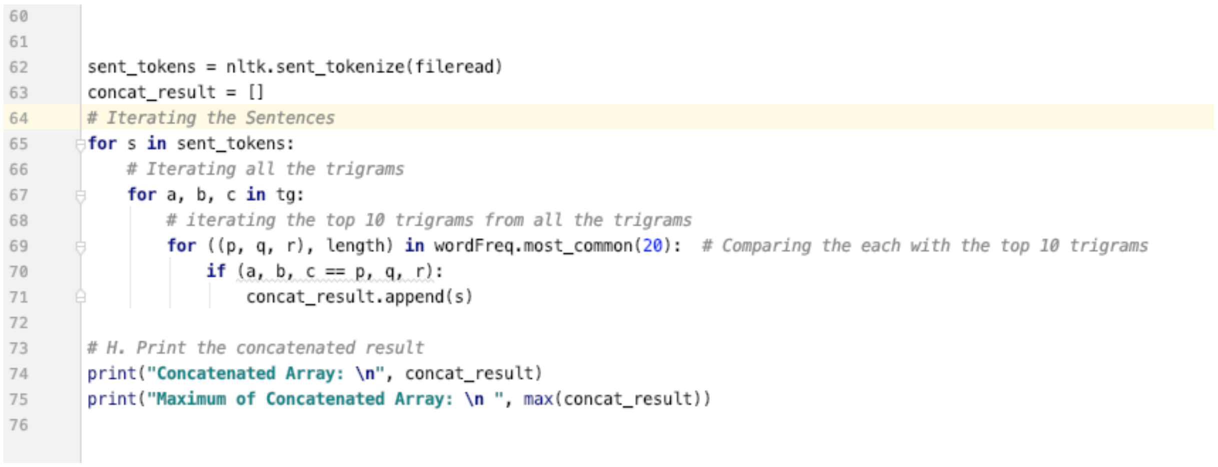
SOLUTION



```python
import nltk
from nltk.stem import WordNetLemmatizer

# A. Read the data from a file
rfile = open("nlp_input.txt", "r", encoding="utf8", errors='ignore')

# opening a file in write mode to store the tokenized words in it
with open("tokenize.txt", "w") as t:
    # reading each sentence
    for sentence in rfile:
        # Performing word tokenization
        wtokens = nltk.word_tokenize(sentence)

        # Writing it to a file
        for w in wtokens:
            t.write(str("\n"))
            t.write(str(w))

# creating a lemmatization object
lemmatizer = WordNetLemmatizer()

# Opening a new file in write mode
with open("lemmatize.txt", "w") as l:
    # Opening the file consisting of words tokenized
    w = open("tokenize.txt", "r")
    for words in w:
        # B. Performing lematization on each word
        le = lemmatizer.lemmatize(words)
        l.write(str(le))

```



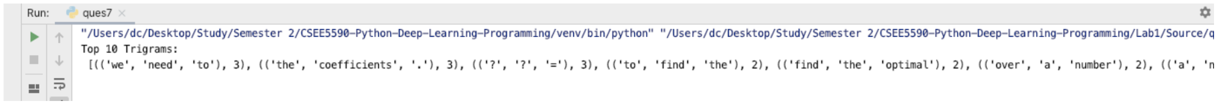
```python

a = open("nlp_input.txt", "r", encoding="utf8", errors='ignore')
with open("trigram.txt", "w") as tri:
    for sentence in a:
        # C. performing trigram on each sentence
        trigram = nltk.trigrams(sentence.split())

        # trigram are written on to a file
        for ti in trigram:
            tri.write(str("\n"))
            tri.write(str(ti))

f1 = open("nlp_input.txt", "r", encoding="utf8", errors='ignore')
fileread = f1.read()

tg = []
word_tokens = nltk.word_tokenize(fileread)
for t in nltk.ngrams(word_tokens, 3):
    tg.append(t)

wordFreq = nltk.FreqDist(tg)
mostCommon = wordFreq.most_common()

# D. Extract the top 10 of the most repeated trigrams based on their count.
Top_ten_trigrams = wordFreq.most_common(10)
print("Top 10 Trigrams:\n", Top_ten_trigrams, "\n")

# E. Go through the text in the file
# F. Find all the sentences with the most repeated tri-grams
# G. Extract those sentences and concatenate

```



```python


sent_tokens = nltk.sent_tokenize(fileread)
concat_result = []
# Iterating the Sentences
for s in sent_tokens:
    # Iterating all the trigrams
    for a, b, c in tg:
        # iterating the top 10 trigrams from all the trigrams
        for ((p, q, r), length) in wordFreq.most_common(20):  # Comparing the each with the top 10 trigrams
            if (a, b, c == p, q, r):
                concat_result.append(s)

# H. Print the concatenated result
print("Concatenated Array: \n", concat_result)
print("Maximum of Concatenated Array: \n ", max(concat_result))

```

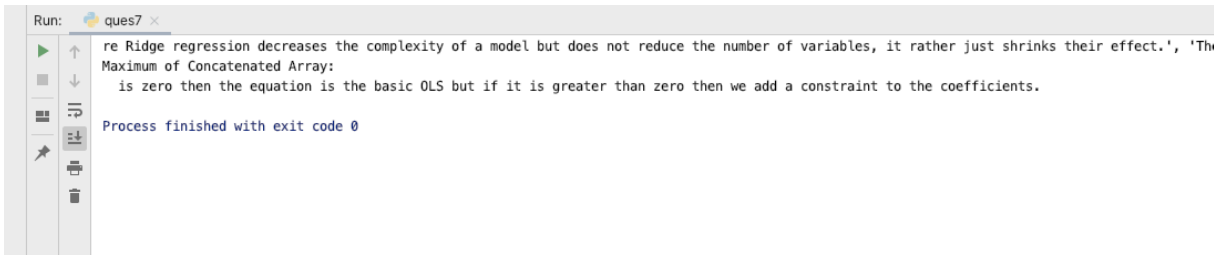
OUTPUT

Top 10 Trigrams:



```
"/Users/dc/Desktop/Study/Semester 2/CSEE5590-Python-Deep-Learning-Programming/venv/bin/python" "/Users/dc/Desktop/Study/Semester 2/CSEE5590-Python-Deep-Learning-Programming/Lab1/Source/q
Top 10 Trigrams:
 [(('we', 'need', 'to'), 3), (('the', 'coefficients', '.'), 3), (('?', '?', '='), 3), (('to', 'find', 'the'), 2), (('find', 'the', 'optimal'), 2), (('over', 'a', 'number'), 2), (('a', 'n
```

Concatenated Array and Maximum of Concatenated Array:

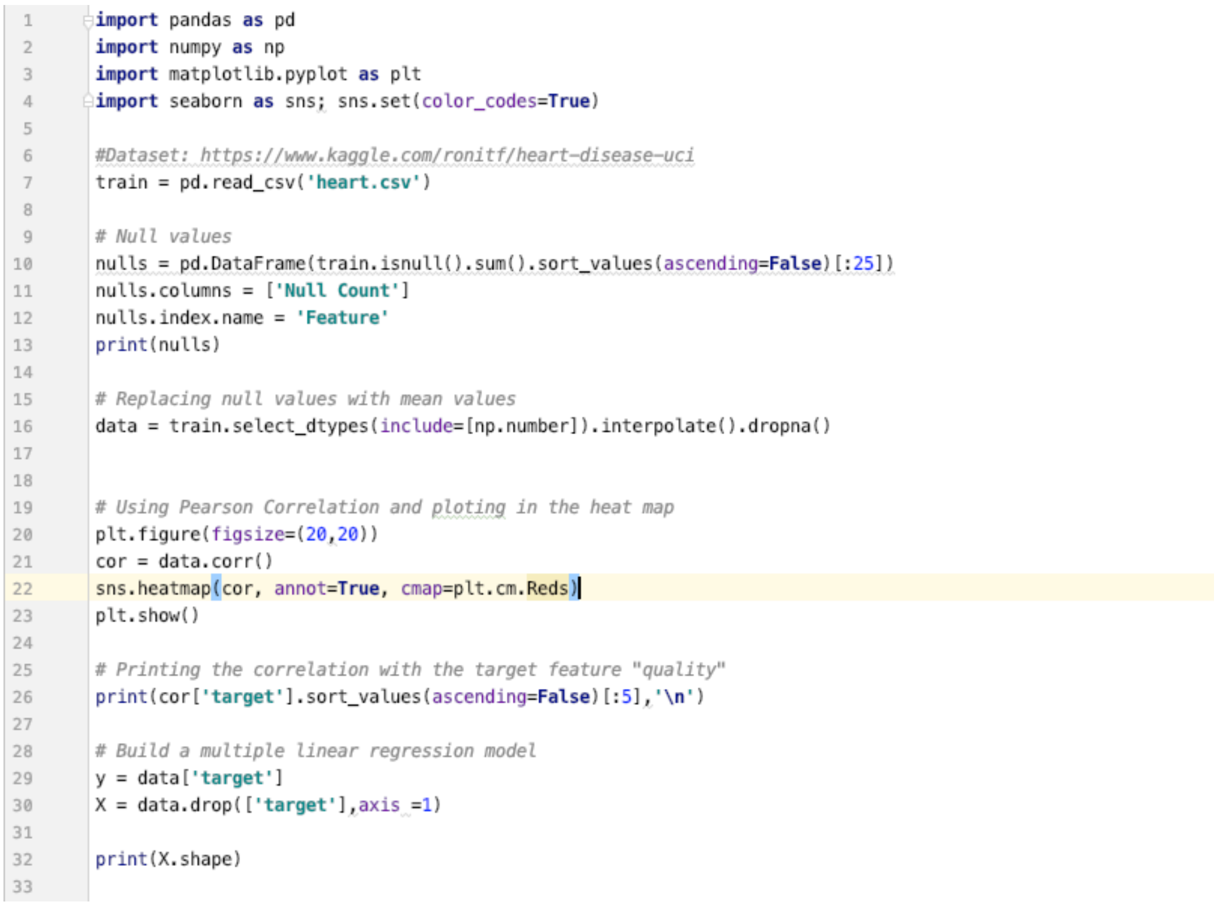


```
 re Ridge regression decreases the complexity of a model but does not reduce the number of variables, it rather just shrinks their effect.', 'The
Maximum of Concatenated Array:
  is zero then the equation is the basic OLS but if it is greater than zero then we add a constraint to the coefficients.

Process finished with exit code 0
```

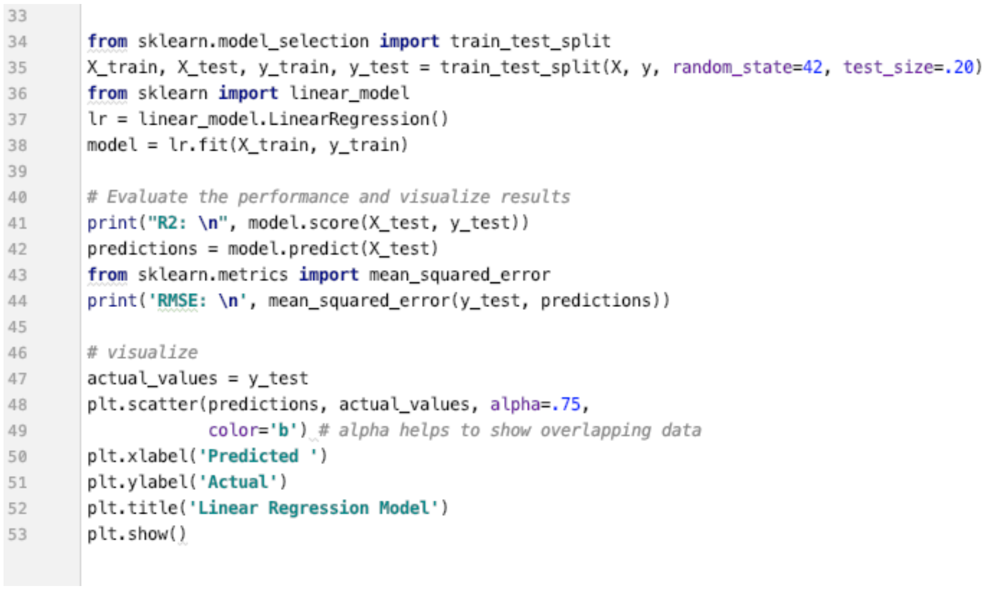
QUESTION 8

Do Multiple Regression on a dataset. Evaluate both RMSE and R2

SOLUTION

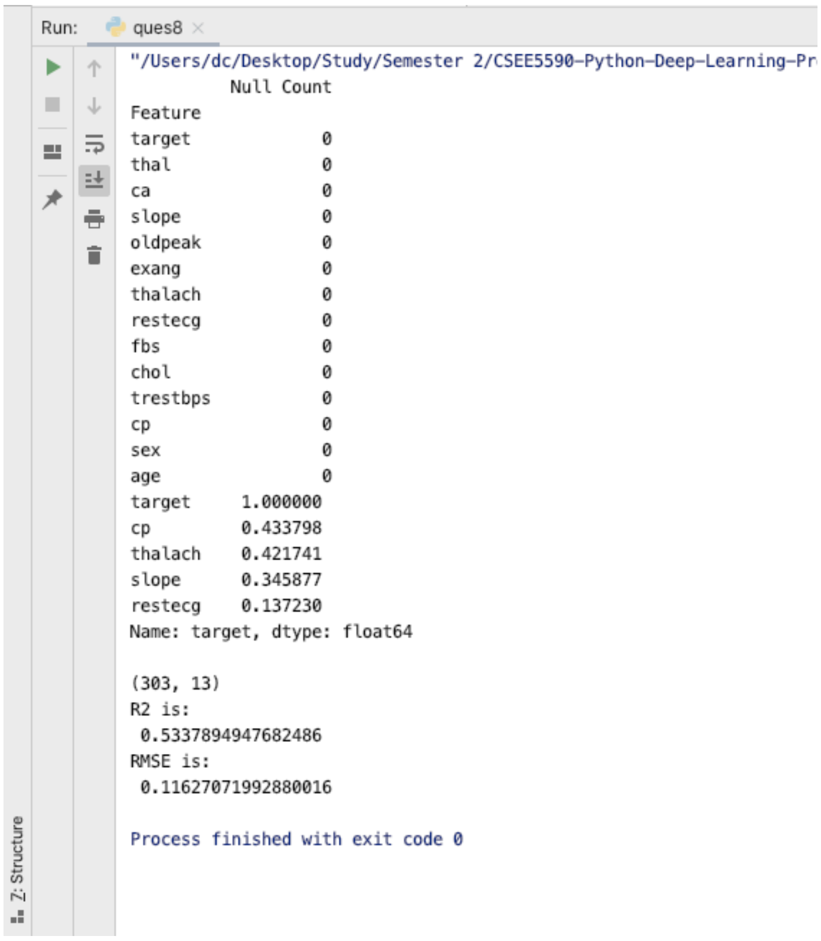


```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set(color_codes=True)

#Dataset: https://www.kaggle.com/ronitf/heart-disease-uci
train = pd.read_csv('heart.csv')

# Null values
nulls = pd.DataFrame(train.isnull().sum().sort_values(ascending=False)[:25])
nulls.columns = ['Null Count']
nulls.index.name = 'Feature'
print(nulls)

# Replacing null values with mean values
data = train.select_dtypes(include=[np.number]).interpolate().dropna()


# Using Pearson Correlation and ploting in the heat map
plt.figure(figsize=(20,20))
cor = data.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()

# Printing the correlation with the target feature "quality"
print(cor['target'].sort_values(ascending=False)[:5],'\n')

# Build a multiple linear regression model
y = data['target']
X = data.drop(['target'],axis =1)

print(X.shape)

```

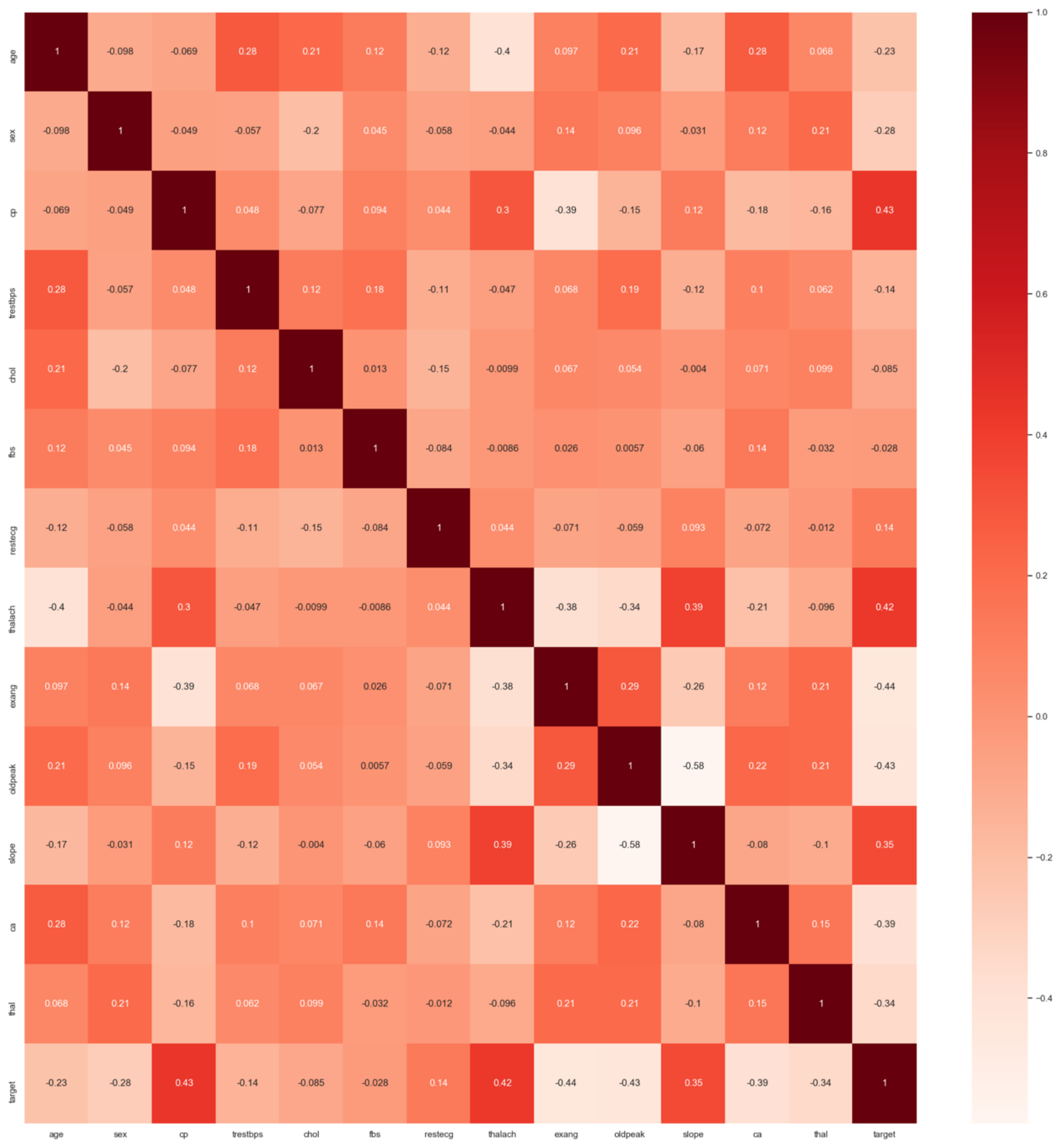


```python

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=.20)
from sklearn import linear_model
lr = linear_model.LinearRegression()
model = lr.fit(X_train, y_train)

# Evaluate the performance and visualize results
print("R2: \n", model.score(X_test, y_test))
predictions = model.predict(X_test)
from sklearn.metrics import mean_squared_error
print('RMSE: \n', mean_squared_error(y_test, predictions))

# visualize
actual_values = y_test
plt.scatter(predictions, actual_values, alpha=.75,
            color='b') # alpha helps to show overlapping data
plt.xlabel('Predicted ')
plt.ylabel('Actual')
plt.title('Linear Regression Model')
plt.show()
```

OUTPUT



```
"/Users/dc/Desktop/Study/Semester 2/CSEE5590-Python-Deep-Learning-Pr
          Null Count
Feature
target             0
thal               0
ca                 0
slope              0
oldpeak            0
exang              0
thalach            0
restecg            0
fbs                0
chol               0
trestbps           0
cp                 0
sex                0
age                0
target     1.000000
cp         0.433798
thalach    0.421741
slope      0.345877
restecg    0.137230
Name: target, dtype: float64

(303, 13)
R2 is:
 0.5337894947682486
RMSE is:
 0.11627071992880016

Process finished with exit code 0
```


age 1 -0.098 -0.069 0.28 0.21 0.12 -0.12 -0.4 0.097 0.21 -0.17 0.28 0.068 -0.23
sex -0.098 1 -0.049 -0.057 -0.2 0.045 -0.058 -0.044 0.14 0.096 -0.031 0.12 0.21 -0.28
cp -0.069 -0.049 1 0.048 -0.077 0.094 0.044 0.3 -0.39 -0.15 0.12 -0.18 -0.16 0.43
trestbps 0.28 -0.057 0.048 1 0.12 0.18 -0.11 -0.047 0.068 0.19 -0.12 0.1 0.062 -0.14
chol 0.21 -0.2 -0.077 0.12 1 0.013 -0.15 -0.0099 0.067 0.054 -0.004 0.071 0.099 -0.085
fbs 0.12 0.045 0.094 0.18 0.013 1 -0.084 -0.0086 0.026 0.0057 -0.06 0.14 -0.032 -0.028
restecg -0.12 -0.058 0.044 -0.11 -0.15 -0.084 1 0.044 -0.071 -0.059 0.093 -0.072 -0.012 0.14
thalach -0.4 -0.044 0.3 -0.047 -0.0099 -0.0086 0.044 1 -0.38 -0.34 0.39 -0.21 -0.096 0.42
exang 0.097 0.14 -0.39 0.068 0.067 0.026 -0.071 -0.38 1 0.29 -0.26 0.12 0.21 -0.44
oldpeak 0.21 0.096 -0.15 0.19 0.054 0.0057 -0.059 -0.34 0.29 1 -0.58 0.22 0.21 -0.43
slope -0.17 -0.031 0.12 -0.12 -0.004 -0.06 0.093 0.39 -0.26 -0.58 1 -0.08 -0.1 0.35
ca 0.28 0.12 -0.18 0.1 0.071 0.14 -0.072 -0.21 0.12 0.22 -0.08 1 0.15 -0.39
thal 0.068 0.21 -0.16 0.062 0.099 -0.032 -0.012 -0.096 0.21 0.21 -0.1 0.15 1 -0.34
target -0.23 -0.28 0.43 -0.14 -0.085 -0.028 0.14 0.42 -0.44 -0.43 0.35 -0.39 -0.34 1
age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target
1.0 0.8 0.6 0.4 0.2 0.0 -0.2 -0.4

CONCLUSION

We have learned how to use python for various tasks like extracting data from website, doing classification such as KNN, SVM and Naïve Bayes.

Also learned all basic usage of python in programming.

YOUTUBE: https://youtu.be/q-cGjCc369Y

GITHUB: https://github.com/dhairyachandra/CSEE5590-Python-Deep-Learning-Programming/tree/master/Lab1

GITHUB WIKI:
https://github.com/dhairyachandra/CSEE5590-Python-Deep-Learning-Programming/wiki/Lab-1