CS 5590 - Python and Deep Learning

Assignment 3

Submitted by

Lakshmi Mrudula chamala

16253401

**Objective:**

To Perform different tasks utilizing machine learning components.

Task 1: To perform direct discriminant examination on our preferred dataset

Task 2: To execute SVM order with straight and Rbf portion.

Task 3: To tokenize, lemmatize content from an info document and to discover top five bigrams from them and furthermore discover the sentences that contain the main 5 bigrams.

Task 4: Analysis over K-closest neighbours calculation.

**Features:**

In this task we get to both python inbuilt highlights and by introducing required bundles

1.Access scikit figure out how to import direct discriminant examination bundle to create the model

2.Access scikit learn bundle to fit svm classifier

3.Access nltk bundle to perform different tasks like tokenization, lemmatization, producing bigrams.

4.Access sklearn to perform K implies closest calculation analysis.

**Configuration:**

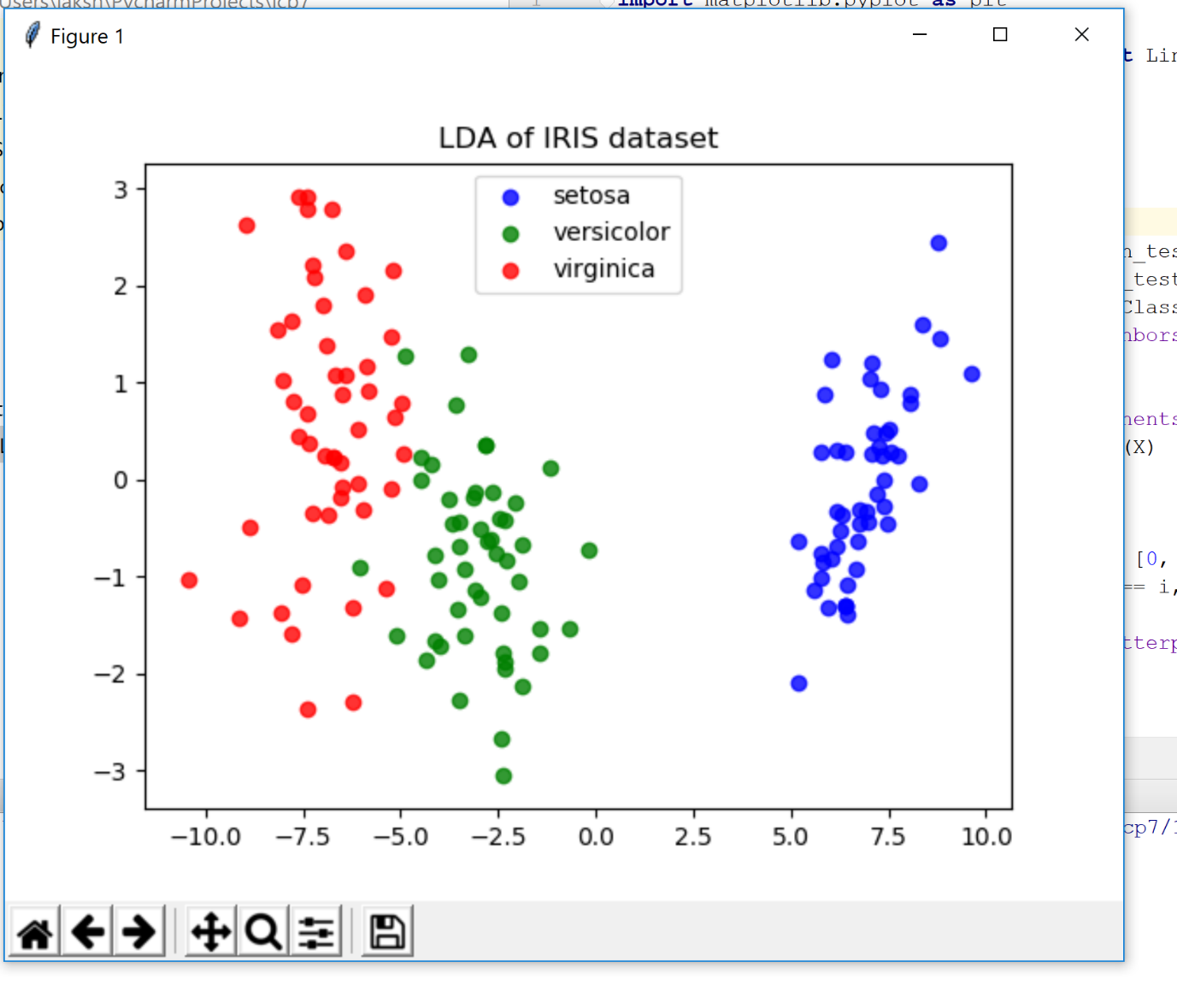
Software used: Python 3.4

IDE: PyCharm

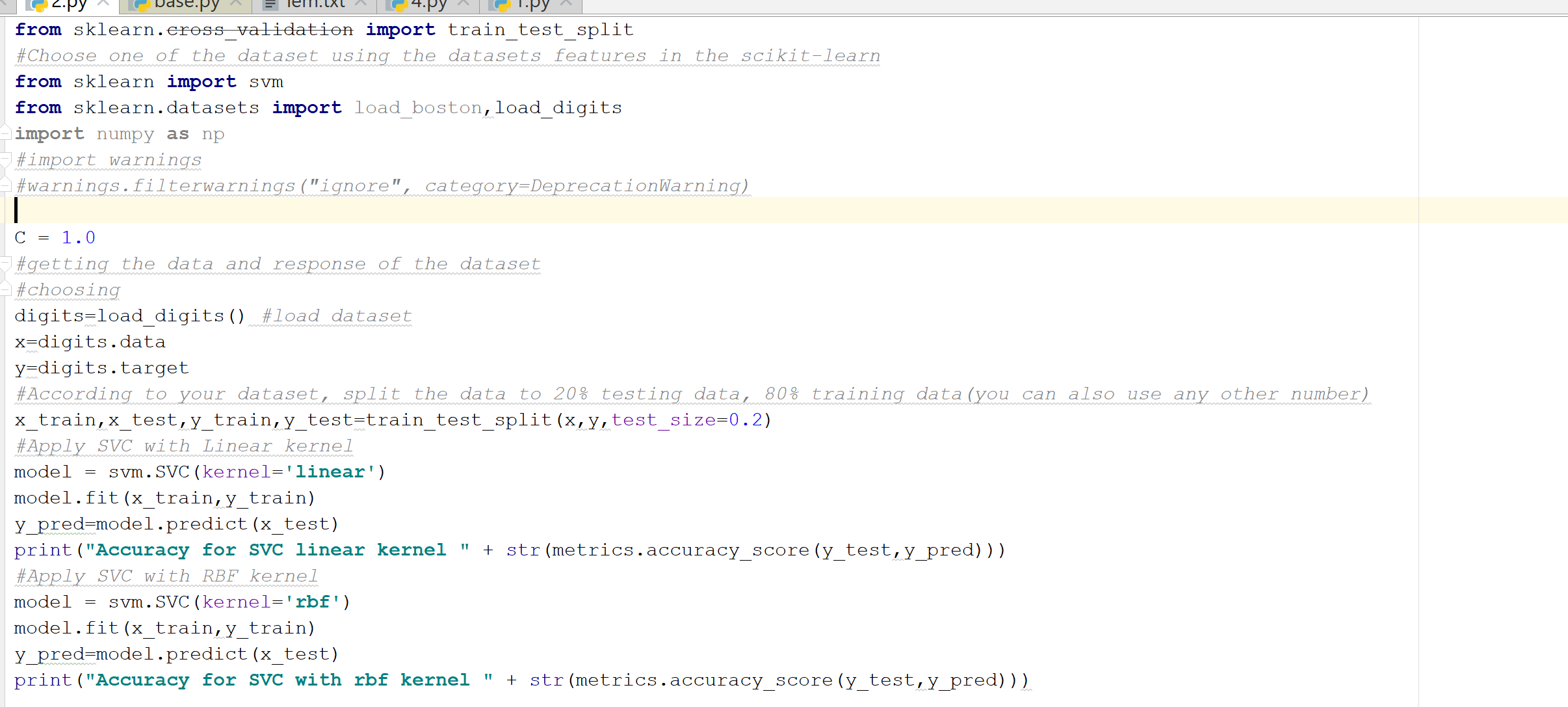
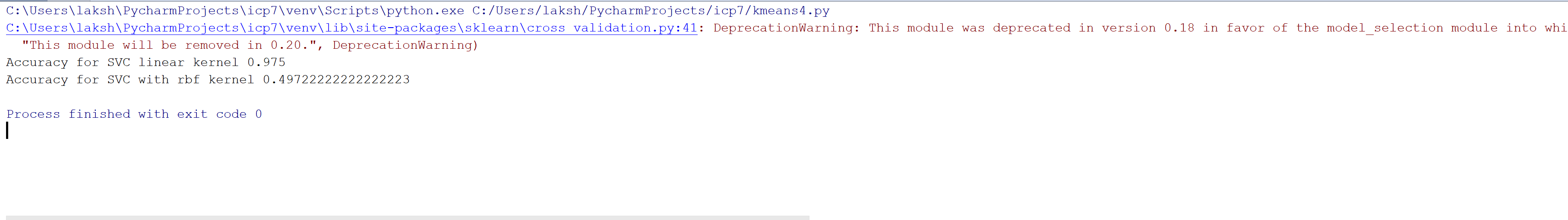
**Input / Output:**

**1.**





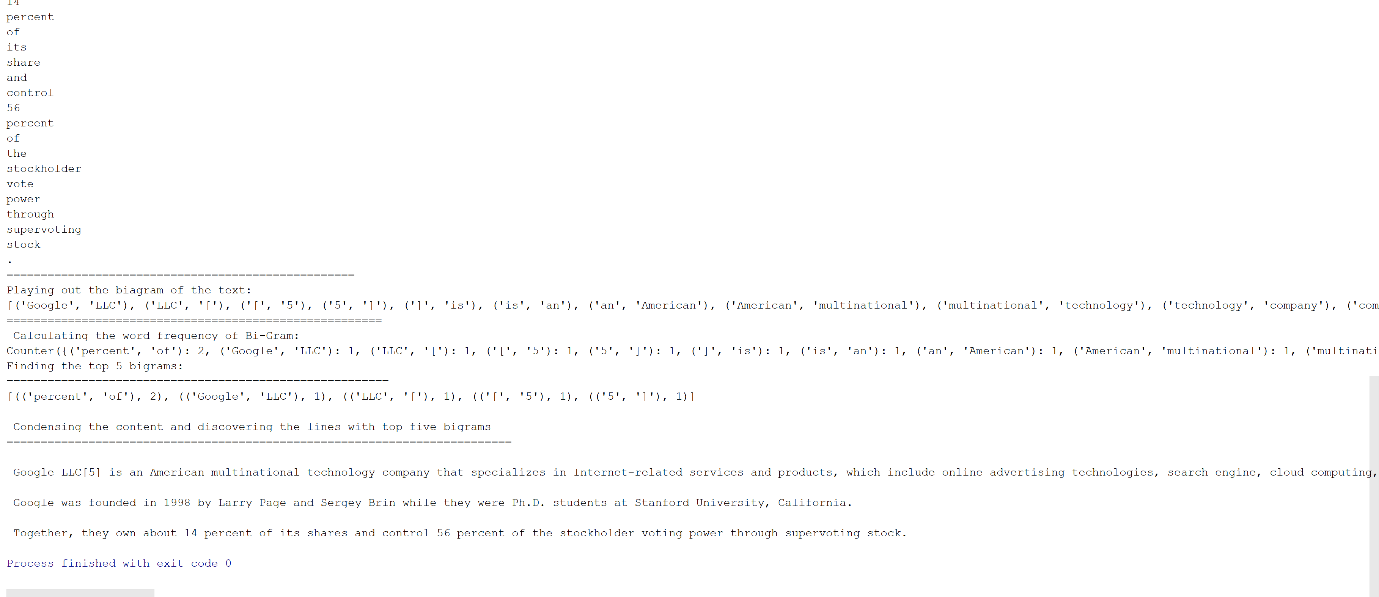
2.

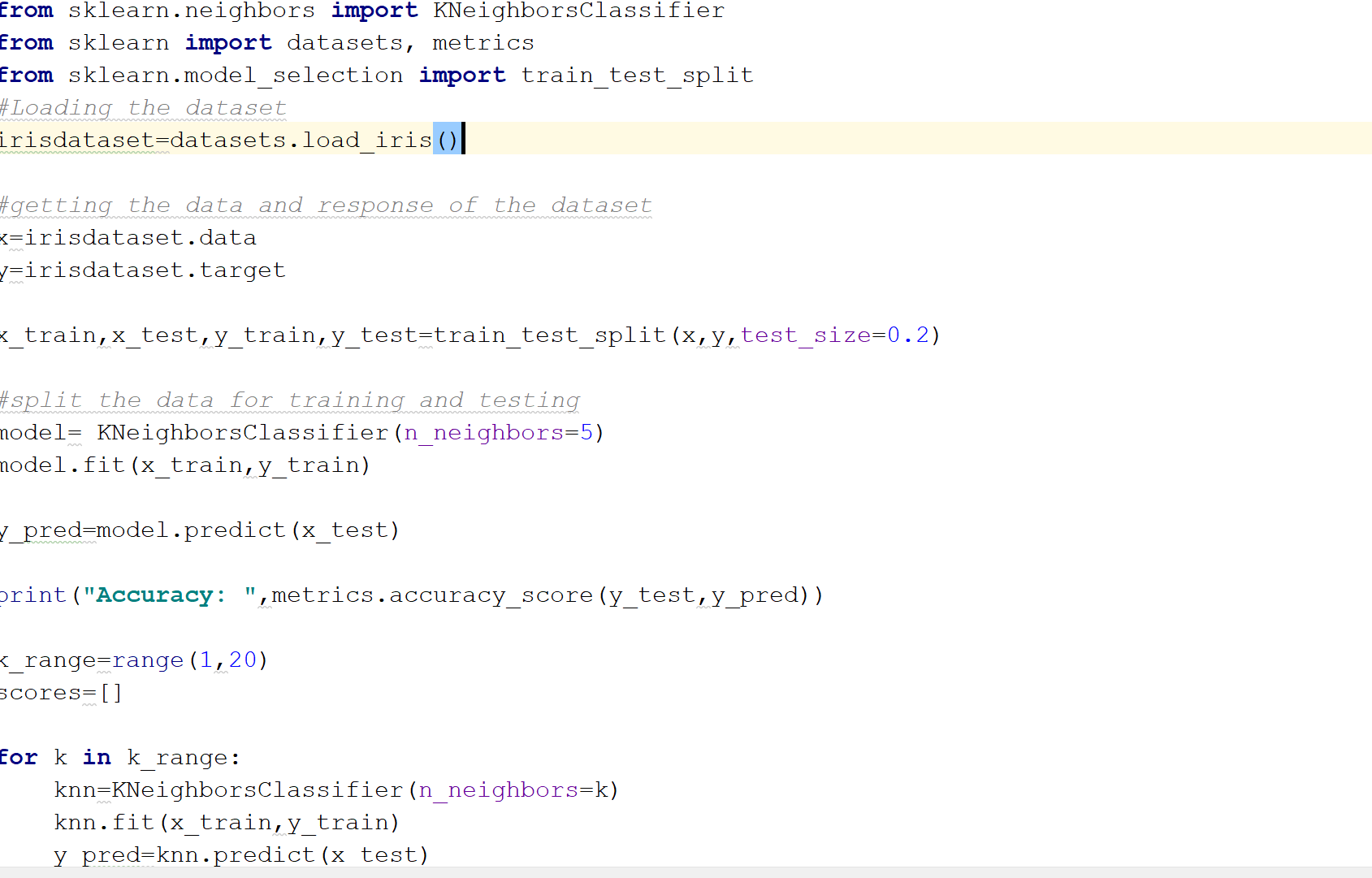
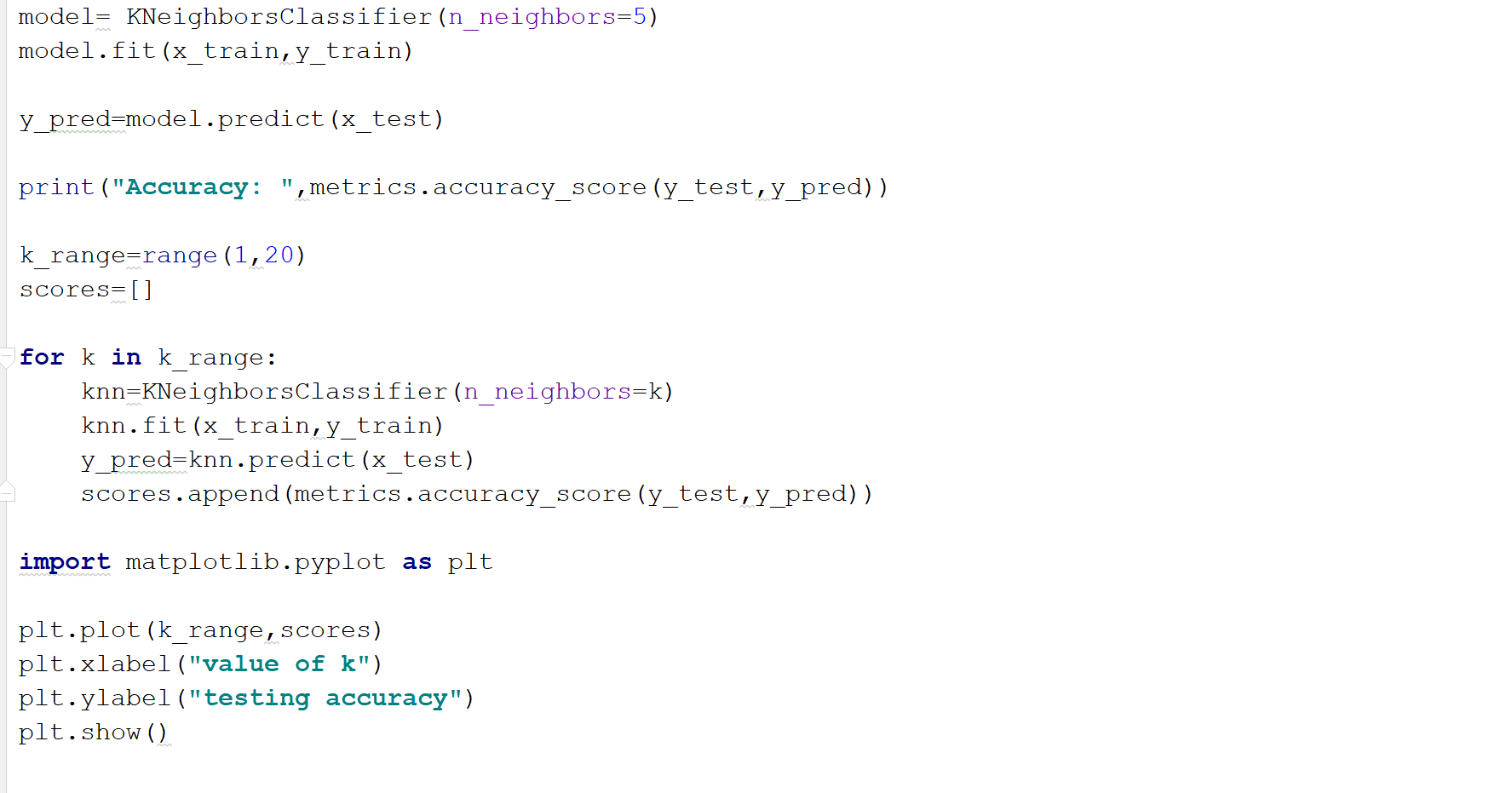
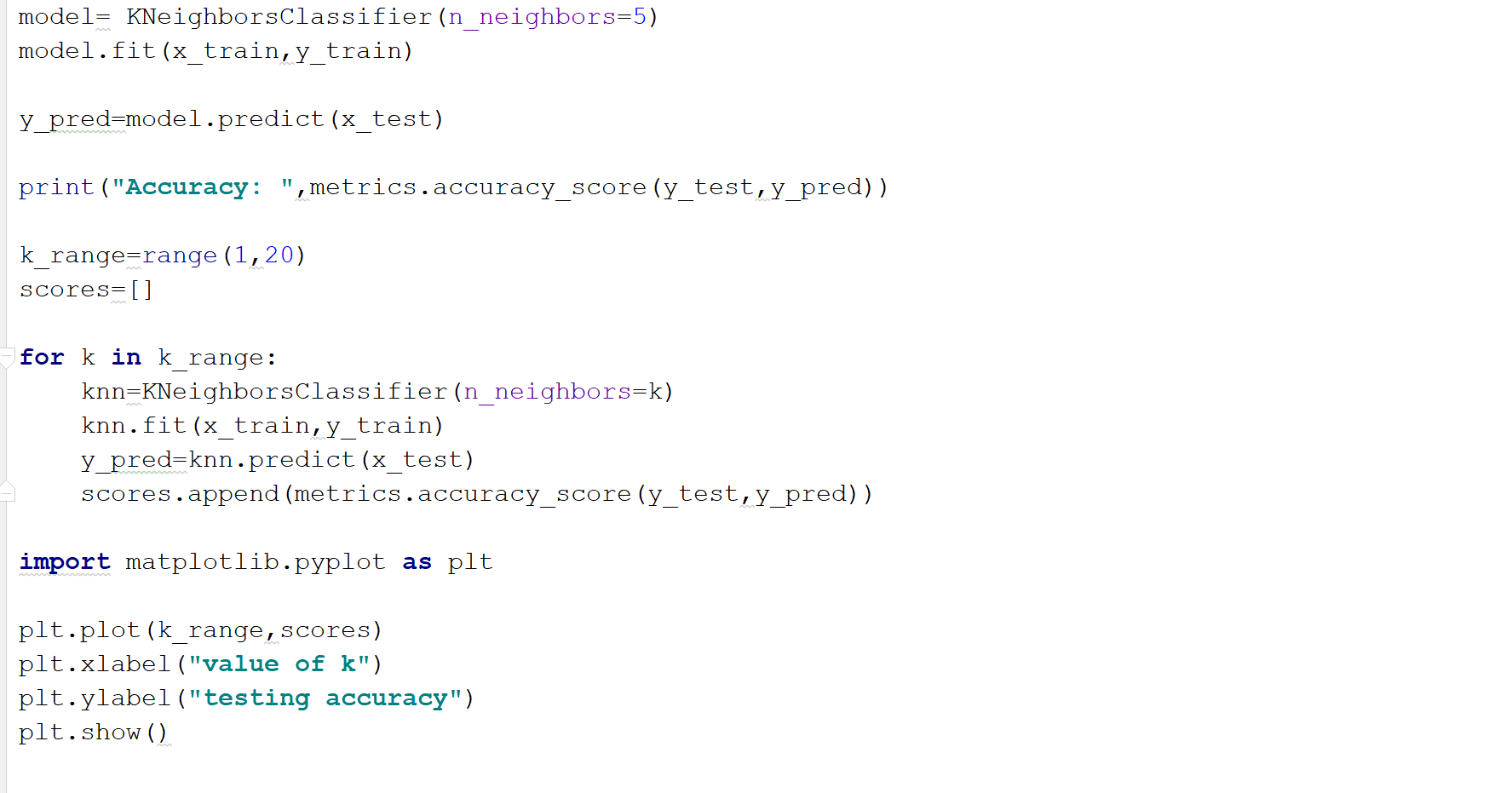
 

3. 







4.   

**Implementation:**

Program 1:

**import** matplotlib.pyplot **as** plt  
**from** sklearn **import** datasets  
**from** sklearn.discriminant\_analysis **import** LinearDiscriminantAnalysis  
  
iris = datasets.load\_iris()  
  
X = iris.data  
y = iris.target  
target\_names = iris.target\_names  
**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)  
**from** sklearn.neighbors **import** KNeighborsClassifier  
classifier = KNeighborsClassifier(n\_neighbors=5)  
classifier.fit(X\_train, y\_train)  
y\_pred = classifier.predict(X\_test)  
lda = LinearDiscriminantAnalysis(n\_components=2)  
X\_r2 = lda.fit(X\_test, y\_pred).transform(X)  
plt.figure()  
colors = [**'blue'**, **'green'**, **'red'**]  
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'**)  
plt.show()

Analysis over LDA and Logistic regression:

On account of both strategic relapse and LDA I.E., Linear discriminant investigation, two of them are from the linear\_model. Them two have likenesses, yet their genuine distinction tolls up when we play out the bit and the way we get to the parameters. Frequently, Logistic relapse is favored over LDA as it is especially strong in nature. In any case, for the situation, when we have every one of the necessities to perform grouping LDA performs better finished Logistic relapse.

2.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 information above, it recommends that straight portion is preferable appropriate over rbf bit to anticipate the information display for the iris informational index. However, we may not affirm the same for the other informational collections. By and large we realize that, when the highlights are increasingly, straight bit beats over rbf piece as there is no need of high dimensional space to foresee the highlights. In any case, this doesn't imply that it turns out to be valid in every one of the cases as an appropriately tuned rbf piece can never beat direct portion also.

3.

Here in this program, we utilize different parts of nltk bundle to perform tasks on the info record. For instance, we utilize nltk.tokenize to perform tokeinization of words. Similarly, perform lemmatization and afterward produce bigrams from the lemmatized words. At that point we discover the most well-known best 5 bigrams from them and contrast and each sentence of the yield record and afterward print out the sentences that contain the best five bigrams.

4. Name itself recommends what k-implies is. Information at n focuses is grouped in k bunches, in which every one of the information thing has a place with the group with the closest mean. Here we have thought about various number of bunches to keep an eye on the precision. Right off the bat, we considered 1 bunch and later we considered 50 groups and performed k-implies grouping to check the execution of precision.

**Deployment & Limitations:**

• In the initial segment of the task, we have composed a program in Pycharm, performing direct discriminant investigation on the considered iris dataset. The significant constraint here is that we are playing out the investigation by considering a little informational collection with respect to the ongoing datasets. At the point when the examination is performed on huge informational indexes, the outcomes may differ considering the execution on the informational collection that is thought about at this point.

• In the second part, we have composed program to perform Support vector characterization for this situation. I have taken the case as specified in the inquiry i.e., 20%test information and 80% preparing information. SVM arrangement is performed on both straight portion and non-direct bit i.e., rbf bit. For the informational index considered here, straight part based svm grouping beats over the non-direct rbf piece based svm arrangement. The fundamental confinement may be the considered informational index. Here we see that we have considered iris dataset which has four highlights like sepal length, width and so on. For this situation, where the highlights are obliged to a specific number, direct part may have beated in precision for svm characterization. Be that as it may, it might fluctuate amid other informational collections with bigger measures of information.

• In the third part, we have composed program to perform different activities utilizing nltk bundle on the record taken as information. The info document comprises of content information. From nltk bundle we import different capacities to tokenize the information, lemmatize the tokens and after that likewise to discover bigrams from the lemmatized information and play out a counter task to discover the recurrence of every bigram. After that we check for the main 5 bigrams and restore the sentences that contain them. I for one don't see confinement on this program. One restriction may be the measure of the content record. On the off chance that the content document is of huge size, say 20gb, it may make a computational test to the framework because of it unnatural size of gigantic information.

• In the fourth part, we are planned to perform K-implies bunching on the thought about dataset. We perform K-implies grouping with various bunch sizes, say 1 and 50. And after that note the different changes that happen when the quantity of bunches change. We locate an extraordinary change in precision in both the instances of various number of bunches. Precision changes radically here when the quantity of bunches change. In any case, it doesn't have any significant bearing to each datum set. As this execution is restricted to this dataset. This may be an impediment. At the point when the dataset transforms, we can't ensure a similar change in the exactness when we think about such number of bunches.

References:

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