

Module1 Lab2

Team: 14

Professor: Yugyung Lee

Name: Sneha Mishra

Class ID: 19

Email: smccr@mail.umkc.edu

[MyGitHub](#)

Technical Partner:

Name: Raju Nekadi

Class ID: 20

Email: [rn8mh@mail.umkc.edu](mailto:rн8mh@mail.umkc.edu)

[GitHub](#)

The source code for this lab work can be found [here](#)

The available datasets formats can be found [here](#)

Objective

This Lab work is divided into 4 sub-parts, to get familiar with python and Machine learning. Few of the topics covered in this lab work are listed below:

1. Introduction to Machine Learning, terminology in Machine Learning, Regression, Clustering.

2. Classification, Evaluating model.
3. Natural Language Processing, Python NLTK package, Stemming, POS, Lemmatization, N-grams, Name Entity Recognition.

Features

The four problem statements can be found below. All the Datasets used for this Lab assignment can be found as follows:

- Cricket Dataset can be found [here](#)
- Wine Dataset can be found [here](#)
- Breast Cancer Dataset can be found [here](#)

Part 1:

Problem Statement:

1. Pick any dataset from the dataset sheet in the class sheet
 - a. plot how many of each category is available in your dataset (you can use seaborn library or matplotlib)
 - b. create one prediction model based on Naïve Bayes Classification and evaluate your model

Objective:

This Program uses the Wine Data from UCI Machine Learning Repository. It has 13 attributes and 1 Class. The purpose of this

program is to take Naive Bayes Classification and create Prediction Model and evaluate same and to plot the Count of Categories on map using seaborn and matplotlib.

Dataset used is Wine Dataset

The Code for the Naive Bayes problem can be found [here](#)

The Code for the Seaborn problem can be found [here](#)

Code Snippet:

Code snippet is attached
below:

```
mod1_lab2 > Part1_1.py
Project  Part2
mod1_lab2 [Python_Lesson]
Part1_1.py
Part1_2.py
Part2.py
winedata.csv
External Libraries
Scratches and Consoles

Part2.py x Part1_1.py x Part1_2.py x
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 # Creating Dataframe Wine using panda libraries
6 wine = pd.read_csv('winedata.csv')
7
8 # Print the first 5 rows from wine data using head() function
9 print(wine.head())
10
11 # Printing the unique Class from values from csv file
12 print('The wine data Contains following Unique Class Value:', wine['Class'].unique())
13
14 # Describe the wine data
15 print(wine.describe())
16
17 # Plot the Categories in wine data using seaborn and matplotlib
18 sns.countplot(wine['Class'], label="Count")
19 plt.show()
```

The screenshot shows the PyCharm IDE interface with the project 'mod1_lab2' open. The 'Part1_2.py' file is the active editor. The code reads a 'winedata.csv' file, checks for missing values, creates a correlation heatmap, drops columns 'Class' and 'Ash', and prints the first two rows of features and labels.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

# Creating Dataframe Wine using panda libraries
wine = pd.read_csv('winedata.csv')

# Check for Missing Values
print('Check if any column have missing value', wine.isnull().sum())

# Identifying the Correlation of column in wine dataframe using corr() and plotting same using seaborn heatmap
# The Pearson correlation coefficient is a widely used approach that measures the linear dependence between 2 variables.
# The correlation coefficient ranges from -1 to 1. A correlation of 1 is a total positive correlation, a correlation
# of -1 is a total negative correlation and a correlation of 0 is non-linear correlation.
corr = wine.corr()
sns.heatmap(corr, cmap="YlGnBu", annot=True)
plt.show()

# print correlation
print(corr)

# As we can Column ASH is least correlated we can remove that column from our analysis and get our Features
# Features
X = wine.drop(['Class', 'Ash'], axis=1)
print('\nThe Features are:\n', X.head(2))

# Labels
Y = wine.iloc[:, :1]
print('\nThe Labels:\n', Y.head(2))
```

The screenshot shows the PyCharm IDE interface with the project 'mod1_lab2' open. The 'Part2.py' file is the active editor. The code performs train-test split, fits a Naive Bayes classifier, predicts test set results, and plots a scatter plot of true vs predicted values.

```
print('Wine_Lab2', wine.head(2))

# Train-Test Split
X_train, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)

# Fitting Naive Bayes Classification to the Training set with linear kernel
nvclassifier = GaussianNB()
nvclassifier.fit(X_train, Y_train)

# Predicting the Test set results
Y_pred = nvclassifier.predict(X_test)

# Accuracy of NavesBayes on Training Sets
print('\nAccuracy of Naive Bayes classifier on Training Set: {:.2f}'.format(nvclassifier.score(X_train, Y_train)))

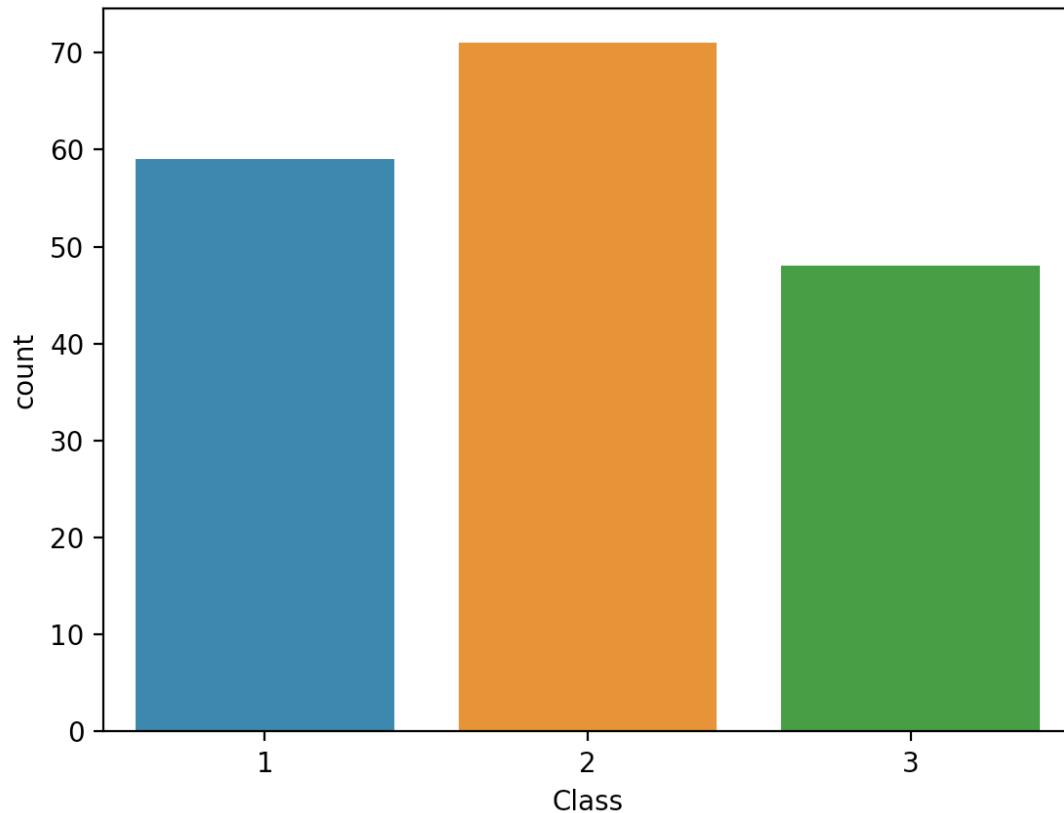
# Accuracy of NavesBayes on Testing Sets
print('\nAccuracy of Naive Bayes Classifier on Test Set: {:.2f}'.format(nvclassifier.score(X_test, Y_test)))

# print the accuracy score of predicted and actual values on test set
print('\nAccuracy of the Naive Bayes Classification is on Test Data Prediction: ', metrics.accuracy_score(Y_test, Y_pred))

# Plot the Test Prediction , Test Value on Graph
plt.scatter(Y_test, Y_pred)
plt.xlabel('True Values')
plt.ylabel('Predictions')
plt.show()
```

Output/ Workflow:

Figure 1



```
Run: seaborn-winedata x
/Users/snehamishra/PycharmProjects/Test/venv/bin/python /Users/snehamishra/Desktop/UMKC/Fall2018/Python/Python-DeepLearning_Fall2
      Class Alcohol MalicAcid ...   Hue OD280/OD315ofdilutedwines Proline
0       1    14.23    1.71 ...   1.04            3.92        1065
1       1    13.20    1.78 ...   1.05            3.40        1050
2       1    13.16    2.36 ...   1.03            3.17        1185
3       1    14.37    1.95 ...   0.86            3.45        1480
4       1    13.24    2.59 ...   1.04            2.93        735
[5 rows x 14 columns]
The wine data Contains following Unique Class Value: [1 2 3]
      Class ...   Proline
count  178.000000 ... 178.000000
mean   1.938202 ... 746.893258
std    0.775035 ... 314.907474
min    1.000000 ... 278.000000
25%   1.000000 ... 500.500000
50%   2.000000 ... 673.500000
75%   3.000000 ... 985.000000
max    3.000000 ... 1680.000000
[8 rows x 14 columns]
Process finished with exit code 0
```

```
: NaiveBayes-WineData x
: /Users/snehamishra/PycharmProjects/Test/venv/bin/python /Users/snehamishra/Desktop/UMKC/Fall2018/Python/Python-DeepLearning_Fa
: Check if any column have missing value Class
:          0
: Alcohol      0
: MalicAcid    0
: Ash          0
: AlcalinityofAsh 0
: Magnesium    0
: TotalPhenols 0
: Flavanoids   0
: Nonflavanoidphenols 0
: Proanthocyanins 0
: ColorIntensity 0
: Hue          0
: OD280/OD315ofdilutedwines 0
: Proline      0
: dtype: int64
objc[10135]: Class FIFinderSyncExtensionHost is implemented in both /System/Library/PrivateFrameworks/FinderKit.framework/Versi
Class ... Proline
```

```
dtype: int64
objc[10135]: Class FIFinderSyncExtensionHost is implemented in both /System/Library/PrivateFrameworks/FinderKit.framework/Versi
          Class ... Proline
Class      1.000000 ... -0.633717
Alcohol    -0.328222 ... 0.643720
MalicAcid   0.437776 ... -0.192011
Ash        -0.049643 ... 0.223626
AlcalinityofAsh 0.517859 ... -0.440597
Magnesium  -0.209179 ... 0.393351
TotalPhenols -0.719163 ... 0.498115
Flavanoids   -0.847498 ... 0.494193
Nonflavanoidphenols 0.489109 ... -0.311385
Proanthocyanins -0.499130 ... 0.330417
ColorIntensity 0.265668 ... 0.316100
Hue        -0.617369 ... 0.236183
OD280/OD315ofdilutedwines -0.788230 ... 0.312761
Proline     -0.633717 ... 1.000000

[14 rows x 14 columns]

The Features are:
Alcohol  MalicAcid ... OD280/OD315ofdilutedwines  Proline
0       14.23    1.71 ...           3.92      1065
1       13.20    1.78 ...           3.40      1050

[2 rows x 12 columns]
```

```
The Labels:
          Class
0         1
1         1

Accuracy of Naive Bayes GaussianNB classifier on Training Set: 0.96

Accuracy of Naive Bayes GausianNbClassifier on Test Set: 0.97

Accuracy of the Naive Bayes GaussianNB Classification is on Test Data Prediction: 0.9722222222222222
/Users/snehamishra/PycharmProjects/Test/venv/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataConversionWarning:
y = column_or_1d(y, warn=True)

Process finished with exit code 0
```

Figure 1

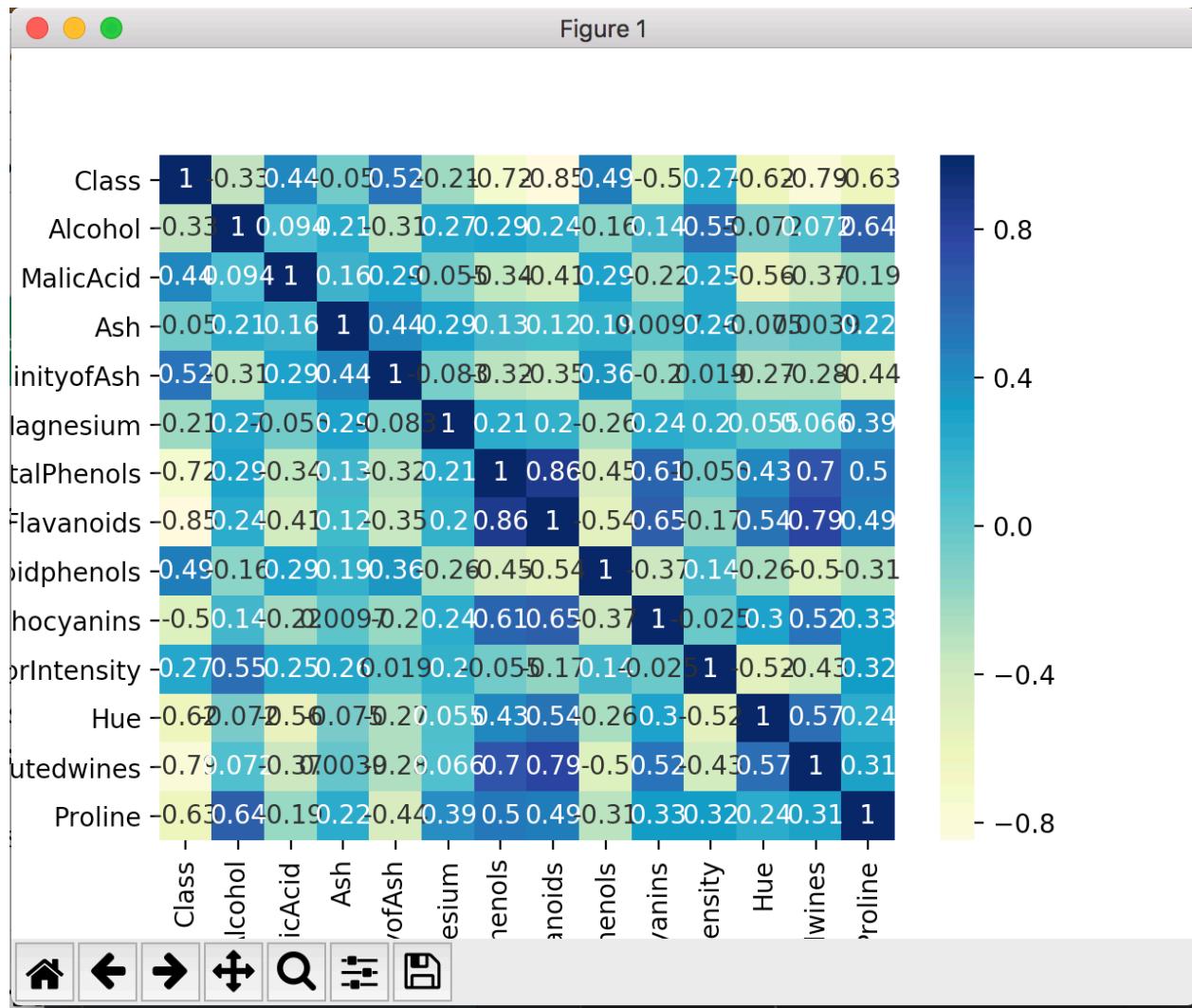
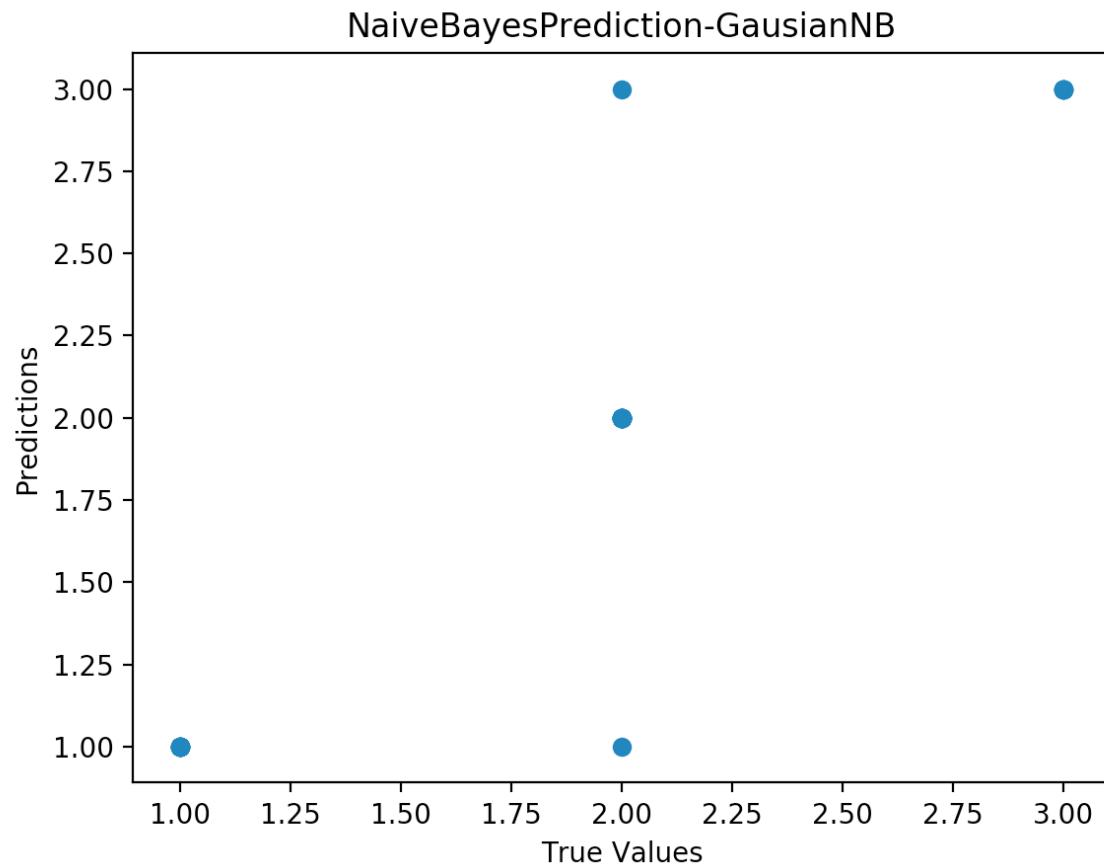




Figure 1



Parameter:

We generated dependent and independent variable from dataset and also train and test datasets with test size of 20%.

Conclusion:

Accuracy of Naive Bayes GaussianNB on train set is less than test set. Thus, the test data prediction is 97.2%.

Part 2:

Problem Statement:

2. Implement Support Vector Machine classification:

- a. Apply SVC with kernel “poly” degree =4
- b. Apply SVC with “rbf” kernel
- c. change gamma and C parameters in the model to see how the result may change
- d. Report the accuracy of the model on both models separately and with which parameters you got better result.

Objective:

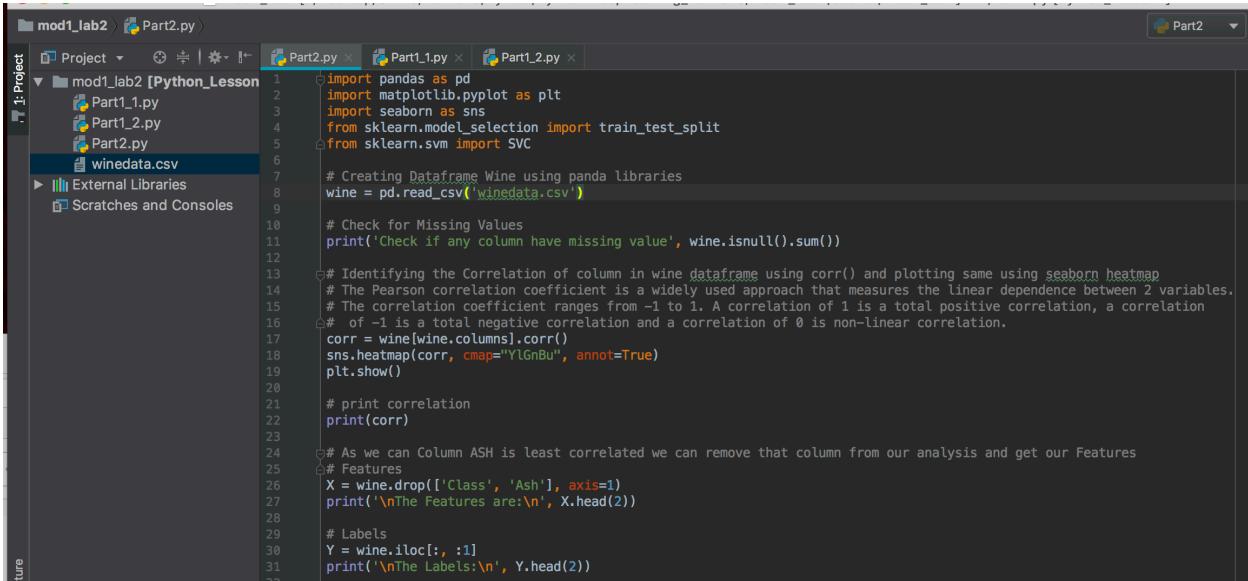
This Program uses the Wine Data from UCI Machine Learning Repository. The is having 13 attribute and 1 Class. The purpose of this program is to apply SVC with kernel “poly” degree =4 and with “rbf” kernel to see how the result change.

Dataset used is Wine Dataset

The Code for the problem can be found [here](#)

Code Snippet:

Code snippet is attached
below:

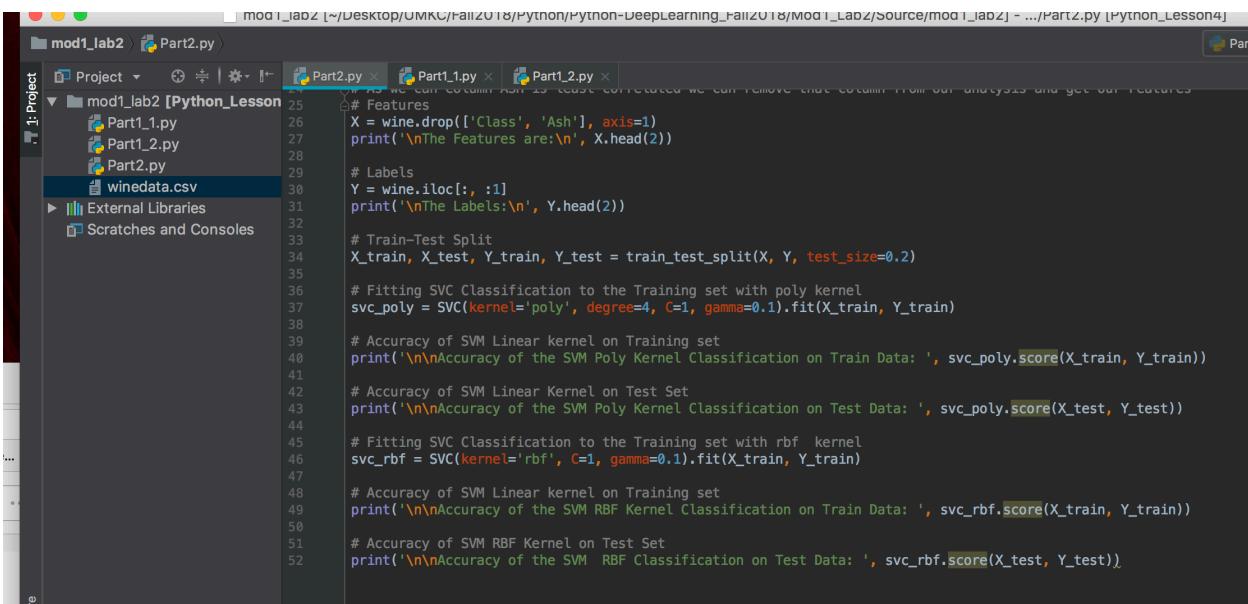


PyCharm IDE showing the project structure and the content of Part2.py. The code reads a wine dataset, checks for missing values, creates a correlation heatmap, drops the 'Ash' column, and prints the first two rows of features and labels.

```
mod1_lab2 [Python_Lesson]
Part1_1.py
Part1_2.py
Part2.py
wine.csv
External Libraries
Scratches and Consoles
```

```
Part2.py
Part1_1.py
Part1_2.py
```

```
1 # Import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 from sklearn.model_selection import train_test_split
5 from sklearn.svm import SVC
6
7 # Creating Dataframe Wine using panda libraries
8 wine = pd.read_csv('wine.csv')
9
10 # Check for Missing Values
11 print('Check if any column have missing value', wine.isnull().sum())
12
13 # Identifying the Correlation of column in wine dataframe using corr() and plotting same using seaborn heatmap
14 # The Pearson correlation coefficient is a widely used approach that measures the linear dependence between 2 variables.
15 # The correlation coefficient ranges from -1 to 1. A correlation of 1 is a total positive correlation, a correlation
16 # of -1 is a total negative correlation and a correlation of 0 is non-linear correlation.
17 corr = wine[wine.columns].corr()
18 sns.heatmap(corr, cmap="YlGnBu", annot=True)
19 plt.show()
20
21 # print correlation
22 print(corr)
23
24 # As we can Column ASH is least correlated we can remove that column from our analysis and get our Features
25 # Features
26 X = wine.drop(['Class', 'Ash'], axis=1)
27 print('\n\nThe Features are:\n', X.head(2))
28
29 # Labels
30 Y = wine.iloc[:, :1]
31 print('\n\nThe Labels:\n', Y.head(2))
```



Continuation of the code in Part2.py. It performs a train-test split, fits an SVC classifier with a poly kernel, and prints accuracy metrics for both training and test sets. It then fits another SVC classifier with an rbf kernel and prints accuracy metrics for both training and test sets.

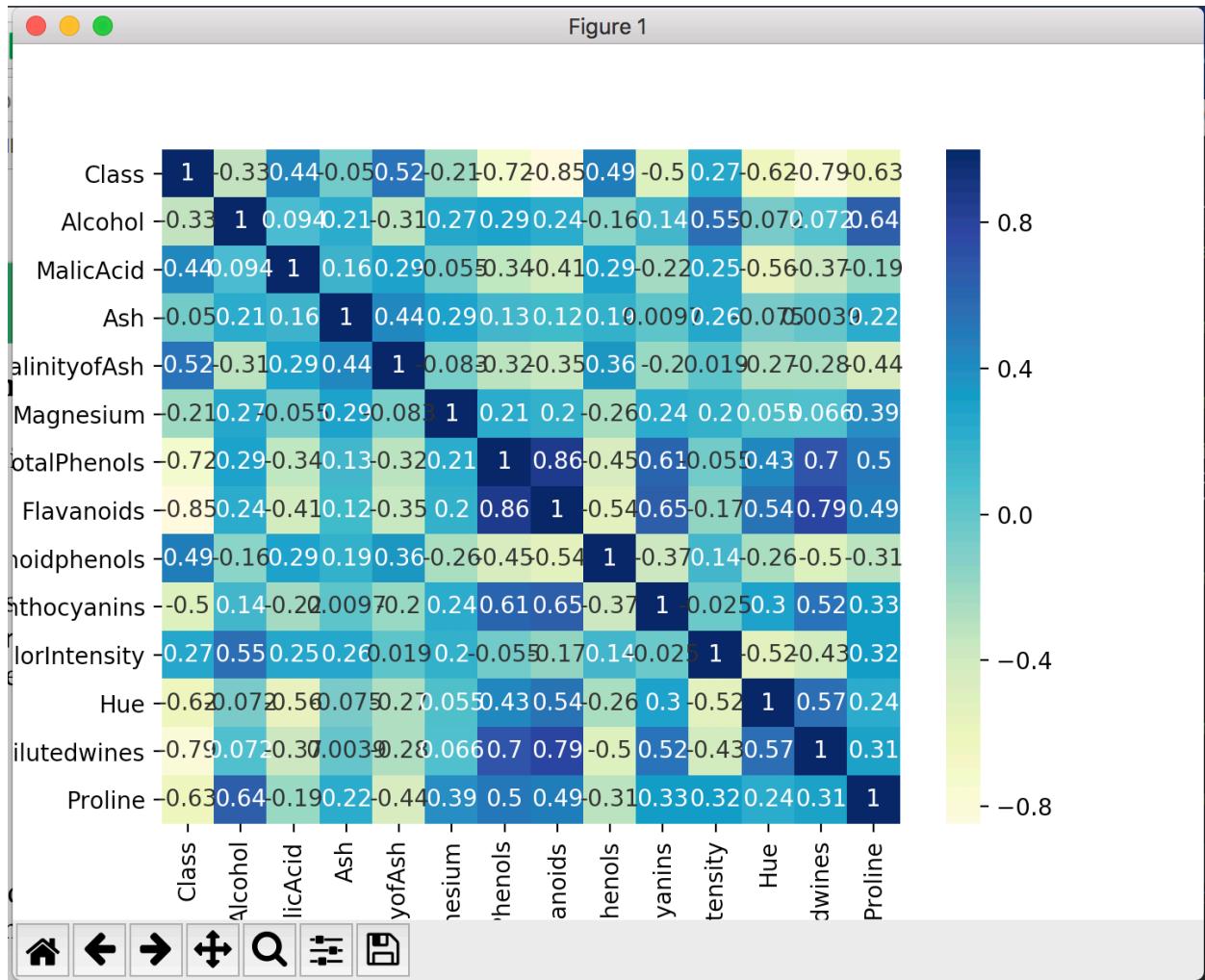
```
mod1_lab2 [Python_Lesson]
Part1_1.py
Part1_2.py
Part2.py
wine.csv
External Libraries
Scratches and Consoles
```

```
Part2.py
Part1_1.py
Part1_2.py
```

```
25 # As we can Column ASH is least correlated we can remove that column from our analysis and get our Features
26 # Features
27 X = wine.drop(['Class', 'Ash'], axis=1)
28 print('\n\nThe Features are:\n', X.head(2))
29
30 # Labels
31 Y = wine.iloc[:, :1]
32 print('\n\nThe Labels:\n', Y.head(2))
33
34 # Train-Test Split
35 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
36
37 # Fitting SVC Classification to the Training set with poly kernel
38 svc_poly = SVC(kernel='poly', degree=4, C=1, gamma=0.1).fit(X_train, Y_train)
39
40 # Accuracy of SVM Linear kernel on Training set
41 print('\n\nAccuracy of the SVM Poly Kernel Classification on Train Data: ', svc_poly.score(X_train, Y_train))
42
43 # Accuracy of SVM Linear Kernel on Test Set
44 print('\n\nAccuracy of the SVM Poly Kernel Classification on Test Data: ', svc_poly.score(X_test, Y_test))
45
46 # Fitting SVC Classification to the Training set with rbf kernel
47 svc_rbf = SVC(kernel='rbf', C=1, gamma=0.1).fit(X_train, Y_train)
48
49 # Accuracy of SVM Linear kernel on Training set
50 print('\n\nAccuracy of the SVM RBF Kernel Classification on Train Data: ', svc_rbf.score(X_train, Y_train))
51
52 # Accuracy of SVM RBF Kernel on Test Set
53 print('\n\nAccuracy of the SVM RBF Classification on Test Data: ', svc_rbf.score(X_test, Y_test))
```

Output/ Workflow:

Figure 1



```

SupportVectorMachine-WineData x
Check if any column have missing value Class          0
Alcohol          0
MalicAcid        0
Ash              0
AlcalinityofAsh  0
Magnesium        0
TotalPhenols     0
Flavanoids       0
Nonflavanoidphenols 0
Proanthocyanins 0
ColorIntensity   0
Hue              0
OD280/OD315ofdilutedwines 0
Proline          0
dtype: int64
      Class    ...    Proline
Class  1.000000  ... -0.633717
Alcohol -0.328222  ...  0.643720
MalicAcid  0.437776  ... -0.192011
Ash     -0.049643  ...  0.223626
AlcalinityofAsh  0.517859  ... -0.440597
Magnesium -0.209179  ...  0.393351
TotalPhenols -0.719163  ...  0.498115
Flavanoids -0.847498  ...  0.494193
Nonflavanoidphenols  0.489109  ... -0.311385
Proanthocyanins -0.499130  ...  0.330417
ColorIntensity  0.265668  ...  0.316100
Hue      -0.617369  ...  0.236183
OD280/OD315ofdilutedwines -0.788230  ...  0.312761
Proline   -0.633717  ...  1.000000

```

```

[14 rows x 14 columns]

The Features are:
   Alcohol  MalicAcid  ...  OD280/OD315ofdilutedwines  Proline
0    14.23      1.71  ...                  3.92      1065
1    13.20      1.78  ...                  3.40      1050

[2 rows x 12 columns]

The Labels:
   Class
0    1
1    1
/Users/snehamishra/PycharmProjects/Test/venv/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataConversionWarning:
y = column_or_1d(y, warn=True)

Accuracy of the SVM Poly Kernel C=1, gamma=0.1 Classification on Train Data: 1.0

Accuracy of the SVM Poly Kernel C=1, gamma=0.1 Classification on Test Data: 0.8888888888888888

Accuracy of the SVM Poly Kernel C=2, gamma=2 Classification on Train Data: 1.0

Accuracy of the SVM Poly Kernel C=2, gamma=2 Classification on Test Data: 0.8888888888888888

Accuracy of the SVM RBF Kernel C=1, gamma=0.1 Classification on Train Data: 1.0

Accuracy of the SVM RBF C=1, gamma=0.1 Classification on Test Data: 0.3333333333333333

Accuracy of the SVM RBF Kernel C=2, gamma=1 Classification on Train Data: 1.0

Accuracy of the SVM RBF C=2, gamma=1 Classification on Test Data: 0.3055555555555556

Process finished with exit code 0

```

Parameter:

None

Conclusion:

Accuracy of SVM Poly kernel & SVM RBF kernel with C=1 and gamma=0.1 is same for Train data, whereas for test data Poly kernel has better accuracy than rbf kernel.

Part 3:

Problem Statement:

3. Write a program in which take an Input file. Use the simple approach below to summarize a text file:
 - a. Read a file
 - b. Apply lemmatization on the words
 - c. Apply the bigram on the text
 - d. Calculate the word frequency (bi-gram frequency) of the words (bi-grams)
 - f. Choose top five bi-grams that have been repeated most
 - g. Go through the original text that you had in the file
 - h. Find all the sentences with those most repeated bi-grams
 - i. Extract those sentences and concatenate
 - j. Enjoy the summarization

Objective:

The aim of this problem is to take input data file and use NLTK library to apply Lemmatization , bigram on text, find top 5 bigrams and find sentence which used to create top most repeated bigrams.

Solution/ Approach:

To apply NLTK library function we used the sample text inputted from user at run time. The we uses various libraries like nltk.stem,Collection,inflect,string, and re to create word token, sentence token,Lemmatization, and bigrams and derive meaningful information from it.

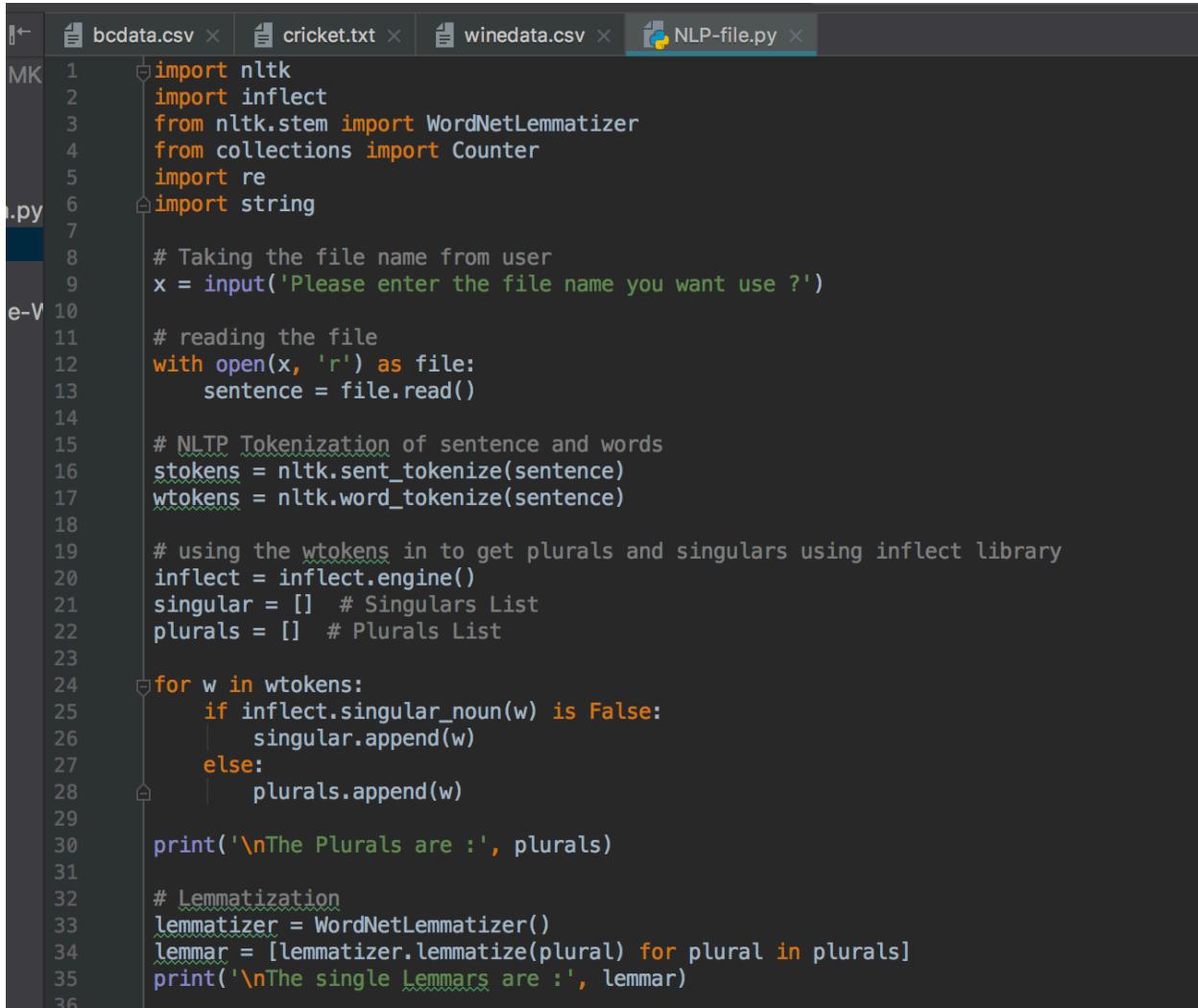
Dataset used is entered by User, in our case we gave 2 input texts - cricket.txt and bcdata.csv

The Code for the problem can be found [here](#)

Code Snippet:

Code snippet is attached

below:



The screenshot shows a code editor window with multiple tabs at the top: bcdata.csv, cricket.txt, winedata.csv, and NLP-file.py (which is the active tab). The code in the editor is as follows:

```
1 import nltk
2 import inflect
3 from nltk.stem import WordNetLemmatizer
4 from collections import Counter
5 import re
6 import string
7
8 # Taking the file name from user
9 x = input('Please enter the file name you want use ?')
10
11 # reading the file
12 with open(x, 'r') as file:
13     sentence = file.read()
14
15 # NLTP Tokenization of sentence and words
16 stokens = nltk.sent_tokenize(sentence)
17 wtokens = nltk.word_tokenize(sentence)
18
19 # using the wtokens in to get plurals and singulars using inflect library
20 inflect = inflect.engine()
21 singular = [] # Singulars List
22 plurals = [] # Plurals List
23
24 for w in wtokens:
25     if inflect.singular_noun(w) is False:
26         singular.append(w)
27     else:
28         plurals.append(w)
29
30 print('\nThe Plurals are :', plurals)
31
32 # Lemmatization
33 lemmatizer = WordNetLemmatizer()
34 lembar = [lemmatizer.lemmatize(plural) for plural in plurals]
35 print('\nThe single Lemmas are :', lembar)
```

```
bcdata.csv x cricket.txt x winedata.csv x NLP-file.py x
MK 36     # Bigrams using Sentence Tokenizer
37     # we need to remove the regular expression from sentences
38     non Speaker = re.compile('[A-Za-z]+: (.*)')
39
40
41     # Function to remove special character and create bigrams
42     def extract_phrases(text, phrase_counter, length):
43         for sent in nltk.sent_tokenize(text):
44             strip Speaker = non Speaker.match(sent)
45             if strip Speaker is not None:
46                 sent = strip Speaker.group(1)
47                 words = nltk.word_tokenize(sent)
48                 for phrase in nltk.ngrams(words, length):
49                     if all(word not in string.punctuation for word in phrase):
50                         phrase_counter[phrase] += 1
51
52     # using the counter for bigrams
53     phrase_counter = Counter()
54
55     # Now taking the sentence tokenizer and generating phrase Counter
56     for stoken in stokens:
57         extract_phrases(stoken, phrase_counter, 2)
58
59     # Printing the Bigram frequency
60     print('\nThe frequency of bigrams are :', phrase_counter)
61
62     most_common_phrases = phrase_counter.most_common(5)
63     for k, v in most_common_phrases:
64         print('\n5 most repeated bigrams are :{0: <5}'.format(v), k)
65
66     # searching sentences for 5 most repeated bi grams
67     bigram = [] # list of 5 most repeated bigram
68
69     for k, v in most_common_phrases:
70         s = ' '.join(k) # Converting bigram into string as it can be used for search sentence
71         bigram.append(s)
72
73         fsentence = '' # Will sentence for top 5 most repeated bigram
74         for b in bigram: # looping bigram string one by one
75             for s in stokens: # looping sentence one by one to search bigram string
76                 if b in s:
77                     fsentence = fsentence + s
78
79     print('\n\nThe final combined sentence are below:\n', fsentence)
```

Output/ Workflow:

```
NLP-file x
/Users/snehamishra/PycharmProjects/Test/venv/bin/python /Users/snehamishra/Desktop/UMKC/Fall2018/Python/Python-DeepLearning_Fall2018/Mod1_Lab2/Source/mod1_lab2/NLP-file.py
Please enter the file name you want to use ?TamilNadu.txt
The Plurals are : ['billions', 'rights', 'millions', 'salaries', 'is', 'is', 'farmers', 'snakes', 'decades', 'states', 'rights', 'conditions', 'farmers', 'demands', 'farmers', 'was'
The single Lemmars are : ['billion', 'right', 'million', 'salary', 'is', 'is', 'farmer', 'snake', 'decade', 'state', 'right', 'condition', 'farmer', 'demand', 'farmer', 'wa', 'ha',
The frequency of bigrams are : Counter({('Tamil', 'Nadu'): 3, ('in', 'the'): 3, ('to', 'be'): 3, ('the', 'stadium'): 3, ('to', 'the'): 2, ('to', 'get'): 2, ('Super', 'Kings'): 2,
5 most repeated bigrams are :3   ('Tamil', 'Nadu')
5 most repeated bigrams are :3   ('in', 'the')
5 most repeated bigrams are :3   ('to', 'be')
5 most repeated bigrams are :3   ('the', 'stadium')
5 most repeated bigrams are :2   ('to', 'the')

The final combined sentence are below:
For decades, two Indian states, Tamil Nadu and Karnataka, have vigorously disputed water rights to the Cauvery River. Pressed by drought conditions, farmers in Tamil Nadu have received
Process finished with exit code 0
```

```
NLP-file x
/Users/snehamishra/PycharmProjects/Test/venv/bin/python /Users/snehamishra/Desktop/UMKC/Fall2018/Python/Python-DeepLearning_Fall2018/Mod1_Lab2/Source/mod1_lab2/NLP-file.py
Please enter the file name you want to use ?nccdata.csv
The Plurals are : ['clump_Thickness', 'bare_nuclei', 'mitoses', 'Class']
The single Lemmars are : ['clump_Thickness', 'bare_nuclei', 'mitoses', 'Class']
The frequency of bigrams are : Counter({('Class', '1000025,5,1,1,2,1,3,1,1,2'): 1, ('1000025,5,1,1,2,1,3,1,1,2', '1002945,5,4,4,5,7,10,3,2,1'): 1,
5 most repeated bigrams are :1   ('Class', '1000025,5,1,1,2,1,3,1,1,2')
5 most repeated bigrams are :1   ('1000025,5,1,1,2,1,3,1,1,2', '1002945,5,4,4,5,7,10,3,2,1,2')
5 most repeated bigrams are :1   ('1002945,5,4,4,5,7,10,3,2,1,2', '1015425,3,1,1,1,2,2,3,1,1,2')
5 most repeated bigrams are :1   ('1015425,3,1,1,1,2,2,3,1,1,2', '1016277,6,8,8,1,3,4,3,7,1,2')
5 most repeated bigrams are :1   ('1016277,6,8,8,1,3,4,3,7,1,2', '1017023,4,1,1,3,2,1,3,1,1,2')

The final combined sentence are below:

Process finished with exit code 0
```

Parameter:

None

Evaluation:

The NLTK library produced the below results and generates Lemmatization,bigram.

Conclusion:

Looking at above results we can conclude that NLTK are very efficient in performing NLP functions and works very well with any kind of data.

Part 4:

Problem Statement:

4. Report your views on the k nearest neighbor algorithm when we change the K how it will affect the accuracy. Provide a good justification for the changes of the accuracy when we change the amount of K.
For example: compare the accuracy when K=1 and K is a big number like 50, why the accuracy will change

Objective:

The aim of this problem is to use KNN algorithm and perform evaluation using the different value of K parameter.

Solution/ Approach:

KNN (K nearest neighbor) is one of the simplest Supervised algorithm, mainly used for Classification. It classifies data points based on how its neighbors are classified. Its stores all cases/ data point and classify cases/data point based on similarity measures.

To apply KNN algorithm we used the breast cancer dataset and generate the accuracy for train,test datasets and prediction. We used various sklearn libraries like pandas,matplotlib, model_selection,metrics,neighbor.

Characteristics of a KNN Model are:

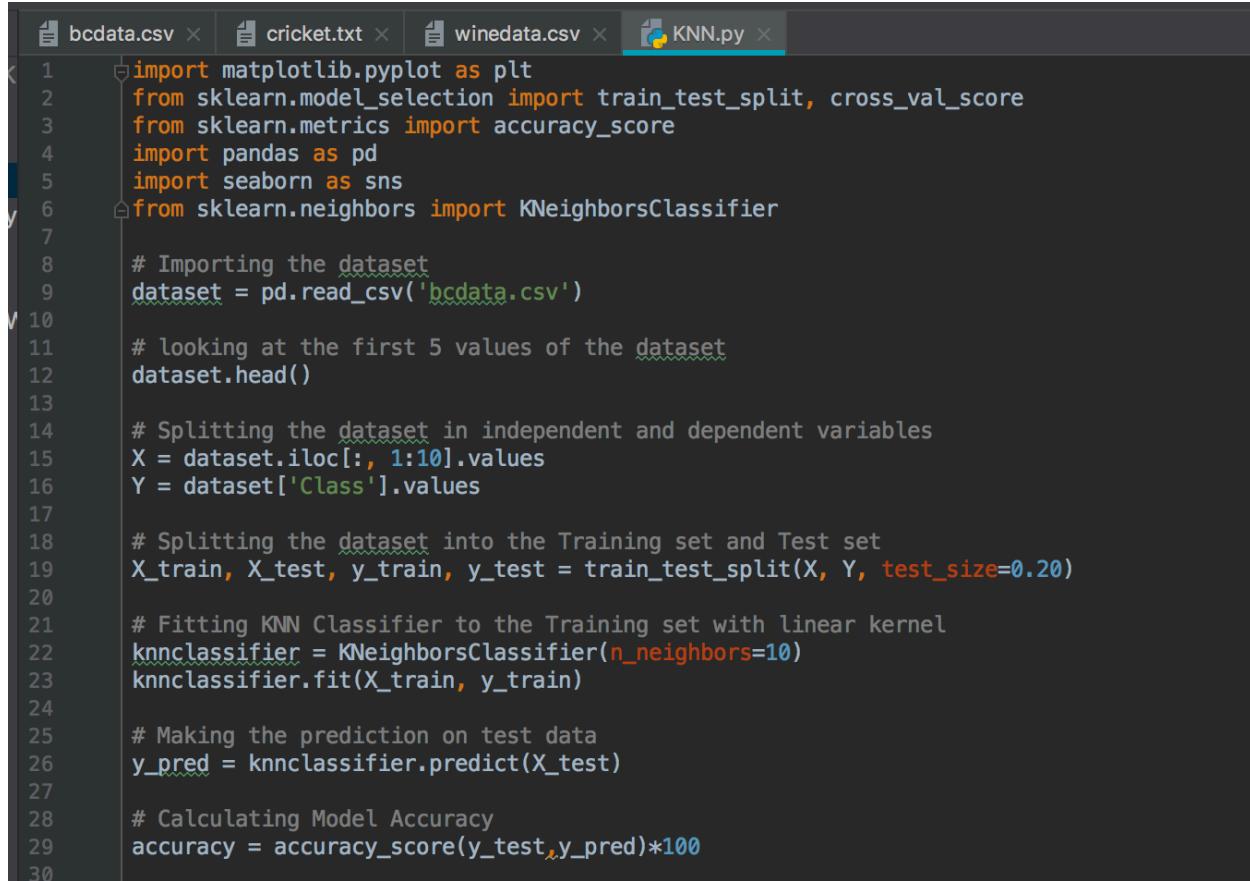
- Fast to create model because it simply stores data
- Slow to predict because many distance calculations
- Can require lots of memory if data set is large

Dataset used is Breast cancer dataset

The Code for the problem can be found [here](#)

Code Snippet:

Code snippet is attached below:



```
1 import matplotlib.pyplot as plt
2 from sklearn.model_selection import train_test_split, cross_val_score
3 from sklearn.metrics import accuracy_score
4 import pandas as pd
5 import seaborn as sns
6 from sklearn.neighbors import KNeighborsClassifier
7
8 # Importing the dataset
9 dataset = pd.read_csv('bcdata.csv')
10
11 # looking at the first 5 values of the dataset
12 dataset.head()
13
14 # Splitting the dataset in independent and dependent variables
15 X = dataset.iloc[:, 1:10].values
16 Y = dataset['Class'].values
17
18 # Splitting the dataset into the Training set and Test set
19 X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)
20
21 # Fitting KNN Classifier to the Training set with linear kernel
22 knnclassifier = KNeighborsClassifier(n_neighbors=10)
23 knnclassifier.fit(X_train, y_train)
24
25 # Making the prediction on test data
26 y_pred = knnclassifier.predict(X_test)
27
28 # Calculating Model Accuracy
29 accuracy = accuracy_score(y_test, y_pred)*100
30
```

```
K 30      # Printing the accuracy score of prediction and train test data
K 31      print('\nAccuracy of KNN classifier on Training Set: {:.2f}'
K 32          .format(knnclassifier.score(X_train, y_train)))
K 33      print('\nAccuracy of KNN classifier on Test Set: {:.2f}'
K 34          .format(knnclassifier.score(X_test, y_test)))
K 35      print('\nAccuracy of Prediction :', str(round(accuracy, 2)) + ' %.')
K 36
K 37
K 38      # Now lets understand the Accuracy of KNN on various value of neighbor by creating list of K for KNN
K 39      k_list = list(range(1, 50, 1))
K 40
K 41      # creating list of cv scores
K 42      cv_scores = []
K 43
K 44      # perform 10-fold cross validation
K 45      for k in k_list:
K 46          knn = KNeighborsClassifier(n_neighbors=k)
K 47          scores = cross_val_score(knn, X_train, y_train, cv=10, scoring='accuracy')
K 48          cv_scores.append(scores.mean())
K 49
K 50      # printing klist and cv_score
K 51      print(k_list)
K 52      print(cv_scores)
K 53
K 54      # changing to misclassification error
K 55      MSE = [1 - x for x in cv_scores]
K 56
K 57      # Calculating best value of K
K 58      best_k = k_list[MSE.index(min(MSE))]
K 59      print("The optimal number of neighbors is %d." % best_k)
K 60
K 61      plt.figure(figsize=(15, 10))
K 62      plt.title('The optimal number of neighbors', fontsize=20, fontweight='bold')
K 63      plt.xlabel('Number of Neighbors K', fontsize=15)
K 64      plt.ylabel('Misclassification Error', fontsize=15)
K 65      sns.set_style("whitegrid")
K 66      plt.plot(k_list, MSE)
K 67      plt.show()
```

Output/ Workflow:

We import various sklearn libraries and imported the breast cancer data.

We generated dependent and independent variable from dataset and also train and test datasets with test size of 20%.

Next we fitted this data into KNN algorithm with value of K =10 and made prediction on test data.

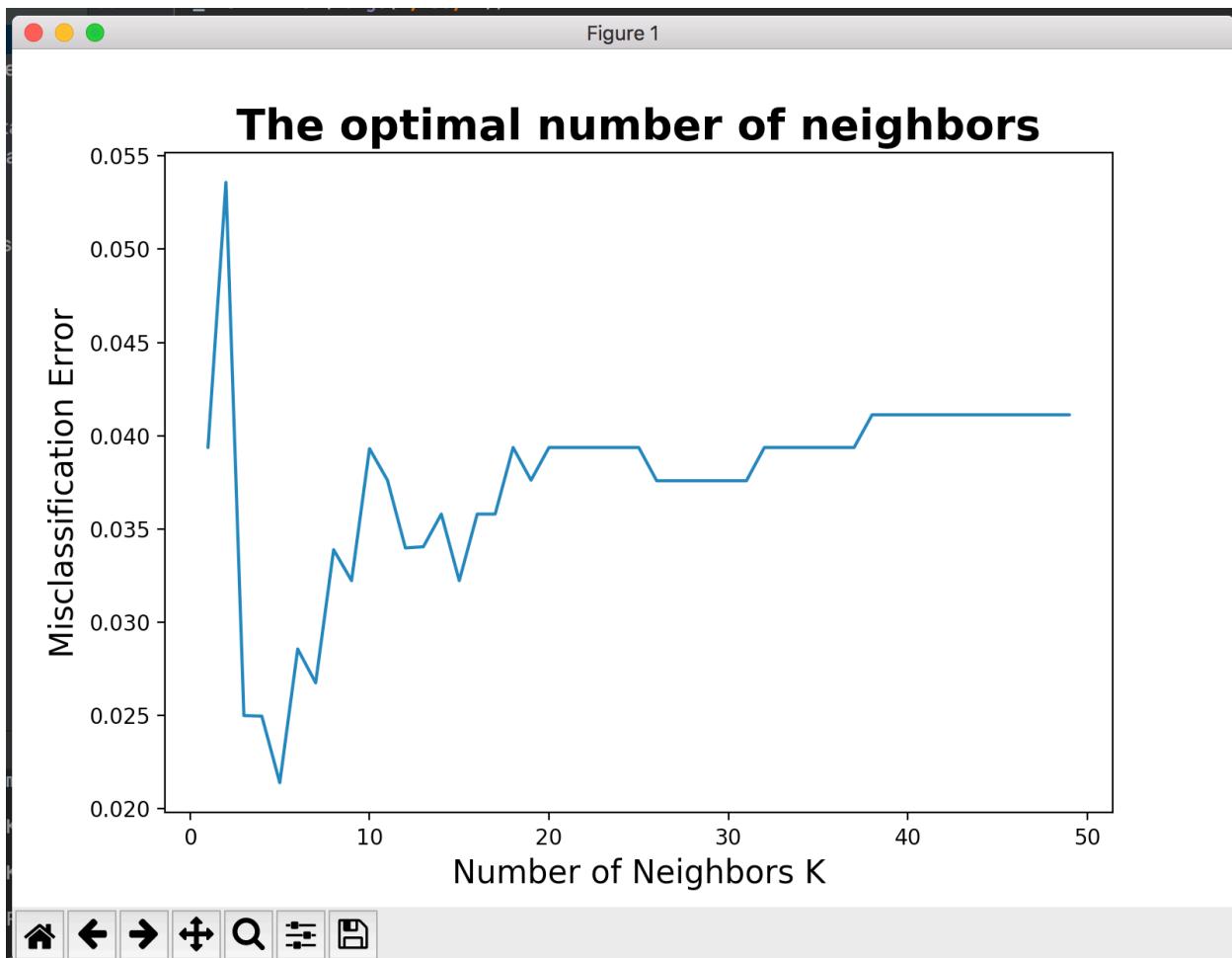
In order evaluate the KNN for different value of K we have used the cross_val_score to generate scores for the K=1 to 50. Then

we calculated the misclassification error to find the optimum value of K neighbors.

Evaluation:

For evaluating the KNN for K = 1 to 50 we have used scores and misclassification error and generated the plot using seaborn and matplotlib.

As we can see that from below graph as value of K increases the accuracy of model decrease. The best value of k for breast cancer dataset is 7.



```
KNN x
/Users/snehamishra/PycharmProjects/Test/venv/bin/python /Users/snehamishra/Desktop/UMKC/Fall2018/Python/Python-DeepLearning_Fall2018/Mod1_Lab2/Source/mo
Accuracy of KNN classifier on Training Set: 0.97
Accuracy of KNN classifier on Test Set: 0.96
Accuracy of Prediction : 96.43 %.
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
0.9606402369560264, 0.9464137616769195, 0.9750244930508087, 0.9750558213716107, 0.9786283891547051, 0.9714519252677147, 0.9732701070858967, 0.966124971
The optimal number of neighbors is 5.

Process finished with exit code 0
```

Parameter:

K from 1 to 50

Evaluation:

For evaluating the KNN for $K = 1$ to 50 we have used scores and misclassification error and generated the plot using seaborn and matplotlib.

As we can see that from below graph as value of K increases the accuracy of model decrease. The best value of k for breast cancer dataset is 7.

Conclusion:

The Choice of K is very critical $K = 1$ means noise will have higher influence on result . Large value of K makes computation expensive and higher misclassification error.

References:

1. <https://stackoverflow.com/questions/1936466/beautifulsoup-grab-visible-webpage-text>
2. https://www.researchgate.net/post/how_to_scrape_text_from_webpage_using_beautifulsoup_python
3. <http://www.intelligent-d2.com/python/web-scraping-saving-file-using-python-beautifulsoup-requests/>