Python and Deep Learning

Lab Assignment:3

-Rakesh Reddy Pallepati

-16231311

Author: Rakesh Reddy Pallepati

Objective-

To implement the following programs using python predefined methods like
1.Linear Discriminant Analysis
2.Support Vector Machines,
3.Natural Language Processing,
4.KnearestNeighbours
5.Machine learning Algorithms
6.Datasets
7.Principal Component Analysis

Features:

- These programs are developed using the Machine learning libraries
- The programs will give the required outputs according to our requirements.
- These Can also be implemented in the real time applications if needed.
- One of the program has been implemented using the Natural Language Processing concepts applying few methods like Lemmatization, Bigrams.
- Importing Sklearn, nltk, matplotlib

Configuration:
Mac OSX 10.13.3
Processor: 2.6 Ghz Intel Core i7
Memory: 16GB Ram
Graphics: Intel HD Graphics
Tool : Pycharm 2017.2.3

Input & Output Screenshots:

1.Input:

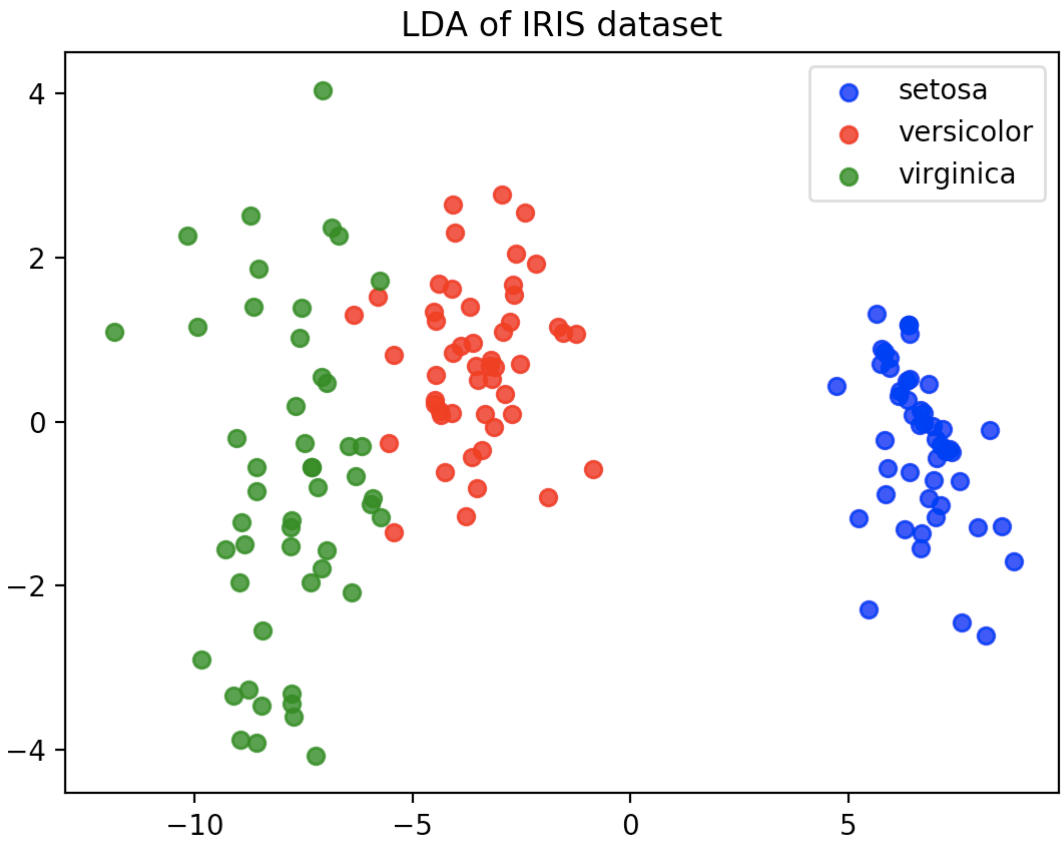
Here, I am considering the IRIS Dataset and performing the LDA Analysis on it and Ialso use the KNN classifier to classify the data

```
#lading iris data sets from all other datasets
iris = datasets.load_iris()
#loading the data
X = iris.data
#target variable
y = iris.target
#Specifying Target names
target_names = iris.target_names
```

1.Output:

Here, Setosa, Versicolor and virginica ar the various types of the Iris DataSet.

It is distributed as shown in the fig.



2.Input :

I have used the Digits data for SVM Classification, Dataset has been loaded and also performning the training split and testing split of the data. I performed the svc with the linear and RBF kernel

```
digits=load_digits()
#loading the data
x=digits.data
#target variable
y=digits.target
```

2.Output:

/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/rakeshreddypallepati/PycharmProjects/lab3/svmc.py
/Users/rakeshreddypallepati/Library/Python/3.6/lib/python/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.
"This module will be removed in 0.20.", DeprecationWarning)

Accuracy for SVC linear kernel 0.975

Accuracy for SVC with rbf kernel 0.516666666667

Process finished with exit 0.

3.Input:

I need to give the input text in file with some text data in it. So that it can find the Lemmatization, Bi grams for the input data. I have also performed for the top-five bigrams from the input text and also the sentences that top-five bigrams.

I/p:

hi this is rakesh

this is rakesh from comp science

from comp science there are different teams participating in the event

in the event is rakesh particpiating a big question to everyone ?

hello everyone how are you doing today

you will not be seen in output

3.Output:

Here Ihave used the predefined nltk libraries for tokenization, Lemmatizaiton, Bigrams. The output will be as follows:

/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/rakeshreddypallepati/PycharmProjects/lab3/task3.py
lines ['hi this is rakesh\n', 'this is rakesh from comp science\n', 'from comp science there are different teams participating in the event\n', 'in the event is rakesh particpiating a big question to everyone ?\n', 'hello everyone how are you doing today\n', 'you will not be seen in output']
hi this is rakesh
this is rakesh from comp science
from comp science there are different teams participating in the event
in the event is rakesh particpiating a big question to everyone ?
hello everyone how are you doing today
you will not be seen in output

----------lemmetaizion----------
['hi', 'this', 'is', 'rakesh', 'this', 'is', 'rakesh', 'from', 'comp', 'science', 'from', 'comp', 'science', 'there', 'are', 'different', 'team', 'participating', 'in', 'the', 'event', 'in', 'the', 'event', 'is', 'rakesh', 'particpiating', 'a', 'big', 'question', 'to', 'everyone', '?', 'hello', 'everyone', 'how', 'are', 'you', 'doing', 'today', 'you', 'will', 'not', 'be', 'seen', 'in', 'output']
----------BIGRAM----------
[('hi', 'this'), ('this', 'is'), ('is', 'rakesh'), ('rakesh', 'this'), ('this', 'is'), ('is', 'rakesh'), ('rakesh', 'from'), ('from', 'comp'), ('comp', 'science'), ('science', 'from'), ('from', 'comp'), ('comp', 'science'), ('science', 'there'), ('there', 'are'),

('are', 'different'), ('different', 'team'), ('team', 'participating'), ('participating', 'in'), ('in', 'the'), ('the', 'event'), ('event', 'in'), ('in', 'the'), ('the', 'event'), ('event', 'is'), ('is', 'rakesh'), ('rakesh', 'particpiating'), ('particpiating', 'a'), ('a', 'big'), ('big', 'question'), ('question', 'to'), ('to', 'everyone'), ('everyone', '?'), ('?', 'hello'), ('hello', 'everyone'), ('everyone', 'how'), ('how', 'are'), ('are', 'you'), ('you', 'doing'), ('doing', 'today'), ('today', 'you'), ('you', 'will'), ('will', 'not'), ('not', 'be'), ('be', 'seen'), ('seen', 'in'), ('in', 'output')]

--------Bi-Grams with word frequency----------

[(('?', 'hello'), 1), (('a', 'big'), 1), (('are', 'different'), 1), (('are', 'you'), 1), (('be', 'seen'), 1), (('big', 'question'), 1), (('comp', 'science'), 2), (('different', 'team'), 1), (('doing', 'today'), 1), (('event', 'in'), 1), (('event', 'is'), 1), (('everyone', '?'), 1), (('everyone', 'how'), 1), (('from', 'comp'), 2), (('hello', 'everyone'), 1), (('hi', 'this'), 1), (('how', 'are'), 1), (('in', 'output'), 1), (('in', 'the'), 2), (('is', 'rakesh'), 3), (('not', 'be'), 1), (('participating', 'in'), 1), (('particpiating', 'a'), 1), (('question', 'to'), 1), (('rakesh', 'from'), 1), (('rakesh', 'particpiating'), 1), (('rakesh', 'this'), 1), (('science', 'from'), 1), (('science', 'there'), 1), (('seen', 'in'), 1), (('team', 'participating'), 1), (('the', 'event'), 2), (('there', 'are'), 1), (('this', 'is'), 2), (('to', 'everyone'), 1), (('today', 'you'), 1), (('will', 'not'), 1), (('you', 'doing'), 1), (('you', 'will'), 1)]

---------Top 5 bi-grams word freq with count--------

[(('is', 'rakesh'), 3), (('this', 'is'), 2), (('from', 'comp'), 2), (('comp', 'science'), 2), (('in', 'the'), 2)]

Sentences with top five Bigrams

hi this is rakesh

this is rakesh from comp science

from comp science there are different teams participating in the event

in the event is rakesh particpiating a big question to everyone ?

Process finished with exit code 0

4.Input:

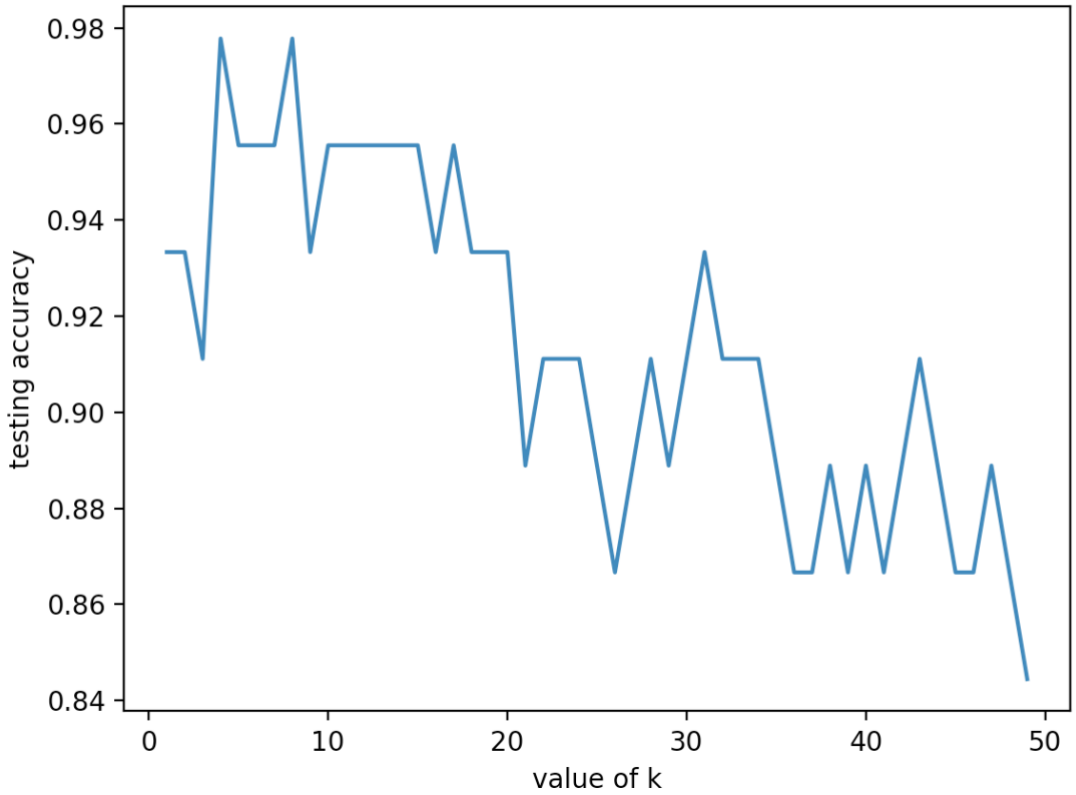
Here I have taken the Iris dataset , and performed the knn classification and here Iare trying to find how the accuracy changes as the K- value varies

```
#Loading the dataset
irisdataset=datasets.load_iris()


#getting the data and response of the dataset
x=irisdataset.data
y=irisdataset.target
```

4.Output:

The output clearly shows that the accuracy is varying as it is moving from low K-value to high K-value. Accuracy depreciates as shown below:



Implementation:

1.Linear Discriminant Analysis:

Here I am importing the matplotlib to plot the data analysis and SKLearn for the datasets and also the Linear Discriminant Analysis and later on the Kneighbours Classifier with it.

Differences:

Logistic Regression	LDA
• **Max Likelihood:** • **Calculates Probability immediately and Conditionally** • **The groups may be different** • **Not so sensitive to the outliers.** • **Latest method** • **Preferred** • **Less exigent/ More Robust**	• **Similar to Linear Regression** • **It estimates the probability with the Naïve Bayes or any other classifier** • **The groups have the similar N** • **Sensitive to the outliers** • **Old Method** • **Requirements need to be satisfied** • **Better than Logistic regression**

Code:

```
#for plotting the graph
import matplotlib.pyplot as plt
#for the datasets
from sklearn import datasets
#for performing the Linear Discriminant analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

#lading iris data sets from all other datasets
iris = datasets.load_iris()
#loading the data
X = iris.data
#target variable
y = iris.target
#Specifying Target names
target_names = iris.target_names

#splitting the train data and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
#Using the knn classifier
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
#predicting the values based on the training data
y_pred = classifier.predict(X_test)

#performing the LDA
lda = LinearDiscriminantAnalysis(n_components=2)
X_r2 = lda.fit(X_test, y_pred).transform(X)
#plotting the figure
plt.figure()
#Selecting the colors
colors = ['blue', 'red', 'green']
lw = 2
```

```
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(X_r2[y == i, 0], X_r2[y == i, 1], alpha=.8, color=color,
                label=target_name)
plt.legend(loc='best', shadow=False, scatterpoints=1)
plt.title('LDA of IRIS dataset')
#displaying the graph
plt.show()
```

2.Support Vector Machines:

SVM: It is used for supervised learning which is used to analyze the data for both classification and Regression Analysis.

Kernel are used to perform nonlinear classification.

Radial Basis Function(RBF) : It is used for scaling with Large Number of features and Training samples

Linear Kernel: SVM training algorithm will build a new

As one group or category and makes it as a Non-probabilistic linear classifier

These are used to classify for more than two types and here Iare using SVM with Linear Kernel and RBF Kernel. The dataset Iused here is an Digits dataset as an input and split the dataset into training and testing. For splitting the data Iuse the K-Fold Cross Validation**.**

```
#importing data sets
from sklearn import datasets, metrics
from scipy import sparse
#using the cross validation
from sklearn.cross_validation import train_test_split
#importing Support vector machine
from sklearn import svm
#Boston dataset is used as one of the datasets features in the scikit-learn
from sklearn.datasets import load_boston,load_digits
import numpy as np
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

C = 1.0
#getting the data and response of the dataset
#loading dataset
digits=load_digits()
#loading the data
x=digits.data
```

```
#target variable
y=digits.target

#According to your dataset, split 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)

#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)
#predicting the values based on the training data
y_pred=model.predict(x_test)
print("Accuracy for SVC with rbf kernel " +
```

3.Natural Language Processing:

Natural language processing is used to summarize the text data and to extract the information about it. It will be done in various stages. Here Ihave Implemented only few techniques and they are as follows:

Tokenization: Iwill be splitting the word documents, read the lines in the code and then process it . It converts into single words from sentences in the lines.

Lemmatization: It means to make the words or text data properly related to a vocabulary into an morphological analysis of words as it means to remove the inflectional endings as -ing, lly,ed….
Finally it returns the dictionary form of a word.

Bigrams: It comes from the n-grams where you make the combination of the words as specified accordingly to n. Here, Bigram refers to n as 2 , Bi grams are mainly used to extract the meaningful words out of it .when processing considering bi gram makes the difference.

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


#- Take an Inputfile. Use the simple approach below to summarize a text file
with open('input.txt' , 'r') as f:
    lines = f.readlines()
print("lines",lines)
fr=''
#Multi- line file into single string
for m in lines:
    fr=fr+m
print(fr)
#tokenize word
fr_word = word_tokenize(fr)
fr_sent = sent_tokenize(fr)

#- Using Lemmatization, apply lemmatization on the remaining words
lemmatizer = WordNetLemmatizer()
fr_lemma = []
for word in fr_word:
    fr_lema = lemmatizer.lemmatize(word.lower())
    fr_lemma.append(fr_lema)

print("\n ----------lemmetaizion---------  ")
print(fr_lemma)

print("-------------BIGRAM-------------")

n = 2
gram=[]
bigrams = ngrams(fr_lemma, n)
```

```
for grams in bigrams:
    gram.append(grams)
print(gram)
print("--------Bi-Grams with word  frequency----------")
fdist1 = nltk.FreqDist(gram)
top_fiv = fdist1.most_common()
top_five = fdist1.most_common(5)

top=sorted(top_fiv,key=itemgetter(0))
print(top)
print('---------Top 5 bi-grams word freq with count--------')
print(top_five)
sent1 = sent_tokenize(fr)
rep_sent1 = []

for sent in sent1:
    for word,words in gram:
        for ((c,m), l) in top_five:
            if (word,words == c,m):
                #considering oly with top-5 bigram
                rep_sent1.append(sent)
print ("\n Sentences with top five Bigrams")
s=max(rep_sent1,key=len)
print(s)
```

4.**KNearestNeighbours:** It comes under the supervised learning algorithm. It means the labelled dataset contains training observations and to form the relation between X&Y. So that when the testing data comes you will be knowing it what need to be there as output with the help of training data.

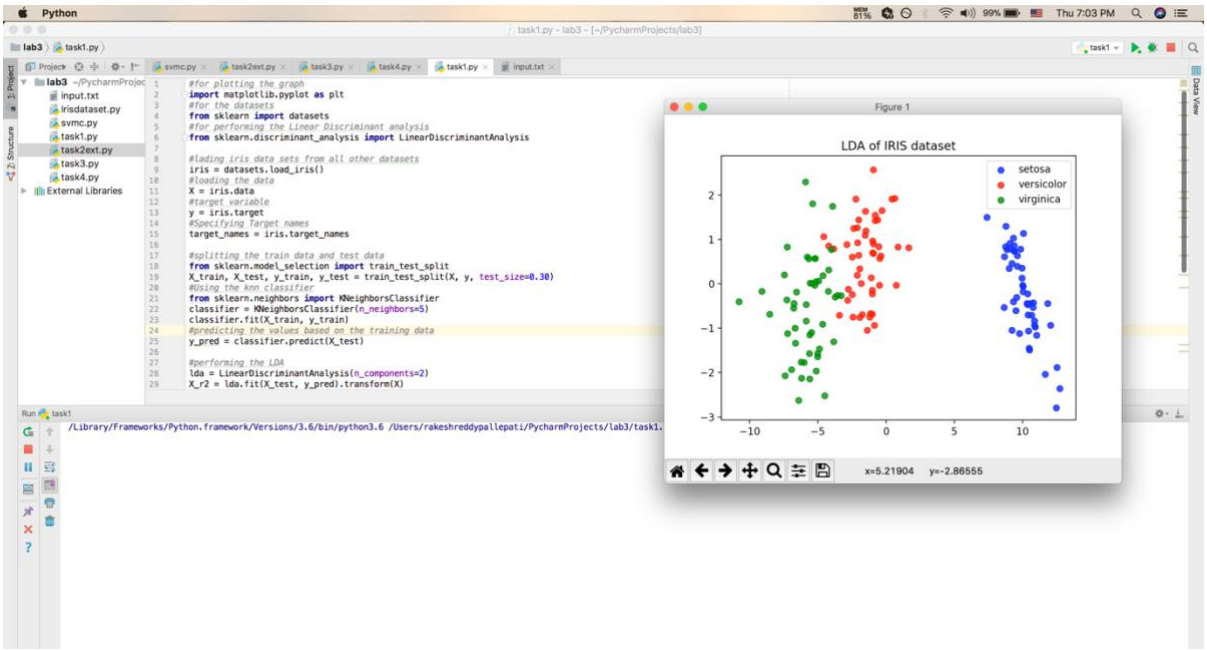
It is a non- Parametric and instance-based algorithm. As KNN is considered as the expensive testing phase which means impractical for industry settings. If there is any category that repeats most will effect the other one because it will be most common .As high value of K is computationally expensive.

Here, Ihave taken the range of K from 1 to 50and tested the accuracy with an Iris Dataset and performed it by giving the testing data of almost 20% and training Data as 80% . with nearest neighbours.

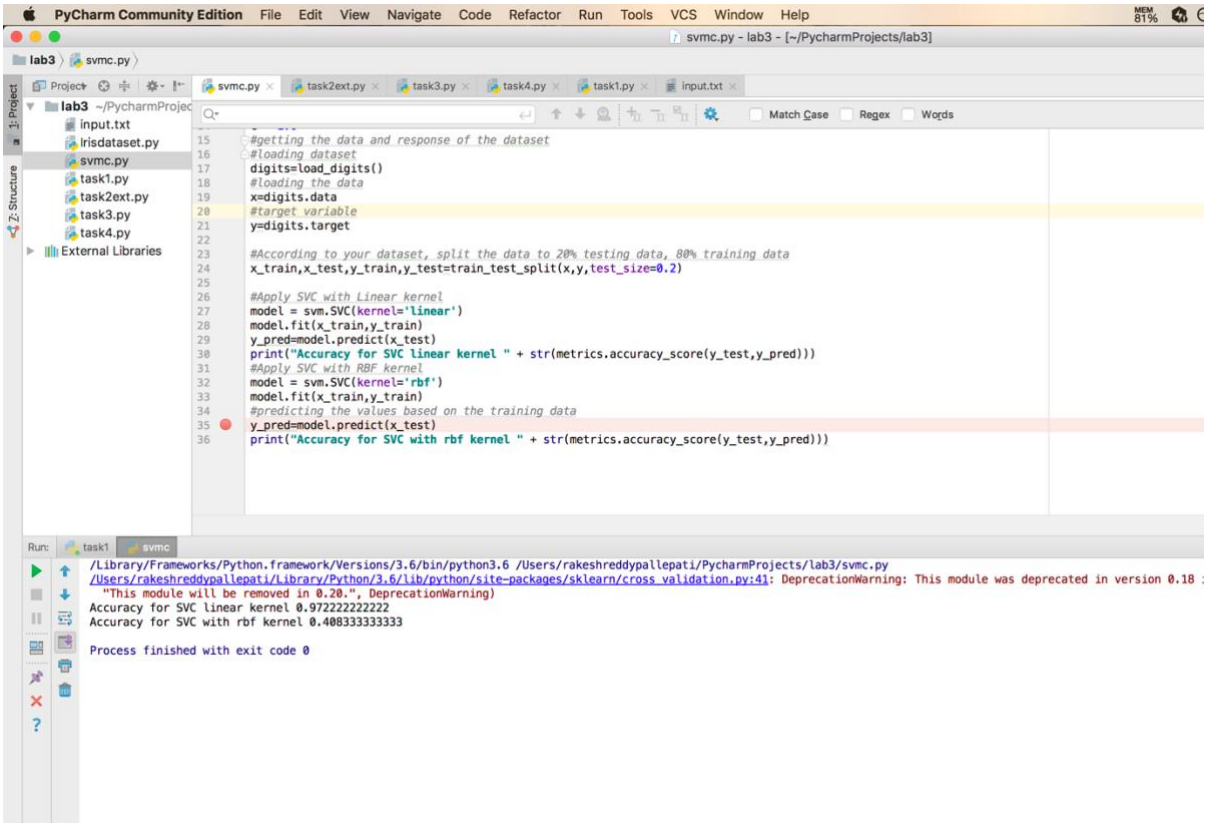
Deployment:

I have written the code in pycharm tool , In python files by importing the libraries

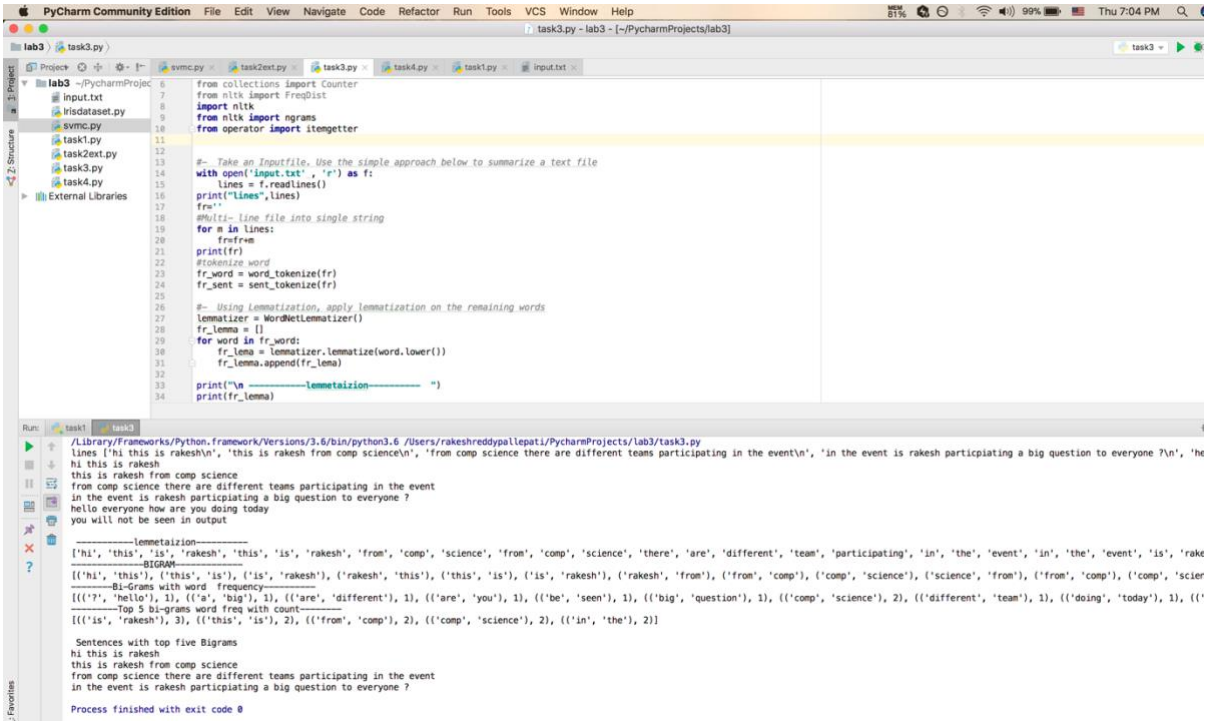
TASK1:



TASK 2:



TASK 3:



```
from collections import Counter
from nltk import FreqDist
import nltk
from nltk import ngrams
from operator import itemgetter


#-  Take an Inputfile. Use the simple approach below to summarize a text file
with open('input.txt' , 'r') as f:
    lines = f.readlines()
print("lines",lines)
fr=''
#Multi- line file into single string
for m in lines:
    fr=fr+m
print(fr)
#tokenize word
fr_word = word_tokenize(fr)
fr_sent = sent_tokenize(fr)

#-  Using Lemmatization, apply lemmatization on the remaining words
lemmatizer = WordNetLemmatizer()
fr_lemma = []
for word in fr_word:
    fr_lema = lemmatizer.lemmatize(word.lower())
    fr_lemma.append(fr_lema)

print("\n ----------lemmetaizion---------- ")
print(fr_lemma)
```

```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/rakeshreddypallepati/PycharmProjects/lab3/task3.py
lines ['hi this is rakesh\n', 'this is rakesh from comp science\n', 'from comp science there are different teams participating in the event\n', 'in the event is rakesh particpiating a big question to everyone ?\n', 'he
hi this is rakesh
this is rakesh from comp science
from comp science there are different teams participating in the event
in the event is rakesh particpiating a big question to everyone ?
hello everyone how are you doing today
you will not be seen in output

 ----------lemmetaizion----------
['hi', 'this', 'is', 'rakesh', 'this', 'is', 'rakesh', 'from', 'comp', 'science', 'from', 'comp', 'science', 'there', 'are', 'different', 'team', 'participating', 'in', 'the', 'event', 'in', 'the', 'event', 'is', 'rake
----------BIGRAM----------
[('hi', 'this'), ('this', 'is'), ('is', 'rakesh'), ('rakesh', 'this'), ('this', 'is'), ('is', 'rakesh'), ('rakesh', 'from'), ('from', 'comp'), ('comp', 'science'), ('science', 'from'), ('from', 'comp'), ('comp', 'scier
---------Bi-Grams with word  frequency----------
[(('?', 'hello'), 1), (('a', 'big'), 1), (('are', 'different'), 1), (('are', 'you'), 1), (('be', 'seen'), 1), (('big', 'question'), 1), (('comp', 'science'), 2), (('different', 'team'), 1), (('doing', 'today'), 1), ((
---------Top 5 bi-grams word freq with count--------
[(('is', 'rakesh'), 3), (('this', 'is'), 2), (('from', 'comp'), 2), (('comp', 'science'), 2), (('in', 'the'), 2)]

 Sentences with top five Bigrams
hi this is rakesh
this is rakesh from comp science
from comp science there are different teams participating in the event
in the event is rakesh particpiating a big question to everyone ?

Process finished with exit code 0
```

TASK 4:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets, metrics
from sklearn.model_selection import train_test_split
#Loading the dataset
irisdataset=datasets.load_iris()

#getting the data and response of the dataset
x=irisdataset.data
y=irisdataset.target

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

#split the data for training and testing
model= KNeighborsClassifier(n_neighbors=5)
model.fit(x_train,y_train)

y_pred=model.predict(x_test)

print("Accuracy: ",metrics.accuracy_score(y_test,y_pred))

k_range=range(1,50)
scores=[]

for k in k_range:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train,y_train)
    y_pred=knn.predict(x_test)
    scores.append(metrics.accuracy_score(y_test,y_pred))
```



```
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/rakeshreddypallepati/PycharmProjects/lab3/task4.py
Accuracy:  0.933333333333

Process finished with exit code 0
```

Limitation:

1.Here, Iare considering few datsets and performing the Data Analysis it will be taking the testing data and training data which has been split from the original data . So , Icouldn't guarantee the consistent or the same accuracy as it varies.

2. In Natural Language Processing, Ihaven't take the final advantage of the Bi-grams or N-grams. Here Iperformed only the bigrams on the data but to get the full advantage of it you need to store the bi-grams output which are meaning full and related to text. It is not performed as not mentioned in the requirements .

3.KNN algorithm will be Computationally expensive to perform when it is having the large K value in real time dataset. KNN algorithm may not find its efficient use if there is a large group of same kind among others as it will be the most common when considering the neighbours and results to decrease in the Accuracy.

4. The limitation in SVM will be the Speed and Size because of its high algorithm complexity and the extensive memory requirements, both in the training and testing the dataset.

References:

1. https://en.wikipedia.org/wiki/Linear_discriminant_analysis
2. https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/
3. https://en.wikipedia.org/wiki/Support_vector_machine
4. http://scikit-learn.org/stable/modules/svm.html