

CS 5590 - Python and Deep Learning

Lab Assignment 3

Submitted by

Vamsi Krishna Challa

16242688

Objective:

Perform various operations using machine learning mechanisms.

- ✓ **Problem 1:** To perform linear discriminant analysis on the dataset of our choice
- ✓ **Problem 2:** To implement SVM classification with linear and Rbf kernel.
- ✓ **Problem 3:** To tokenize, lemmatize content from an input file and to find top five bigrams from them and also find the sentences that contain the top 5 bigrams.
- ✓ **Problem 4:** Analysis over K-nearest neighbours algorithm.

Features:

In this assignment we access both python inbuilt features and by installing required packages

1. Access scikit learn to import linear discriminant analysis package to generate the model
2. Access scikit learn package to fit svm classifier
3. Access nltk package to perform various operations like tokenization, lemmatization, generating bigrams.
4. Access sklearn to perform K means nearest algorithm analysiss.

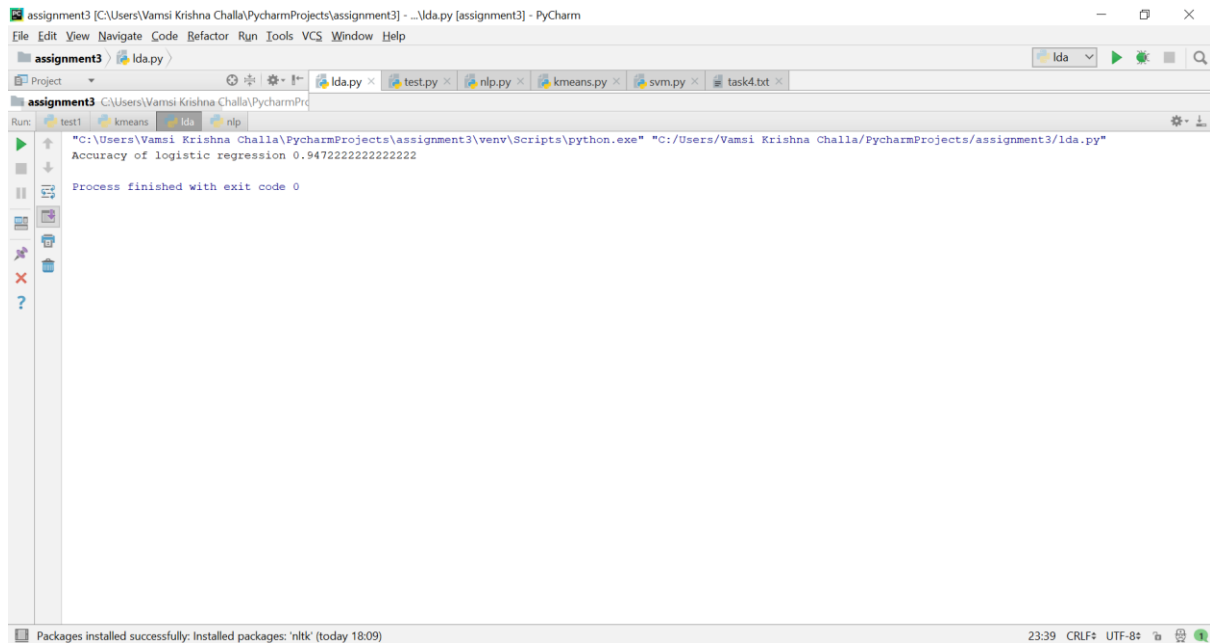
Configuration:

Software used: Python 3.4

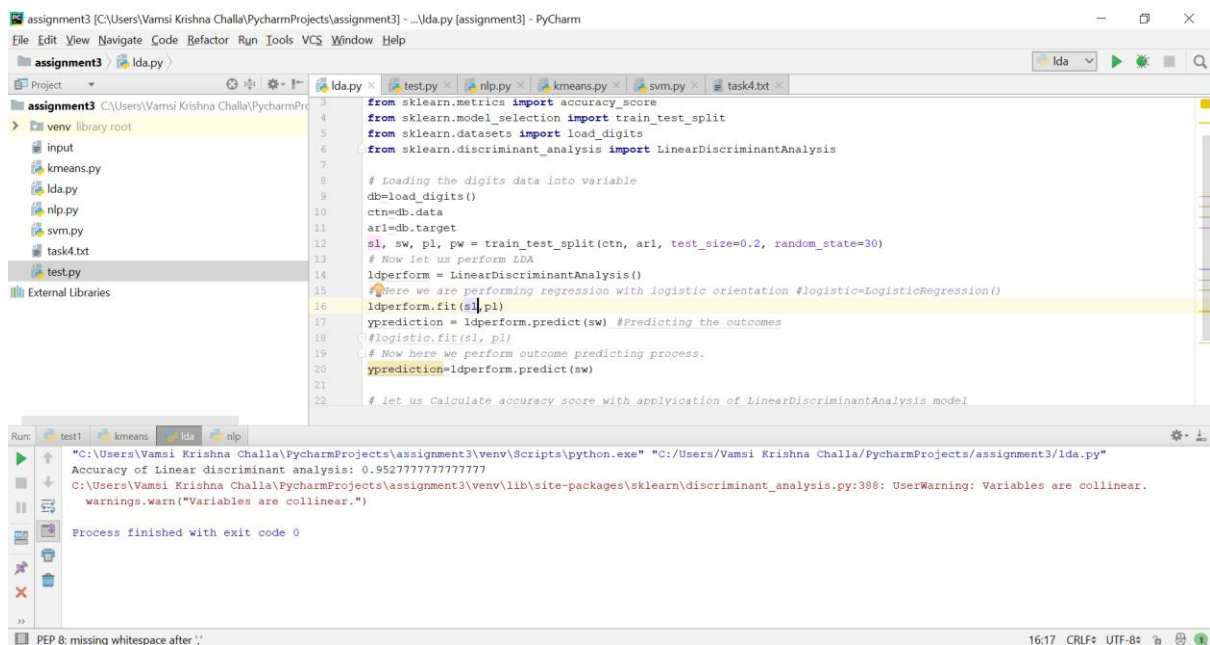
IDE: PyCharm

Input / Output:

1. Program for performing linear discriminant analysis on the Data set we choose:



```
assignment3 [C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3] - ...lda.py [assignment3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project assignment3 C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3
Run: test1 kmeans lda nlp
"C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\Scripts\python.exe" "C:/Users/Vamsi Krishna Challa/PycharmProjects/assignment3/lda.py"
Accuracy of logistic regression 0.9472222222222222
Process finished with exit code 0
Packages installed successfully. Installed packages: 'nltk' (today 18:09) 23:39 CRLF+ UTF-8+
```

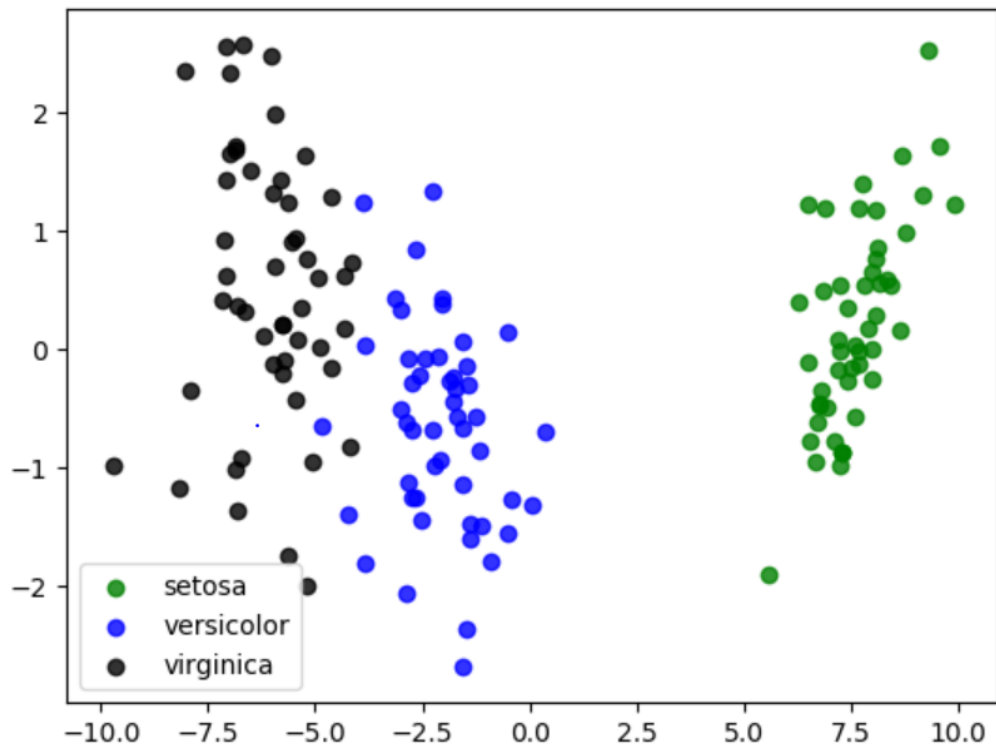


```
assignment3 [C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3] - ...lda.py [assignment3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project assignment3 C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3
Run: test1 kmeans lda nlp
"C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\Scripts\python.exe" "C:/Users/Vamsi Krishna Challa/PycharmProjects/assignment3/lda.py"
Accuracy of Linear discriminant analysis: 0.9527777777777777
C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning: Variables are collinear.
warnings.warn("Variables are collinear.")
Process finished with exit code 0
PEP 8: missing whitespace after ' ' 16:17 CRLF+ UTF-8+
```

```
3 from sklearn.metrics import accuracy_score
4 from sklearn.model_selection import train_test_split
5 from sklearn.datasets import load_digits
6 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
7
8 # Loading the digits data into variable
9 db=load_digits()
10 ctn=db.data
11 ari=db.target
12 sl, sw, pl, pw = train_test_split(ctn, ari, test_size=0.2, random_state=30)
13 # Now let us perform LDA
14 ldperform = LinearDiscriminantAnalysis()
15 # Here we are performing regression with logistic orientation #logistic=LogisticRegression()
16 ldperform.fit(sl, pl)
17 yprediction = ldperform.predict(sw) #Predicting the outcomes
18 #logistic.fit(sl, pl)
19 # Now here we perform outcome predicting process.
20 yprediction=ldperform.predict(sw)
21
22 # let us Calculate accuracy_score with application of LinearDiscriminantAnalysis model
```

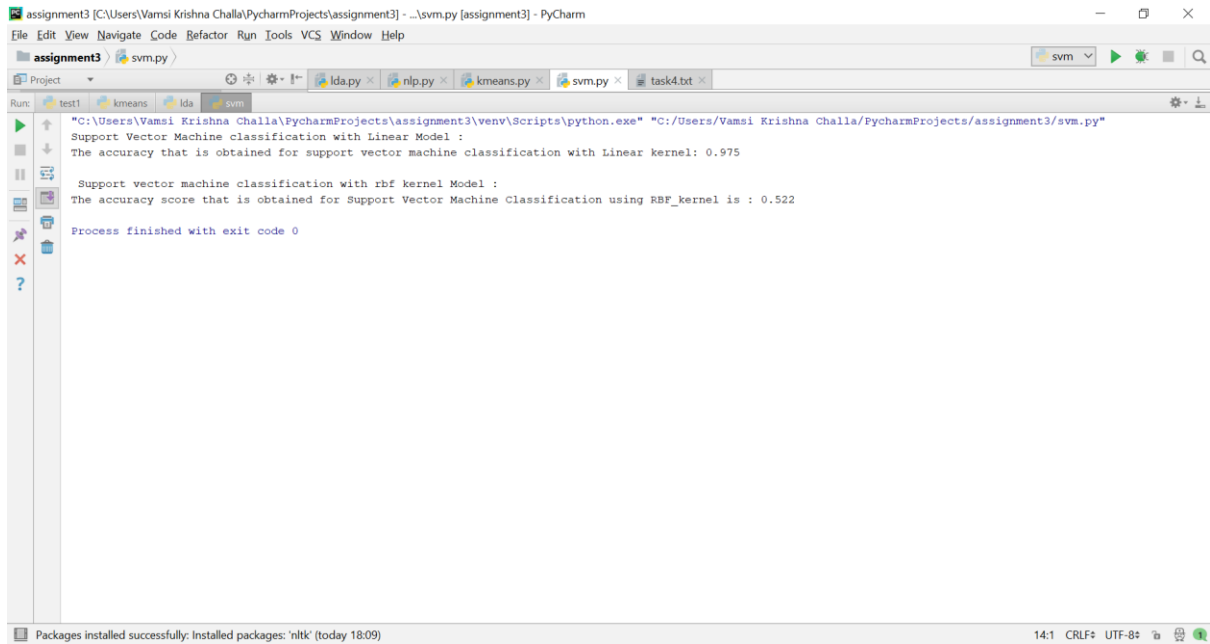
Figure 1

Linear Discriminant Analysis results performed on Iris data set.



x=-10.0553 y=0.910501

2. Program to perform support vector classification on linear kernel and rbf kernel:



The image shows a PyCharm IDE window titled 'assignment3 [C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3] - svm.py [assignment3] - PyCharm'. The Run console displays the output of a Python script. The output text is as follows:

```
"C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\Scripts\python.exe" "C:/Users/Vamsi Krishna Challa/PycharmProjects/assignment3/svm.py"  
Support Vector Machine classification with Linear Model :  
The accuracy that is obtained for support vector machine classification with Linear kernel: 0.975  
  
Support vector machine classification with rbf kernel Model :  
The accuracy score that is obtained for Support Vector Machine Classification using RBF_kernel is : 0.522  
  
Process finished with exit code 0
```

At the bottom of the console, a status bar indicates 'Packages installed successfully: Installed packages: 'nltk' (today 18:09)' and the system clock shows '14:1 CRLF UTF-8'.

3. Program to take input text file and perform lemmatization, tokenization, generating bigrams and searching for the sentences with top 5 bigrams.

```
assignment3 [C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3] - ...nlp.py [assignment3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help

assignment3 > nlp.py >
Run: test1 kmeans lda nlp
"C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\Scripts\python.exe" "C:/Users/Vamsi Krishna Challa/PycharmProjects/assignment3/nlp.py"
Contents of the file:
This is python and deep learning assignment 3. This is Vamsi Krishna Challa, Submitting the assignment 3. We have gone through various types of machine learning tec
-----
Lemmatization with verb form of the words:
This
be
python
and
deep
learn
assignment
3
.
This
be
Vamsi
Krishna
Challa
.
Submitting
the
assignment
3
.
We
have
go
through
various
type
-----
Packages installed successfully: Installed packages: 'nltk' (today 18:09) 17:34 CRLF+ UTF-8+ 1
```

```
assignment3 [C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3] - ...nlp.py [assignment3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help

assignment3 > nlp.py >
Run: test1 kmeans lda nlp
SUU
company
.
-----
Performing bi-gram on the text:
[('This', 'is'), ('is', 'python'), ('python', 'and'), ('and', 'deep'), ('deep', 'learning'), ('learning', 'assignment'), ('assignment', '3'), ('3', '.'), ('.', 'This'), ('This', 'is'), ('is', 'python'), ('python', 'and'), ('and', 'deep'), ('deep', 'learning'), ('learning', 'assignment'), ('assignment', '3'), ('3', '.'), ('.', 'This')]
-----
Calculating the word frequency of Bi-Gram:
Counter({'This', 'is': 3, ('This', 'is'): 2, ('assignment', '3'): 2, ('3', '.'): 2, ('machine', 'learning'): 2, ('of', 'python'): 2, ('python', 'programming'): 2, ('programming', 'techniques'): 2, ('techniques', 'throughout'): 2, ('throughout', 'this'): 2, ('this', 'course'): 2, ('course', 'enabled'): 2, ('enabled', 'me'): 2, ('me', 'to'): 2, ('to', 'learn'): 2, ('learn', 'basics'): 2, ('basics', 'of'): 2, ('of', 'python'): 2, ('python', 'programming'): 2, ('programming', 'with'): 2, ('with', 'machine'): 2, ('machine', 'learning'): 2, ('learning', 'orientation'): 2, ('orientation', 'helped'): 2, ('helped', 'me'): 2, ('me', 'brush'): 2, ('brush', 'up'): 2, ('up', 'my'): 2, ('my', 'skills'): 2, ('skills', 'in'): 2, ('in', 'programming'): 2})
-----
Finding the top 5 bigrams:
[ (('This', 'is'), 3), (('This', 'is'), 2), (('assignment', '3'), 2), (('3', '.'), 2), (('machine', 'learning'), 2) ]
-----
Summarising the text and finding out the lines with top five bigrams

This is python and deep learning assignment 3.

This is Vamsi Krishna Challa, Submitting the assignment 3.

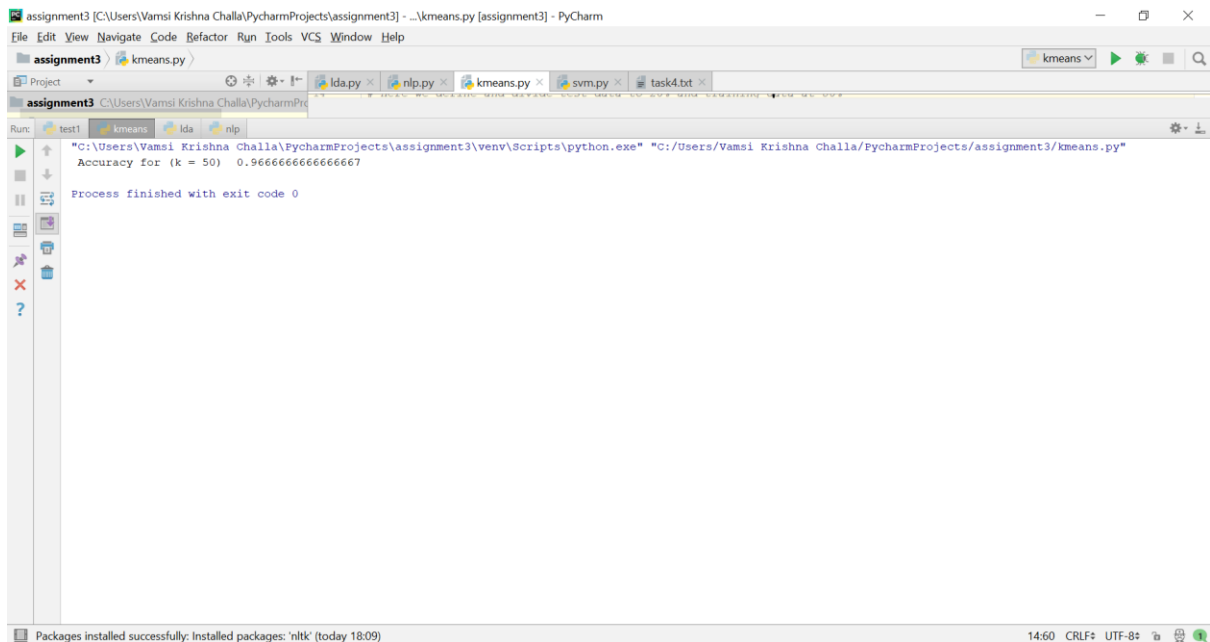
We have gone through various types of machine learning techniques throughout this course.

This course enabled me to learn basics of python programming with machine learning orientation.

This had helped me brush up my skills in programming.

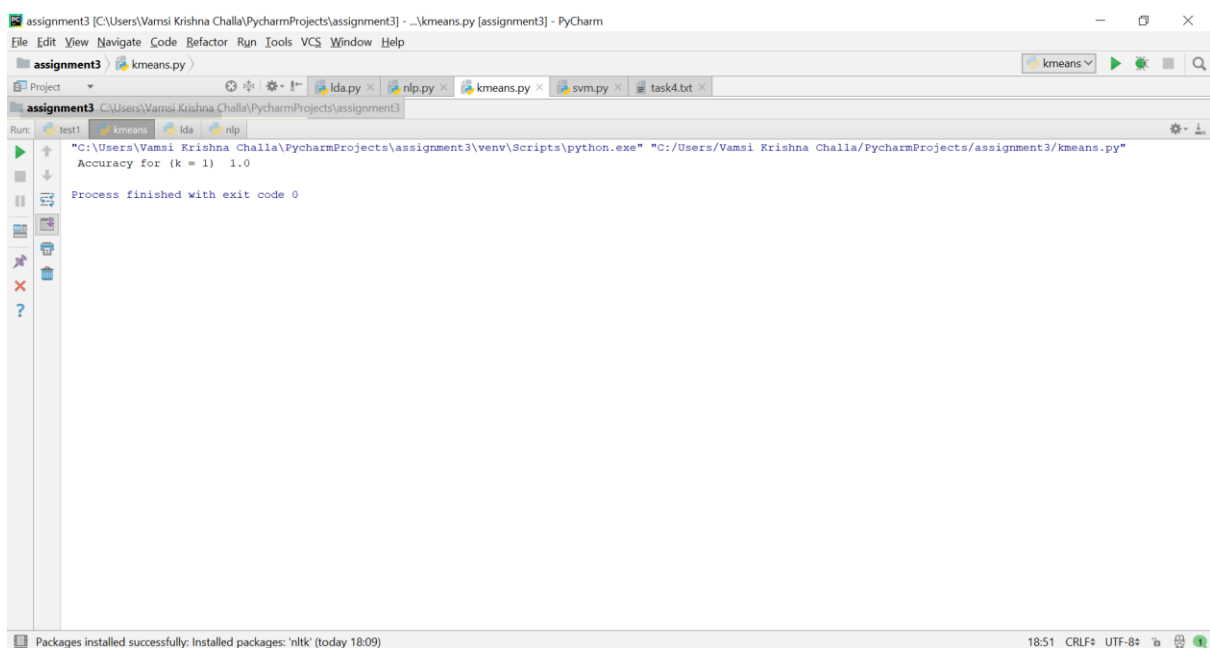
Process finished with exit code 0
-----
Packages installed successfully: Installed packages: 'nltk' (today 18:09) 17:34 CRLF+ UTF-8+ 1
```

4. Program to perform K Nearest Neighbours to generate accuracy for one cluster and 50 clusters over the given data set:



This screenshot shows the PyCharm IDE interface for a project named 'assignment3'. The 'Run' console at the bottom displays the output of a Python script. The script has executed successfully, showing the accuracy for k=50 as 0.9666666666666667. The status bar at the bottom indicates that packages were installed successfully, including 'nltk'.

```
assignment3 [C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3] - ...kmeans.py [assignment3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project assignment3 kmeans.py
Run: test1 kmeans.py
"C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\Scripts\python.exe" "C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\kmeans.py"
Accuracy for (k = 50) 0.9666666666666667
Process finished with exit code 0
Packages installed successfully. Installed packages: 'nltk' (today 18:09) 14:60 CRLF+ UTF-8+
```



This screenshot shows the PyCharm IDE interface for the same project. The 'Run' console displays the output for k=1, showing an accuracy of 1.0. The status bar at the bottom indicates that packages were installed successfully, including 'nltk'.

```
assignment3 [C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3] - ...kmeans.py [assignment3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project assignment3 kmeans.py
Run: test1 kmeans.py
"C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\Scripts\python.exe" "C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\kmeans.py"
Accuracy for (k = 1) 1.0
Process finished with exit code 0
Packages installed successfully. Installed packages: 'nltk' (today 18:09) 18:51 CRLF+ UTF-8+
```

Implementation:

Program 1:

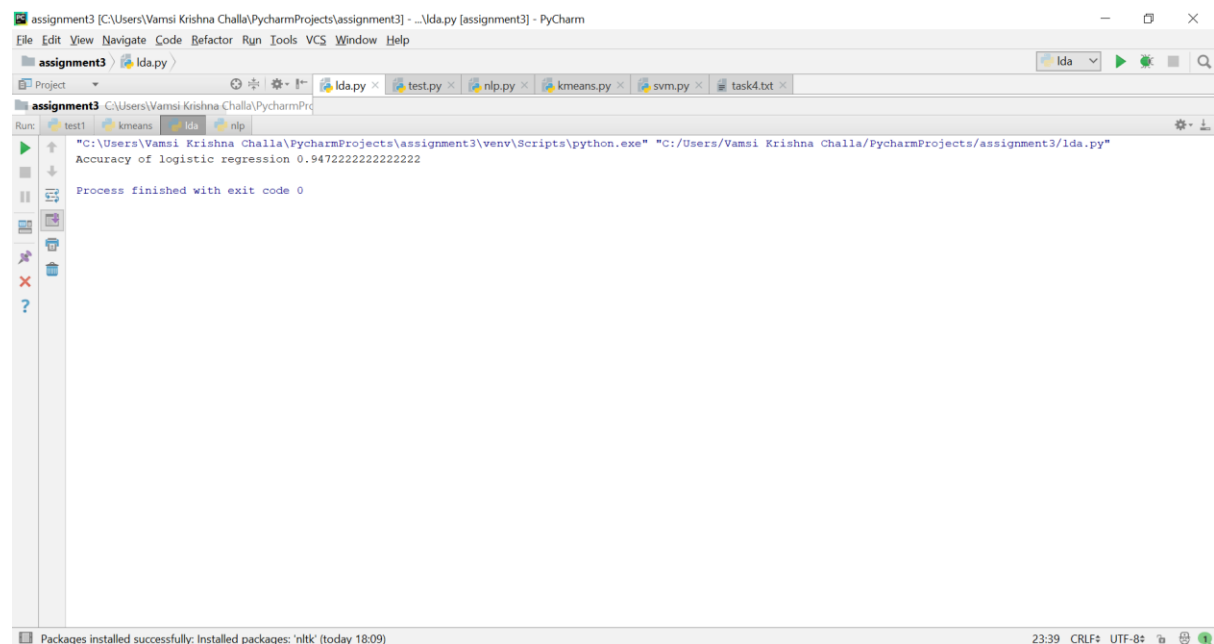
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_digits
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
import matplotlib.pyplot as plot

# Loading the digits data into variable
db=load_digits()
ctn=db.data
arl=db.target
sl, sw, pl, pw = train_test_split(ctn, arl, test_size=0.2, random_state=30)
# Now let us perform LDA
ldperform = LinearDiscriminantAnalysis()
# Here we are performing regression with logistic orientation
#logistic=LogisticRegression()
a = ldperform.fit(sl,pl)
yprediction = ldperform.predict(sw) #Predicting the outcomes
#logistic.fit(sl, pl)
# Now here we perform outcome predicting process.
yprediction=ldperform.predict(sw)

# let us Calculate accuracy score with applycation of LinearDiscriminantAnalysis
model
print("Accuracy of Linear discriminant analysis:", accuracy_score(yprediction, pw))

plot.figure()
col = ['green', 'red', 'blue']
for x, y, z in zip(col, [0, 1, 2], db):
    plot.scatter(a[arl == y, 0], a[arl == y, 1], alpha=.8, color=x, label=z)
plot.legend(loc='best', shadow=False, scatterpoints=1)
plot.title('Linear Discriminant Analysis results performed on Iris data set:')
plot.show()
```

Result:

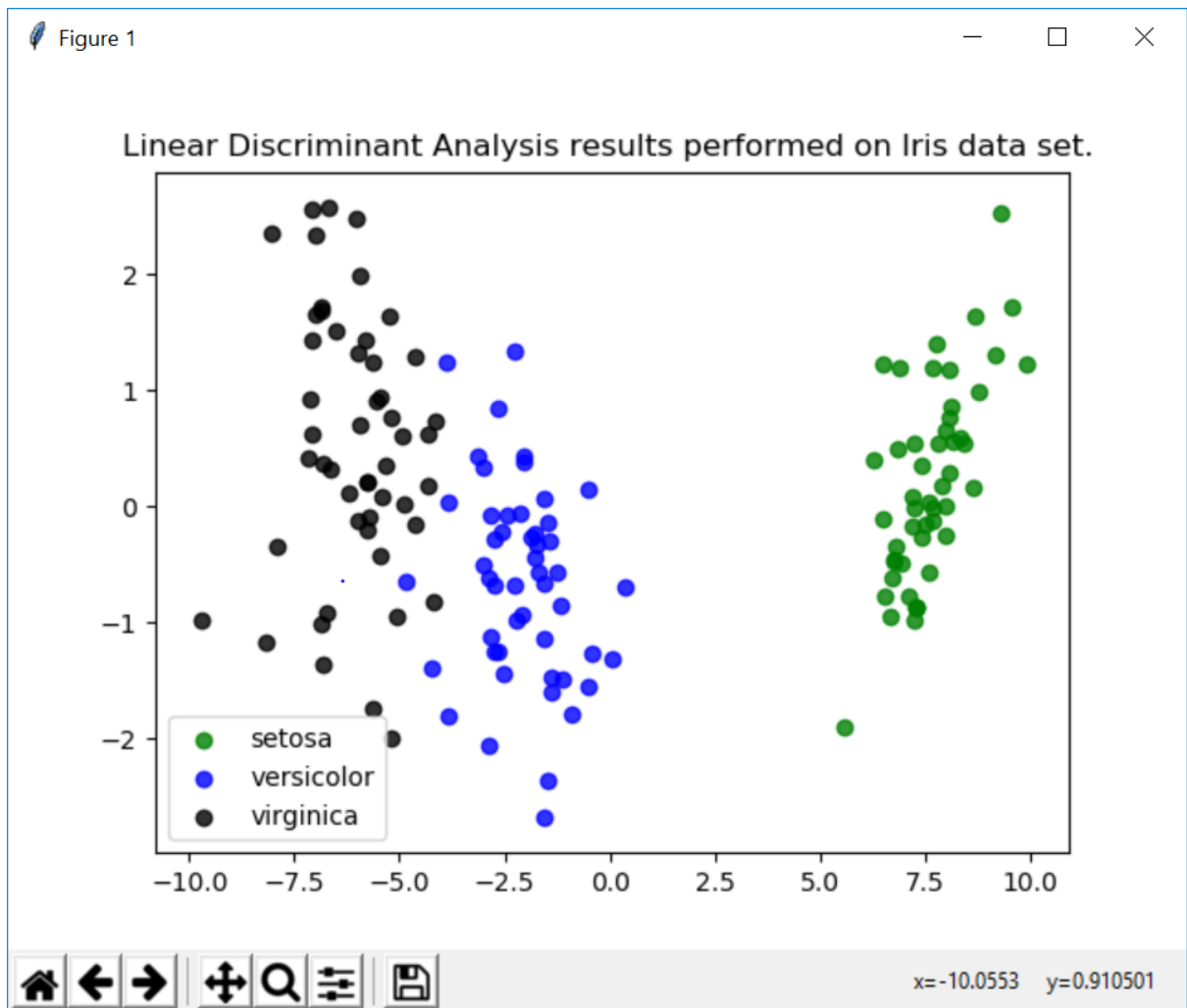



```
assignment3 [C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3] - ... \lda.py [assignment3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help

assignment3 > lda.py
Project
assignment3
  venv library root
  input
  kmeans.py
  lda.py
  nlp.py
  svm.py
  task4.txt
  test.py
  External Libraries

Run:
  test1 kmeans lda nlp
  "C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\Scripts\python.exe" "C:/Users/Vamsi Krishna Challa/PycharmProjects/assignment3/lda.py"
  Accuracy of Linear discriminant analysis: 0.9527777777777777
  C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning: Variables are collinear.
    warnings.warn("Variables are collinear.")
  Process finished with exit code 0

PEP 8: missing whitespace after ',' 16:17 CRLF: UTF-8
```



Analysis over LDA and Logistic regression:

In the case of both logistic regression and LDA I.E., Linear discriminant analysis, two of them are from the linear_model. Both have similarities, but their actual difference tolls up when we perform the bit and the way we access the parameters. Quite often, Logistic regression is preferred over LDA as it is very much robust in nature. But in the case, when we have all the requirements to perform classification LDA performs better over Logistic regression.

Program 2:

Program to perform Support vector classification over linear kernel and rbf kernel:

```
5. from sklearn import svm
from sklearn import datasets
from sklearn import metrics
from sklearn.model_selection import train_test_split

# Default data set from the available, digits is being loaded
iris = datasets.load_digits()

# loading data into the variable
xload = iris.data

# loading target into the variable
yload = iris.target

# We are going to divide the test data and training data in
xtrainer, xtesting, ytrainer, ytesting = train_test_split(xload, yload,
test_size=0.2)

print("Support Vector Machine classification with Linear Model :")

# We are Creating model for Support vector machine with linear_kernel
linearkernel = svm.SVC(kernel="linear")
linearkernel.fit(xtrainer, ytrainer)
yprediction = linearkernel.predict(xtesting)

print("The accuracy that is obtained for support vector machine
classification with Linear kernel: %.3f" % metrics.accuracy_score(ytesting,
yprediction))

print("\n Support vector machine classification with rbf kernel Model :")

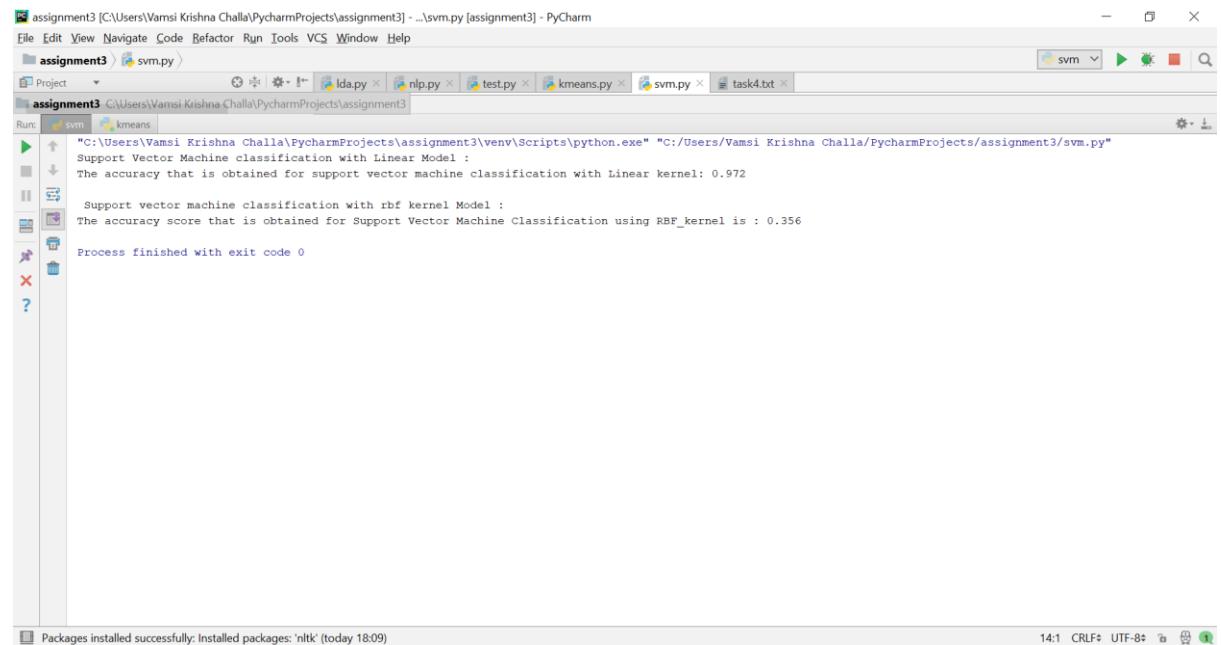
# Creating model for Support Vecctor Machinde using RBF_kernel
rbfmodel = svm.SVC(kernel="rbf")

# Fitting the data
rbfmodel.fit(xtrainer, ytrainer)

yprediction = rbfmodel.predict(xtesting)

# Checking the accuracy score on comparing ytesting data and ypredicting
data
print("The accuracy score that is obtained for Support Vector Machine
Classification using RBF_kernel is : %.3f" %
metrics.accuracy_score(ytesting, yprediction))
```

Result Summarization:



The screenshot shows the PyCharm IDE interface. The top toolbar has a dropdown menu set to 'svm'. The 'Run' console on the left displays the following output:

```
"C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\Scripts\python.exe" "C:/Users/Vamsi Krishna Challa/PycharmProjects/assignment3/svm.py"
Support Vector Machine classification with Linear Model :
The accuracy that is obtained for support vector machine classification with Linear kernel: 0.972

Support vector machine classification with rbf kernel Model :
The accuracy score that is obtained for Support Vector Machine Classification using RBF_kernel is : 0.356

Process finished with exit code 0
```

At the bottom of the IDE, a status bar indicates 'Packages installed successfully: Installed packages: 'nltk' (today 18:09)' and the encoding is '14:1 CRLF UTF-8'.

Analysis over rbf and liner kernel accuracies:

The accuracy of SVM classification over linear kernel:

The accuracy of SVM classification over RBF kernel:

From the data above, it suggests that linear kernel is better suitable than rbf kernel to predict the data model for the iris data set. But we may not confirm the same for the other data sets. In general we know that, when the features are more, linear kernel outperforms over rbf kernel as there is no need of high dimensional space to predict the features. But this doesn't mean that it becomes true in all the cases as a properly tuned rbf kernel can never outperform linear kernel as well.

Program 3:

Program to take input text file and perform lemmatization, tokenization, generating bigrams and searching for the sentences with top 5 bigrams.

```
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.collocations import ngrams
from nltk.util import bigrams
import collections
from collections import Counter
import nltk
from operator import itemgetter

# Opening the input text file with read mode.
f = open("task4.txt", "r")

# Reading the text file
contentsoffile = f.read()

# Printing the contents of the file
print("Contents of the file:", "\n", contentsoffile)
print("-----")
print("-----")

# Performing word lemmatization
lem = WordNetLemmatizer()

# Tokenizing the contents of the file.
words = word_tokenize(contentsoffile)

print("Lemmatization with verb form of the words:")
for word in words:
    print(lem.lemmatize(word, pos='v'))

print("-----")
print("-----")
print("Performing bi-gram on the text:")
cp = word_tokenize(contentsoffile)
li = []

# Accessing ngram function to find bigrams from the given text.
bigramfinder = ngrams(cp, 2)
for a in bigramfinder:
    li.append(a)
print(li)

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

# Using counter function to count the number of occurrences of each bigram.
word_count = Counter(li)
print(" Calculating the word frequency of Bi-Gram:")
print(word_count)

print("-----")
print("-----")
print("Finding the top 5 bigrams:")
freqcnt = nltk.FreqDist(li)
tf = freqcnt.most_common(5)

# Printing top five bigrams which are found out using most_common function.
print(tf)
```

```

with open('task4.txt' , 'r') as file:
    ln = file.readlines()
    fit= ''

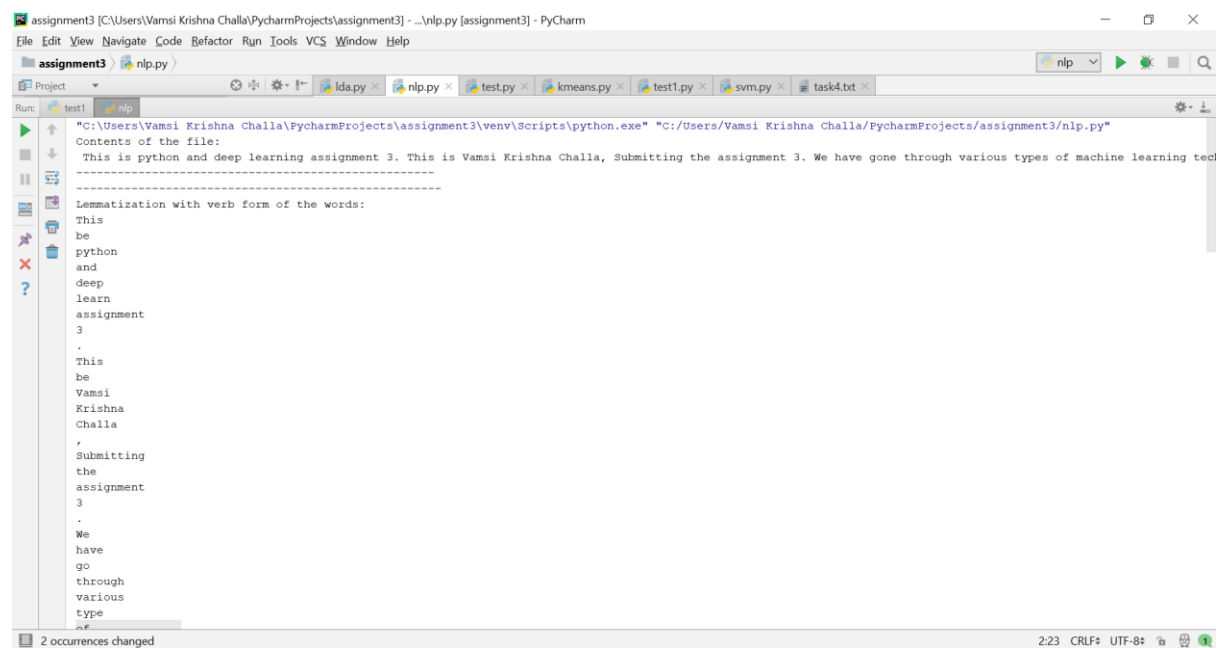
for mp in ln:
    fit= fit + mp
senti = sent_tokenize(fit)
rep_sen1 = []

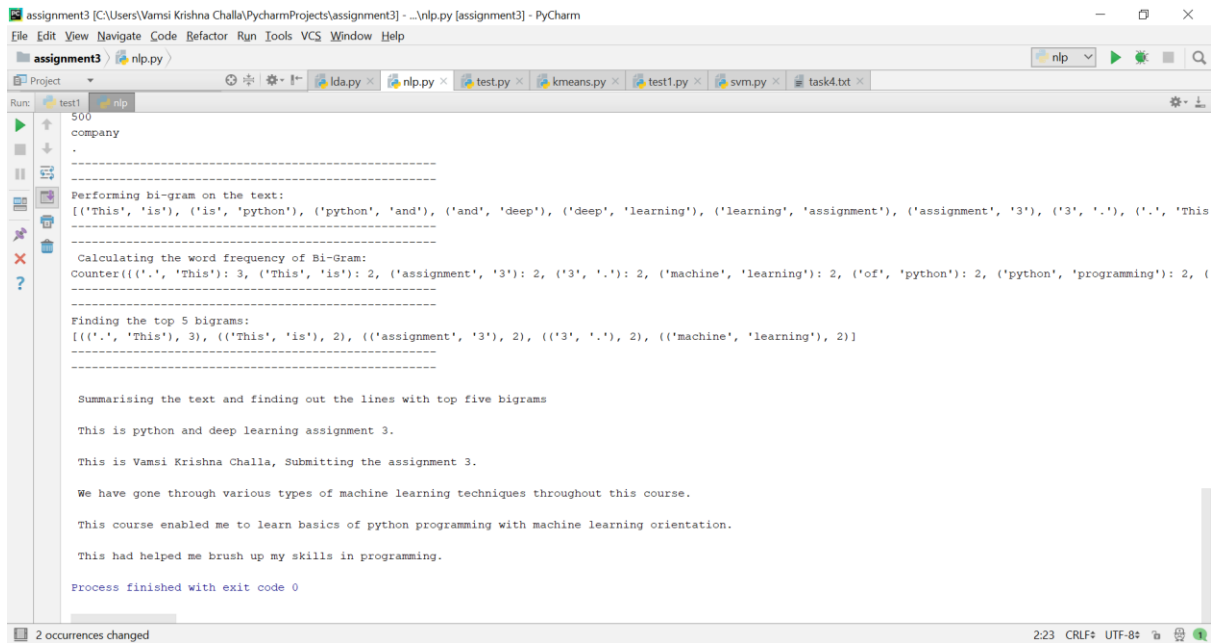
for sentences in senti:
    for word,words in li:
        for ((cp, mp), l) in tf:
            if (word, words == cp, mp):
                rep_sen1.append(sentences)

# Printing the lines of text that contain the most common top five bigrams.
print("-----")
print("-----")
print ("\n Summarising the text and finding out the lines with top five bigrams")
fc = nltk.FreqDist(rep_sen1)
fff = fc.most_common(5)
for ke,val in fff:
    print ("\n",ke)

```

results summary:





```
assignment3 [C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3] - \nlp.py [assignment3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
assignment3 \nlp.py
Project
Run test1 nlp
500
company
.
-----
Performing bi-gram on the text:
[('This', 'is'), ('is', 'python'), ('python', 'and'), ('and', 'deep'), ('deep', 'learning'), ('learning', 'assignment'), ('assignment', '3'), ('3', '.'), ('.', 'This')]
-----
Calculating the word frequency of Bi-Gram:
Counter({'.', 'This': 3, ('This', 'is'): 2, ('assignment', '3'): 2, ('3', '.'): 2, ('machine', 'learning'): 2, ('of', 'python'): 2, ('python', 'programming'): 2, ('python', 'and'): 2, ('is', 'python'): 2})
-----
Finding the top 5 bigrams:
[({'.', 'This'}, 3), (('This', 'is'), 2), (('assignment', '3'), 2), (('3', '.'), 2), (('machine', 'learning'), 2)]
-----
Summarizing the text and finding out the lines with top five bigrams

This is python and deep learning assignment 3.

This is Vamsi Krishna Challa, Submitting the assignment 3.

We have gone through various types of machine learning techniques throughout this course.

This course enabled me to learn basics of python programming with machine learning orientation.

This had helped me brush up my skills in programming.

Process finished with exit code 0
```

Here in this program, we use various parts of nltk package to perform operations on the input file. For example, we use `nltk.tokenize` to perform tokenization of words. In the same way, perform lemmatization and then generate bigrams from the lemmatized words. Then we find out the most common top 5 bigrams from them and compare with every sentence of the output file and then print out the sentences that contain the top five bigrams.

Program 4:

Program on K-Means clustering algorithm:

```
4. from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import neighbors, datasets

# loading the default dataset IRIS.
k1 = datasets.load_iris()

# loading data into the variable
k2 = k1.data

# Loading target into the variable
labd1=k1.target

# Here we define and divide test data to 20% and training data at 80%
xtrainer, xttester, ytraining, ytesting = train_test_split(k2, labd1,
test_size=0.2, random_state=42)

# Here we are seen setting the n-neighbours to the perform KNN
knn = neighbors.KNeighborsClassifier(n_neighbors=1)
knn.fit(xtrainer, ytraining)

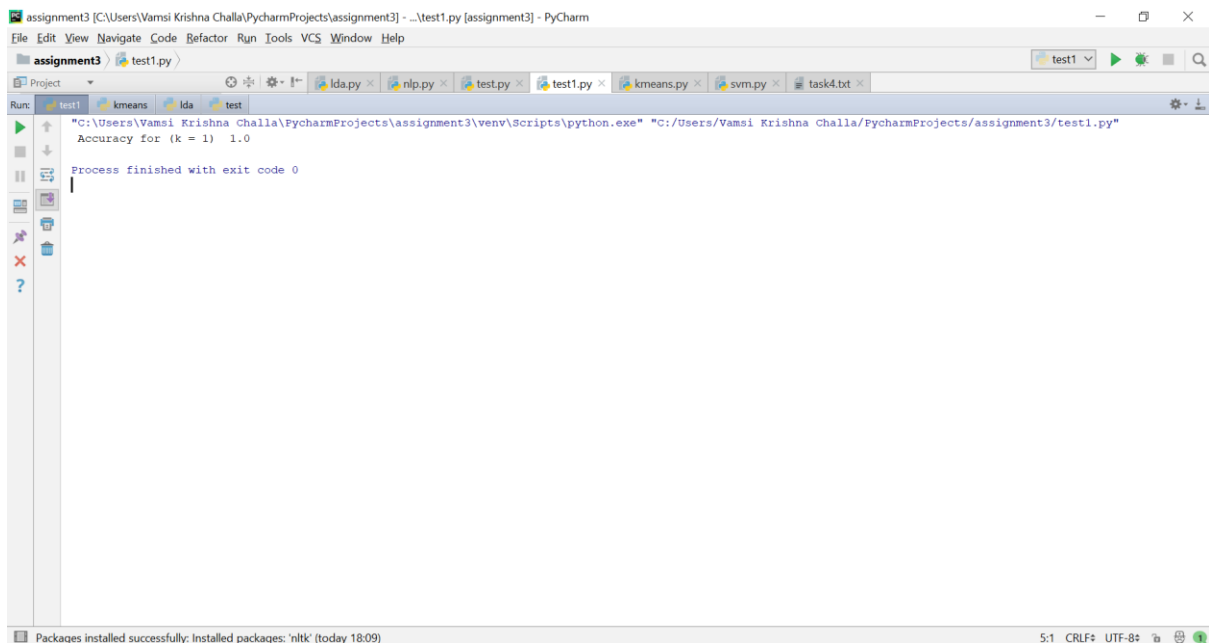
# Now let us perform prediction
yprediction=knn.predict(xttester)

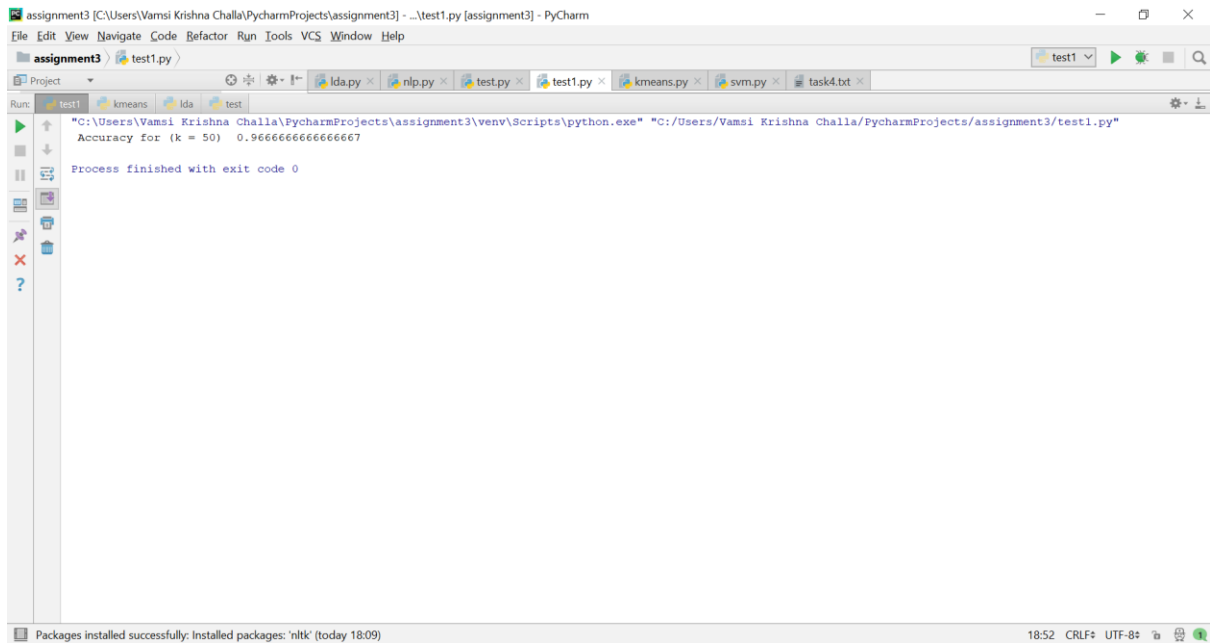
# now we perform accuracy calculation over knn algorithm
print(" Accuracy for (k = 1) ", accuracy_score(yprediction, ytesting))
```

5.

We run the above code again with new number of clusters with k = 50.

Result Summarization:





The image shows a PyCharm Run window for a project named 'assignment3'. The run configuration is 'test1.py'. The output shows the execution of a k-means clustering script. The accuracy for k=50 is 0.9666666666666667. The process finished with exit code 0. The status bar at the bottom indicates that packages were installed successfully, including 'nltk'.

```
"C:\Users\Vamsi Krishna Challa\PycharmProjects\assignment3\venv\Scripts\python.exe" "C:/Users/Vamsi Krishna Challa/PycharmProjects/assignment3/test1.py"
Accuracy for (k = 50) 0.9666666666666667
Process finished with exit code 0
```

Packages installed successfully: Installed packages: 'nltk' (today 18:09) 18:52 CRLF UTF-8

Name itself suggests what k-means is. Data at n points is clustered in k clusters, in which each of the data item belongs to the cluster with the nearest mean. Here we have considered different number of clusters to check on the accuracy. Firstly, we considered 1 cluster and later we considered 50 clusters and performed k-means clustering to verify the performance of accuracy.

Deployment & Limitations:

- In the first part of the assignment, we have written a program in Pycharm, performing linear discriminant analysis on the considered iris dataset. The major limitation here is that we are performing the analysis by considering a small data set relative to the real time datasets. When the analysis is performed on large data sets, the results might vary considering the performance on the data set that is considered now.
- In the second part, we have written program to perform Support vector classification in this case. I have taken the case as mentioned in the question i.e., 20%test data and 80% training data. SVM classification is performed on both linear kernel and non-linear kernel i.e., rbf kernel. For the data set considered here, linear kernel based svm classification outperforms over the non-linear rbf kernel based svm classification. The main limitation might be the considered data set. Here we see that we have considered iris dataset which has four features like sepal length, width etc. In this case, where the features are constrained to a certain number, linear kernel might have outperformed in accuracy for svm classification. But it may vary during other data sets with larger amounts of data.
- In the third part, we have written program to perform various operations using nltk package on the file taken as input. The input file consists of text data. From nltk package we import various functions to tokenize the data, lemmatize the tokens and then also to find bigrams from the lemmatized data and perform a counter operation to find the frequency of each bigram. After that we check for the top 5 bigrams and return the sentences that contain them. I personally don't see limitation on this program. One limitation might be the size of the text file. If the text file is of large size, say 20gb, it might create a computational challenge to the system due to its unnatural size of enormous data.
- In the fourth part, we are intended to perform K-means clustering on the considered dataset. We perform K-means clustering with different cluster sizes, say 1 and 50. And then note the various changes that occur when the number of clusters change. We find a great change in accuracy in both the cases of different number of clusters. Accuracy changes drastically here when the number of clusters change. But it doesn't apply to every data set. As this performance is confined to this dataset. This might be a limitation. When the dataset changes, we cannot guarantee the same variance in the accuracy when we consider such number of clusters.

References:

- scikitlearn.org/stable/modules/generated/sklearn.svm.SVC.html
- <https://scikitlearn.org/0.16/modules/generated/sklearn.lda.LDA.html>
- <http://scikitlearn.org/stable/modules/generated/sklearn.cluster.KMeans.html>
- <https://stackoverflow.com/questions/19145332/nltk-counting-frequency-of-bigram>