This document contains only code portion as requested in the file:

https://umkc.app.box.com/s/c3pim9503j9usgdqieuot468ps9ftrnb

If you need the file with full documentation. please check the gitub lab3 folder. It is quite confusing as lab3 qsns are updated so I am keeping only the code part here.

Q1.Code:

```
ir = datasets.load_iris()
inp = ir.data

out= ir.target

trget_name = ir.target_names

X_trn, X_tst, y_trn, y_tst = train_test_split(inp, out, test_size=0.30)

clasifir = KNeighborsClassifier(n_neighbrs=5)

clasifir.fit(X_trn, y_trn)

y_prd = clasifir.predict(X_tst)

lda = LinearDiscriminantAnalysis(n_components=2)

X_r2 = lda.fit(X_test, y_pred).transform(X)

plt.figure()

colors = [‘black’, 'blue', 'orange']

lw = 2
for clr, val, trget_name in zip(colors, [0, 1, 2], trget_names):

    plt.scatter(inp_r2[out== val, 0], inp_r2[y == val, 1], alpha=.9, color=clr,

                label=trget_name)
```

```
plt.legend(loc='best',shadw=False, scatterpoints=1)

plt.title('Linear-Discriminant-Analysis with Iris-Dataset')

plt.show()
```

Q2.Code:

```
C =  1 . 0

dgit=load_digits()

in=dgit.data

out=dgit.target

x_trn, x_tst,y_trin,y_tst=train_test_split(in,out,test_size=0.3)

mdl = svm.SVC(kernel='linear')

mdl.fit(x_train,y_train)

y_prd=mdl.predict(x_test)

print("Acc for SVC-linear kernel " + str(metrics.accuracy_score(y_tst,y_prd)))

mdl2 = svm.SVC(kernel='rbf')

mdl2.fit(x_trn,y_trn)

y_prd=mdl2.predict(x_tst)

print("Acc for SVC-rbf-kernel " + str(metrics.accuracy_score(y_tst,y_prd)))
```

O/p:

```
Acc for SVC-linear kernel 0.977777777778
Acc for SVC-rbf kernel 0.513888888889
```

Q3.code:

```
with open('inpt.txt' , 'r') as f:

    lns = f.readlines()
```

```
f=''
for c in lines:
   f=f+c

print(f)

f_wrd = word_tokenize(f)

f_snt = sent_tokenize(f)

lema = WordNetLemmatizer()

f_lema = []

for wrd in f_wrd:

   f_lema = lema.lemmatize(wrd.lower())

   f_lema.append(f_lema)

print("\n Applying lemmetaizion for input text  ")

print(f_lema)
f_pos = pos_tag(f_lema)

print("Applying bigram for the text ")

n = 2
grm=[]

bigrm = ngrams(f_lema, n)

for grm in bigrm:

   grm.append(grm)

print(grm)

s1 = " ".join(str(d) for d,e in f_pos)

s1_wrd = word_tokenize(str1)

print("--------Bi-Grams with word  frequency----------")
```

```
fdist1 = nltk.FreqDist(gram)

top_fiv = fdist1.most_common()

top_five = fdist1.most_common(5)

top=sorted(top_fiv,key=itemgetter(0))
print(top)

print('---------Top 5 bi-grams word freq with count--------')

print(top_five)

sent1 = sent_tokenize(fr)

rep_sent1 = []

for sent in sent1:
   for word,words in gram:
      for ((c,m), l) in top_five:
         if (word,words == c,m):

            rep_sent1.append(sent)
print ("\n Sentences with top five Bigrams")

s=max(rep_sent1,key=len)

print(s)
```

Input:

hi this is rakesh

this is rakesh from comp science

from comp science there are different teams participating in the event

in the event is rakesh particpiating a big question to everyone ?

hello everyone how are you doing today

you will not be seen in output

Output:

hi this is rakesh
this is rakesh from comp science
from comp science there are different teams participating in the event

in the event is rakesh particpiating a big question to everyone ?
hello everyone how are you doing today
you will not be seen in output

----------lemmetaizion----------
['hi', 'this', 'is', 'rakesh', 'this', 'is', 'rakesh', 'from', 'comp', 'science', 'from', 'comp', 'science', 'there', 'are', 'different', 'team', 'participating', 'in', 'the', 'event', 'in', 'the', 'event', 'is', 'rakesh', 'particpiating', 'a', 'big', 'question', 'to', 'everyone', '?', 'hello', 'everyone', 'how', 'are', 'you', 'doing', 'today', 'you', 'will', 'not', 'be', 'seen', 'in', 'output']

----------BIGRAM----------
[('hi', 'this'), ('this', 'is'), ('is', 'rakesh'), ('rakesh', 'this'), ('this', 'is'), ('is', 'rakesh'), ('rakesh', 'from'), ('from', 'comp'), ('comp', 'science'), ('science', 'from'), ('from', 'comp'), ('comp', 'science'), ('science', 'there'), ('there', 'are'), ('are', 'different'), ('different', 'team'), ('team', 'participating'), ('participating', 'in'), ('in', 'the'), ('the', 'event'), ('event', 'in'), ('in', 'the'), ('the', 'event'), ('event', 'is'), ('is', 'rakesh'), ('rakesh', 'particpiating'), ('particpiating', 'a'), ('a', 'big'), ('big', 'question'), ('question', 'to'), ('to', 'everyone'), ('everyone', '?'), ('?', 'hello'), ('hello', 'everyone'), ('everyone', 'how'), ('how', 'are'), ('are', 'you'), ('you', 'doing'), ('doing', 'today'), ('today', 'you'), ('you', 'will'), ('will', 'not'), ('not', 'be'), ('be', 'seen'), ('seen', 'in'), ('in', 'output')]

--------Bi-Grams with word frequency----------
[(('?', 'hello'), 1), (('a', 'big'), 1), (('are', 'different'), 1), (('are', 'you'), 1), (('be', 'seen'), 1), (('big', 'question'), 1), (('comp', 'science'), 2), (('different', 'team'), 1), (('doing', 'today'), 1), (('event', 'in'), 1), (('event', 'is'), 1), (('everyone', '?'), 1), (('everyone', 'how'), 1), (('from', 'comp'), 2), (('hello', 'everyone'), 1), (('hi', 'this'), 1), (('how', 'are'), 1), (('in', 'output'), 1), (('in', 'the'), 2), (('is', 'rakesh'), 3), (('not', 'be'), 1), (('participating', 'in'), 1), (('particpiating', 'a'), 1), (('question', 'to'), 1), (('rakesh', 'from'), 1), (('rakesh', 'particpiating'), 1), (('rakesh', 'this'), 1), (('science', 'from'), 1), (('science', 'there'), 1), (('seen', 'in'), 1), (('team', 'participating'), 1), (('the', 'event'), 2), (('there', 'are'), 1), (('this', 'is'), 2), (('to', 'everyone'), 1), (('today', 'you'), 1), (('will', 'not'), 1), (('you', 'doing'), 1), (('you', 'will'), 1)]

---------Top 5 bi-grams word freq with count--------
[(('is', 'rakesh'), 3), (('this', 'is'), 2), (('from', 'comp'), 2), (('comp', 'science'), 2), (('in', 'the'), 2)]

Sentences with top five Bigrams

hi this is rakesh
this is rakesh from comp science
from comp science there are different teams participating in the event
in the event is rakesh particpiating a big question to everyone ?

Q4.code:
Irsdata = datasets.load_iris()

```
inp=irsdataset.data

out=irisdataset.target

x_trn,x_tst,y_trn,y_tst=train_test_split(inp,out,test_size=0.3)

mdl= KNeighborsClassifier(n_neighbors=5)

mdl.fit(x_trn,y_trn)

y_prd=mdl.predict(x_tst)

print("Acc: ",metrics.accuracy_score(y_tst,y_prd))

krng=range(1,50)

scrs=[]

for m in krng:

   knn5=KNeighborsClassifier(n_neighbors=m)

   knn5.fit(x_trn,y_trn)
   y_prd=knn5.predict(x_tst)

   scrs.append ( metrics  . accuracy_score(y_tst,y_prd))


plt.plot(krng,scrs)

plt.xlabel("k-value")

plt.ylabel("Accuracy")

plt.show()
```