CSEE 5590 0001
Special Topics SPRING 2018

PHYTHON LAB ASSIGNMENT - 3

Submitted On 3/7/2018

Name: Satya Sai Deepthi Katta

Class ID: 21

Department of Computer Science and Electrical Engineering

DOCUMENT CONTENTS

AUTHOR

This document is a part of Lab Assignment #2 submitted by SATYA SAI DEEPTHI KATTA (ID: 21), a graduate student majoring in computer science at University of Missouri Kansas City. The lab assignment carried out under Python and Deep Learning course(CS5590) taught by Dr. Yungyug Lee along with Rashmi, Saira and Vijaya Yeruva.

OBJECTIVE

The main aim of the lab work is to create an exposure to some concepts in Python language like:

- Linear Discriminant Analysis
- Support Vector Machine: linear and rbf kernel
- K – nearest neighbor's algorithm
- Concepts of Lemmatization, Bi grams etc

The assignment is divided into four tasks which focuses to make one familiar with the python concepts listed above.

- To plot Linear Discriminant Analysis on any loaded dataset.
- To implement SVM classification using Linear and RBF Kernel.
- To read a file and perform operations listed like Lemmatization, Bi-grams etc.
- To report the views by varying the K value from 1 to 50.

FEATURES

The features involved in each task are documented below.

Task 1:

LDA on Wine Dataset

The task is to perform the Linear Discriminant analysis on wine dataset by loading the values and give the classification of 3 classes in the dataset. Also to state the convincible differences between logistic regression and linear discrimination analysis.

Task 2:

SVM classification on Linear and RBF

The task mainly features the differences between the SVM classifications on any dataset which is split into 20% testing data and 80% training data.

To apply SVC with Linear Kernel and RBF kernel and report which is better for the dataset taken in the program.

Task 3:

NPL Features

This task involves applying NLP features on a input text

- to use lemmatization on the words
- to apply bi-grams on the text
- to print the top five bi-grams mostly repeated in the text
- to print the sentences which has top five bi-grams

Task 4:

Variation in K Value

This task mainly reports the views on the accuracy of KNN algorithm by changing the value of K from 1 to 50

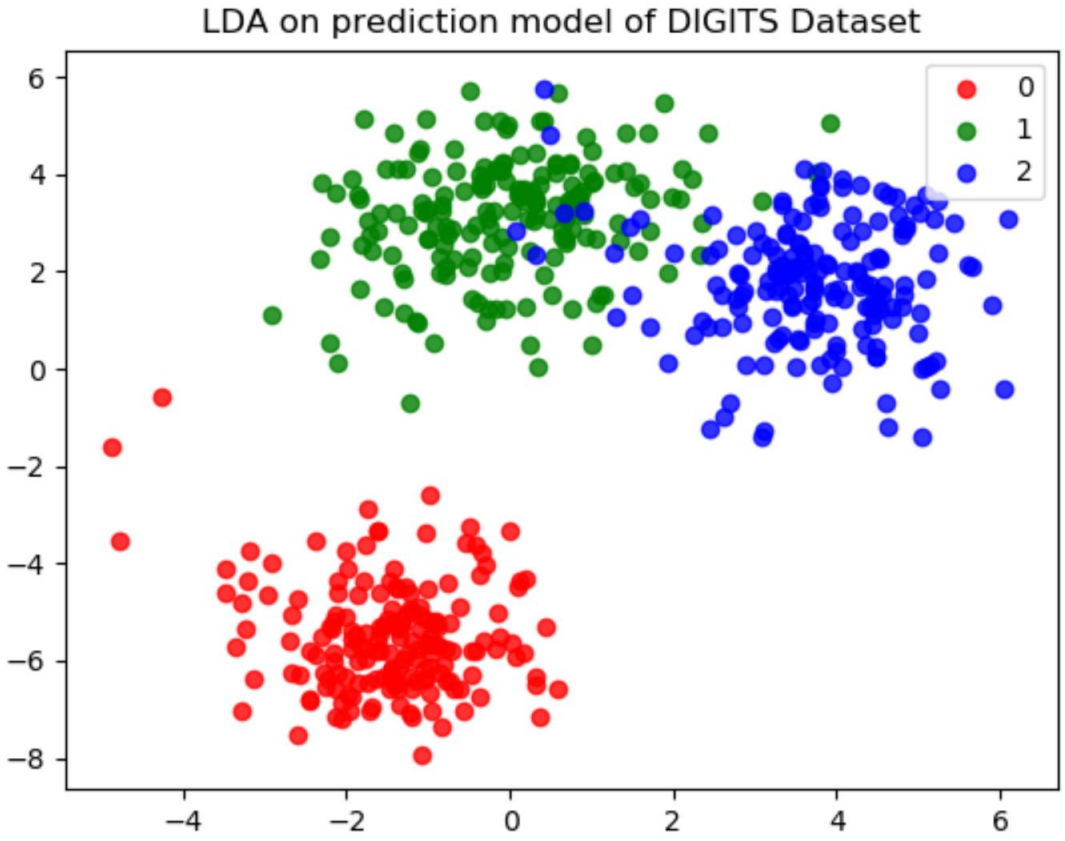
CONFIGURATION

For executing the tasks given in the lab assignment, advanced version of **Python 3.6.4** is used and the code is built in **PYCHARM** Software.

INPUT/OUTPUT SCREENSHOTS

Task 1:

The Digits Dataset is taken as input and a prediction model is built with 30%testing data and 70% training data. The output is given as the figure plotting the linear discriminant analysis on the digits dataset.



Explanation:

Logistic Regression(LR) weights are estimated using maximum likelihood, whereas LDA(Linear Discriminant Analysis) parameters are computed using the estimated mean and variance from the normal distribution.

- When the classes are well separated, the parameter estimates for LR model would give unstable results.
- If the number of observations is small ad distribution of feature si normal in each class, then LDA gives stable results.

In the task 1, LR makes no assumptions on the distribution of the explanatory data, LDA has been developed for normally distributed explanatory variables. It is therefore reasonable to expect LDA to give better results in the case when the normality assumptions are fulfilled, but in all other situations LR should be more appropriate.

Task 2:

The digits dataset is given as the input for the task. The data and target are loaded form the digits dataset into the variables and the SVC is applied on Linear and RBF Kernel.

The accuracy values from the support vector classifications are given as the output.

```
C:\Users\satyasaideepthi\PycharmProjects\LAB3\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/LAB3/t2.py
Accuracy for SVC linear kernel 0.9833333333333333
Accuracy for SVC with rbf kernel 0.4944444444444446

Process finished with exit code 0
```

Explanation:

Here in this task, we have considered the randomized linear data with less dataset. Linear kernel is a parametric model whose complexity increases with the increasing of the dataset. Here only 3-dimensional data [0,1,2] is considered. So linear kernel gives more accuracy than RBF(Radial Basis Function) which is basically for non linear problems with larger dimensional datasets.

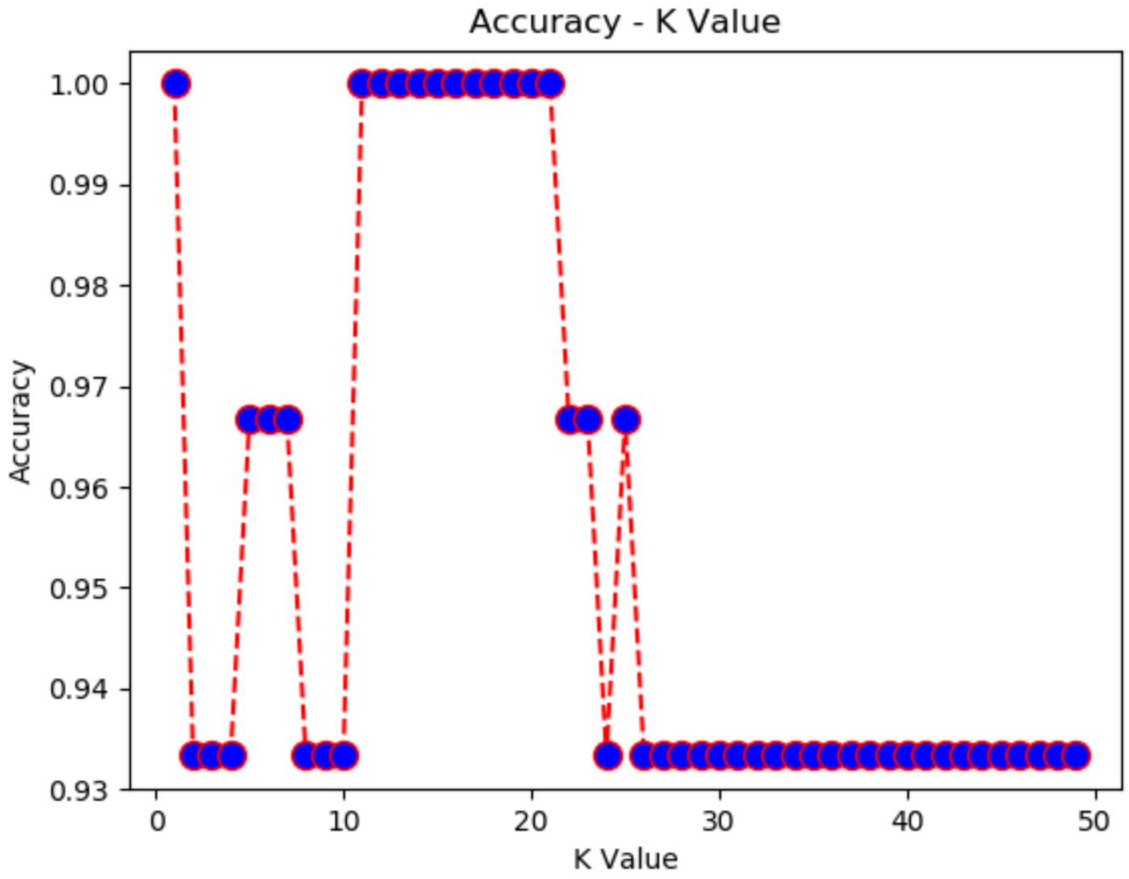
Task 3:

A text file is given as the input. Different NLP operations are taken as output like lemmatization of the words, getting the bi-grams for the text file, printing most commonly repeated five bi-grams and finally printing the sentences with commonly repeated top five bi-grams.

```
C:\Users\satyasaideepthi\PycharmProjects\LAB3\venv\Scripts\python.exe C:/Users/satyasaideepthi/PycharmProjects/LAB3/t3.py
_____INPUT_____
['Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design p
Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design phi


_____LEMMATIZATION_____
['python', 'is', 'an', 'interpreted', 'high-level', 'programming', 'language', 'for', 'general-purpose', 'programming', '.', 'created', 'by', 'guido', 'van', 'rossum', '


_____BI-GRAMS_____
[('python', 'is'), ('is', 'an'), ('an', 'interpreted'), ('interpreted', 'high-level'), ('high-level', 'programming'), ('programming', 'language'), ('language', 'for'),


_____Bi-Grams with Word frequency_____
[(('!', ','), 5), (('%', ')'), 5), (('%', 'b'), 10), (('&', 'informatica'), 5), (("'", ')'), 5), (("'", ','), 15), (("'", 'is'), 5), (("'", '('), 5), (("'", '.'), 10),


_____Most Commonly Repeated top five Bi-grams_____
[(('.', '['), 330), ((',', 'and'), 150), ((']', 'python'), 114), (('of', 'the'), 105), ((',', 'the'), 95)]


_____Top five sentences with commonly repeated Bi-grams_____
However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.As of SeptemberÂ 2

Process finished with exit code 0
```

Task 4:

The iris dataset is loaded taking its data and responses as input in this task. The output is a figure plotting the accuracy obtained from KNN algorithm for the K value ranging from 1 to 50.



Explanation:

When the K value increases to higher value, the model is simplest and all test data point will belong to the same class as the majority class. This is under fit with high bias and low variance, so the graph shows lesser accuracy.

When the K value decreases, say K =1 then the resolution is too fine. This is overfit with low bias and high variance, so the graph shows the higher accuracy of 100%.

IMPLEMENTATION & CODE SNIPPET

Task 1:

To implement LDA on a dataset, libraries are imported from scikit-learn. Here digits dataset is loaded as the input and where the variable X takes the data and y takes target from dataset. Target names are loaded as well to names.

```
#importing required libraries
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

#loading from the dataset
digi = datasets.load_digits()
X = digi.data
y = digi.target
names = digi.target_names #getting the target names from dataset
```

Now, a prediction model is designed by splitting the dataset with the training data of 70% and testing data of 30%, so test_size given as 0.30.

K-neighbors classification is used which fits the model according to the training data and predicts the value with the neighbors given as 5 which is stored in y_pred.

```
#importing scikit library to get the prediction model
from sklearn.model_selection import train_test_split
X_t_r, X_t, y_t_r, y_t = train_test_split(X, y, test_size=0.30)

#using K- neihbours classification
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_t_r, y_t_r)
#implementing prediction model with 30% testing data and 70% training data
y_pred = classifier.predict(X_t)
```

Now, Linear Discriminant Analysis is performed on X-testing data and y- prediction data by transforming in variation to the data collected from digits dataset.

A figure is plotted which represents 3 regions in red, blue and green showing the linear grouped data of 0,1,2.

```
#Linear Discriminant Analysis
lda = LinearDiscriminantAnalysis(n_components=2)
output = lda.fit(X_t, y_pred).transform(X)

##plotting LDA
plt.figure()
colors = ['red', 'green', 'blue']
lw = 2
for color, i, name in zip(colors, [0, 1, 2], names):
 plt.scatter(output[y == i, 0], output[y == i, 1], alpha=.8, color=color, label=name)
plt.legend(loc='best', shadow=False, scatterpoints=1)
plt.title('LDA on prediction model of DIGITS Dataset')
plt.show()
```

Task 2:

Similar to task, digits dataset is split into 80% training data and 20% testing data by importing the libraries form scikit learn.

```
from sklearn import datasets, metrics
from sklearn.cross_validation import train_test_split
#Choose one of the dataset using the datasets features in the scikit-learn
from sklearn import svm
from sklearn.datasets import load_digits

C = 1.0
digits=load_digits() #load dataset
#getting the data and response of the dataset
x=digits.data
y=digits.target

#prediction model: splitting the data to 20% testing data, 80% training data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

For applying support vector classification(SVC) on dataset, kernel is specified as the parameter. Different types of kernel are used in classification. This task involves the differences between the ‘Linear Kernel’ and ‘RBF Kernel’.

```
#Apply SVC with Linear kernel
model = svm.SVC(kernel='linear')
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print("Accuracy for SVC linear kernel " + str(metrics.accuracy_score(y_test,y_pred)))

#Apply SVC with RBF kernel
model = svm.SVC(kernel='rbf')
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print("Accuracy for SVC with rbf kernel " + str(metrics.accuracy_score(y_test,y_pred)))
```

The classification is implemented by considering the model from SVM with the specification being Linear or RBF.

Then model is fit into training co-ordinates and the output prediction is calculated. Accuracy method from the metrics library is used in calculating the difference between the test data and the predicted data.

Task 3:

Natural Language Toolkit(nltk) is imported, to get all the functions which helps to perform the functions. Word_tokenize is used in splitting the words either individually or sentence wise. WordNetLemmatizer helps in lemmatizing the words, ngrams is used for getting the bi-grams of the text. 'lines' contains all the text present in the text file which is given as the input.

```
from nltk.tokenize import word_tokenize,wordpunct_tokenize,sent_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.tag import pos_tag
import nltk
from nltk import ngrams
from operator import itemgetter

# Text file input is taken
with open('input.txt', 'r') as f:
    lines = f.readlines()
print("_____INPUT_____")
print(lines)
file_read=''
```

To get all the sentences into a single strings which helps in tokenizing the words separately and also divides sentences separately.

Lemmatization normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as lemma.

```
#Multiple lines  into single string
for m in lines:
    file_read=file_read+m
print(file_read)
#tokenize word individually and sentence wise
f_r_w = word_tokenize(file_read)
f_r_s = sent_tokenize(file_read)

#-  Using Lemmatization, apply lemmatization on the remaining words
lemmatizer = WordNetLemmatizer()
f_r_l = []
for word in f_r_w:
    frl = lemmatizer.lemmatize(word.lower())
    f_r_l.append(frl)
print("\n\n")
print("_____LEMMATIZATION_____")
print(f_r_l)
```

Bi-grams are splitting of words in text file in groups of two, with consecutive words like (one, two), (two, three),(three, four)… Ngrams is considered where 'n'value can be changes based on type of grams expecting, here n=2 being bi-grams.

```
print("\n\n")
print("_____BI-GRAMS_____")
n = 2 #giving n value as 2 for getting bi-grams
gram=[]
bigrams = ngrams(f_r_l, n)
for grams in bigrams: #for every 2 grams are grouped into a set creating bi-grams
    gram.append(grams)
print(gram)
```

For each word, pos tag is attached by using .join() method. The whole string is tokenized into words with their pos tags and FreqDist() gives the frequency of the grams containing in the text file.

Most frequently repeated grams are takes as t_c and top five bi-grams are taken into t_c_f are printed.

```
print("\n\n")
print("_____Bi-Grams with Word frequency_____")
f_d = nltk.FreqDist(gram)
t_c = f_d.most_common() #most common bi-grams
t_c_f = f_d.most_common(5) #top most common five bi-grams
top=sorted(t_c,key=itemgetter(0))
print(top)
print("\n\n")
print('_____Most Commonly Repeated top five Bi-grams_____')
print(t_c_f)
```

To get the top five sentences, text file is tokenized into sentences and then for each sentence bi-grams are generated comparing each with top five commonly repeated bi-grams deduced in the above. If both are a match then they are appended to the final string (f_s).

```
s = sent_tokenize(file_read)
f_s = []
for each in s: #for each sentence in the list of sentences
    for word,words in gram:  #bi-grams are taken
        for ((c,m), l) in t_c_f: #each bi-gram is compared with top five bi-grams
            if (word,words == c,m):
                f_s.append(each)  #if its a match then sentence is appended with the final sentence
print("\n\n")
print ("_____Top five sentences with commonly repeated Bi-grams_____")
print(max(f_s,key=len))
```

Task 4:

To implement KNN algorithm, the data is taken from IRIS dataset by importing suitable libraries from scikit learn.

```
import matplotlib.pyplot as plt
from sklearn import datasets, metrics

#Loading the dataset
irisdataset=datasets.load_iris()
#getting the data and response of the dataset
X=irisdataset.data
y=irisdataset.target
```

A prediction model is designed with 20% testing data and 80% training data and standardized scalers are imported from scikilearn preprocessing library to normalize the training and testing data of X variable.

```
#building a prediction model with 20% testing data and 80% training data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
# standard scaler methods are used for normalizing the values of training data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

A for loop takes the variations of k value by changing the n_neighbors in KNeighborsClassifier. The changing K value gives different accuracy values which are plotted in the graph.

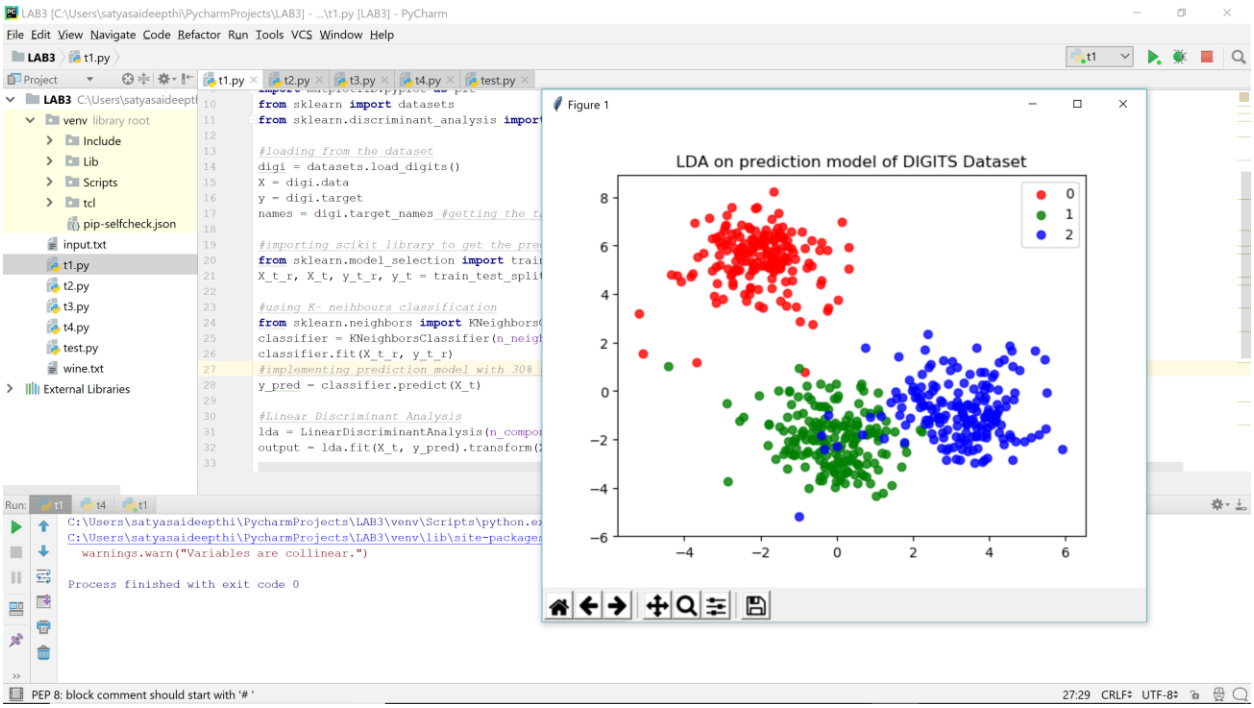
```
#Using KNN algorithm to predict the output values with neighbours changing with K value
from sklearn.neighbors import KNeighborsClassifier
scores = []
# Calculating tha accuracy for K values ranging from 1 to 50
for i in range(1, 50):
    knn = KNeighborsClassifier(n_neighbors=i) #here the algorithm is run for every k value
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test) #prediction is calculated
    scores.append(metrics.accuracy_score(y_test, pred_i)) #accuracy is appended to the scores list

#plotting the graph using matplotlib
plt.plot(range(1, 50), scores, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Accuracy - K Value')
plt.xlabel('K Value')
plt.ylabel('Accuracy')
plt.show()
```

DEPLOYMENT

For this assignment, all the code is deployed on PYCHARM software saving all the program files in .py format. The console window is used for giving the inputs and for getting the outputs from the code.

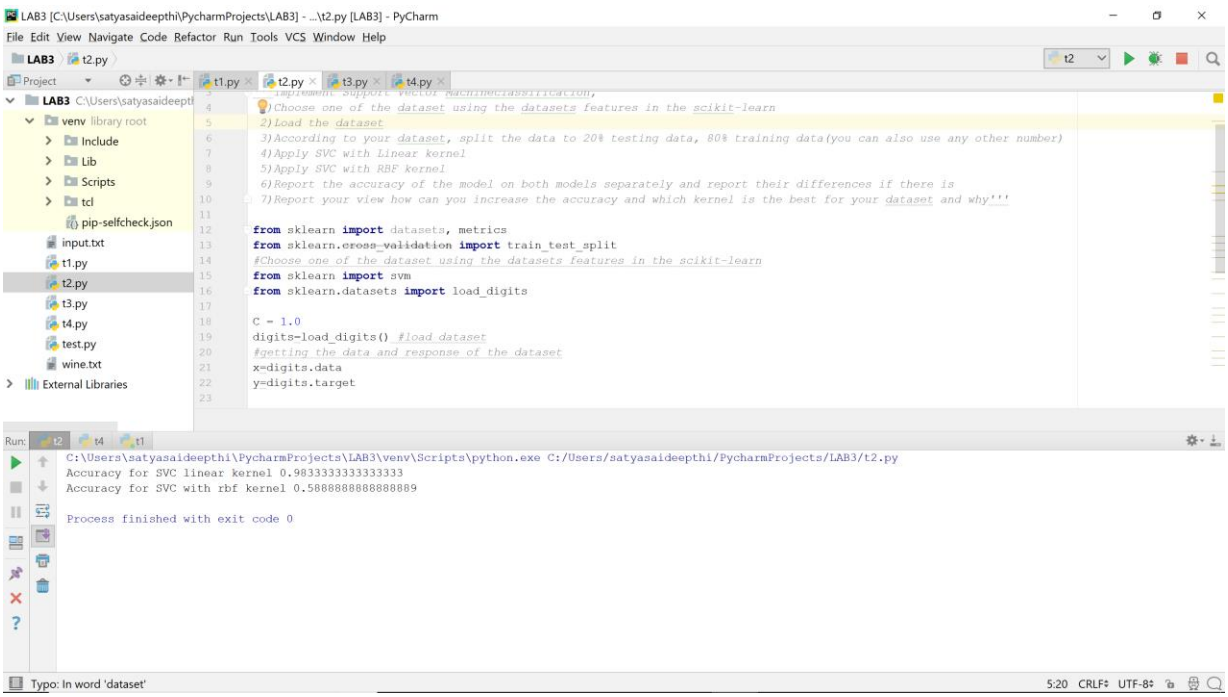
Task 1 Deployment:

The input is loading dataset which is given while implementing the code and the output is plot which is shown as a figure.



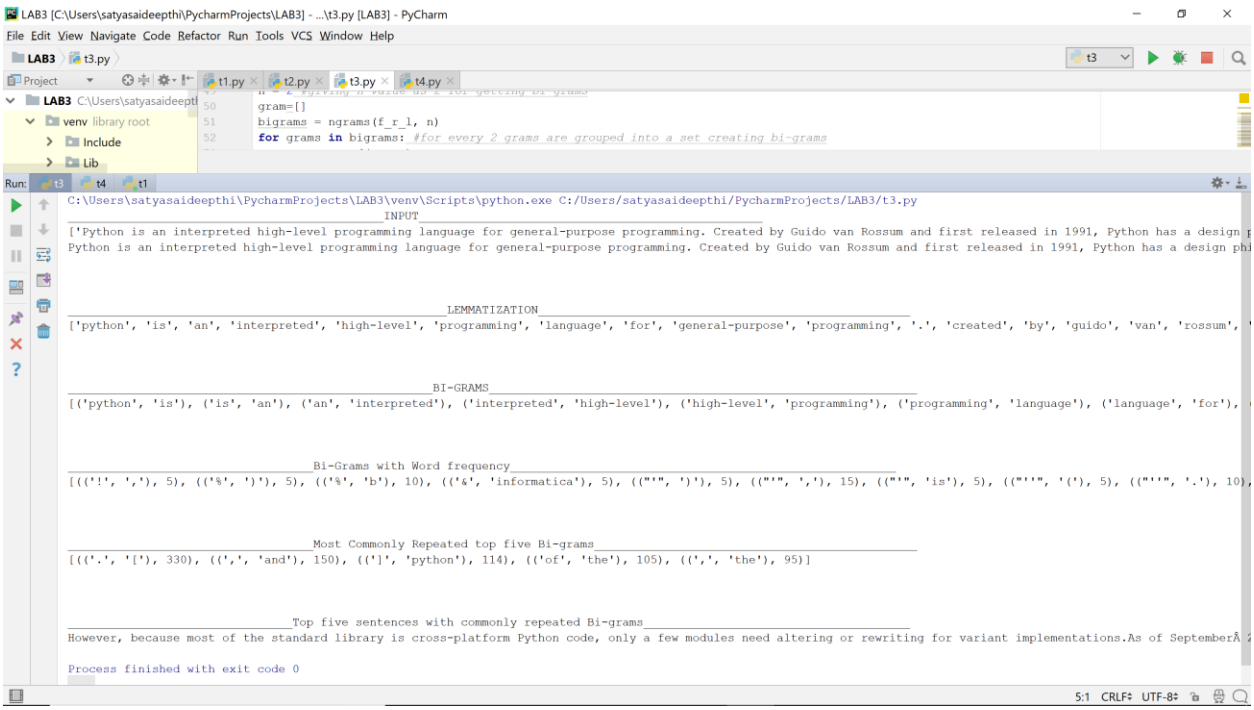
Task 2 Deployment:

The data and target responses are loaded from digits dataset as input. The accuracy values of both the classifications are taken from the console window of pycharm software.



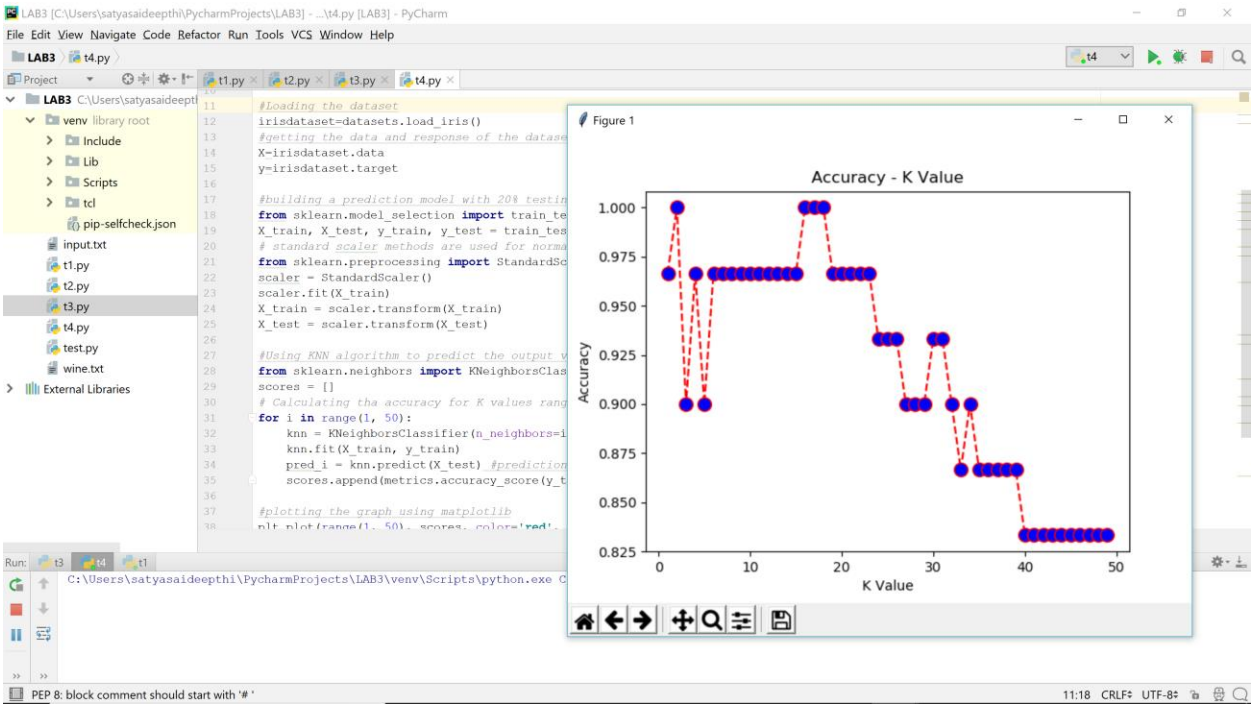
Task 3 Deployment:

Input text file reads all the lines of text and the opeartions performed on the test like lemmatiation, bi-grams on the text, word frequency of the bi-grams and the listing of top five sentences which contains most commonly repeated top five bi-grams are shown in console window.



Task 4 Deployment:

Considered input data is loaded form iris dataset and the figure depicting the accuracy and k value variation is shown as output in the form of a graph.



LIMITATIONS

The provided code meets all the requirements of the tasks given. But few limitations of the code are listed below:

- The limitation commonly observed in all the tasks is the data being loaded from the datasets randomly takes input every time when the task is executed which gives the varied results and accuracy values. But to note that each time the values depicts the same exact conclusion from each execution.
- In Task 2, limitation is considering the dataset of linear data and performing linear kernel which shows higher accuracy. Instead RBF kernel gives the accurate results for the real time non-linear problems.

- In Task 3, the input text can be limitation. Here 120KB text file is taken as input which takes execution time of about 5 secs to give the output. In the real-time applications, if the document content is in GB then the computational time would be a challenging task.
- In Task 4, using higher K value leads to less accuracy but can be used for larger datasets which is different in here where the considered dataset is very less comparably.

REFERENCES

[1] http://www.apnorton.com/blog/2016/12/19/Visualizing-Multidimensional-Data-in-Python/

[2] https://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html

[3] http://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/