LAB ASSIGNMENT-2

KARTHIK GUNDALAPALLI

SANTOSH GANDHI DIDDI

MOURYA PRAHARSHA BOBBILI

SAI TEJA PENUGONDA

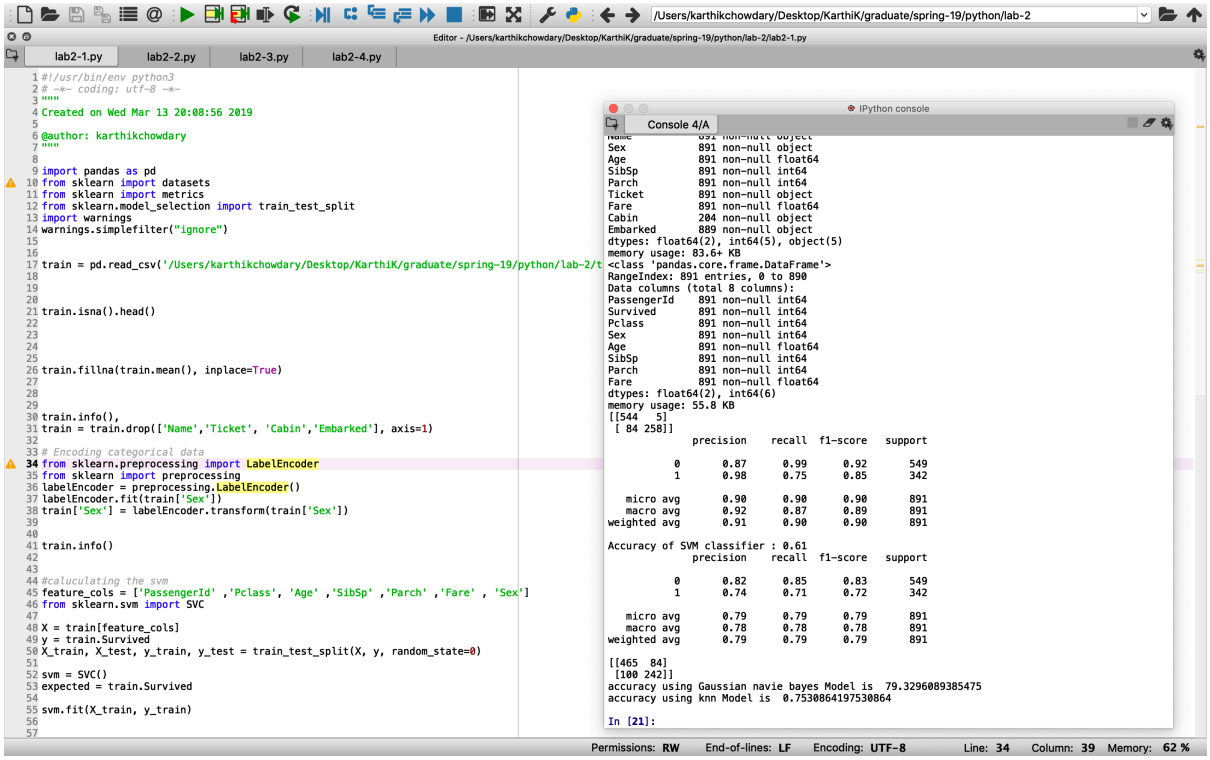
(1Q) Pick any dataset from the dataset sheet in the class sheet or online which includes both numeric and non-numeric features

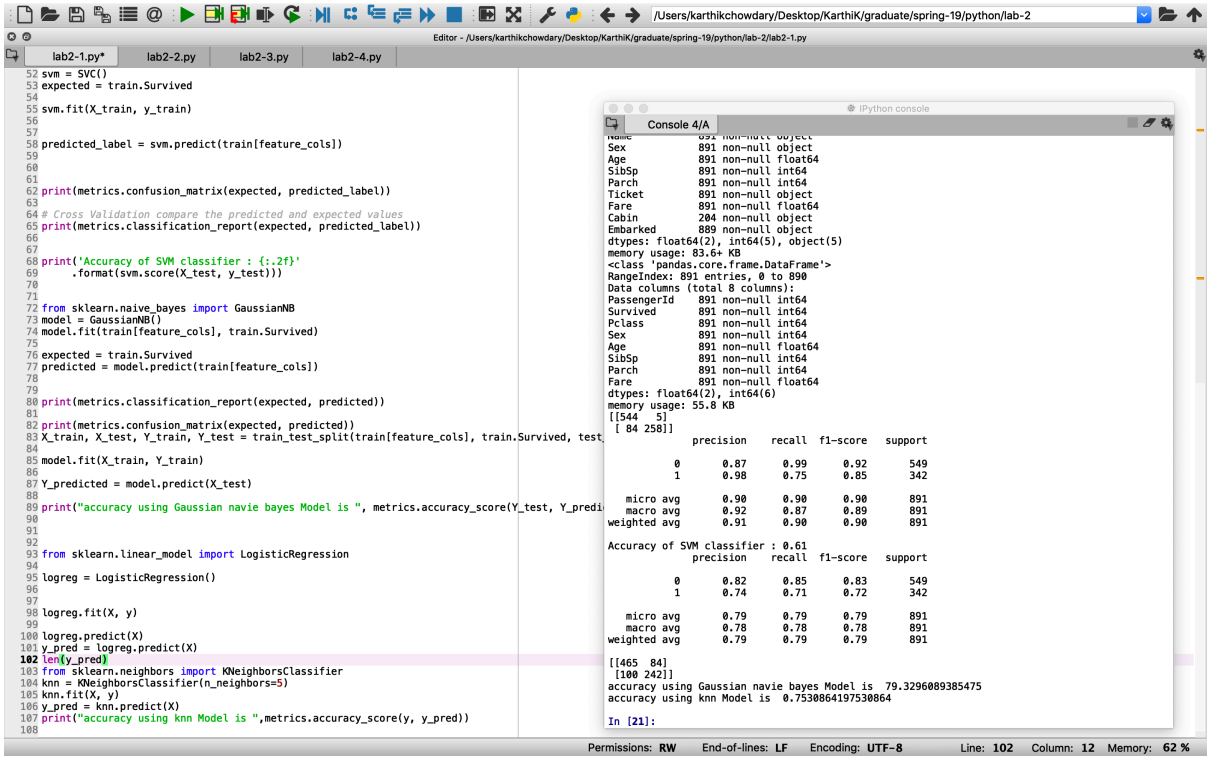
a. Perform exploratory data analysis on the data set (like Handling null values, removing the features not correlated to the target class, encoding the categorical features, ...)

b. Apply the three classification algorithms Naïve Baye's, SVM and KNN on the chosen data set and report which classifier gives better result.

Approach: we have applied Naive bayes ,SVM and KNN on the train dataset in .csv format and the results are displayed below:

Code with output:

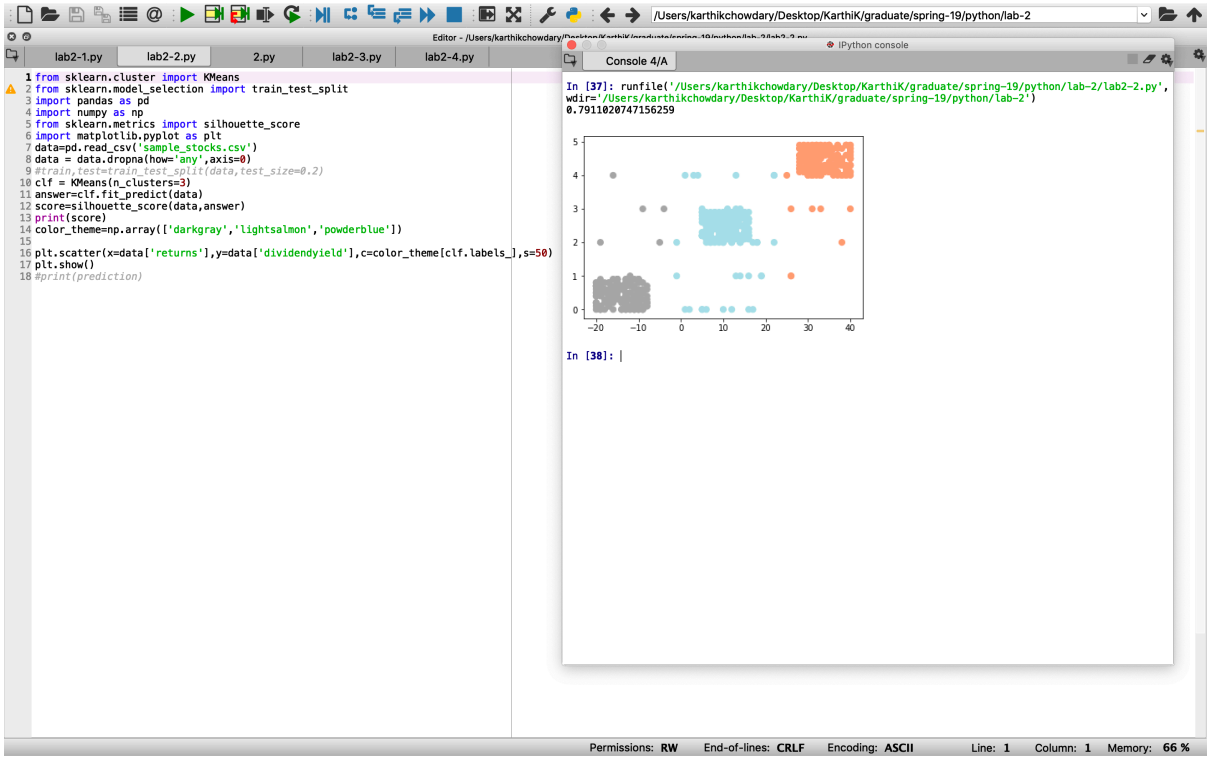


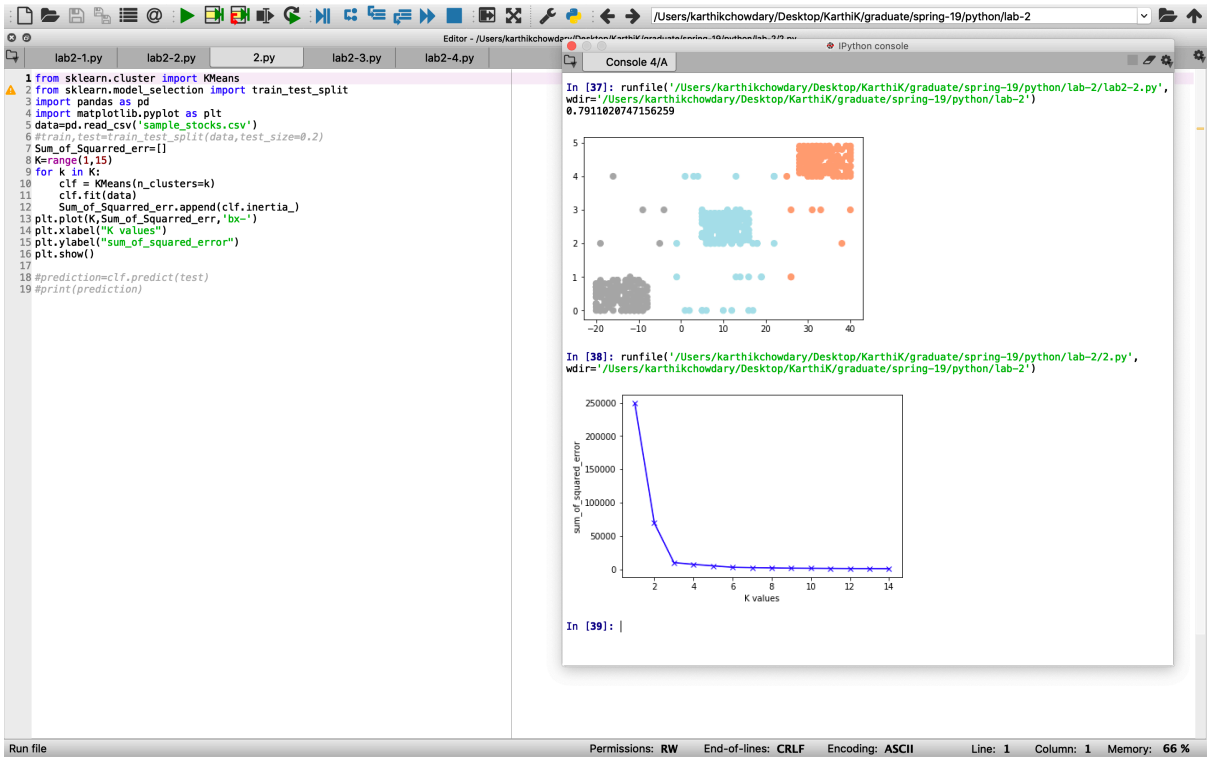

/Users/karthikchowdary/Desktop/KarthiK/graduate/spring-19/python/lab-2
Editor - /Users/karthikchowdary/Desktop/KarthiK/graduate/spring-19/python/lab-2/lab2-1.py
lab2-1.py*
lab2-2.py
lab2-3.py
lab2-4.py
svm = SVC()
expected = train.Survived
svm.fit(X_train, y_train)
predicted_label = svm.predict(train[feature_cols])
print(metrics.confusion_matrix(expected, predicted_label))
# Cross Validation compare the predicted and expected values
print(metrics.classification_report(expected, predicted_label))
print('Accuracy of SVM classifier : {:.2f}'
.format(svm.score(X_test, y_test)))
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(train[feature_cols], train.Survived)
expected = train.Survived
predicted = model.predict(train[feature_cols])
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
X_train, X_test, Y_train, Y_test = train_test_split(train[feature_cols], train.Survived, test
model.fit(X_train, Y_train)
Y_predicted = model.predict(X_test)
print("accuracy using Gaussian navie bayes Model is ", metrics.accuracy_score(Y_test, Y_predi
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X, y)
logreg.predict(X)
y_pred = logreg.predict(X)
len(y_pred)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X, y)
y_pred = knn.predict(X)
print("accuracy using knn Model is ",metrics.accuracy_score(y, y_pred))
IPython console
Console 4/A
Sex 891 non-null object
Age 891 non-null float64
SibSp 891 non-null int64
Parch 891 non-null int64
Ticket 891 non-null object
Fare 891 non-null float64
Cabin 204 non-null object
Embarked 889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
PassengerId 891 non-null int64
Survived 891 non-null int64
Pclass 891 non-null int64
Sex 891 non-null int64
Age 891 non-null float64
SibSp 891 non-null int64
Parch 891 non-null int64
Fare 891 non-null float64
dtypes: float64(2), int64(6)
memory usage: 55.8 KB
[[544 5]
[84 258]]
precision recall f1-score support
0 0.87 0.99 0.92 549
1 0.98 0.75 0.85 342
micro avg 0.90 0.90 0.90 891
macro avg 0.92 0.87 0.89 891
weighted avg 0.91 0.90 0.90 891
Accuracy of SVM classifier : 0.61
precision recall f1-score support
0 0.82 0.85 0.83 549
1 0.74 0.71 0.72 342
micro avg 0.79 0.79 0.79 891
macro avg 0.78 0.78 0.78 891
weighted avg 0.79 0.79 0.79 891
[[465 84]
[100 242]]
accuracy using Gaussian navie bayes Model is 79.3296089385475
accuracy using knn Model is 0.7530864197530864
In [21]:
Permissions: RW
End-of-lines: LF
Encoding: UTF-8
Line: 102
Column: 12
Memory: 62 %

(2Q) Choose any dataset of your choice. Apply K-means on the dataset and visualize the clusters using matplotlib or seaborn.
a. Report which K is the best using the elbow method.
b. Evaluate with silhouette score or other scores relevant for unsupervised approaches (before applying clustering clean the data set with the EDA learned in the class)

Approach: we applied k-means on the stock-data and the output is vizualized

Code and output:

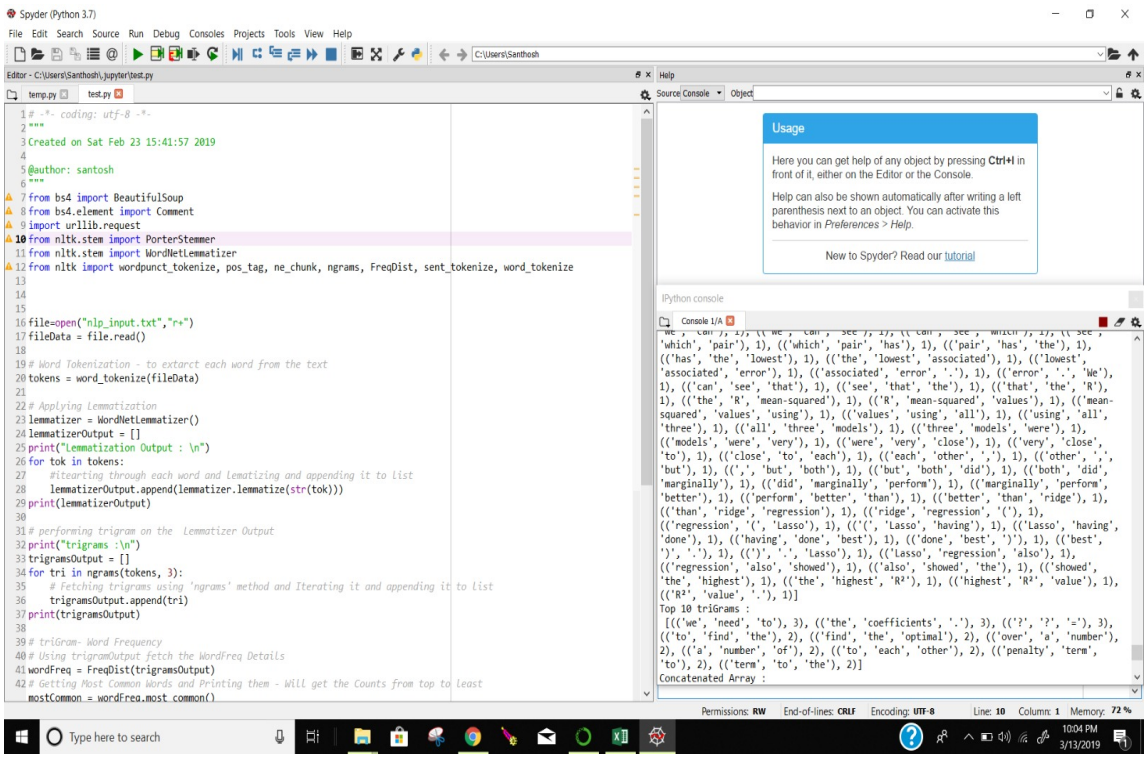


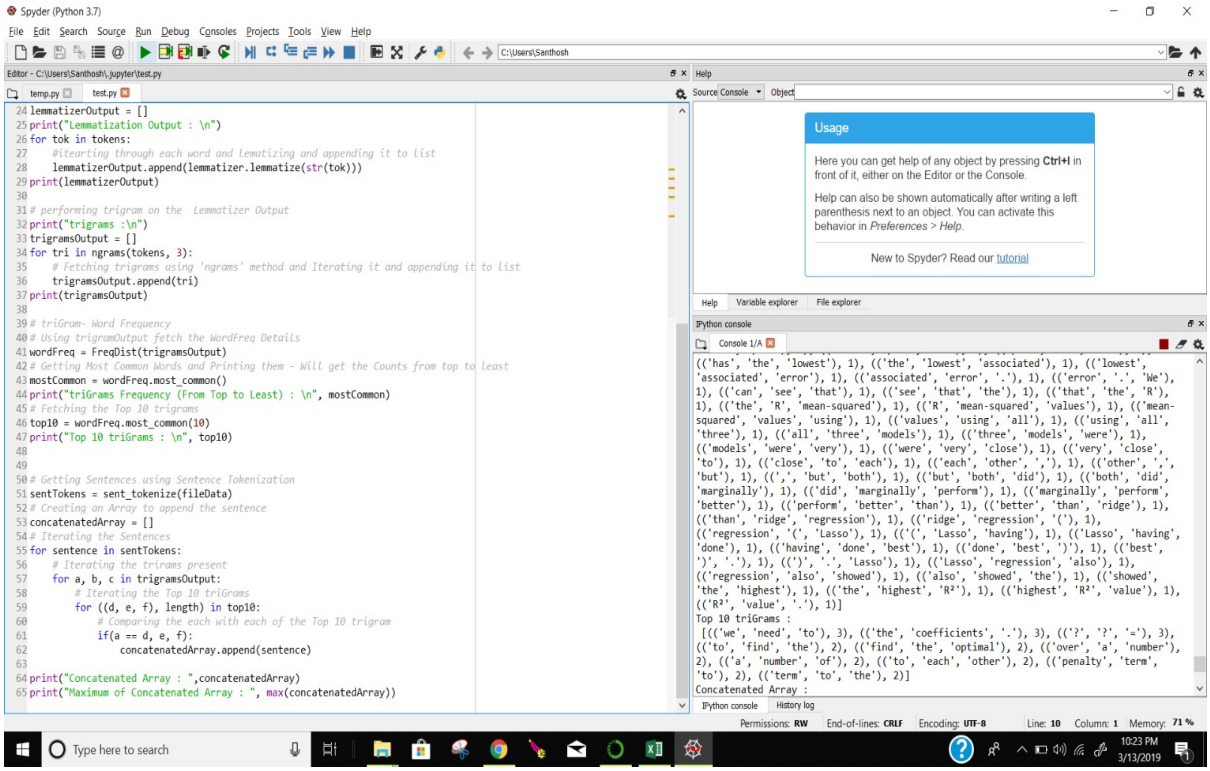

/Users/karthikchowdary/Desktop/KarthiK/graduate/spring-19/python/lab-2
lab2-1.py
lab2-2.py
2.py
lab2-3.py
lab2-4.py
1 from sklearn.cluster import KMeans
2 from sklearn.model_selection import train_test_split
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 data=pd.read_csv('sample_stocks.csv')
6 #train,test=train_test_split(data,test_size=0.2)
7 Sum_of_Squarred_err=[]
8 K=range(1,15)
9 for k in K:
10 clf = KMeans(n_clusters=k)
11 clf.fit(data)
12 Sum_of_Squarred_err.append(clf.inertia_)
13 plt.plot(K,Sum_of_Squarred_err,'bx-')
14 plt.xlabel("K values")
15 plt.ylabel("sum_of_squared_error")
16 plt.show()
17
18 #prediction=clf.predict(test)
19 #print(prediction)
IPython console
Console 4/A
In [37]: runfile('/Users/karthikchowdary/Desktop/KarthiK/graduate/spring-19/python/lab-2/lab2-2.py',
wdir='/Users/karthikchowdary/Desktop/KarthiK/graduate/spring-19/python/lab-2')
0.7911020747156259
In [38]: runfile('/Users/karthikchowdary/Desktop/KarthiK/graduate/spring-19/python/lab-2/2.py',
wdir='/Users/karthikchowdary/Desktop/KarthiK/graduate/spring-19/python/lab-2')
sum_of_squared_error
K values
In [39]:
Run file
Permissions: RW
End-of-lines: CRLF
Encoding: ASCII
Line: 1
Column: 1
Memory: 66 %

(3) Write a program in which take an Input file, use the simple approach below to summarize
a text file:Link to input file: https://umkc.box.com/s/7by0f4540cdbdp3pm60h5fxxffefsvrwa.
Read the data from a file
b. Tokenize the text into words and apply lemmatization technique on each word.
c. Find all the trigrams for the words.
d. Extract the top 10 of the most repeated trigrams based on their count.
e. Go through the text in the file
f. Find all the sentences with the most repeated tri-grams
g. Extract those sentences and concatenate
h. Print the concatenated result

Approach: All the techniques are executed in the program and can be seen in following screen shots

Code and Output:




Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Santhosh
Editor - C:\Users\Santhosh\.jupyter\test.py
temp.py
test.py
lemmatizerOutput = []
print("Lemmatization Output : \n")
for tok in tokens:
#itearting through each word and lematizing and appending it to list
lemmatizerOutput.append(lemmatizer.lemmatize(str(tok)))
print(lemmatizerOutput)
# performing trigram on the Lemmatizer Output
print("trigrams :\n")
trigramsOutput = []
for tri in ngrams(tokens, 3):
# Fetching trigrams using 'ngrams' method and Iterating it and appending it to list
trigramsOutput.append(tri)
print(trigramsOutput)
# triGram- Word Frequency
# Using trigramOutput fetch the WordFreq Details
wordFreq = FreqDist(trigramsOutput)
# Getting Most Common Words and Printing them - Will get the Counts from top to least
mostCommon = wordFreq.most_common()
print("triGrams Frequency (From Top to Least) : \n", mostCommon)
# Fetching the Top 10 trigrams
top10 = wordFreq.most_common(10)
print("Top 10 triGrams : \n", top10)
# Getting Sentences using Sentence Tokenization
sentTokens = sent_tokenize(fileData)
# Creating an Array to append the sentence
concatenatedArray = []
# Iterating the Sentences
for sentence in sentTokens:
# Iterating the trirams present
for a, b, c in trigramsOutput:
# Iterating the Top 10 triGrams
for ((d, e, f), length) in top10:
# Comparing the each with each of the Top 10 trigram
if(a == d, e, f):
concatenatedArray.append(sentence)
print("Concatenated Array : ",concatenatedArray)
print("Maximum of Concatenated Array : ", max(concatenatedArray))
Help
Source Console
Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Help
Variable explorer
File explorer
IPython console
Console 1/A
Top 10 triGrams :
Concatenated Array :
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: UTF-8
Line: 10
Column: 1
Memory: 71 %
Type here to search
10:23 PM
3/13/2019

(4) Create Multiple Regression by choosing a dataset of your choice (again before evaluating, clean the data set with the EDA learned in the class). Evaluate the model using RMSE and R2 and also report if you saw any improvement before and after the EDA.

Approach: We applied multiple regression on the diabetes data set and also evaluated using RMSE and R2 and the results can be seen as follows.

Code and Output: