

CSEE5590/490: Python and Deep Learning Programming (2018 Fall)

LAB ASSIGNMENT 2

Team ID:- 13

Partner 1: Kamal Tej Veerapaneni Id - 31

Partner 2: Vinay Maturi Id - 17

YouTube Link: <https://youtu.be/xSwrpo3Yhlo>

Introduction:

In this Lab Assignment we have worked on the following tasks.

- 1) Pick a dataset and plot each category that is available along with creating a prediction model and evaluating it
- 2) Implement Support Vector Machine Classification
- 3) Take an Input file and perform lemmatization, bigram and its frequency...etc
- 4) Reporting views on the KNN algorithm and how it changes with K

Objectives:

Applying dictionaries, array, and iterations in strings for finding the first non-repeated characters in string.

Looping, splitting and respective methods are used for removing contents from file_1 which are present in also file_2

Creating 2 lists for each class and then using a loop we can remove the members from python class list who are in common with the web development class.

Using various classes, constructors inheritance and attribute keywords like `**kw` different sections of the task were completed.

Downloaded the html webpages which contain a table and parsed it using BeautifulSoup library.

Question 1:

Pick any dataset from the dataset sheet in the class sheet

a) Plot how many of each category is available in your dataset (you can use seaborn library or matplotlib)

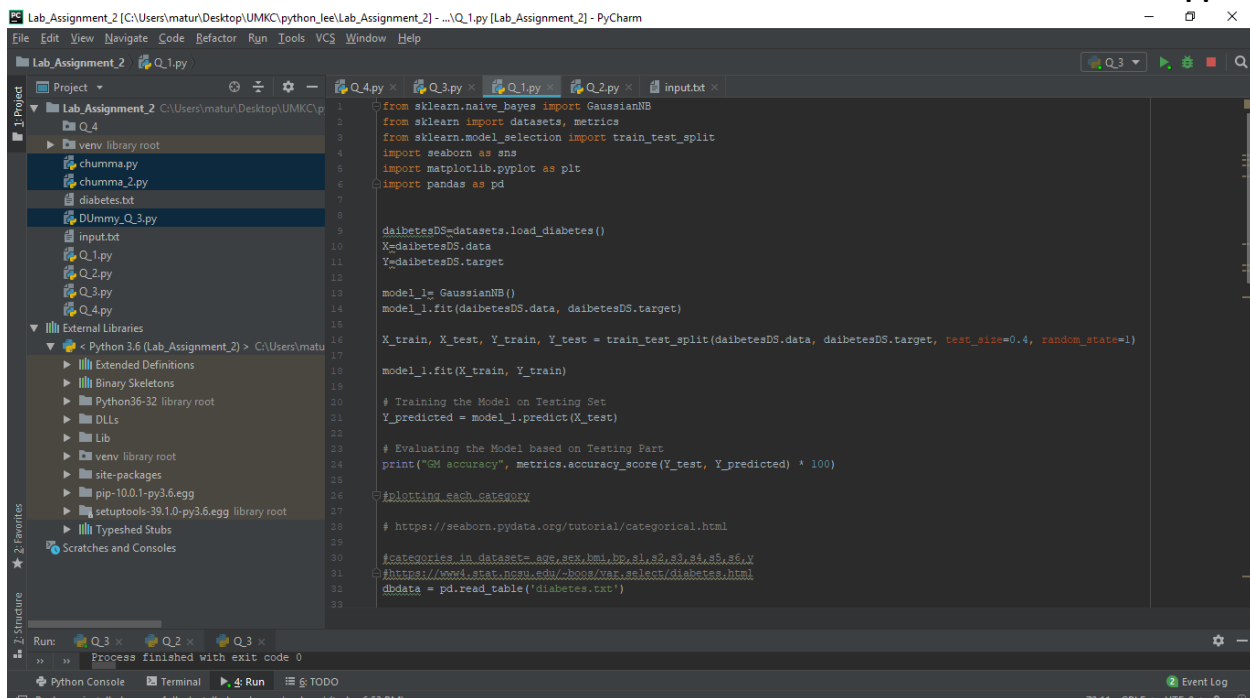
b) Create one prediction model based on Naïve Bayes Classification and evaluate your model

Workflow:

The result is predicted using a Naive Bayes and the resultant prediction is plotted with all the categories that are present in the dataset

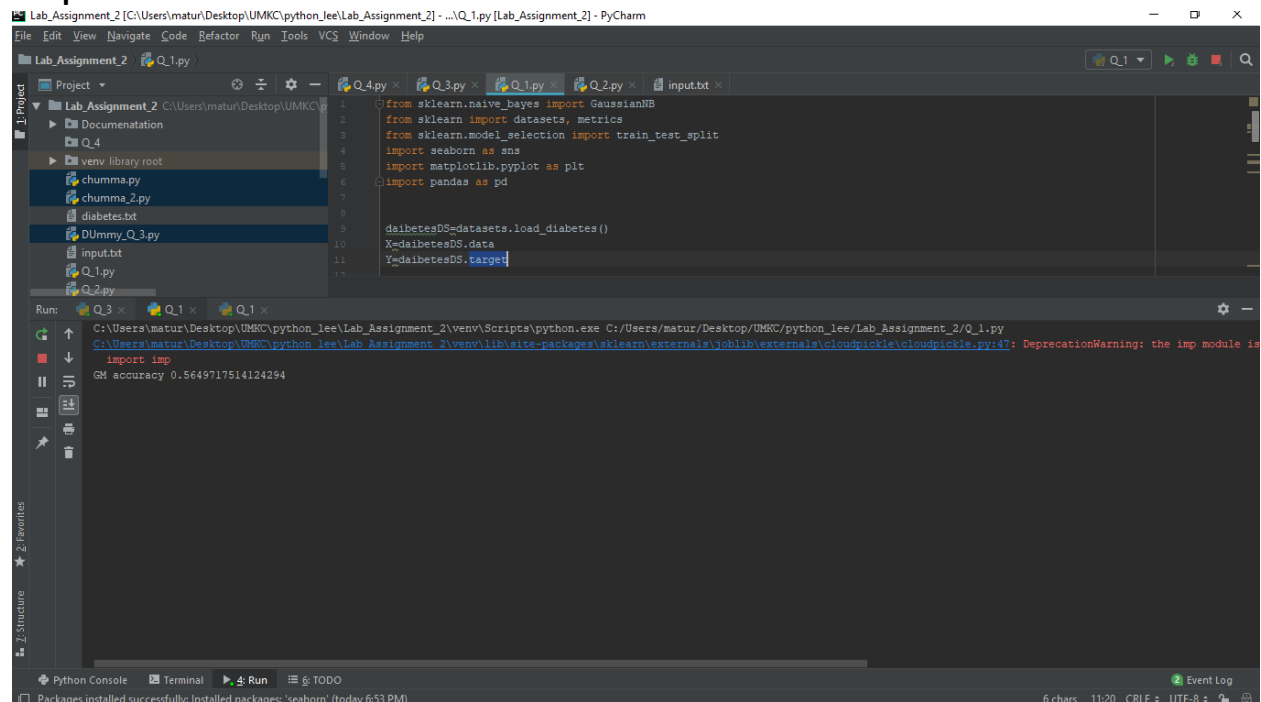
Code:

Snippet:



```
1 from sklearn.naive_bayes import GaussianNB
2 from sklearn import datasets, metrics
3 from sklearn.model_selection import train_test_split
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 diabetesDS=datasets.load_diabetes()
9 X=diabetesDS.data
10 Y=diabetesDS.target
11
12 model_1= GaussianNB()
13 model_1.fit(diabetesDS.data, diabetesDS.target)
14
15 X_train, X_test, Y_train, Y_test = train_test_split(diabetesDS.data, diabetesDS.target, test_size=0.4, random_state=1)
16
17 model_1.fit(X_train, Y_train)
18
19 # Training the Model on Testing Set
20 Y_predicted = model_1.predict(X_test)
21
22 # Evaluating the Model based on Testing Part
23 print("GM accuracy", metrics.accuracy_score(Y_test, Y_predicted) * 100)
24
25 #plotting each category
26
27 # https://seaborn.pydata.org/tutorial/categorical.html
28
29 #categories in dataset= age,sex,bmi,bp,s1,s2,s3,s4,s5,s6,y
30 #https://www1.stat.ncsu.edu/~boone/var.select/diabetes.html
31 dbdata = pd.read_table('diabetes.txt')
32
```

Output:



```
1 from sklearn.naive_bayes import GaussianNB
2 from sklearn import datasets, metrics
3 from sklearn.model_selection import train_test_split
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6 import pandas as pd
7
8 diabetesDS=datasets.load_diabetes()
9 X=diabetesDS.data
10 Y=diabetesDS.target
```

Run: Q3 x Q1 x Q1 x

C:\Users\matur\Desktop\UMKC\python_lee\Lab_Assignment_2\venv\Scripts\python.exe C:\Users\matur\Desktop\UMKC\python_lee\Lab_Assignment_2\Q_1.py

import imp

GM accuracy 0.5649717514124294

Python Console Terminal Run TODO

Package installed successfully. Installed packages: 'seaborn' (today 6:53 PM)

6 chars 11:20 CRLF : UTF-8 : 9

Question 2:

Implement Support Vector Machine classification:

- Apply SVC with kernel “poly” degree =4
- Apply SVC with “rbf” kernel
- change gamma and C parameters in the model to see how the result may change
- Report the accuracy of the model on both models separately and with which parameters you got better result.

Workflow: SVM model is implemented on diabetes dataset by importing it and then by applying SVC with kernel “poly” and “RBF”, the results are compared. From the outputs we can say that the RBF model gives more accuracy than poly model.

CODE:

```
Lab_Assignment_2 [C:\Users\matur\Desktop\UMKC\python_lee\Lab_Assignment_2] - ...Q_2.py [Lab_Assignment_2] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help

Lab_Assignment_2 Q_2.py
Project
  Lab_Assignment_2 C:\Users\matur\Desktop\UMKC\python_lee\Lab_Assignment_2
    Documentation
    Q_4
    venv library root
    chumma.py
    chumma_2.py
    diabetes.txt
    DUummy_Q_3.py
    input.txt
    Q_1.py
    Q_2.py
    Q_3.py
    Q_4.py
  External Libraries
    < Python 3.6 (Lab_Assignment_2) > C:\Users\matur\Desktop\UMKC\python_lee\Lab_Assignment_2\venv\Scripts\python.exe
    Extended Definitions
    Binary Skeletons
    Python36-32 library root
    DLLs
    Lib
    venv library root
    site-packages
    pip-10.0.1-py3.6.egg
    setuptools-39.1.0-py3.6.egg library root
    Typed stubs
    Scratches and Consoles

Run: Q_3 x Q_1 x Q_1 x
C:\Users\matur\Desktop\UMKC\python_lee\Lab_Assignment_2\venv\Scripts\python.exe C:\Users\matur\Desktop\UMKC\python_lee\Lab_Assignment_2\Q_1.py
C:\Users\matur\Desktop\UMKC\python_lee\Lab_Assignment_2\venv\lib\site-packages\sklearn\externals\joblib\externals\cloudpickle\cloudpickle.py:47: DeprecationWarning: the imp module is deprecated in favour of importlib; See https://docs.python.org/3.7/library/importlib.html#deprecating-the-imp-module
Python Console Terminal Run TODO Event Log

import matplotlib.pyplot as plot
import numpy
from sklearn import datasets, linear_model, metrics
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

# Loading diabetes Dataset
diabetesdataset = datasets.load_diabetes()

X = diabetesdataset.data
Y = diabetesdataset.target

# Training and Testing Data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=21)

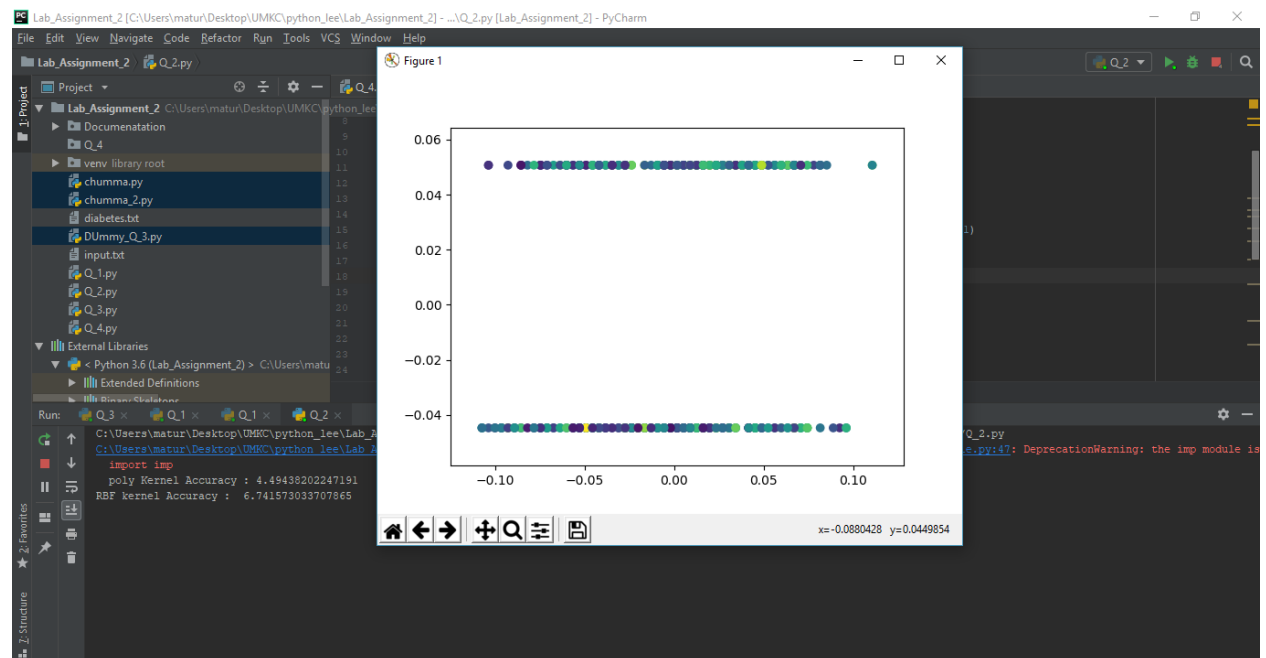
# Cross-Validation
# C = 1.0 ---> SVM Regularization Parameter

# Poly Kernel
clf = SVC(kernel='poly', degree=4, C=1.0, gamma=0.1).fit(X_train, Y_train)
clf.fit(X_test, Y_test)
y_pred = clf.predict(X_test)
print(" poly Kernel Accuracy :", metrics.accuracy_score(Y_test, y_pred) *100)

# Increasing Random State increases accuracy

# RBF Kernel
clf1 = SVC(kernel='rbf', C=1.0, gamma=0.4).fit(X_train, Y_train)
```

Output:



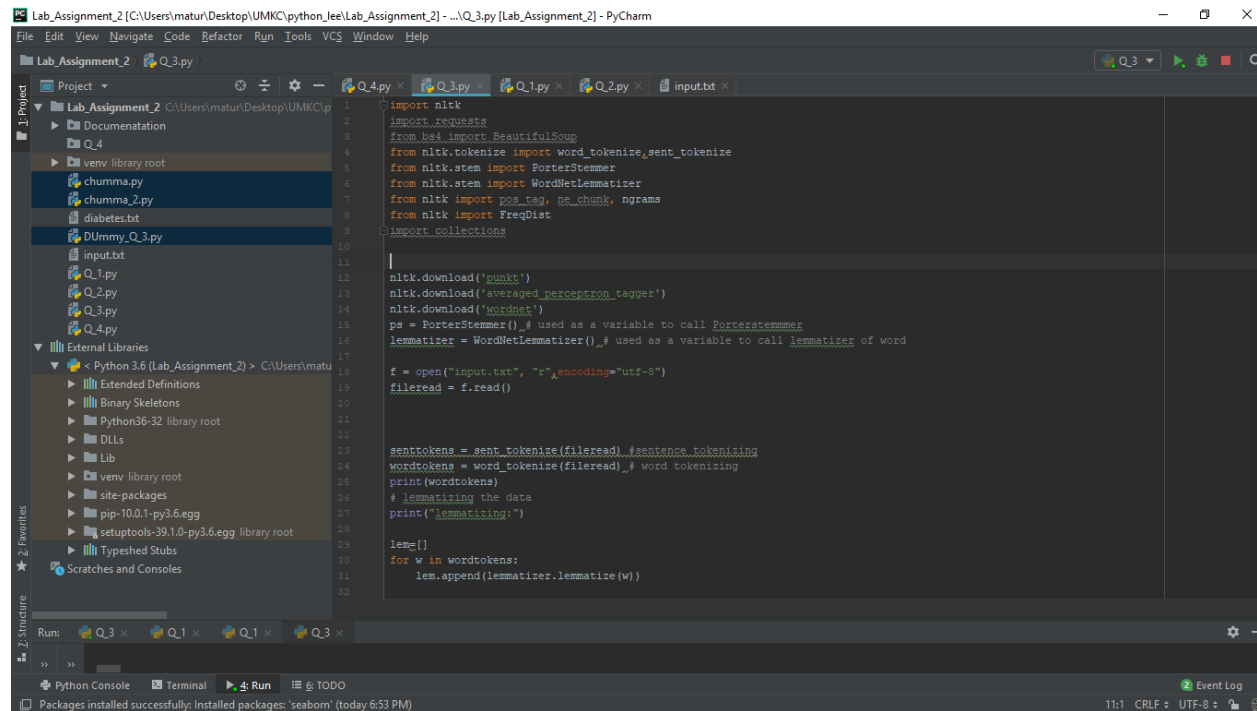
Question 3:

Write a program in which take an Input file. Use the simple approach below to summarize a text file:

- a. Read a file
- b. Apply lemmatization on the words
- c. Apply the bigram on the text
- d. Calculate the word frequency (bi-gram frequency) of the words (bi-grams)
- f. Choose top five bi-grams that have been repeated most
- g. Go through the original text that you had in the file
- h. Find all the sentences with those most repeated bi-grams
- i. Extract those sentences and concatenate
- j. Enjoy the summarization

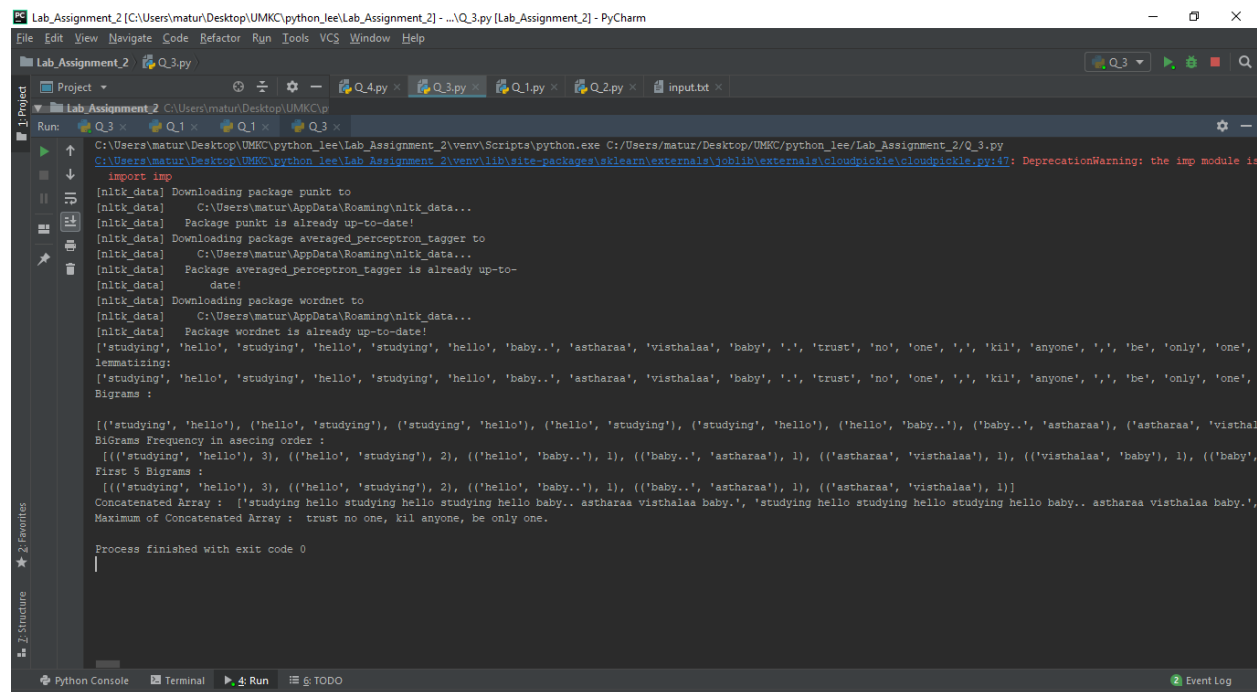
Workflow: A file is taken as an input and limmitization is applied to the word present in the text file, then bigrams are generated from the file. The top most repeated bigrams are found and then the respective sentences containing the top bigrams are concatenated.

CODE:



```
1 import nltk
2 import requests
3 from bs4 import BeautifulSoup
4 from nltk.tokenize import word_tokenize, sent_tokenize
5 from nltk.stem import PorterStemmer
6 from nltk.stem import WordNetLemmatizer
7 from nltk import pos_tag, pos_chunk, ngrams
8 from nltk import FreqDist
9 import collections
10
11
12 nltk.download('punkt')
13 nltk.download('averaged_perceptron_tagger')
14 nltk.download('wordnet')
15 ps = PorterStemmer() # used as a variable to call PorterStemmer
16 lemmatizer = WordNetLemmatizer() # used as a variable to call lemmatizer of word
17
18 f = open("input.txt", "r", encoding="utf-8")
19 file_read = f.read()
20
21
22 senttokens = sent_tokenize(file_read) # sentence tokenizing
23 wordtokens = word_tokenize(file_read) # word tokenizing
24 print(wordtokens)
25 # lemmatizing the data
26 print("lemmatizing:")
27
28 lemmatizer = WordNetLemmatizer()
29 for w in wordtokens:
30     lem.append(lemmatizer.lemmatize(w))
31
32
```

Output:



```
1 import imp
2
3 [nltk_data] Downloading package punkt to
4 [nltk_data] C:\Users\matur\AppData\Roaming\nltk_data...
5 [nltk_data] Package punkt is already up-to-date!
6 [nltk_data] Downloading package averaged_perceptron_tagger to
7 [nltk_data] C:\Users\matur\AppData\Roaming\nltk_data...
8 [nltk_data] Package averaged_perceptron_tagger is already up-to-
9 [nltk_data] date!
10 [nltk_data] Downloading package wordnet to
11 [nltk_data] C:\Users\matur\AppData\Roaming\nltk_data...
12 [nltk_data] Package wordnet is already up-to-date!
13 ['studying', 'hello', 'studying', 'hello', 'studying', 'hello', 'baby..', 'astharaa', 'visthalaa', 'baby', 'trust', 'no', 'one', 'kill', 'anyone', 'be', 'only', 'one',
14 lemmatizing:
15 ['studying', 'hello', 'studying', 'hello', 'studying', 'hello', 'baby..', 'astharaa', 'visthalaa', 'baby', 'trust', 'no', 'one', 'kill', 'anyone', 'be', 'only', 'one',
16 Bigrams :
17 [(['studying', 'hello'), ('hello', 'studying'), ('studying', 'hello'), ('hello', 'studying'), ('studying', 'hello'), ('hello', 'baby..'), ('baby..', 'astharaa'), ('astharaa', 'visthalaa'), ('visthalaa', 'baby'), ('baby', 'trust'), ('trust', 'no'), ('no', 'one'), ('one', 'kill'), ('kill', 'anyone'), ('anyone', 'be'), ('be', 'only'), ('only', 'one')]
18 Bigrams Frequency in ascending order :
19 [(['studying', 'hello'), 3], [(['hello', 'studying'), 2], [(['hello', 'baby..'), 1], [(['baby..', 'astharaa'), 1], [(['astharaa', 'visthalaa'), 1], [(['visthalaa', 'baby'), 1], [(['baby', 'trust'), 1], [(['trust', 'no'), 1], [(['no', 'one'), 1], [(['one', 'kill'), 1], [(['kill', 'anyone'), 1], [(['anyone', 'be'), 1], [(['be', 'only'), 1], [(['only', 'one'), 1]]
20 First 5 Bigrams :
21 [(['studying', 'hello'), 3], [(['hello', 'studying'), 2], [(['hello', 'baby..'), 1], [(['baby..', 'astharaa'), 1], [(['astharaa', 'visthalaa'), 1]]
22 Concatenated Array : ['studying hello studying hello studying hello baby.. astharaa visthalaa baby.', 'studying hello studying hello studying hello baby.. astharaa visthalaa baby.',
23 Maximum of Concatenated Array : trust no one, kill anyone, be only one.
24
25 Process finished with exit code 0
```

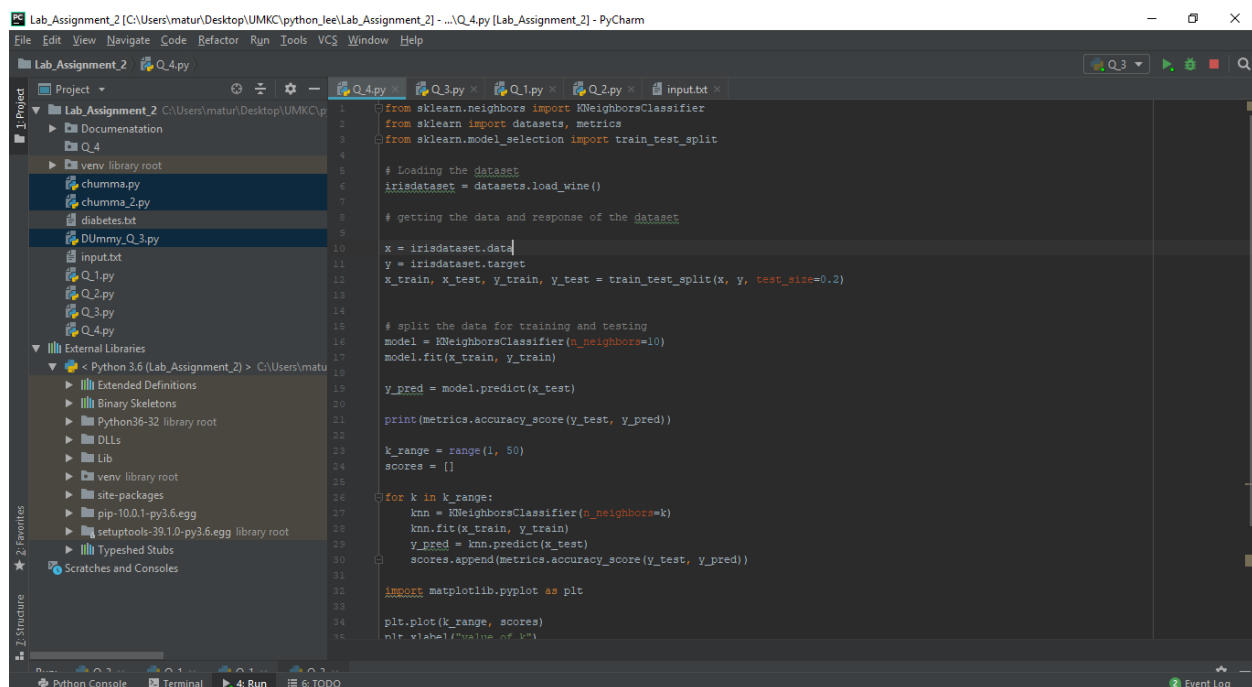
Question 4:

Report your views on the k nearest neighbor algorithm when we change the K how it will affect the accuracy. Provide a good justification for the changes of the accuracy when we change the amount of K. For example: compare the accuracy when K=1 and K is a big number like 50, why the accuracy will change.

WorkFlow:

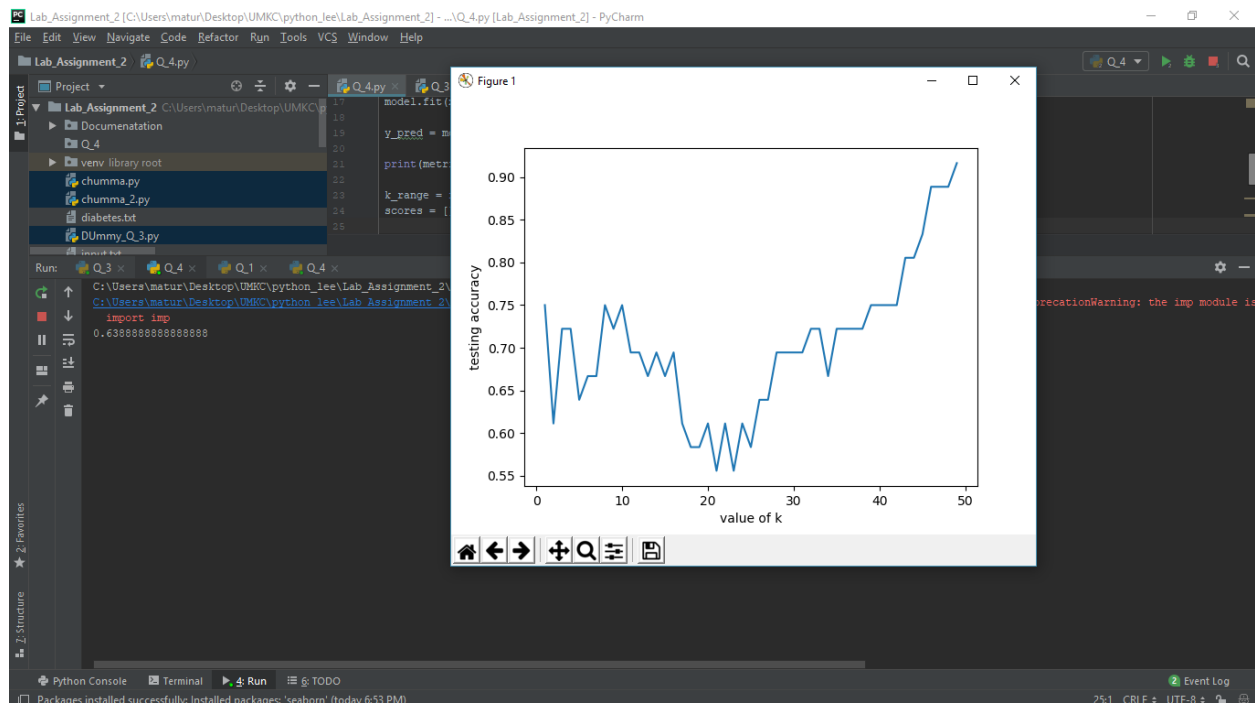
As the knn gives different accuracies with different k-values. The accuracies are observed as follows. The accuracies completely depends on how the model divides the dataset into training set and test set and k values. One of such outputs is as follows.

Code Snippet:



```
1 from sklearn.neighbors import KNeighborsClassifier
2 from sklearn import datasets, metrics
3 from sklearn.model_selection import train_test_split
4
5 # Loading the dataset
6 irisdataset = datasets.load_wine()
7
8 # getting the data and response of the dataset
9
10 x = irisdataset.data
11 y = irisdataset.target
12 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
13
14 # split the data for training and testing
15 model = KNeighborsClassifier(n_neighbors=10)
16 model.fit(x_train, y_train)
17
18 y_pred = model.predict(x_test)
19
20 print(metrics.accuracy_score(y_test, y_pred))
21
22 k_range = range(1, 50)
23 scores = []
24
25 for k in k_range:
26     knn = KNeighborsClassifier(n_neighbors=k)
27     knn.fit(x_train, y_train)
28     y_pred = knn.predict(x_test)
29     scores.append(metrics.accuracy_score(y_test, y_pred))
30
31 import matplotlib.pyplot as plt
32
33 plt.plot(k_range, scores)
34 plt.xlabel("Value of k")
```

Output:



Thank you..