

# Python Lab Assignment-3

Name: Saketh Garuda

ID: 16

## Objective:

The main objective for this lab assignment is to know the working of different classification models such Logistic Regression, Linear Discriminant Analysis, Support Vector Machine, KNN and natural language processing in python using NLTK. By using the above methods we met the following objectives,

- Compare contrast logistic regression and linear discriminant analysis
- Calculating best accuracy for the given dataset using the above models
- Applying SVC to different kernels such as Linear and RBF for predicting accuracy
- How KNN algorithm affecting the accuracy of the model

## Features:

The code snippets are executed and debugged for purpose of software environment. The code snippets are written in such way that they won't affect the performance of the system when they are executed in multiple environments.

## Configuration:

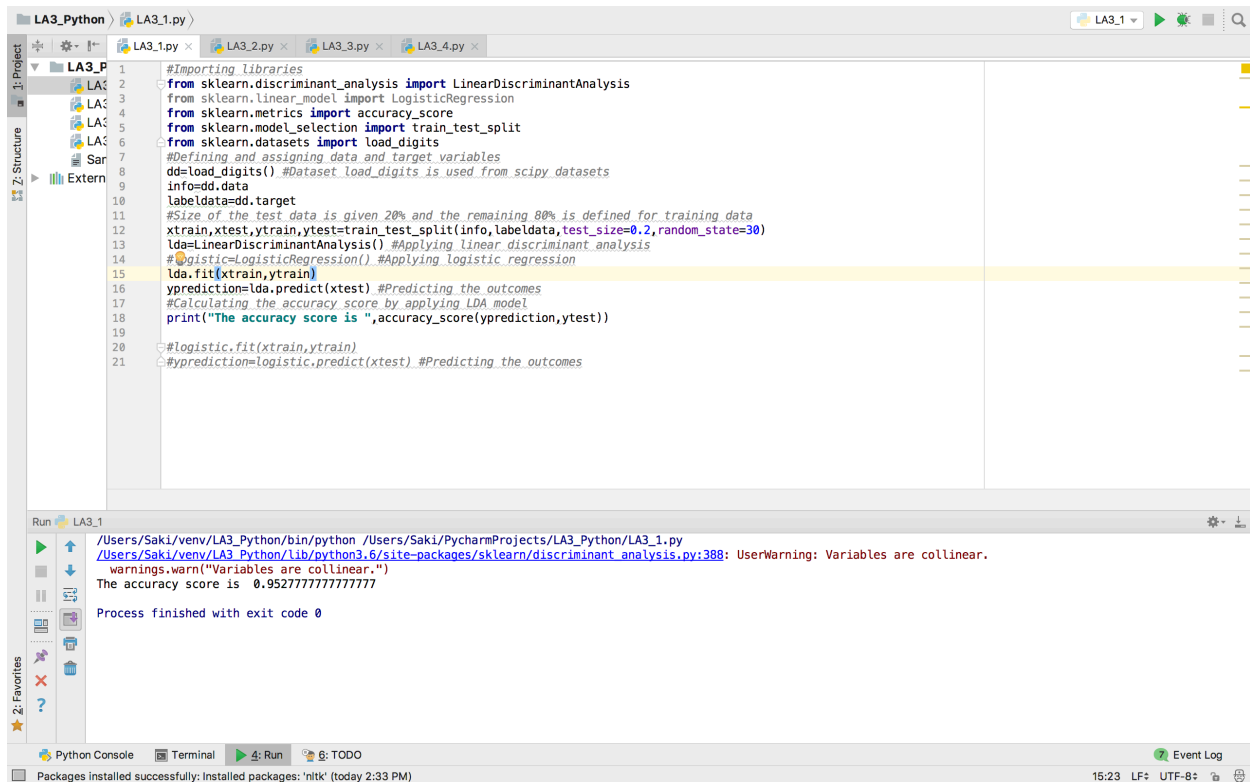
- PyCharm IDE
- Python 3.6.4
- NLTK

## Screenshots:

### 1) Choosing a dataset and making a prediction model using Linear Discriminant Analysis

## Output:

- Accuracy of the model using Linear Discriminant Analysis model



```
1 #Importing libraries
2 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
3 from sklearn.linear_model import LogisticRegression
4 from sklearn.metrics import accuracy_score
5 from sklearn.model_selection import train_test_split
6 from sklearn.datasets import load_digits
7 #Defining and assigning data and target variables
8 dd=load_digits() #Dataset load_digits is used from scipy datasets
9 info=dd.data
10 labeldata=dd.target
11 #Size of the test data is given 20% and the remaining 80% is defined for training data
12 xtrain,xtest,ytrain,ytest=train_test_split(info,labeldata,test_size=0.2,random_state=30)
13 lda=LinearDiscriminantAnalysis() #Applying Linear discriminant analysis
14 #Logistic=LogisticRegression() #Applying logistic regression
15 lda.fit(xtrain,ytrain)
16 yprediction=lda.predict(xtest) #Predicting the outcomes
17 #Calculating the accuracy score by applying LDA model
18 print("The accuracy score is ",accuracy_score(yprediction,ytest))
19
20 #Logistic.fit(xtrain,ytrain)
21 #yprediction=Logistic.predict(xtest) #Predicting the outcomes
```

Run LA3\_1

```
/Users/Saki/venv/LA3_Python/bin/python /Users/Saki/PycharmProjects/LA3_Python/LA3_1.py
/Users/Saki/venv/LA3_Python/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388: UserWarning: Variables are collinear.
warnings.warn("Variables are collinear.")
The accuracy score is  0.9527777777777777

Process finished with exit code 0
```

Python Console | Terminal | Run | TODO | Event Log

Packages installed successfully: Installed packages: 'nltk' (today 2:33 PM)

15:23 LF UTF-8

- Accuracy of the model using Logistic Regression model

```

1 #Importing libraries
2 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
3 from sklearn.linear_model import LogisticRegression
4 from sklearn.metrics import accuracy_score
5 from sklearn.model_selection import train_test_split
6 from sklearn.datasets import load_digits
7 #Defining and assigning data and target variables
8 d=load_digits() #Dataset load_digits is used from scipy.datasets
9 info=d.data
10 labeldata=d.target
11 #Size of the test data is given 20% and the remaining 80% is defined for training data
12 xtrain,xtest,ytrain,ytest=train_test_split(info,labeldata,test_size=0.2,random_state=30)
13 #def=LinearDiscriminantAnalysis() #Applying Linear discriminant analysis
14 logistic=LogisticRegression() #Applying logistic regression
15 #ld=fit(xtrain,ytrain)
16 #prediction=ld.predict(xtest) #Predicting the outcomes
17
18 logistic.fit(xtrain,ytrain)
19 yprediction=logistic.predict(xtest) #Predicting the outcomes
20 #Calculating the accuracy score by applying LDA model
21 print("The accuracy score is ",accuracy_score(prediction,ytest))

```

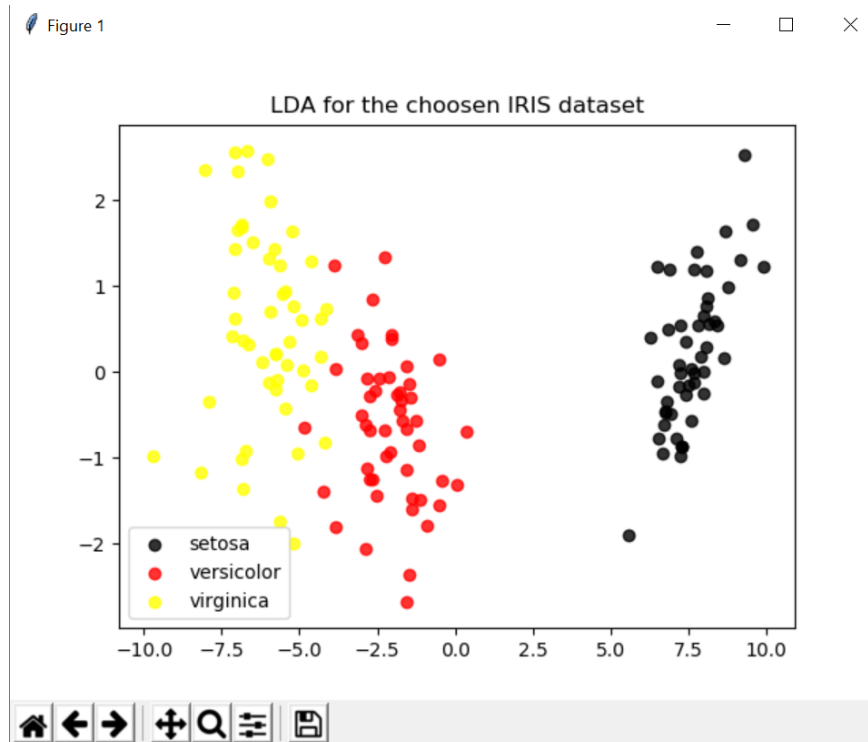
Run LA3\_1

/Users/Saki/venv/LA3\_Python/bin/python /Users/Saki/PycharmProjects/LA3\_Python/LA3\_1.py

The accuracy score is 0.9472222222222222

Process finished with exit code 0

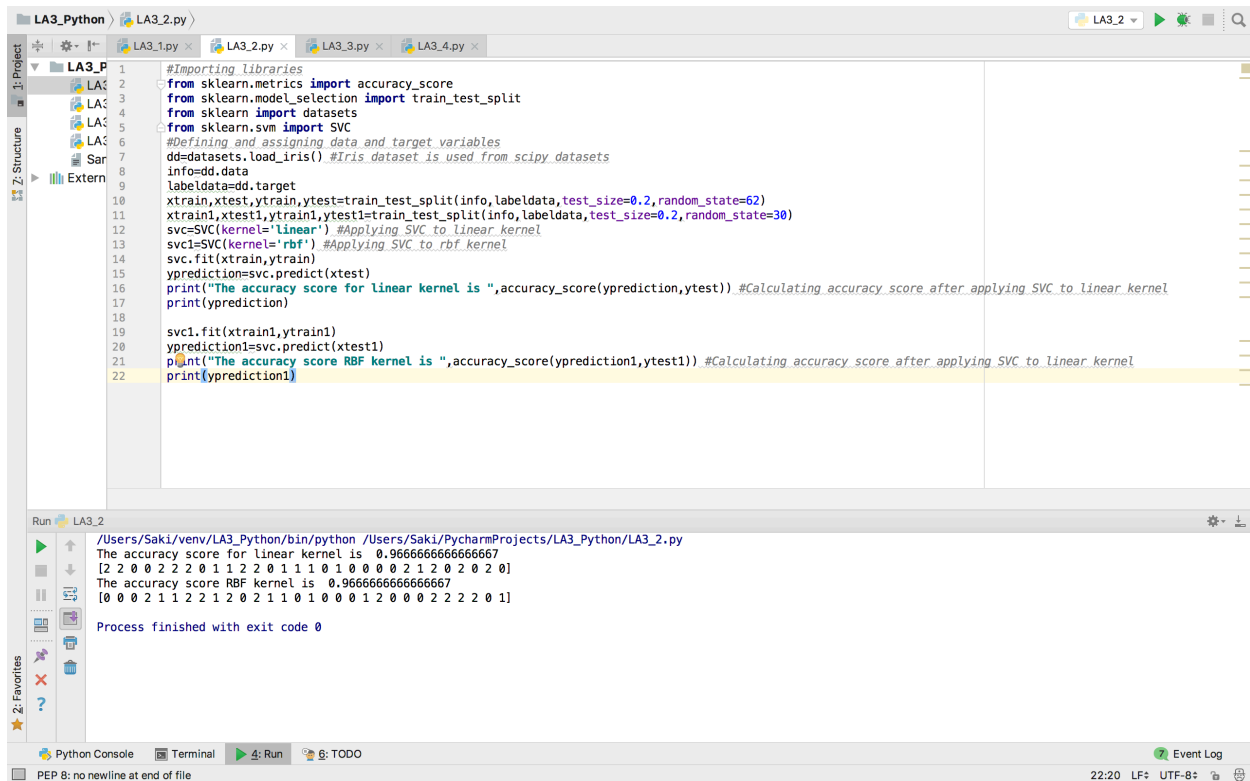
- Scatter plot for LDA model



2) Implementing Support Vector Machine classification model on the given dataset using Linear and RBF kernels.

### Output:

- Accuracy with linear kernel and accuracy with RBF kernel



The screenshot displays the PyCharm IDE interface. The main editor window shows a Python script named `LA3_2.py` with the following code:

```
1 #Importing libraries
2 from sklearn.metrics import accuracy_score
3 from sklearn.model_selection import train_test_split
4 from sklearn import datasets
5 from sklearn.svm import SVC
6 #Defining and assigning data and target variables
7 dd=datasets.load_iris() #Iris dataset is used from scipy datasets
8 info=dd.data
9 labeldata=dd.target
10 xtrain,xtest,ytrain,ytest=train_test_split(info,labeldata,test_size=0.2,random_state=62)
11 xtrain1,xtest1,ytrain1,ytest1=train_test_split(info,labeldata,test_size=0.2,random_state=30)
12 svc=SVC(kernel='linear') #Applying SVC to linear kernel
13 svc1=SVC(kernel='rbf') #Applying SVC to rbf kernel
14 svc.fit(xtrain,ytrain)
15 yprediction=svc.predict(xtest)
16 print("The accuracy score for linear kernel is ",accuracy_score(yprediction,ytest)) #Calculating accuracy score after applying SVC to linear kernel
17 print(yprediction)
18
19 svc1.fit(xtrain1,ytrain1)
20 yprediction1=svc1.predict(xtest1)
21 print("The accuracy score RBF kernel is ",accuracy_score(yprediction1,ytest1)) #Calculating accuracy score after applying SVC to linear kernel
22 print(yprediction1)
```

The Run window at the bottom shows the execution output:

```
/Users/Saki/venv/LA3_Python/bin/python /Users/Saki/PycharmProjects/LA3_Python/LA3_2.py
The accuracy score for linear kernel is  0.9666666666666667
[2 2 0 0 2 2 2 0 1 1 2 2 0 1 1 0 1 0 0 0 0 2 1 2 0 2 0]
The accuracy score RBF kernel is  0.9666666666666667
[0 0 0 2 1 1 2 2 1 2 0 2 1 1 0 1 0 0 0 1 2 0 0 2 2 2 0 1]
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8 and the time is 22:20.

3) Applying lemmatization on a text file and find the word frequency of bi-grams. Also return the top five bigrams from the frequency of bigrams and return the concatenated sentence with the list of top five bigrams.

## Output:

- The output file contains the lemmatized words and list of top five bigrams from the word frequency bigrams. The program also produces the concatenated sentence with the top five bigrams in the text content.

```
LA3_Python > LA3_3.py
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk import ngrams

#Reading file content
f = open('Sample1.txt', 'r')
filecontent = f.read()

wordlemmatizer=nltk.WordNetLemmatizer() #Lemmatizing the sentence using WordNetLemmatizer
words=word_tokenize(filecontent) #Tokenizes the given sentence to words
sentencetokenize=sent_tokenize(filecontent) #Tokenizes the text file to sentences
print(sentencetokenize)
samplelist=[]
for line2 in words:
    lwords=wordlemmatizer.lemmatize(line2)
    samplelist.append(lwords) #Appends every lemmatize word to list
print(samplelist)

print("Bigrams")
samplelist2=[]
bigrams=ngrams(words,2) #Bigram operation performed by using ngrams function
for a in bigrams:
    samplelist2.append(a)
print(samplelist2)

frequencydistribution=nltk.FreqDist(samplelist2)
wordfrequency=frequencydistribution.most_common() #Returns all the word frequency of bigrams
firstfive=frequencydistribution.most_common(5) #Returns the top 5 bigrams from bigrams list
print("Word frequency of bigrams")
print(wordfrequency)
print("Top 5 bigrams")
print(firstfive)
concatenatesentence=[]
for sentence in sentencetokenize:
    for x,y in samplelist2:
        for((word1,word2),count) in firstfive:
            if(x,y == word1,word2): #If the given sentence has the top most frequency bigrams then this loop is executed else it breaks
                concatenatesentence.append(sentencetokenize) #Each sentence with one of the top 5 bigram is appended to a list
for line2 in words

Run LA3_2
/Users/Saki/venv/LA3_Python/bin/python /Users/Saki/PycharmProjects/LA3_Python/LA3_2.py
The accuracy score for linear kernel is 0.9666666666666667
[2 2 0 0 2 2 2 0 1 1 2 2 0 1 1 1 0 1 0 0 0 0 2 1 2 0 2 0 2 0]
The accuracy score RBF kernel is 0.9666666666666667
[0 0 0 2 1 1 2 2 1 2 0 2 1 1 0 1 0 0 0 1 2 0 0 0 2 2 2 2 0 1]

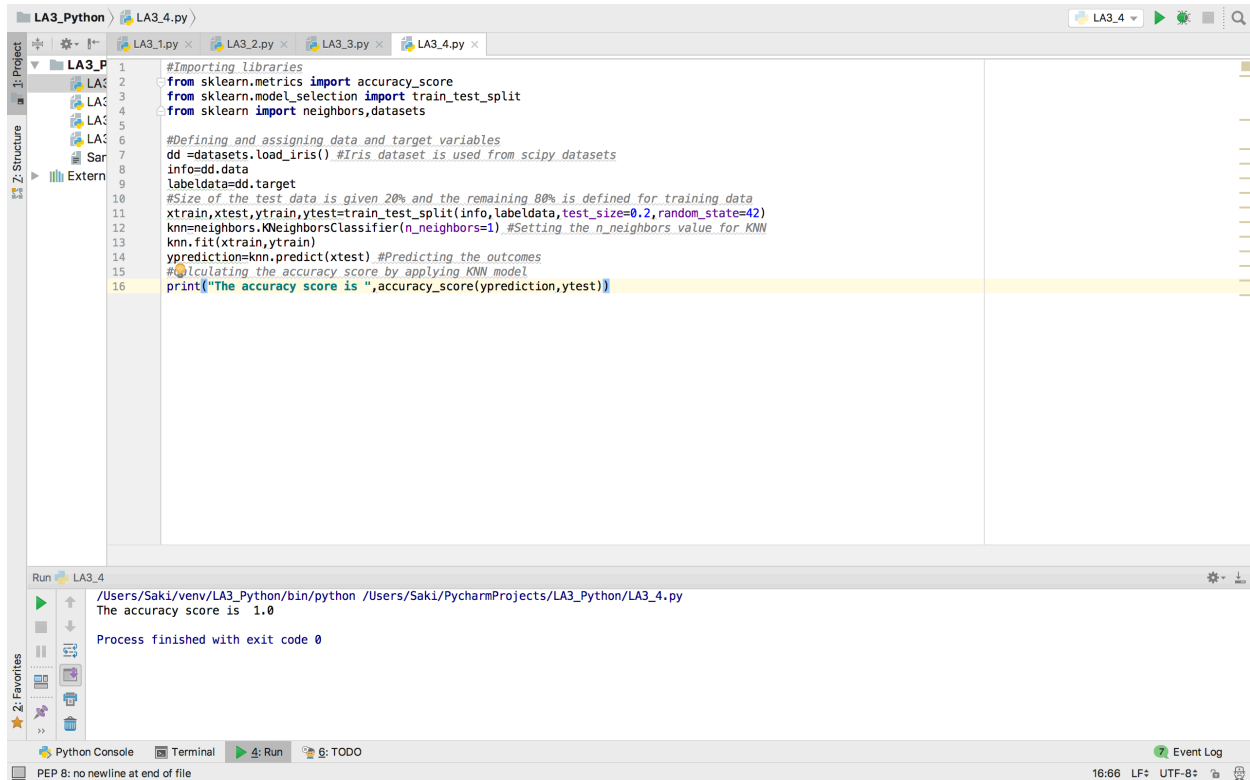
Python Console
Terminal
Run
TODO
Event Log
16:43 LF UTF-8

Run LA3_3
/Users/Saki/venv/LA3_Python/bin/python /Users/Saki/PycharmProjects/LA3_Python/LA3_3.py
['Much of what we know about the Venom F5 are the stats that Hennessey Special Vehicles, the small company behind the car, has already shared.', 'Teased for nearly a full']
Bigrams
[('Much', 'of'), ('of', 'what'), ('what', 'we'), ('we', 'know'), ('know', 'about'), ('about', 'the'), ('the', 'Venom'), ('Venom', 'F5'), ('F5', 'are'), ('are', 'the'), ('the', 'small'), ('small', 'company'), ('company', 'be')]
Word frequency of bigrams
[('the', 'small'), 4], [('behind', 'the'), 3], [('the', 'car'), 3], [('miles', 'per'), 3], [('per', 'hour'), 3], [(',', 'and'), 3], [('of', 'what'), 2], [('about', 'the'), 2], [('small', 'company'), 2], [('behind', 'the'), 2], [('the', 'car'), 2], [('miles', 'per'), 2], [('per', 'hour'), 2]]
Top 5 bigrams
[('the', 'small'), 4], [('behind', 'the'), 3], [('the', 'car'), 3], [('miles', 'per'), 3], [('per', 'hour'), 3]]
Concatenated sentence
['Much of what we know about the Venom F5 are the stats that Hennessey Special Vehicles, the small company behind the car, has already shared.', 'Teased for nearly a full']
Process finished with exit code 0
```

4) Report your views on the k nearest neighbor algorithm and how it affects the accuracy if value of k changes.

### Output:

- Accuracy calculated using knn model with neighbors=1



```
1 #Importing libraries
2 from sklearn.metrics import accuracy_score
3 from sklearn.model_selection import train_test_split
4 from sklearn import neighbors, datasets
5
6 #Defining and assigning data and target variables
7 dd = datasets.load_iris() #Iris dataset is used from scipy datasets
8 info = dd.data
9 labeldata = dd.target
10
11 #Size of the test data is given 20% and the remaining 80% is defined for training data
12 xtrain, xtest, ytrain, ytest = train_test_split(info, labeldata, test_size=0.2, random_state=42)
13 knn = neighbors.KNeighborsClassifier(n_neighbors=1) #Setting the n_neighbors value for KNN
14 knn.fit(xtrain, ytrain)
15 yprediction = knn.predict(xtest) #Predicting the outcomes
16 #Calculating the accuracy score by applying KNN model
17 print("The accuracy score is ", accuracy_score(yprediction, ytest))
```

Run LA3\_4

/Users/Saki/venv/LA3\_Python/bin/python /Users/Saki/PycharmProjects/LA3\_Python/LA3\_4.py

The accuracy score is 1.0

Process finished with exit code 0

Python Console Terminal 4: Run 6: TODO

PEP 8: no newline at end of file

16:06 LF UTF-8

- Accuracy calculated using knn model with neighbors=50

The screenshot displays the PyCharm IDE interface. The main editor window shows a Python script named `LA3_4.py` with the following code:

```
1 #Importing libraries
2 from sklearn.metrics import accuracy_score
3 from sklearn.model_selection import train_test_split
4 from sklearn import neighbors, datasets
5
6 #Defining and assigning data and target variables
7 dd = datasets.load_iris() #Iris dataset is used from scipy datasets
8 info = dd.data
9 labeldata = dd.target
10
11 #Size of the test data is given 20% and the remaining 80% is defined for training data
12 xtrain, xtest, ytrain, ytest = train_test_split(info, labeldata, test_size=0.2, random_state=42)
13 knn = neighbors.KNeighborsClassifier(n_neighbors=50) #Setting the n_neighbors value for KNN
14 knn.fit(xtrain, ytrain)
15 yprediction = knn.predict(xtest) #Predicting the outcomes
16 #Calculating the accuracy score by applying KNN model
17 print("The accuracy score is ", accuracy_score(yprediction, ytest))
```

The Run tool window at the bottom shows the execution output for `LA3_4`:

```
/Users/Saki/venv/LA3_Python/bin/python /Users/Saki/PycharmProjects/LA3_Python/LA3_4.py
The accuracy score is  0.9666666666666667
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8 and the line length is 16:66.



## Code Snippets:

### Solution for Problem 1:

Linear discriminant analysis LDA is applicable where the condition is mutually exclusive such that a dependent variable has two or more groups. Whereas in logistic regression the model is based on the combination of predictors and it provide only conditional distribution. Both LDA and Logistic regression models are based on linear-odd assumption but they estimate coefficients in different techniques.

### Code Snippet 1:

```
#Importing libraries
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_digits
#Defining and assigning data and target variables
dd=load_digits() #Dataset load_digits is used from scipy datasets
info=dd.data
labeldata=dd.target
#Size of the test data is given 20% and the remaining 80% is defined for training data
xtrain,xtest,ytrain,ytest=train_test_split(info,labeldata,test_size=0.2,random_state=30)
lda=LinearDiscriminantAnalysis() #Applying linear discriminant analysis
#logistic=LogisticRegression() #Applying logistic regression
a=lda.fit(xtrain,ytrain)
yprediction=lda.predict(xtest) #Predicting the outcomes

#a=logistic.fit(xtrain,ytrain)
#yprediction=logistic.predict(xtest) #Predicting the outcomes
#Calculating the accuracy score by applying LDA model
print("The accuracy score is ",accuracy_score(yprediction,ytest))

plt.figure()
colours = ['red', 'black', 'yellow']
for x, y, z in zip(colours, [0, 1, 2], dd):
    plt.scatter(a[labeldata == y, 0], a[labeldata == y, 1], alpha=.8, color=x,
               label=z)
plt.legend(loc='best', shadow=False, scatterpoints=1)
```

```
plt.title('LDA for the given dataset is')
plt.show()
```

## Solution for Problem 2:

The accuracy for both the models is different where accuracy score for linear kernel is 0.93333333 and accuracy for RBF kernel i.e. non-linear kernel is 1.0. Using linear when number of features is large whereas RBF kernel can be used when number of features is comparably smaller in size and expecting a predictive performance. For given iris dataset if the random state value is set higher for linear model than RBF kernel, then accuracy results of RBF model are best, vice-versa.

## Code Snippet 2:

```
#Importing libraries
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.svm import SVC
#Defining and assigning data and target variables
dd=datasets.load_iris() #Iris dataset is used from scipy datasets
info=dd.data
labeldata=dd.target
xtrain,xtest,ytrain,ytest=train_test_split(info,labeldata,test_size=0.2,random_state=62)
xtrain1,xtest1,ytrain1,ytest1=train_test_split(info,labeldata,test_size=0.2,random_state=30)
svc=SVC(kernel='linear') #Applying SVC to linear kernel
svc1=SVC(kernel='rbf') #Applying SVC to rbf kernel
svc.fit(xtrain,ytrain)
yprediction=svc.predict(xtest)
print("The accuracy score for linear kernel is ",accuracy_score(yprediction,ytest)) #Calculating accuracy score after applying SVC to linear kernel
print(yprediction)

svc1.fit(xtrain1,ytrain1)
yprediction1=svc1.predict(xtest1)
print("The accuracy score RBF kernel is ",accuracy_score(yprediction1,ytest1)) #Calculating accuracy score after applying SVC to linear kernel
print(yprediction1)
```

## Code Snippet 3:

```
#Importing libraries
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
```

```

from nltk import ngrams

#Reading file content
f = open('Sample1.txt', 'r')
filecontent = f.read();

wordlemmatizer=nltk.WordNetLemmatizer() #Lemmatizing the sentence using WordNetLemmatizer
words=word_tokenize(filecontent) #Tokenizes the given sentence to words
sentencetokenize=sent_tokenize(filecontent) #Tokenizes the text file to sentences
print(sentencetokenize)
samplelist=[]
for line2 in words:
    lwords=wordlemmatizer.lemmatize(line2)
    samplelist.append(lwords) #Appends every lemmatize word to list
print(samplelist)

print("Bigrams")
samplelist2=[]
bigrams=ngrams(words,2) #Bigram operation performed by using ngrams function
for a in bigrams:
    samplelist2.append(a)
print(samplelist2)

frequencydistribution=nltk.FreqDist(samplelist2)
wordfrequency=frequencydistribution.most_common() #Returns all the word frequency of bigrams
firstfive=frequencydistribution.most_common(5) #Returns the top 5 bigrams from bigrams list
print("Word frequency of bigrams")
print(wordfrequency)
print("Top 5 bigrams")
print(firstfive)
concatenatesentence=[]
for sentence in sentencetokenize:
    for x,y in samplelist2:
        for((word1,word2),count) in firstfive:
            if(x,y == word1,word2): #If the given sentence has the top most frequency bigrams then this loop
is executed else it breaks
                concatenatesentence.append(sentencetokenize) #Each sentence with one of the top 5 bigram
is appended to a list

print("Concatenated sentence") #The final concatenated sentence
print(max(concatenatesentence))

```

## Solution for Problem 4:

The accuracy is affected whenever the K value is increased or decreased as it affects the test data point that belongs to the same class or different class. When K increases i.e. K=50, the resolution is too fine which makes the model under fit and results in less accuracy. Whereas K decreases i.e. K=1, then the model is said to be over fit and provide correct classification which results in best accuracy.

## Code Snippet 4:

```
#Importing libraries
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import neighbors, datasets

#Defining and assigning data and target variables
dd = datasets.load_iris() #Iris dataset is used from scipy datasets
info=dd.data
labeldata=dd.target
#Size of the test data is given 20% and the remaining 80% is defined for training data
xtrain,xtest,ytrain,ytest=train_test_split(info,labeldata,test_size=0.2,random_state=42)
knn=neighbors.KNeighborsClassifier(n_neighbors=1) #Setting the n_neighbors value for KNN
knn.fit(xtrain,ytrain)
yprediction=knn.predict(xtest) #Predicting the outcomes
#Calculating the accuracy score by applying KNN model
print("The accuracy score is ",accuracy_score(yprediction,ytest))
```

## Deployment:

The code snippets are written using Python IDE and executed with the help of python 3.6.4 interpreter. Outputs are shown in the Python IDE console.

## Limitations:

The given code snippets doesn't have any limitations as they have met all rules and conditions.

## References:

- [http://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
- <http://scikit-learn.org/stable/modules/svm.html>
- <http://scikit-learn.org/stable/modules/neighbors.html>
- <http://www.nltk.org/book/ch01.html>
- <http://scikit-learn.org/0.16/modules/generated/sklearn.lda.LDA.html>