CS5590/490-Python/DeepLearning

ASSIGNMENT-3

Mohammad Mohsin Ali

16246618

Class ID: 26

**Author: Mohammad Mohsin ali**

**Objective:** Get familiar with python

**Features:**

The objective is to implement the lab task of finding difference between logistic regression and linear regression, support vector machine classification, display lemmatization and k nearnest

**Configuration:**

Python 3.6.4

IDE: JetBrains PyCharm community Edition 2017.3.3

Numpy 1.14.0

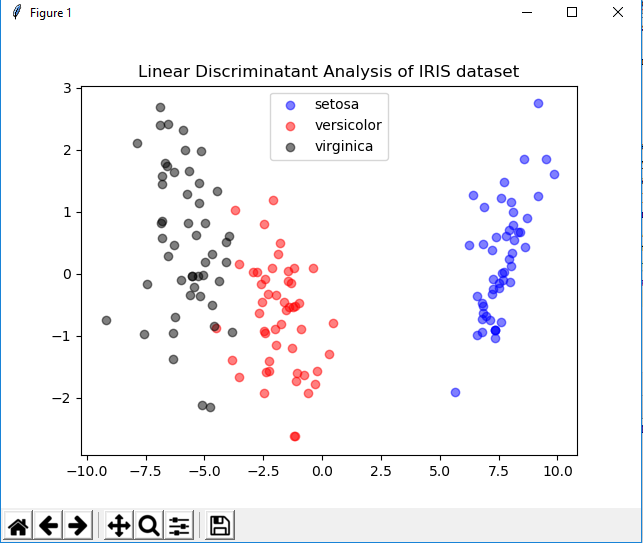
**Input/output :**

**1**. difference between logistic regression

**from** sklearn **import** datasets  
**import** matplotlib.pyplot **as** pt  
**from** sklearn.discriminant\_analysis **import** LinearDiscriminantAnalysis  
s = datasets.load\_iris()  
  
a = s.data  
b = s.target  
target\_names = s.target\_names  
da = LinearDiscriminantAnalysis(n\_components=2)  
X = da.fit(a, b).transform(a)  
c = [**'blue'**, **'red'**, **'black'**]  
**for** color, k, target\_name **in** zip(c, [0, 1, 2], target\_names):  
 pt.scatter(X[b == k, 0], X[b == k, 1], alpha=.5, color=color,  
 label=target\_name)  
pt.title(**'Linear Discriminatant Analysis of IRIS dataset'**)  
pt.legend(loc=**'best'**, shadow=**False**, scatterpoints=1)  
pt.show()

we get output as and explanation,

lda is mostly categorization when compared to logistic regression is used in uneven ratio’s

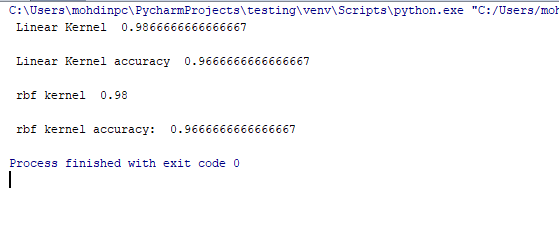


**2**. support vector machine classification

**from** sklearn **import** datasets,metrics   
**from** sklearn **import** svm   
**from** sklearn.model\_selection **import** train\_test\_split   
  
s=datasets.load\_iris()   
a=s.data   
b=s.target  
a\_train,a\_test,b\_train,b\_test=train\_test\_split(a,b,test\_size=0.20)   
model\_linear=svm.SVC(kernel=**'linear'**)   
b\_predict = model\_linear.fit(a\_train,b\_train).predict(a\_test)   
print(**' Linear Kernel '**,(model\_linear.score(a,b)))   
print(**'\n'**,**'Linear Kernel accuracy '**, metrics.accuracy\_score(b\_test,b\_predict))   
d=svm.SVC(kernel=**'rbf'**)   
b\_rbf = d.fit(a\_train,b\_train).predict(a\_test)   
print(**'\n'**,**'rbf kernel '**,(d.score(a,b)))   
print(**'\n'**,**'rbf kernel accuracy: '**, metrics.accuracy\_score(b\_test,b\_rbf))

the outputs and explanation is

rbf is better for non linear problems where as kinear kernel works best for data sets.rbf is used here

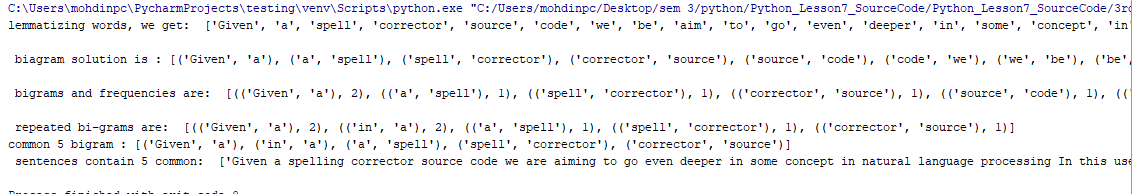


**3**.lemetizaion

**import** nltk  
**from** nltk.tokenize **import** sent\_tokenize,word\_tokenize  
**from** nltk.util **import** ngrams  
**from** collections **import** Counter  
**from** nltk.stem **import** WordNetLemmatizer  
  
f= (open(**'input.txt'**).read())  
z = sent\_tokenize(f)  
word\_token = []  
**for** a **in** z:  
 word\_token.append(word\_tokenize(f))  
  
l = []  
Lm=WordNetLemmatizer()  
**for** i **in** word\_token:  
 **for** x **in** i:  
 l.append(Lm.lemmatize(x,**'v'**))  
print(**'lemmatizing words, we get: '**, l)  
  
  
b = []  
biagram\_logic = ngrams(l,2)  
**for** j **in** biagram\_logic:  
 b.append(j)  
print(**'\n'**,**'biagram solution is :'**, b)  
  
count = nltk.FreqDist(b)  
freq= []  
**for** i, j **in** count.items():  
 freq.append((i,j))  
print(**'\n'**, **'bigrams and frequencies are: '**,freq)  
common5= []  
common5=count.most\_common(5)  
print(**'\n'**,**'repeated bi-grams are: '**,common5)  
text = []  
**for** i **in** common5:  
 text.append(i[0])  
print(**'common 5 bigram :'**,text)  
list=[]  
**for** i **in** f.splitlines():  
 **for** m **in** range(5):  
 **if** text[m][0] **in** i.split() **and** text[m][1] **in** i.split():  
 list.append(i)  
  
print(**' sentences contain 5 common: '**,list)

the outputs and explanation is,

we are implementing lemmatization and then the performing various tasks required



**4**.views on k nearest algorithm and explanation.

When k is bigger it is computationally expensive, and middle may suffer offer fitting and when k is smaller it may be underfitting. We have make sure k is bigger to avoid cv error and smaller to reduce computational complexity.

**Deployment:**

1. Save the file in your computer
2. Install Python 3.6.4 and PyCharm IDE on your machine.
3. click open-files -location of the saved folder on pycharm.
4. Make sure the code running has .py extension.
5. Right click on the code screen and then click on run .
6. Enter input to see output.

**limitation**

1.Task 2 runs for only certain datasets, and doesn’t run for boston etc .

**References**

* <https://stackoverflow.com/>
* <https://www.tutorialspoint.com/python3/python_classes_objects.htm>
* <https://stats.stackexchange.com/questions/126051/choosing-optimal-k-for-knn>