CSEE5590 Python and Deep Learning Programming
Lab Assignment #1

Submitted By:

Name: Saran Akkiraju
Class ID: 1
Name: Koushik Katakam
Class ID: 11

Objective:

The main objective of this lab assignment is to get the understanding of python and introduction to machine learning algorithms in both supervised and unsupervised techniques.

Technologies/IDE's used:

- Python 3.7
- Pycharm IDE

Workflow:

The workflow of the any machine learning algorithm in this lab assignment is as follows:

- Choose a dataset
- Pre - process the data
- Split into train and test
- Fitting the model
- Metrics calculation

Program 1:

Suppose you have a list of tuples as follows:

[(‘John’, (‘Physics’, 80)) , (‘ Daniel’, (‘Science’, 90)), (‘John’, (‘Science’, 95)), (‘Mark’,(‘Maths’, 100)), (‘Daniel’, (’History’, 75)), (‘Mark’, (‘Social’, 95))]

Create a dictionary with keys as names and values as list of (subjects, marks) in sorted order.

{John: [(‘Physics’, 80), (‘Science’, 95)] Daniel: [(’History’, 75), (‘Science’, 90)] Mark: [(‘Maths’, 100), (‘Social’, 95)]}

Python code:

```
for student in student_list:
    if student[0] not in student_dict:
        student_dict[student[0]] = list()
        student_dict[student[0]].append(student[1])
    else:
        student_dict[student[0]].append(student[1])
```

From the above code,

We have initialized the empty dictionary and then using the conditional and a looping statement, appending the values to the dictionary in a sorted order.

Output:

```
John : [('Physics', 80), ('Science', 95)]
Daniel : [('History', 75), ('Science', 90)]
Mark : [('Maths', 100), ('Social', 95)]
```

Program 2:

Given a string, find the longest substrings without repeating characters along with the length as a tuple Input:

"pwwkew" Output: (wke,3), (kew,3)

Python code:

```
def long_substr(str):
    temp = ""
    dict = {}
    for j in range(len(str)):
        for i in range(j,len(str)):
            if not(str[i] in temp):
                temp += str[i]
            else :
                dict[temp] = len(temp)
                temp = ''
                break
    max_val = max(dict.values())
    list1=[]
    for key, val in dict.items():
        if max_val == val:
            list1.append((key, val))
```

- Firstly, an empty string is created to store all the non-repeating characters.
- Now, iterate through the input, if the particular character is not available in the empty string just append the character.

Output:

```
[('wke', 3), ('kew', 3)]

Process finished with exit code 0
```

Program 3:

Write a python program to create any one of the following management systems.

1. Airline Booking Reservation System (e.g. classes Flight, Person, Employee, Passenger etc.)

2. Library Management System (e.g. Student, Book, Faculty, Department etc.)

Library management System with 5 classes

- Person - Base class
- Student - Inherited class (single inheritance)
- Librarian - Inherited class (single)
- Book - Contains the private variable.
- Borrow_book - Multiple inheritance

Python code:

The Person class is as follows:

```
class Person:
    def __init__(self,name,email):
        self.name = name
        self.email = email

    def display(self):
        print("Name: ", self.name)
        print("Email: ", self.email)
```

Inheritance by Student class:

```
# Inheritance concept where student is inheriting the Person class


class Student(Person):
    StudentCount = 0

    def __init__(self,name,email,student_id):
        Person.__init__(self,name,email)
        self.student_id = student_id
        Student.StudentCount +=1
```

Super call:

```
class Librarian(Person):
    StudentCount = 0

    def __init__(self,name,email,employee_id):
        # super call where Librarian class is inheriting the Person class
        super().__init__(name,email)
        self.employee_id = employee_id
```

private member:

```
class Book():
    __numBooks = 0   # private member
    def __init__(self,book_name,author,book_id):
        self.book_name = book_name
        self.author = author
        self.book_id = book_id
        Book.__numBooks += 1   # keeps track of which student or staff has book checked
```

Multiple inheritance:

```
class Borrow_Book(Student,Book):

    def __init__(self,name,email,student_id,book_name,author,book_id):
        Student.__init__(self,name,email,student_id)
        Book.__init__(self,book_name,author,book_id)
```

Instances:

```
# creating instances of all classes
Records = []
Records.append(Student('xyz','xyz@gmail.com',123))
Records.append(Librarian('abc','xyz@gmail.com',789))
Records.append(Book('davinci code','leo',123456))
Records.append(Borrow_Book('def','pqr@gmail.com',456,'wings of fire','kalam',67890))
```

Output:

```
Student Details:
Name:  xyz
Email:  xyz@gmail.com
Student Id:  123


Employee Details:
Name:  abc
Email:  xyz@gmail.com
Employee Id:  789


Book Details
Book_Name:  davinci code
Author:  leo
Book_ID:  123456
```

```
Borrowed Book Details:
Student Details:
Name:  def
Email:  pqr@gmail.com
Student Id:  456
Book Details
Book_Name:  wings of fire
Author:  kalam
Book_ID:  67890


Total Number of Students: 2

Process finished with exit code 0
```

Program 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.

Dataset: sklearn.datasets.load_boston

Dataset link: https://github.com/scikit-learn/scikit-learn/blob/7813f7efb5b2012412888b69e73d76f2df2b50b6/sklearn/datasets/data/boston_house_prices.csv

Python code:

Loading the dataset using pandas:

```
import numpy as np # linear algebra
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston
house = load_boston()
bos = pd.DataFrame(house.data)
bos.columns = house.feature_names
bos['Price']=house.target
print(bos.head())
bos.describe()
```

Correlation in between features:

```
correl = bos.corr()
print(correl['Price'].sort_values(ascending=False)[:6], '\n')
print(correl['Price'].sort_values(ascending=False)[-6:])
```

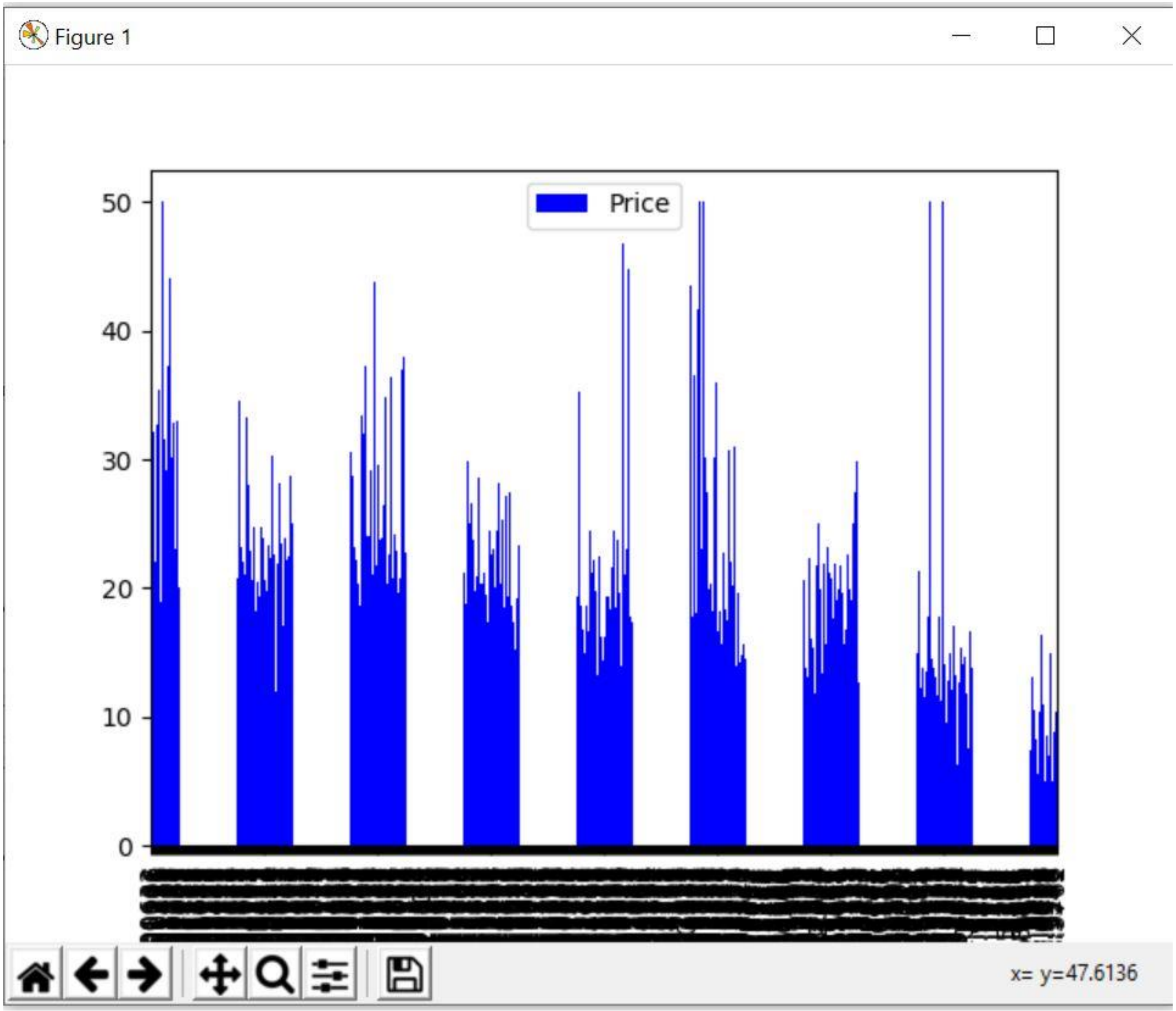
Output:

```
      CRIM    ZN  INDUS  CHAS    NOX  ...    TAX  PTRATIO       B  LSTAT  Price
0  0.00632  18.0   2.31   0.0  0.538  ...  296.0     15.3  396.90   4.98   24.0
1  0.02731   0.0   7.07   0.0  0.469  ...  242.0     17.8  396.90   9.14   21.6
2  0.02729   0.0   7.07   0.0  0.469  ...  242.0     17.8  392.83   4.03   34.7
3  0.03237   0.0   2.18   0.0  0.458  ...  222.0     18.7  394.63   2.94   33.4
4  0.06905   0.0   2.18   0.0  0.458  ...  222.0     18.7  396.90   5.33   36.2

[5 rows x 14 columns]
Price    1.000000
RM       0.695360
ZN       0.360445
B        0.333461
DIS      0.249929
CHAS     0.175260
Name: Price, dtype: float64

CRIM      -0.388305
NOX       -0.427321
TAX       -0.468536
INDUS     -0.483725
PTRATIO   -0.507787
LSTAT     -0.737663
Name: Price, dtype: float64
```

Creating the pivot plots:

```
quality_pivot = bos.pivot_table(index='CRIM', values='Price', aggfunc=np.median)
quality_pivot.plot(kind='bar', color='blue')
plt.show()

quality_pivot = bos.pivot_table(index='INDUS', values='Price', aggfunc=np.median)
quality_pivot.plot(kind='bar', color='blue')
plt.show()

quality_pivot = bos.pivot_table(index='NOX', values='Price', aggfunc=np.median)
quality_pivot.plot(kind='bar', color='blue')
plt.show()

quality_pivot = bos.pivot_table(index='AGE', values='Price', aggfunc=np.median)
quality_pivot.plot(kind='bar', color='blue')
plt.show()

quality_pivot = bos.pivot_table(index='CRIM', values='Price', aggfunc=np.median)
quality_pivot.plot(kind='bar', color='blue')
plt.show()

quality_pivot = bos.pivot_table(index='RAD', values='Price', aggfunc=np.median)
quality_pivot.plot(kind='bar', color='blue')
plt.show()
```

Output of pivot plots:



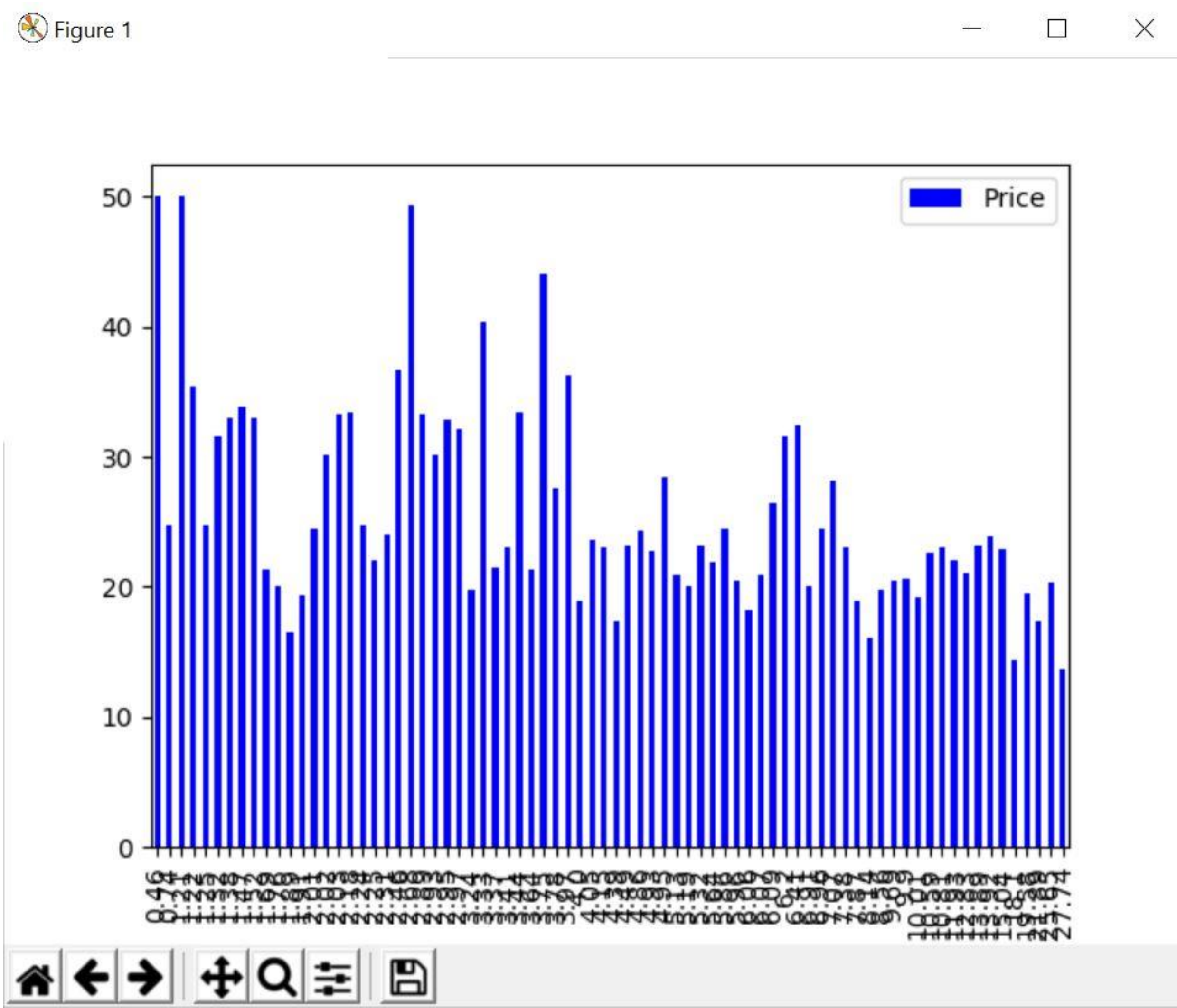
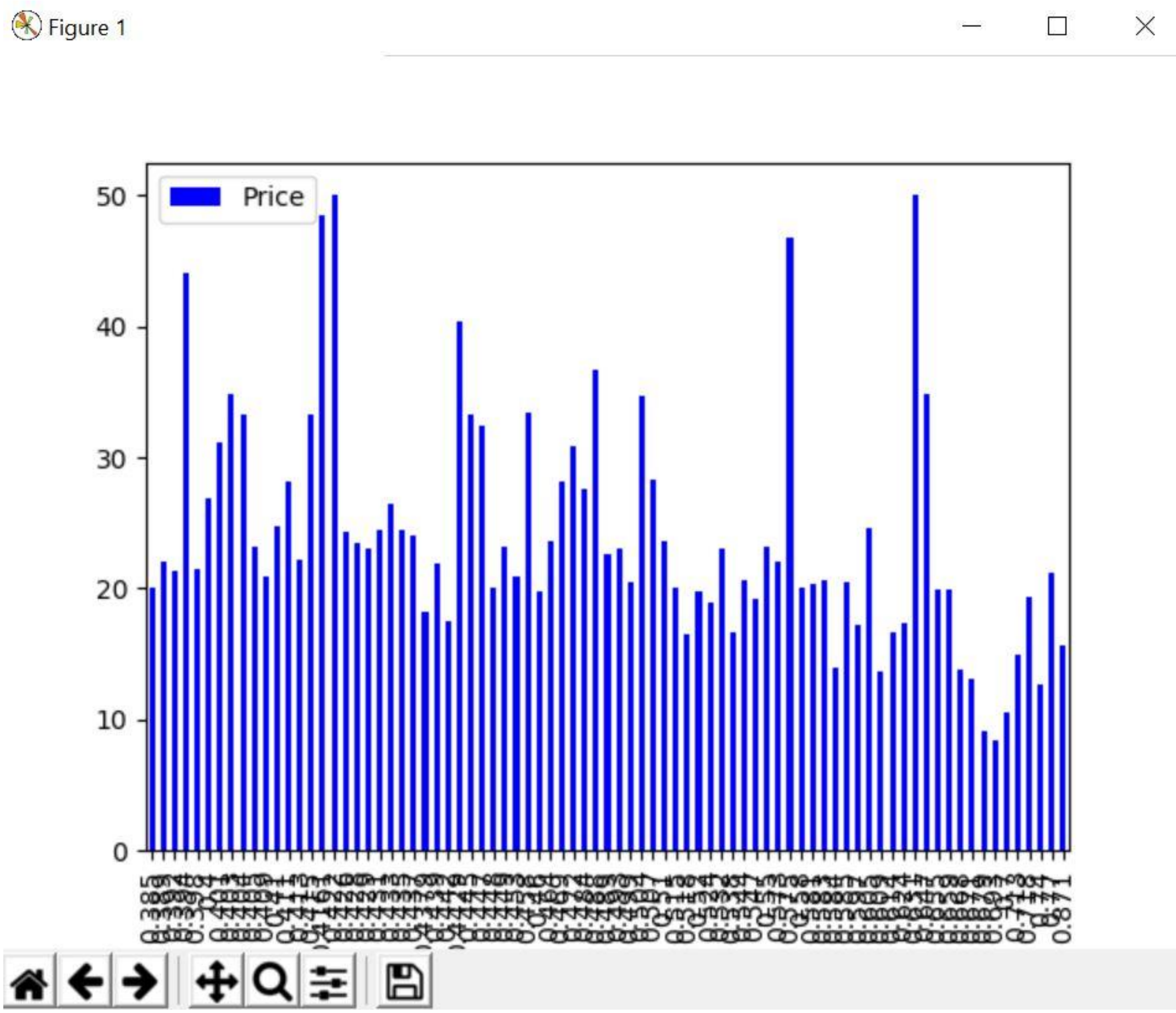


Figure 1
Price
50
40
30
20
10
0



Fitting the model and calculating the scores:

```
from sklearn import linear_model
lr1 = linear_model.LinearRegression()
model = lr1.fit(x_train, y_train)

print('r2 is: ', model.score(x_test, y_test))
prediction = model.predict(x_test)
from sklearn.metrics import mean_squared_error
print('rmse: ', mean_squared_error(y_test, prediction))
```

Output before eliminating the features:

```
r2 is:  0.7789207451814417
rmse:  18.495420122448404

Process finished with exit code 0
```

Output after eliminating the features:

I observe an increase in the score as we are eliminating the data that are less correlated to the target variable.

Common steps for Question 5 and Question 6

- Read the data from dataset
- Perform EDA (Cleaning the data)
- Split the data into training and test data
- fit the model on training data
- Predict the response on test data
- Evaluate the performance of the Model

Question 5:

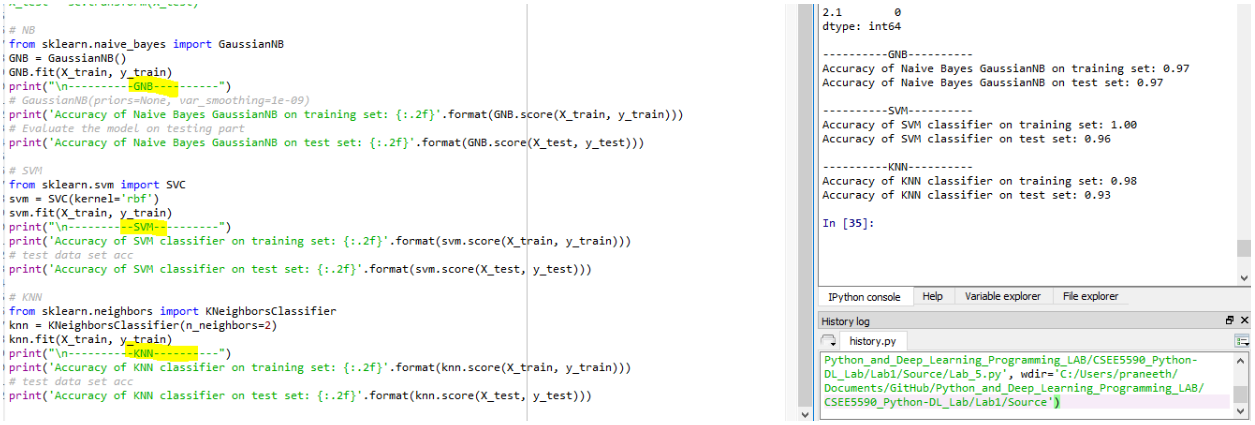
Dataset Chosen: Cancer

- Perform exploratory data analysis on the data set

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#importing our cancer.csv dataset
dataset = pd.read_csv('cancer.csv')
X = dataset.iloc[:, 1:9].values
Y = dataset.iloc[:, 10].values
dataset.head()
print("Cancer data set dimensions : {}".format(dataset.shape))
dataset.isnull().sum()
print(dataset.isna().sum())

from sklearn.preprocessing import LabelEncoder
```

```
Cancer data set dimensions : (698, 11)
1000025    0
5          0
1          0
1.1        0
1.2        0
2          0
1.3        0
3          0
1.4        0
1.5        0
2.1        0
```

- Apply the three classification algorithms Naïve Baye’s, SVM and KNN on the chosen data set

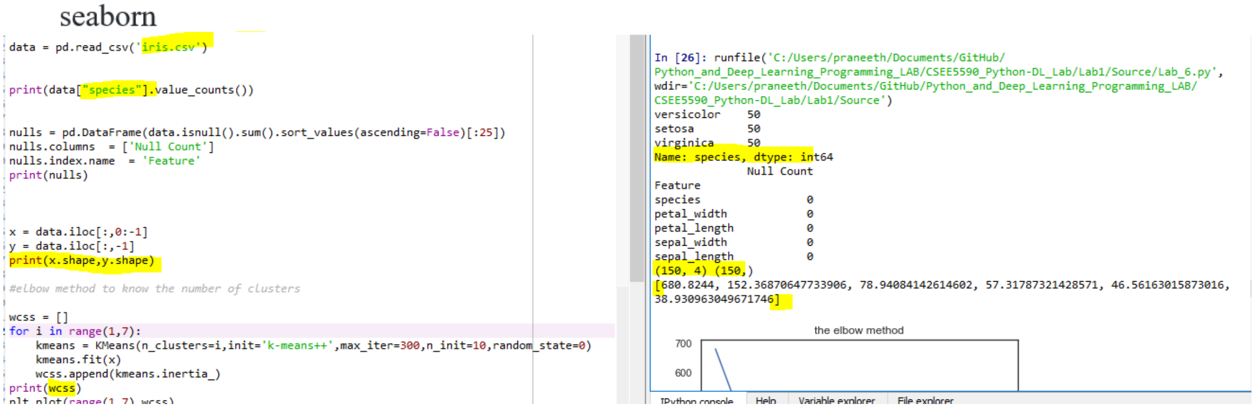


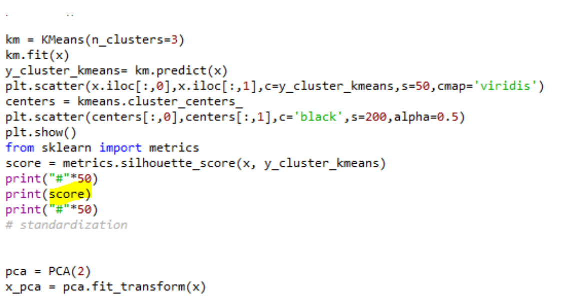
- Better Result: **Naive Baye's**

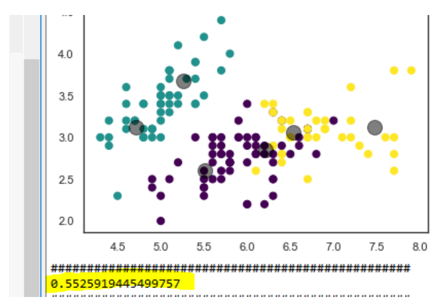
Question 6:

Dataset Chosen: Iris

- Apply K-means on the dataset
- Visualize the clusters using matplotlib or seaborn

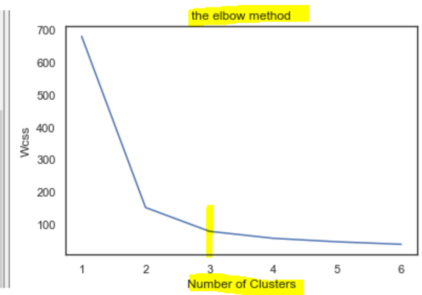






- Report which K is the best using the elbow method

```
#elbow method to know the number of clusters

wcss = []
for i in range(1,7):
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
print(wcss)
plt.plot(range(1,7),wcss)
plt.title('the elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()

km = KMeans(n_clusters=3)
km.fit(x)
```



Best K: 3

- Standardization(PCA)

```
# standardization

pca = PCA(2)
x_pca = pca.fit_transform(x)
df2 = pd.DataFrame(data=x_pca)
finaldf = pd.concat([df2,data[['species']]],axis=1)
print("#"*50)
print(finaldf)
print("#"*50)
```

```
##################################################
          0         1 species
0 -2.684207  0.326607  setosa
1 -2.715391 -0.169557  setosa
2 -2.889820 -0.137346  setosa
3 -2.746437 -0.311124  setosa
4 -2.728593  0.333925  setosa
5 -2.279897  0.747783  setosa
6 -2.820891 -0.082105  setosa
7 -2.626482  0.170405  setosa
8 -2.887959 -0.570798  setosa
```

- Evaluate with silhouette score

```
# KMeans after standarization

km = KMeans(n_clusters=4)
km.fit(x_pca)
y_cluster_kmeans= km.predict(x_pca)
from sklearn import metrics
score = metrics.silhouette_score(x_pca, y_cluster_kmeans)
print("#"*50)
print(score)
print("#"*50)
```

```
[150 rows x 3 columns]
##################################################
##################################################
0.5581660400375023
##################################################

In [27]:
```