PYTHON LAB ASSIGNMENT – 3

Objective:

The main objective of this lab assignment is to get familiar with Machine Learning Algorithms and scikit in python.

Features:

1. Linear Discrimination Analysis

2. SVM Application

3. Lemmatization and Bi-grams

4. KNN

Configuration:

Python 3.6 interpreter

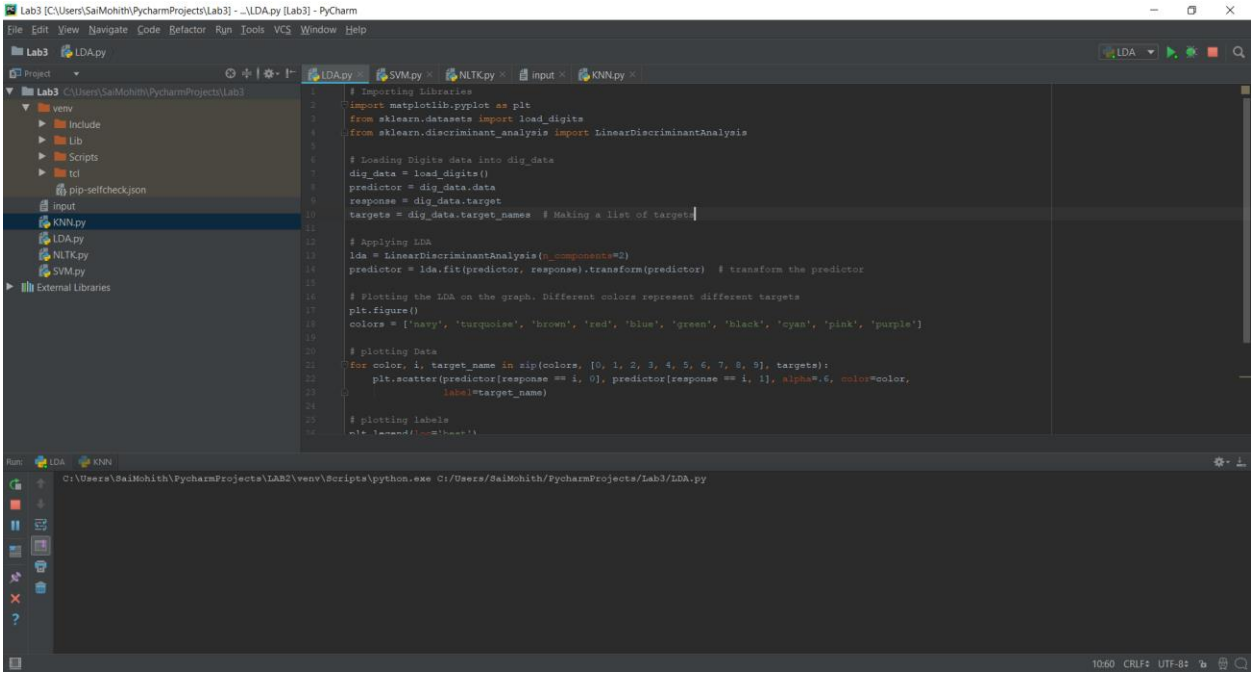
Sklearn Library

Matplotlib.pyplot Library

JetBrains Pycharm Community Edition

Screenshots:

1. **LDA:**


Lab3 [C:\Users\SaiMohith\PycharmProjects\Lab3] - ...\LDA.py [Lab3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Lab3 LDA.py
LDA
Project
LDA.py SVM.py NLTK.py input KNN.py
Lab3 C:\Users\SaiMohith\PycharmProjects\Lab3
venv
Include
Lib
Scripts
tcl
pip-selfcheck.json
input
KNN.py
LDA.py
NLTK.py
SVM.py
External Libraries
# Importing Libraries
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
# Loading Digits data into dig_data
dig_data = load_digits()
predictor = dig_data.data
response = dig_data.target
targets = dig_data.target_names # Making a list of targets
# Applying LDA
lda = LinearDiscriminantAnalysis(n_components=2)
predictor = lda.fit(predictor, response).transform(predictor) # transform the predictor
# Plotting the LDA on the graph. Different colors represent different targets
plt.figure()
colors = ['navy', 'turquoise', 'brown', 'red', 'blue', 'green', 'black', 'cyan', 'pink', 'purple']
# plotting Data
for color, i, target_name in zip(colors, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], targets):
plt.scatter(predictor[response == i, 0], predictor[response == i, 1], alpha=.6, color=color,
label=target_name)
# plotting labels
Run: LDA KNN
C:\Users\SaiMohith\PycharmProjects\LAB2\venv\Scripts\python.exe C:/Users/SaiMohith/PycharmProjects/Lab3/LDA.py
10:60 CRLF UTF-8

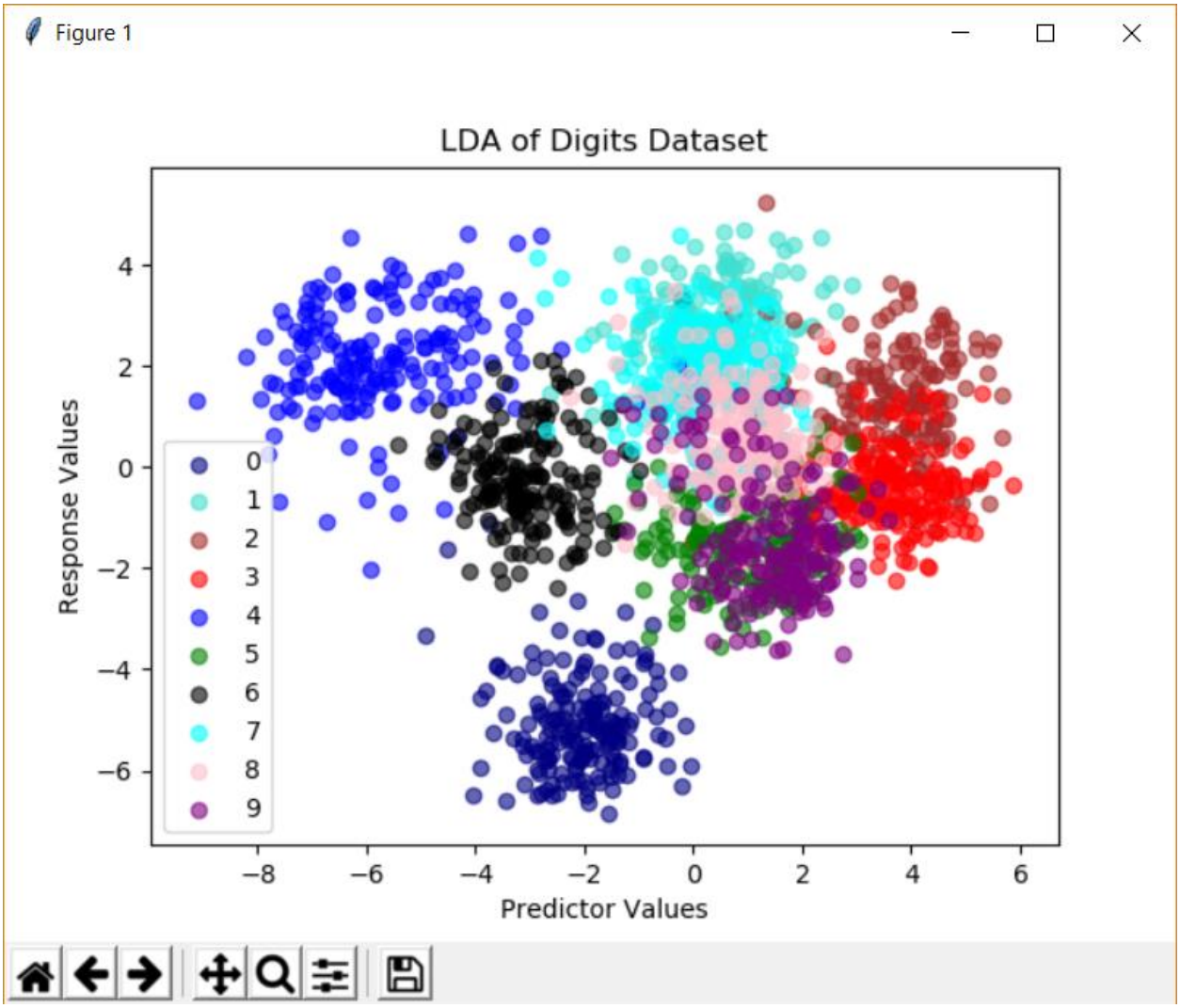
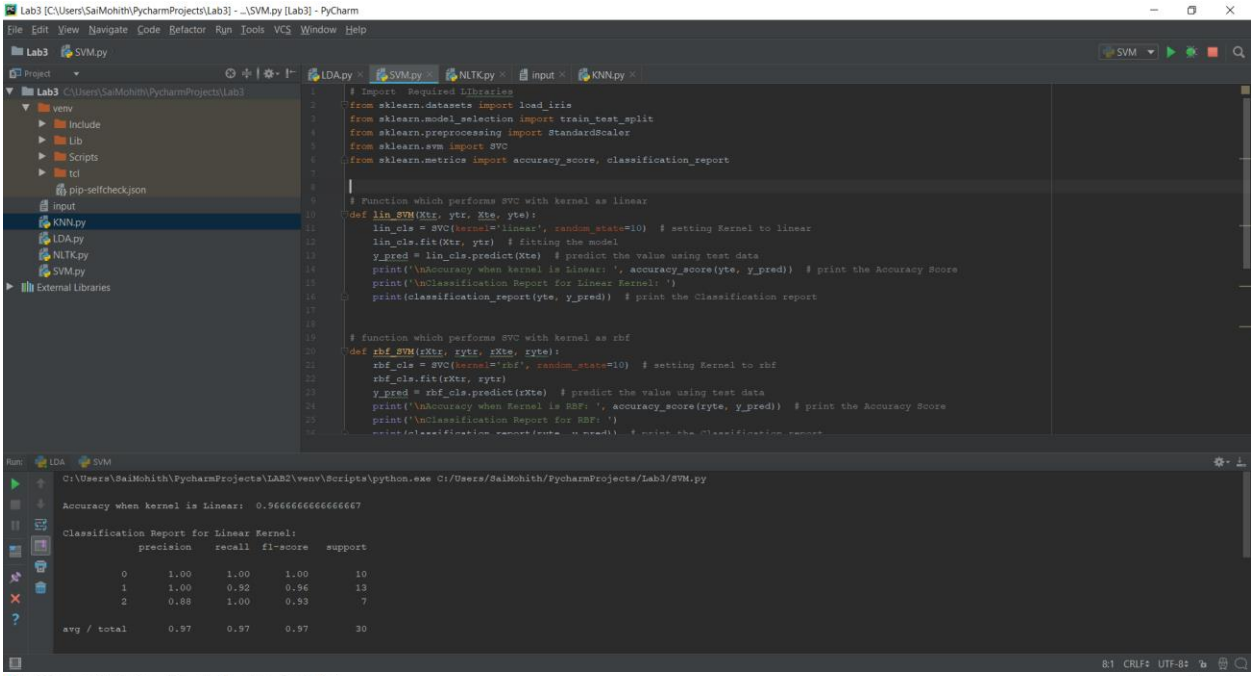
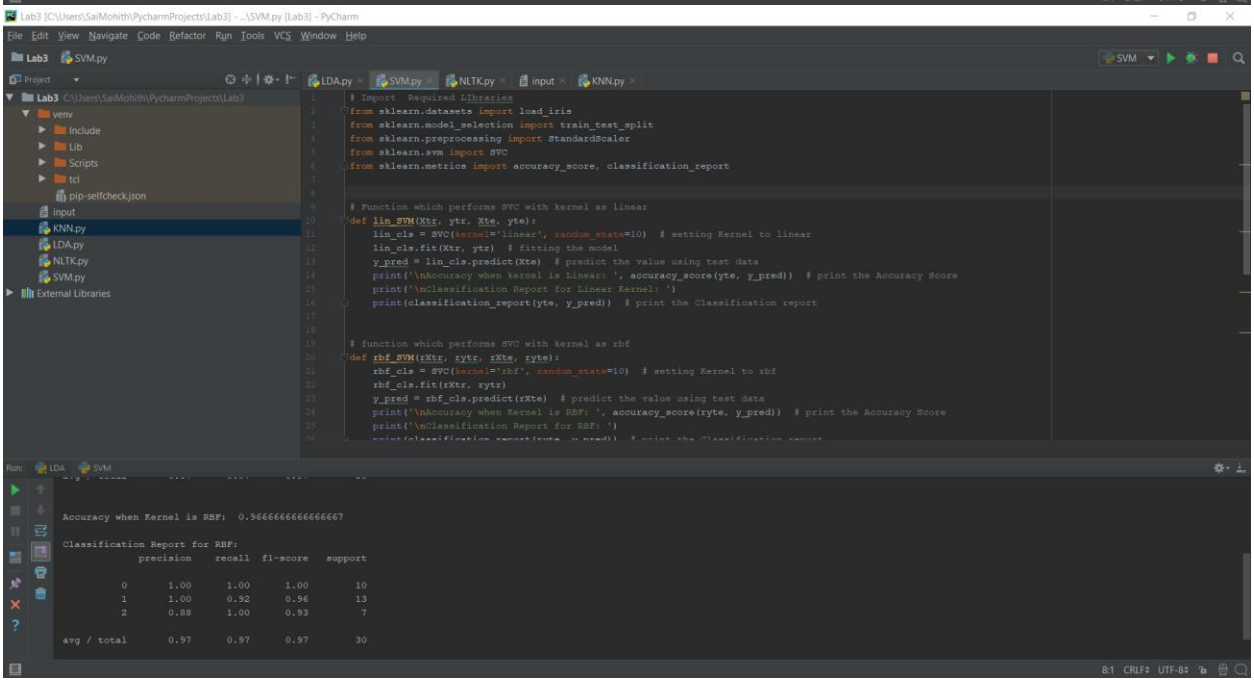


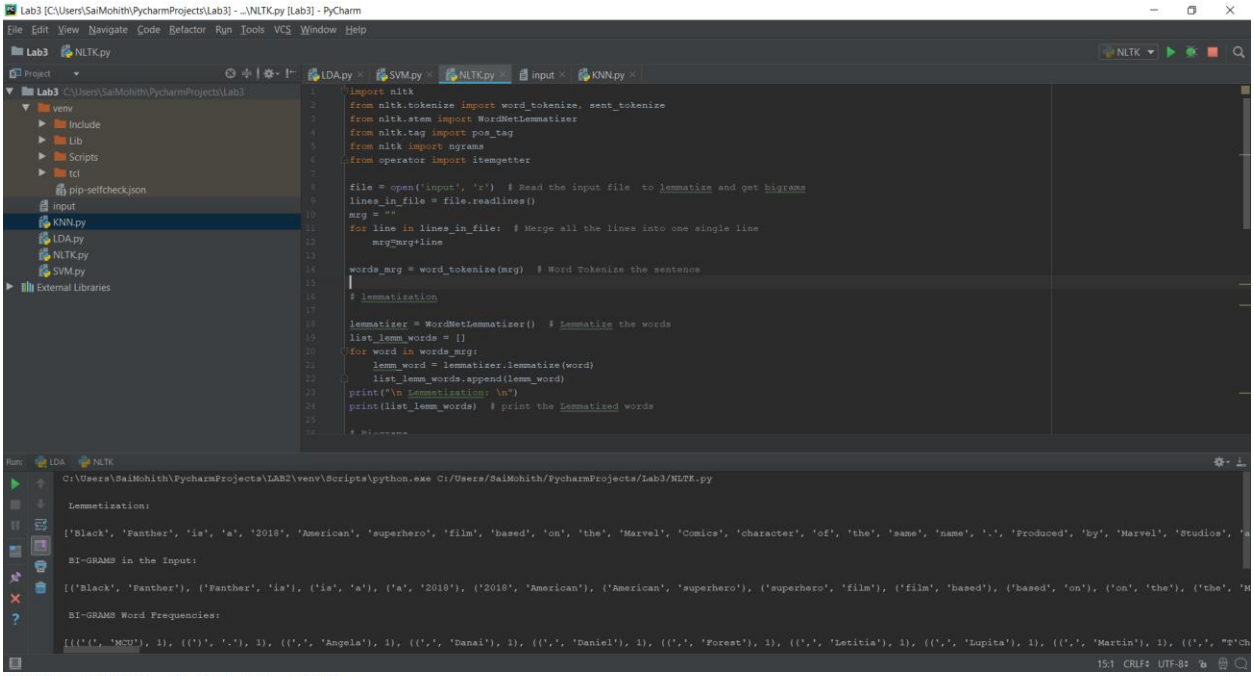
Figure 1
LDA of Digits Dataset
Response Values
Predictor Values
0
1
2
3
4
5
6
7
8
9
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6

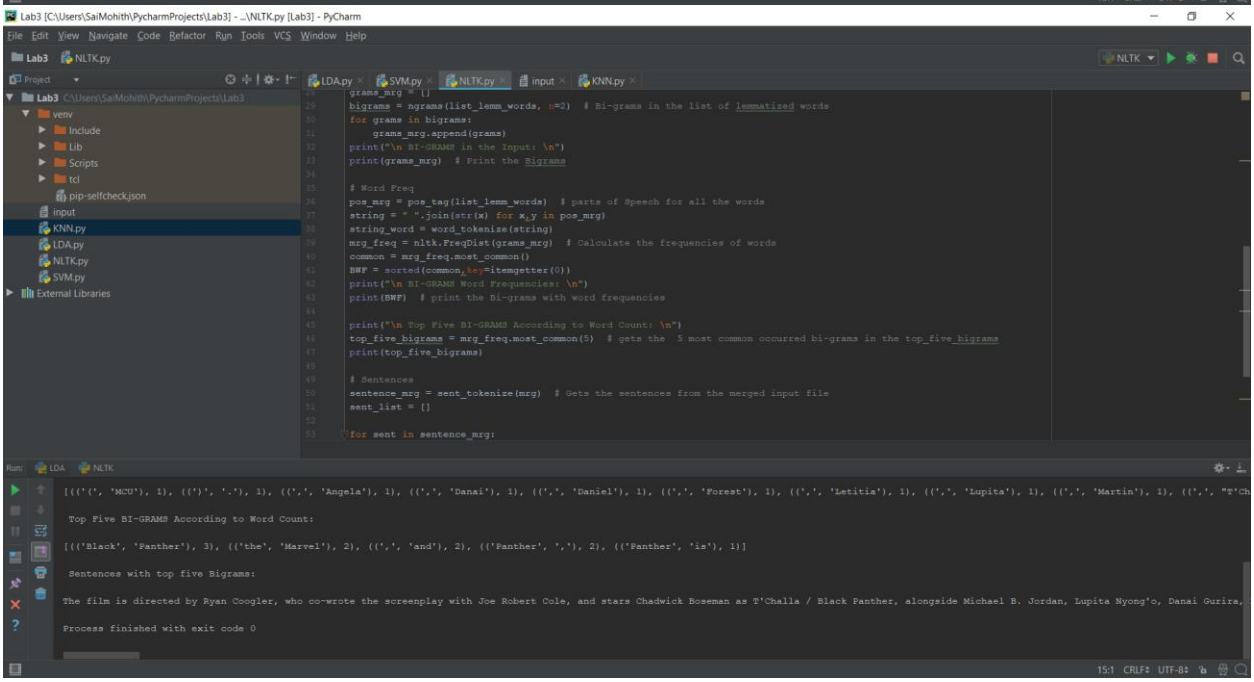
2. **SVM:**



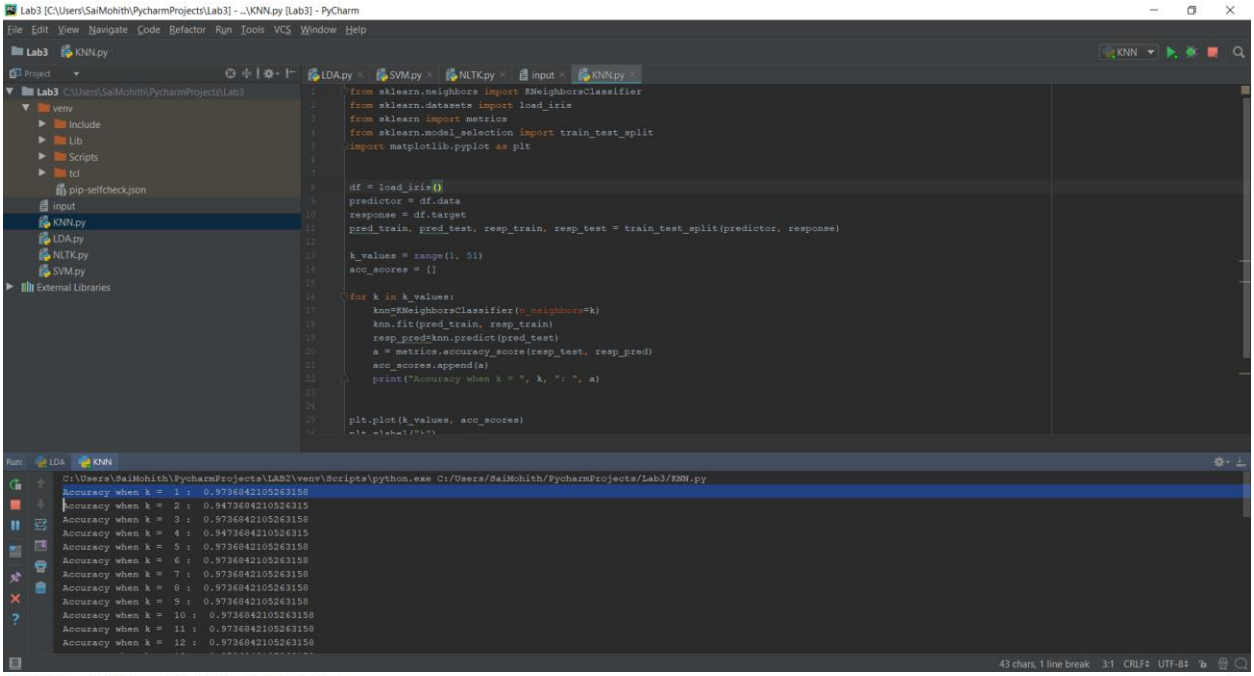


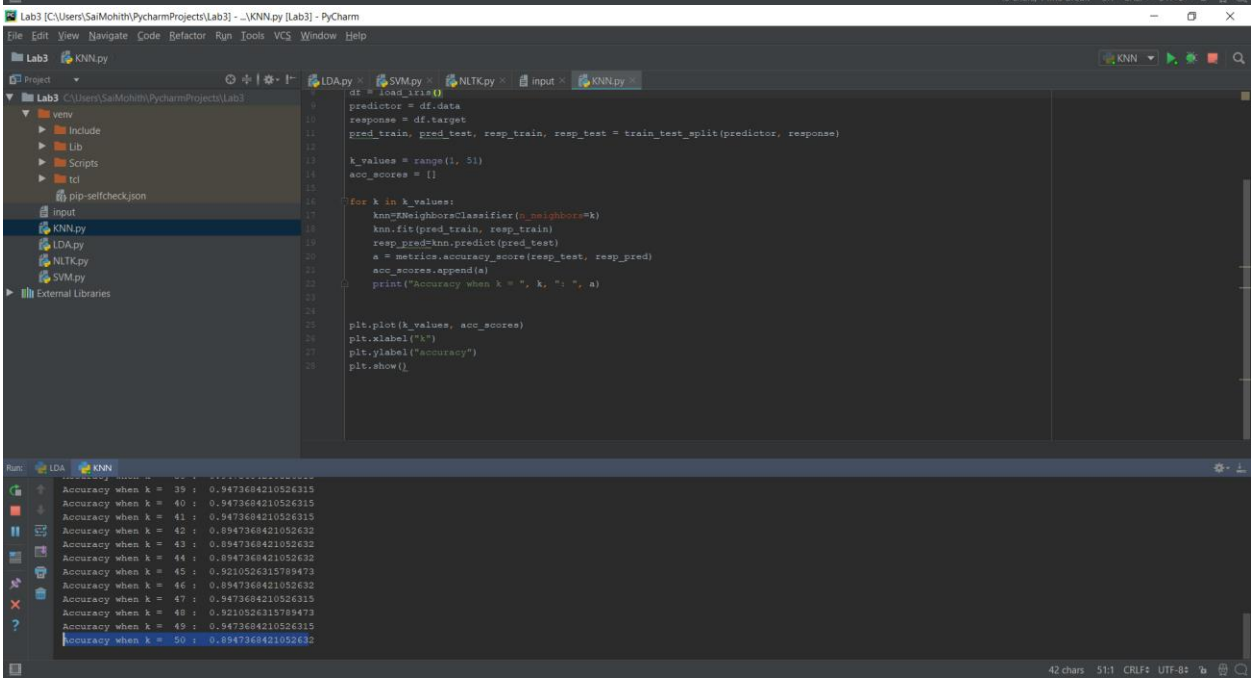
3. **Lemmatization and Bi-grams:**


Lab3 [C:\Users\SaiMohith\PycharmProjects\Lab3] - ...\NLTK.py [Lab3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Lab3 NLTK.py
LDA.py SVM.py NLTK.py input KNN.py
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.tag import pos_tag
from nltk import ngrams
from operator import itemgetter
file = open('input', 'r') # Read the input file to lemmatize and get bigrams
lines_in_file = file.readlines()
mrg = ""
for line in lines_in_file: # Merge all the lines into one single line
mrg=mrg+line
words_mrg = word_tokenize(mrg) # Word Tokenize the sentence
# lemmatization
lemmatizer = WordNetLemmatizer() # Lemmatize the words
list_lemm_words = []
for word in words_mrg:
lemm_word = lemmatizer.lemmatize(word)
list_lemm_words.append(lemm_word)
print("\n Lemmetization: \n")
print(list_lemm_words) # print the Lemmatized words
C:\Users\SaiMohith\PycharmProjects\LAB2\venv\Scripts\python.exe C:/Users/SaiMohith/PycharmProjects/Lab3/NLTK.py
Lemmetization:
['Black', 'Panther', 'is', 'a', '2018', 'American', 'superhero', 'film', 'based', 'on', 'the', 'Marvel', 'Comics', 'character', 'of', 'the', 'same', 'name', '.', 'Produced', 'by', 'Marvel', 'Studios',
BI-GRAMS in the Input:
[('Black', 'Panther'), ('Panther', 'is'), ('is', 'a'), ('a', '2018'), ('2018', 'American'), ('American', 'superhero'), ('superhero', 'film'), ('film', 'based'), ('based', 'on'), ('on', 'the'), ('the', '
BI-GRAMS Word Frequencies:
[(('(', 'MCU'), 1), ((')', '.'), 1), ((',', 'Angela'), 1), ((',', 'Danai'), 1), ((',', 'Daniel'), 1), ((',', 'Forest'), 1), ((',', 'Letitia'), 1), ((',', 'Lupita'), 1), ((',', 'Martin'), 1), ((',', "T'Ch
15:1 CRLF UTF-8


Lab3 [C:\Users\SaiMohith\PycharmProjects\Lab3] - ...\NLTK.py [Lab3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Lab3 NLTK.py
LDA.py SVM.py NLTK.py input KNN.py
grams_mrg = []
bigrams = ngrams(list_lemm_words, n=2) # Bi-grams in the list of lemmatized words
for grams in bigrams:
grams_mrg.append(grams)
print("\n BI-GRAMS in the Input: \n")
print(grams_mrg) # Print the Bigrams
# Word Freq
pos_mrg = pos_tag(list_lemm_words) # parts of Speech for all the words
string = " ".join(str(x) for x,y in pos_mrg)
string_word = word_tokenize(string)
mrg_freq = nltk.FreqDist(grams_mrg) # Calculate the frequencies of words
common = mrg_freq.most_common()
BWF = sorted(common,key=itemgetter(0))
print("\n BI-GRAMS Word Frequencies: \n")
print(BWF) # print the Bi-grams with word frequencies
print("\n Top Five BI-GRAMS According to Word Count: \n")
top_five_bigrams = mrg_freq.most_common(5) # gets the 5 most common occurred bi-grams in the top_five_bigrams
print(top_five_bigrams)
# Sentences
sentence_mrg = sent_tokenize(mrg) # Gets the sentences from the merged input file
sent_list = []
for sent in sentence_mrg:
[(('(', 'MCU'), 1), ((')', '.'), 1), ((',', 'Angela'), 1), ((',', 'Danai'), 1), ((',', 'Daniel'), 1), ((',', 'Forest'), 1), ((',', 'Letitia'), 1), ((',', 'Lupita'), 1), ((',', 'Martin'), 1), ((',', "T'Ch
Top Five BI-GRAMS According to Word Count:
[(('Black', 'Panther'), 3), (('the', 'Marvel'), 2), ((',', 'and'), 2), (('Panther', ','), 2), (('Panther', 'is'), 1)]
Sentences with top five Bigrams:
The film is directed by Ryan Coogler, who co-wrote the screenplay with Joe Robert Cole, and stars Chadwick Boseman as T'Challa / Black Panther, alongside Michael B. Jordan, Lupita Nyong'o, Danai Gurira,
Process finished with exit code 0
15:1 CRLF UTF-8

4. **KNN:**


Lab3 [C:\Users\SaiMohith\PycharmProjects\Lab3] - ...\KNN.py [Lab3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Lab3 KNN.py
KNN
Project
Lab3 C:\Users\SaiMohith\PycharmProjects\Lab3
venv
Include
Lib
Scripts
tcl
pip-selfcheck.json
input
KNN.py
LDA.py
NLTK.py
SVM.py
External Libraries
LDA.py SVM.py NLTK.py input KNN.py
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
from sklearn import metrics
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
df = load_iris()
predictor = df.data
response = df.target
pred_train, pred_test, resp_train, resp_test = train_test_split(predictor, response)
k_values = range(1, 51)
acc_scores = []
for k in k_values:
knn=KNeighborsClassifier(n_neighbors=k)
knn.fit(pred_train, resp_train)
resp_pred=knn.predict(pred_test)
a = metrics.accuracy_score(resp_test, resp_pred)
acc_scores.append(a)
print("Accuracy when k = ", k, ": ", a)
plt.plot(k_values, acc_scores)
Run: LDA KNN
C:\Users\SaiMohith\PycharmProjects\LAB2\venv\Scripts\python.exe C:/Users/SaiMohith/PycharmProjects/Lab3/KNN.py
Accuracy when k = 1 : 0.9736842105263158
Accuracy when k = 2 : 0.9473684210526315
Accuracy when k = 3 : 0.9736842105263158
Accuracy when k = 4 : 0.9473684210526315
Accuracy when k = 5 : 0.9736842105263158
Accuracy when k = 6 : 0.9736842105263158
Accuracy when k = 7 : 0.9736842105263158
Accuracy when k = 8 : 0.9736842105263158
Accuracy when k = 9 : 0.9736842105263158
Accuracy when k = 10 : 0.9736842105263158
Accuracy when k = 11 : 0.9736842105263158
Accuracy when k = 12 : 0.9736842105263158
43 chars, 1 line break 3:1 CRLF UTF-8


Lab3 [C:\Users\SaiMohith\PycharmProjects\Lab3] - ...\KNN.py [Lab3] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Lab3 KNN.py
KNN
Project
Lab3 C:\Users\SaiMohith\PycharmProjects\Lab3
venv
Include
Lib
Scripts
tcl
pip-selfcheck.json
input
KNN.py
LDA.py
NLTK.py
SVM.py
External Libraries
LDA.py SVM.py NLTK.py input KNN.py
df = load_iris()
predictor = df.data
response = df.target
pred_train, pred_test, resp_train, resp_test = train_test_split(predictor, response)
k_values = range(1, 51)
acc_scores = []
for k in k_values:
knn=KNeighborsClassifier(n_neighbors=k)
knn.fit(pred_train, resp_train)
resp_pred=knn.predict(pred_test)
a = metrics.accuracy_score(resp_test, resp_pred)
acc_scores.append(a)
print("Accuracy when k = ", k, ": ", a)
plt.plot(k_values, acc_scores)
plt.xlabel("k")
plt.ylabel("accuracy")
plt.show()
Run: LDA KNN
Accuracy when k = 39 : 0.9473684210526315
Accuracy when k = 40 : 0.9473684210526315
Accuracy when k = 41 : 0.9473684210526315
Accuracy when k = 42 : 0.8947368421052632
Accuracy when k = 43 : 0.8947368421052632
Accuracy when k = 44 : 0.8947368421052632
Accuracy when k = 45 : 0.9210526315789473
Accuracy when k = 46 : 0.8947368421052632
Accuracy when k = 47 : 0.9473684210526315
Accuracy when k = 48 : 0.9210526315789473
Accuracy when k = 49 : 0.9473684210526315
Accuracy when k = 50 : 0.8947368421052632
42 chars 51:1 CRLF UTF-8

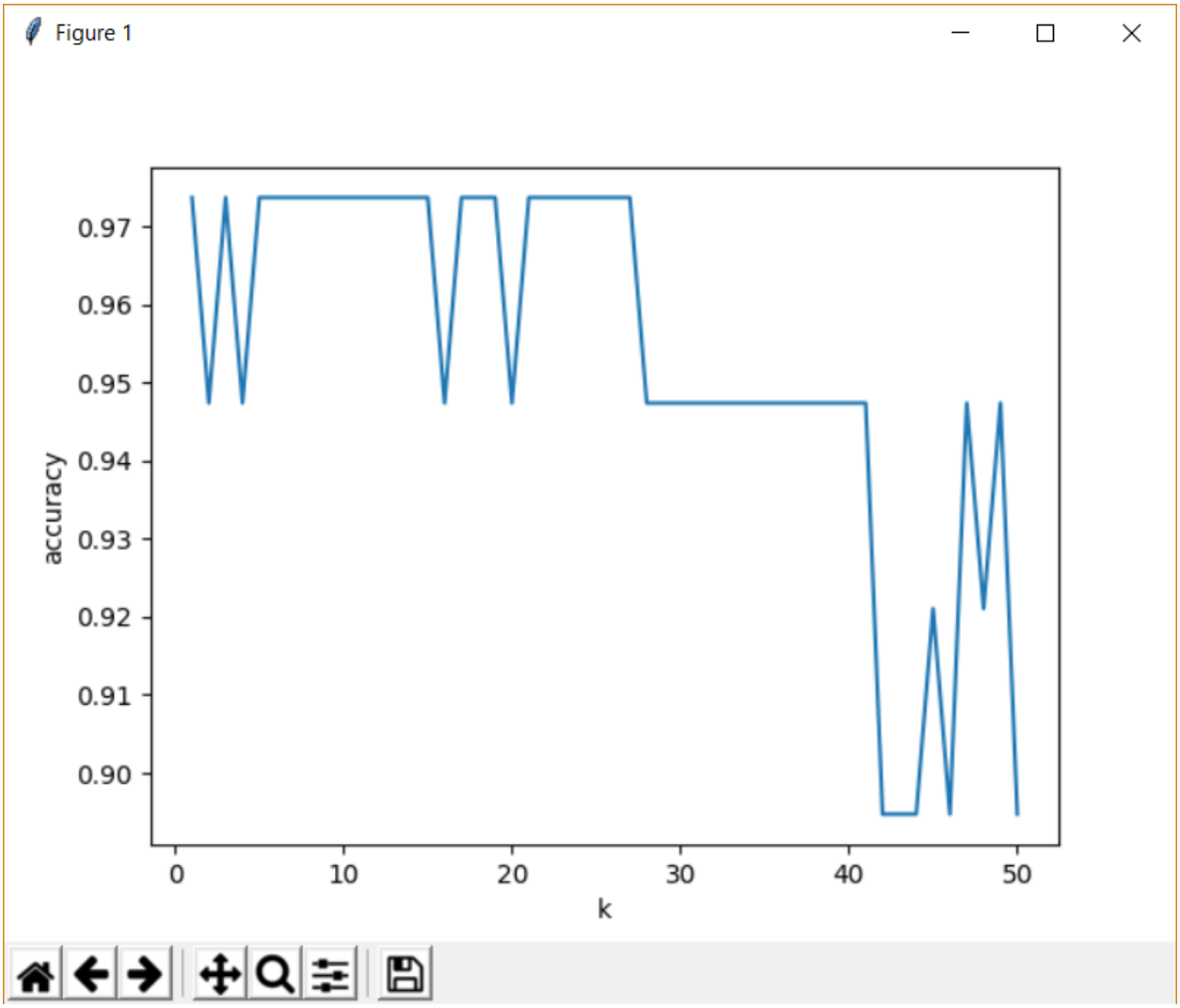


Figure 1
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.90
accuracy
0
10
20
30
40
50
k

Code Implementation:

1. LDA:

In this program we perform LDA on the Digits dataset. This dataset had 10 classes. We first load the data and then apply the lda to the data. We then plot the results using Matplotlib.

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Loading Digits data into dig_data
dig_data = load_digits()
predictor = dig_data.data
response = dig_data.target
targets = dig_data.target_names  # Making a list of targets
```

Now we apply LDA to the predictor and response and then transform the predictor.

```
lda = LinearDiscriminantAnalysis(n_components=2)
predictor = lda.fit(predictor, response).transform(predictor)  # transform the predictor
```

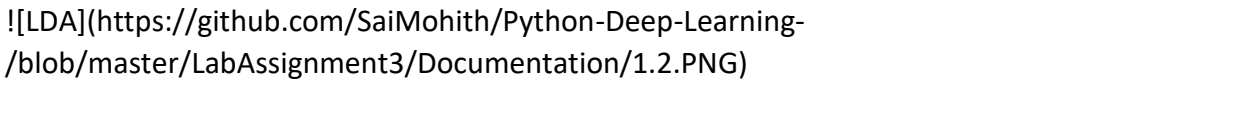
Now we plot the results on the graph. As, we got 10 targets,i used 10 colors to represent each target. We can see the classification in the output.

```
# Plotting the LDA on the graph. Different colors represent different targets
plt.figure()
colors = ['navy', 'turquoise', 'brown', 'red', 'blue', 'green', 'black', 'cyan', 'pink', 'purple']
```

```
    # plotting Data
    for color, i, target_name in zip(colors, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], targets):
        plt.scatter(predictor[response == i, 0], predictor[response == i, 1], alpha=.6, color=color,
                    label=target_name)

    # plotting labels
    plt.legend(loc='best')
    plt.title('LDA of Digits Dataset')
    plt.xlabel('Predictor Values')
    plt.ylabel('Response Values')
    plt.show()
```

Output:



Logistic Regression and LDA Differences:

LDA reduces the number of dimensions for the data. If a n independent variable data set is considered, LDA can show the data in p<n dimensions. Both LDA and Logistic Regression produce linear boundaries. Data is assumed to be normally distributed and with common variance in LDA. But, Logistic Regression doesn't have this assumption. LDA performs better that Logistic regression as long as this assumption hold for the data. LDA performs better that Logistic Regression in the case of two or more classes.

**2. SVM Classification: **

In this program, we will use SVM classification on iris dataset. We will use 20% of the data as testing data and remaining as training data. I created two functions here, one for the linear and other for rbf Simple vector classification.

We calculate the accuracy using accuracy_score and get the precision, recall, f1 score using classification_report.

```
# Import  Required LIbraries

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, classification_report



# Function which performs SVC with kernel as linear

def lin_SVM(Xtr, ytr, Xte, yte):

    lin_cls = SVC(kernel='linear', random_state=10)  # setting Kernel to linear

    lin_cls.fit(Xtr, ytr)  # fitting the model

    y_pred = lin_cls.predict(Xte)  # predict the value using test data

    print('\nAccuracy when kernel is Linear: ', accuracy_score(yte, y_pred))  # print the
Accuracy Score

    print('\nClassification Report for Linear Kernel: ')

    print(classification_report(yte, y_pred))  # print the Classification report



# function which performs SVC with kernel as rbf
```

```
    def rbf_SVM(rXtr, rytr, rXte, ryte):
        rbf_cls = SVC(kernel='rbf', random_state=10)  # setting Kernel to rbf
        rbf_cls.fit(rXtr, rytr)
        y_pred = rbf_cls.predict(rXte)  # predict the value using test data
        print('\nAccuracy when Kernel is RBF: ', accuracy_score(ryte, y_pred))  # print the Accuracy Score
        print('\nClassification Report for RBF: ')
        print(classification_report(ryte, y_pred))  # print the Classification report



    #Load Iris Data
    df = load_iris()
    predictor = df.data
    response = df.target

    # Split the test and train data. Here 20% of the data is test data
    pred_train, pred_test, resp_train, resp_test = train_test_split(predictor, response, test_size = 0.20, random_state =
    10)

    # Feature Scaling for scaling the values
    scale = StandardScaler()
    pred_train = scale.fit_transform(pred_train)
    pred_test = scale.transform(pred_test)

    # Calling Functions
    lin_SVM(pred_train, resp_train, pred_test, resp_test)
    rbf_SVM(pred_train, resp_train, pred_test, resp_test)
```

Output:

```
C:\Users\SaiMohith\PycharmProjects\LAB2\venv\Scripts\python.exe
C:/Users/SaiMohith/PycharmProjects/Lab3/SVM.py

Accuracy when kernel is Linear:  0.9666666666666667

Classification Report for Linear Kernel:
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        10
          1       1.00      0.92      0.96        13
          2       0.88      1.00      0.93         7

avg / total       0.97      0.97      0.97        30


Accuracy when Kernel is RBF:  0.9666666666666667

Classification Report for RBF:
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        10
          1       1.00      0.92      0.96        13
          2       0.88      1.00      0.93         7
```

```
avg / total       0.97      0.97      0.97       30


Process finished with exit code 0
```

If we see, for IRIS data, i got the same accuracy for both linear and rbf kernels. In the case of the Iris dataset, SVC with rbf or linear kernel is a good fit. The accuracy is 96.6% which says that this model is a good fit.

3. NLTK:

```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.tag import pos_tag
from nltk import ngrams
from operator import itemgetter

file = open('input', 'r')  # Read the input file  to lemmatize and get bigrams
lines_in_file = file.readlines()
mrg = ""
for line in lines_in_file:  # Merge all the lines into one single line
    mrg=mrg+line

words_mrg = word_tokenize(mrg)  # Word Tokenize the sentence

# lemmatization
```

```
lemmatizer = WordNetLemmatizer()  # Lemmatize the words
list_lemm_words = []
for word in words_mrg:
   lemm_word = lemmatizer.lemmatize(word)
   list_lemm_words.append(lemm_word)
print("\n Lemmetization: \n")
print(list_lemm_words)  # print the Lemmatized words

# Bi-grams

grams_mrg = []
bigrams = ngrams(list_lemm_words, n=2)  # Bi-grams in the list of lemmatized words
for grams in bigrams:
   grams_mrg.append(grams)
print("\n BI-GRAMS in the Input: \n")
print(grams_mrg)  # Print the Bigrams

# Word Freq
pos_mrg = pos_tag(list_lemm_words)  # parts of Speech for all the words
string = " ".join(str(x) for x,y in pos_mrg)
string_word = word_tokenize(string)
mrg_freq = nltk.FreqDist(grams_mrg)  # Calculate the frequencies of words
common = mrg_freq.most_common()
BWF = sorted(common,key=itemgetter(0))
print("\n BI-GRAMS Word Frequencies: \n")
print(BWF)  # print the Bi-grams with word frequencies
```

```
print("\n Top Five BI-GRAMS According to Word Count: \n")

top_five_bigrams = mrg_freq.most_common(5)  # gets the  5 most common occurred bi-grams in the top_five_bigrams

print(top_five_bigrams)


# Sentences

sentence_mrg = sent_tokenize(mrg)  # Gets the sentences from the merged input file

sent_list = []


for sent in sentence_mrg:

    for word, words in grams_mrg:

        for ((o, p), l) in top_five_bigrams:

            if (word, words) == (o, p):

                sent_list.append(sent)  # Appending the sentences with the most common words.

print ("\n Sentences with top five Bigrams: \n")

print(max(sent_list, key=len))
```

Input:

Black Panther is a 2018 American superhero film based on the Marvel Comics character of the same name. Produced by

Marvel Studios and distributed by Walt Disney Studios Motion Pictures, it is the eighteenth film in the Marvel

Cinematic Universe (MCU). The film is directed by Ryan Coogler, who co-wrote the screenplay with Joe Robert Cole, and

stars Chadwick Boseman as T'Challa / Black Panther, alongside Michael B. Jordan, Lupita Nyong'o, Danai Gurira, Martin

Freeman, Daniel Kaluuya, Letitia Wright, Winston Duke, Angela Bassett, Forest Whitaker, and Andy Serkis.

In Black Panther, T'Challa returns home as king of Wakanda but finds his sovereignty challenged by a new adversary, in

a conflict with global consequences.

Output:

Lemmetization:

['Black', 'Panther', 'is', 'a', '2018', 'American', 'superhero', 'film', 'based', 'on', 'the', 'Marvel', 'Comics',

'character', 'of', 'the', 'same', 'name', '.', 'Produced', 'by', 'Marvel', 'Studios', 'and', 'distributed', 'by',

'Walt', 'Disney', 'Studios', 'Motion', 'Pictures', ',', 'it', 'is', 'the', 'eighteenth', 'film', 'in', 'the',

'Marvel', 'Cinematic', 'Universe', '(', 'MCU', ')', '.', 'The', 'film', 'is', 'directed', 'by', 'Ryan', 'Coogler',

',', 'who', 'co-wrote', 'the', 'screenplay', 'with', 'Joe', 'Robert', 'Cole', ',', 'and', 'star', 'Chadwick',

'Boseman', 'a', "T'Challa", '/', 'Black', 'Panther', ',', 'alongside', 'Michael', 'B.', 'Jordan', ',', 'Lupita',

"Nyong'o", ',', 'Danai', 'Gurira', ',', 'Martin', 'Freeman', ',', 'Daniel', 'Kaluuya', ',', 'Letitia', 'Wright', ',',

'Winston', 'Duke', ',', 'Angela', 'Bassett', ',', 'Forest', 'Whitaker', ',', 'and', 'Andy', 'Serkis', '.', 'In',

'Black', 'Panther', ',', "T'Challa", 'return', 'home', 'a', 'king', 'of', 'Wakanda', 'but', 'find', 'his',

'sovereignty', 'challenged', 'by', 'a', 'new', 'adversary', ',', 'in', 'a', 'conflict', 'with', 'global',

'consequence', '.']

BI-GRAMS in the Input:

[('Black', 'Panther'), ('Panther', 'is'), ('is', 'a'), ('a', '2018'), ('2018', 'American'), ('American', 'superhero'),

('superhero', 'film'), ('film', 'based'), ('based', 'on'), ('on', 'the'), ('the', 'Marvel'), ('Marvel', 'Comics'),

('Comics', 'character'), ('character', 'of'), ('of', 'the'), ('the', 'same'), ('same', 'name'), ('name', '.'), ('.',

'Produced'), ('Produced', 'by'), ('by', 'Marvel'), ('Marvel', 'Studios'), ('Studios', 'and'), ('and', 'distributed'),

('distributed', 'by'), ('by', 'Walt'), ('Walt', 'Disney'), ('Disney', 'Studios'), ('Studios', 'Motion'), ('Motion',

'Pictures'), ('Pictures', ','), (',', 'it'), ('it', 'is'), ('is', 'the'), ('the', 'eighteenth'), ('eighteenth',

'film'), ('film', 'in'), ('in', 'the'), ('the', 'Marvel'), ('Marvel', 'Cinematic'), ('Cinematic', 'Universe'),

('Universe', '('), ('(', 'MCU'), ('MCU', ')'), (')', '.'), ('.', 'The'), ('The', 'film'), ('film', 'is'), ('is',

'directed'), ('directed', 'by'), ('by', 'Ryan'), ('Ryan', 'Coogler'), ('Coogler', ','), (',', 'who'), ('who', 'co-

wrote'), ('co-wrote', 'the'), ('the', 'screenplay'), ('screenplay', 'with'), ('with', 'Joe'), ('Joe', 'Robert'),

('Robert', 'Cole'), ('Cole', ','), (',', 'and'), ('and', 'star'), ('star', 'Chadwick'), ('Chadwick', 'Boseman'),

('Boseman', 'a'), ('a', "T'Challa"), ("T'Challa", '/'), ('/', 'Black'), ('Black', 'Panther'), ('Panther', ','), (',',

'alongside'), ('alongside', 'Michael'), ('Michael', 'B.'), ('B.', 'Jordan'), ('Jordan', ','), (',', 'Lupita'),

('Lupita', "Nyong'o"), ("Nyong'o", ','), (',', 'Danai'), ('Danai', 'Gurira'), ('Gurira', ','), (',', 'Martin'),

('Martin', 'Freeman'), ('Freeman', ','), (',', 'Daniel'), ('Daniel', 'Kaluuya'), ('Kaluuya', ','), (',', 'Letitia'),

('Letitia', 'Wright'), ('Wright', ','), (',', 'Winston'), ('Winston', 'Duke'), ('Duke', ','), (',', 'Angela'),

('Angela', 'Bassett'), ('Bassett', ','), (',', 'Forest'), ('Forest', 'Whitaker'), ('Whitaker', ','), (',', 'and'),

('and', 'Andy'), ('Andy', 'Serkis'), ('Serkis', '.'), ('.', 'In'), ('In', 'Black'), ('Black', 'Panther'), ('Panther',

','), (',', "T'Challa"), ("T'Challa", 'return'), ('return', 'home'), ('home', 'a'), ('a', 'king'), ('king', 'of'),

('of', 'Wakanda'), ('Wakanda', 'but'), ('but', 'find'), ('find', 'his'), ('his', 'sovereignty'), ('sovereignty',

'challenged'), ('challenged', 'by'), ('by', 'a'), ('a', 'new'), ('new', 'adversary'), ('adversary', ','), (',', 'in'),

('in', 'a'), ('a', 'conflict'), ('conflict', 'with'), ('with', 'global'), ('global', 'consequence'), ('consequence',

'.')]

BI-GRAMS Word Frequencies:

[(('(', 'MCU'), 1), ((')', '.'), 1), ((',', 'Angela'), 1), ((',', 'Danai'), 1), ((',', 'Daniel'), 1), ((',',

'Forest'), 1), ((',', 'Letitia'), 1), ((',', 'Lupita'), 1), ((',', 'Martin'), 1), ((',', "T'Challa"), 1), ((',',

'Winston'), 1), ((',', 'alongside'), 1), ((',', 'and'), 2), ((',', 'in'), 1), ((',', 'it'), 1), ((',', 'who'), 1),

(('.', 'In'), 1), (('.', 'Produced'), 1), (('.', 'The'), 1), (('/', 'Black'), 1), (('2018', 'American'), 1),

(('American', 'superhero'), 1), (('Andy', 'Serkis'), 1), (('Angela', 'Bassett'), 1), (('B.', 'Jordan'), 1),

(('Bassett', ','), 1), (('Black', 'Panther'), 3), (('Boseman', 'a'), 1), (('Chadwick', 'Boseman'), 1), (('Cinematic',

'Universe'), 1), (('Cole', ','), 1), (('Comics', 'character'), 1), (('Coogler', ','), 1), (('Danai', 'Gurira'), 1),

(('Daniel', 'Kaluuya'), 1), (('Disney', 'Studios'), 1), (('Duke', ','), 1), (('Forest', 'Whitaker'), 1), (('Freeman',

','), 1), (('Gurira', ','), 1), (('In', 'Black'), 1), (('Joe', 'Robert'), 1), (('Jordan', ','), 1), (('Kaluuya', ','),

1), (('Letitia', 'Wright'), 1), (('Lupita', "Nyong'o"), 1), (('MCU', ')'), 1), (('Martin', 'Freeman'), 1), (('Marvel',

'Cinematic'), 1), (('Marvel', 'Comics'), 1), (('Marvel', 'Studios'), 1), (('Michael', 'B.'), 1), (('Motion',

'Pictures'), 1), (("Nyong'o", ','), 1), (('Panther', ','), 2), (('Panther', 'is'), 1), (('Pictures', ','), 1),

(('Produced', 'by'), 1), (('Robert', 'Cole'), 1), (('Ryan', 'Coogler'), 1), (('Serkis', '.'), 1), (('Studios',

'Motion'), 1), (('Studios', 'and'), 1), (("T'Challa", '/'), 1), (("T'Challa", 'return'), 1), (('The', 'film'), 1),

(('Universe', '('), 1), (('Wakanda', 'but'), 1), (('Walt', 'Disney'), 1), (('Whitaker', ','), 1), (('Winston',

'Duke'), 1), (('Wright', ','), 1), (('a', '2018'), 1), (('a', "T'Challa"), 1), (('a', 'conflict'), 1), (('a', 'king'),

1), (('a', 'new'), 1), (('adversary', ','), 1), (('alongside', 'Michael'), 1), (('and', 'Andy'), 1), (('and',

'distributed'), 1), (('and', 'star'), 1), (('based', 'on'), 1), (('but', 'find'), 1), (('by', 'Marvel'), 1), (('by',

'Ryan'), 1), (('by', 'Walt'), 1), (('by', 'a'), 1), (('challenged', 'by'), 1), (('character', 'of'), 1), (('co-wrote',

'the'), 1), (('conflict', 'with'), 1), (('consequence', '.'), 1), (('directed', 'by'), 1), (('distributed', 'by'), 1),

(('eighteenth', 'film'), 1), (('film', 'based'), 1), (('film', 'in'), 1), (('film', 'is'), 1), (('find', 'his'), 1),

(('global', 'consequence'), 1), (('his', 'sovereignty'), 1), (('home', 'a'), 1), (('in', 'a'), 1), (('in', 'the'),

1), (('is', 'a'), 1), (('is', 'directed'), 1), (('is', 'the'), 1), (('it', 'is'), 1), (('king', 'of'), 1), (('name',

'.'), 1), (('new', 'adversary'), 1), (('of', 'Wakanda'), 1), (('of', 'the'), 1), (('on', 'the'), 1), (('return',

'home'), 1), (('same', 'name'), 1), (('screenplay', 'with'), 1), (('sovereignty', 'challenged'), 1), (('star',

'Chadwick'), 1), (('superhero', 'film'), 1), (('the', 'Marvel'), 2), (('the', 'eighteenth'), 1), (('the', 'same'), 1),

(('the', 'screenplay'), 1), (('who', 'co-wrote'), 1), (('with', 'Joe'), 1), (('with', 'global'), 1)]

Top Five BI-GRAMS According to Word Count:

[(('Black', 'Panther'), 3), (('the', 'Marvel'), 2), ((',', 'and'), 2), (('Panther', ','), 2), (('Panther', 'is'), 1)]

Sentences with top five Bigrams:

The film is directed by Ryan Coogler, who co-wrote the screenplay with Joe Robert Cole, and stars Chadwick Boseman as

T'Challa / Black Panther, alongside Michael B. Jordan, Lupita Nyong'o, Danai Gurira, Martin Freeman, Daniel Kaluuya,

Letitia Wright, Winston Duke, Angela Bassett, Forest Whitaker, and Andy Serkis.

Process finished with exit code 0

4. KNN :

In this program we implemented KNN with k values ranging from 1 to 50 and observe the accuracy of the model for each value of k.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
from sklearn import metrics
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt


df = load_iris()  # load Iris Data
predictor = df.data
response = df.target
pred_train, pred_test, resp_train, resp_test = train_test_split(predictor, response)

acc_scores = []

for k in range(1,51):
```

```
    knn=KNeighborsClassifier(n_neighbors=k)  # KNN for different values of k
    knn.fit(pred_train, resp_train)
    resp_pred=knn.predict(pred_test)
    a = metrics.accuracy_score(resp_test, resp_pred)  # get the accuracy
    acc_scores.append(a)
    print("Accuracy when k = ", k, ": ", a)


#Plot the Graph with Accuracies
plt.plot(k_values, acc_scores)
plt.xlabel("k")
plt.ylabel("accuracy")
plt.show()
```

Output:

```
Accuracy when k =  1 :  0.9473684210526315
Accuracy when k =  2 :  0.8947368421052632
Accuracy when k =  3 :  0.9736842105263158
Accuracy when k =  4 :  0.8947368421052632
Accuracy when k =  5 :  0.9210526315789473
Accuracy when k =  6 :  0.9210526315789473
Accuracy when k =  7 :  0.9210526315789473
Accuracy when k =  8 :  0.9210526315789473
Accuracy when k =  9 :  0.9210526315789473
Accuracy when k =  10 :  0.9210526315789473
Accuracy when k =  11 :  0.9473684210526315
```

Accuracy when k = 12 : 0.9473684210526315

Accuracy when k = 13 : 0.9473684210526315

Accuracy when k = 14 : 0.9210526315789473

Accuracy when k = 15 : 0.9473684210526315

Accuracy when k = 16 : 0.9210526315789473

Accuracy when k = 17 : 0.9210526315789473

Accuracy when k = 18 : 0.9210526315789473

Accuracy when k = 19 : 0.9210526315789473

Accuracy when k = 20 : 0.9210526315789473

Accuracy when k = 21 : 0.9210526315789473

Accuracy when k = 22 : 0.9210526315789473

Accuracy when k = 23 : 0.9210526315789473

Accuracy when k = 24 : 0.9210526315789473

Accuracy when k = 25 : 0.9473684210526315

Accuracy when k = 26 : 0.8947368421052632

Accuracy when k = 27 : 0.9210526315789473

Accuracy when k = 28 : 0.8947368421052632

Accuracy when k = 29 : 0.8947368421052632

Accuracy when k = 30 : 0.8947368421052632

Accuracy when k = 31 : 0.9210526315789473

Accuracy when k = 32 : 0.8947368421052632

Accuracy when k = 33 : 0.8947368421052632

Accuracy when k = 34 : 0.8947368421052632

Accuracy when k = 35 : 0.8947368421052632

Accuracy when k = 36 : 0.8947368421052632

Accuracy when k = 37 : 0.8947368421052632

Accuracy when k = 38 : 0.8947368421052632

Accuracy when k = 39 : 0.8947368421052632

Accuracy when k = 40 : 0.8947368421052632

Accuracy when k = 41 : 0.8947368421052632

Accuracy when k = 42 : 0.8947368421052632

Accuracy when k = 43 : 0.8947368421052632

Accuracy when k = 44 : 0.8947368421052632

Accuracy when k = 45 : 0.8947368421052632

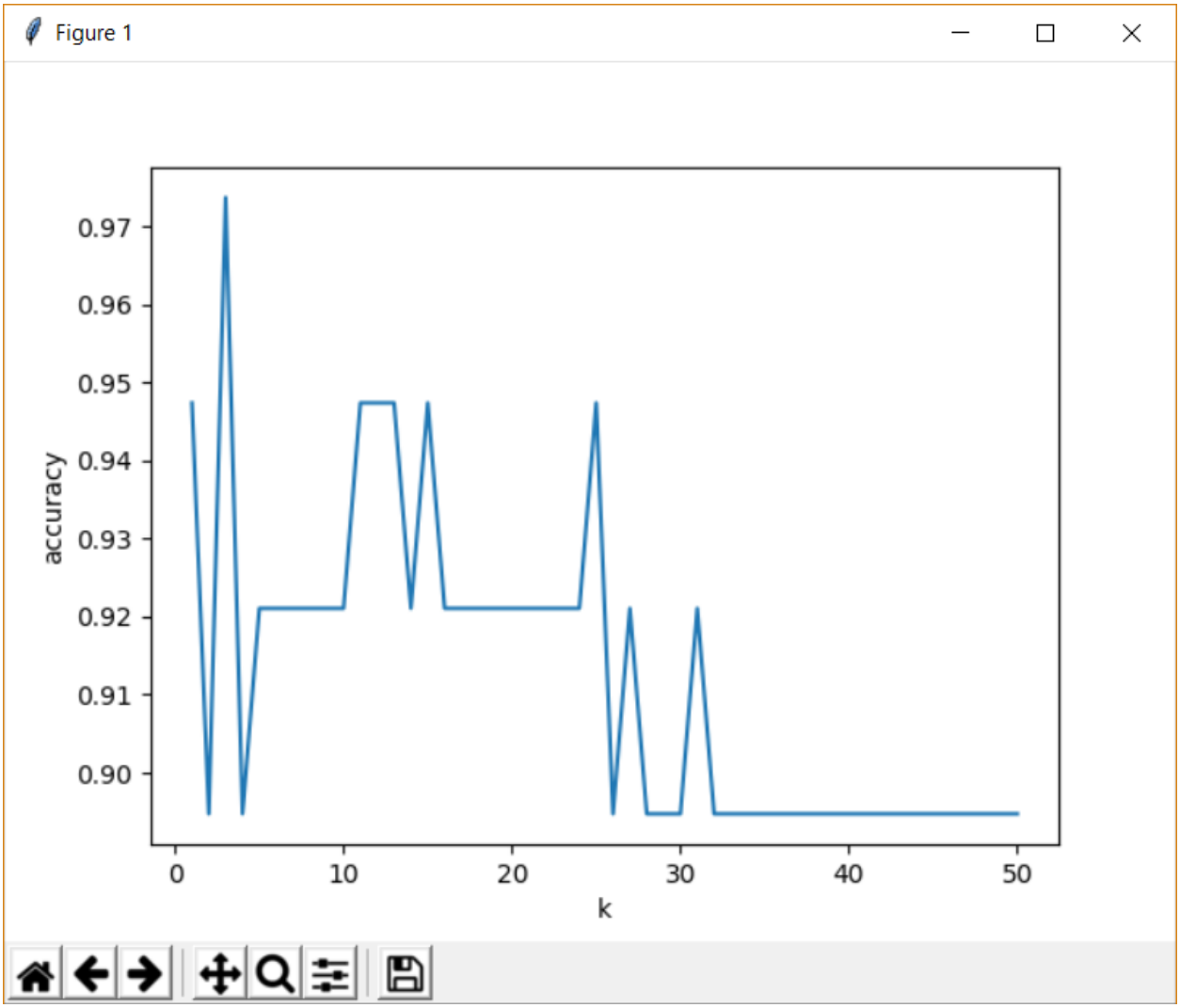
Accuracy when k = 46 : 0.8947368421052632

Accuracy when k = 47 : 0.8947368421052632

Accuracy when k = 48 : 0.8947368421052632

Accuracy when k = 49 : 0.8947368421052632

Accuracy when k = 50 : 0.8947368421052632



If we observe the accuracy when k =1 is 94.7% and when k = 50 for the same model, the accuracy is 89.4%. The accuracy decreased as the k value increases. Generally K value can be selected by using cross validation. If we see in our output, k =3 has more accuracy and stands as good fit for the model. Generally k = 1, 3, 5 are known to be good values to get the best fit for most of the models. The k value is used in controlling the decision boundary. In our case when k=1 or k=3 we got high accuracy, which means there is a less bias between the neighbors, but more variance. In the case of k = 50, we can see that there is more bias but decreased variance. Accuracy decreased when k = 50 because, in the give data the data points are closely related and can be divided into less categories. So, increasing the distance made the accuracy to fall down.

Limitations:

Python 3.6 is used for writing this code. Some functions may not work properly when run on python 2(2.x) version.

The datasets used in the lab assignment are small datasets. These cannot be used for datasets which in case of real time applications.

References:

http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits

https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

http://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html

https://stackoverflow.com/questions/26225344/why-feature-scaling

https://en.wikipedia.org/wiki/Black_Panther_(film)