```
import warnings
warnings.filterwarnings('ignore')
```

K-Nearest-Neighbors

KNN falls in the supervised learning family of algorithms. Informally, this means that we are given a labelled dataset consiting of training observations (x,y) and would like to capture the relationship between x and y. More formally, ou goal is to learn a function h:X→Y so that given an unseen observation x, h(x) can confidently predict the correspond output y.

In this module we will explore the inner workings of KNN, choosing the optimal K values and using KNN from scikit-learn.

Overview

1.Read the problem statement.

2.Get the dataset.

3.Explore the dataset.

4.Pre-processing of dataset.

5.Visualization

6.Transform the dataset for building machine learning model.

7.Split data into train, test set.

7.Build Model.

8.Apply the model.

9.Evaluate the model.

10.Finding Optimal K value

11.Repeat 7,8,9 steps.

Problem statement

Dataset

The data set we'll be using is the Iris Flower Dataset which was first introduced in 1936 by the famous statistician Ronald Fisher and consists of 50 observations from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals.

Attributes of the dataset: https://archive.ics.uci.edu/ml/datasets/Iris

Train the KNN algorithm to be able to distinguish the species from one another given the measurements of the 4 features.

Question 1

Import the data set and print 10 random rows from the data set

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier,NearestNeighbors
from sklearn.model_selection import train_test_split,cross_val_score
from scipy.stats import zscore
from sklearn.preprocessing import Imputer
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
import seaborn as sns
import os
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473189

Enter your authorization code:
··········
Mounted at /content/drive
```

```
file = '/content/drive/My Drive/PGAIML/26-28 July 2019/Lab/External/Iris.csv'
file
dframe = pd.read_csv(file)
randomRows = dframe.sample(n=10, random_state=2)
randomRows
```

	Sepal Length (in cm)	Sepal Width in (cm)	Petal length (in cm)	Petal width (in cm)	Cl
6	4.6	3.4	1.4	0.3	Iris-se
3	4.6	3.1	1.5	0.2	Iris-se
113	5.7	2.5	5.0	2.0	virgi
12	4.8	3.0	1.4	0.1	Iris-se
24	4.8	3.4	1.9	0.2	Iris-se
129	7.2	3.0	5.8	1.6	virgi
25	5.0	3.0	1.6	0.2	Iris-se
108	6.7	2.5	5.8	1.8	virgi

Data Pre-processing

Question 2 - Estimating missing values

*Its not good to remove the records having missing values all the time. We may end up loosing some data points. So will have to see how to replace those missing values with some estimated values (median) *

```
#There is no missing values
dframe.isna().any()
```

```
Sepal Length (in cm)    False
Sepal Width in (cm)     False
Petal length (in cm)    False
Petal width (in cm)     False
Class                   False
dtype: bool
```

There is no missing value in the dataset, so we dont need to replace anything

Question 3 - Dealing with categorical data

Change all the classes to numericals (0to2).

```
dframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
Sepal Length (in cm)    150 non-null float64
Sepal Width in (cm)     150 non-null float64
Petal length (in cm)    150 non-null float64
```

```
dframe.describe()
```

	Sepal Length (in cm)	Sepal Width in (cm)	Petal length (in cm)	Petal width (in
count	150.000000	150.000000	150.000000	150.000
mean	5.843333	3.054000	3.758667	1.198
std	0.828066	0.433594	1.764420	0.763
min	4.300000	2.000000	1.000000	0.100
25%	5.100000	2.800000	1.600000	0.300
50%	5.800000	3.000000	4.350000	1.300
75%	6.400000	3.300000	5.100000	1.800
max	7.900000	4.400000	6.900000	2.500

```
print(dframe['Class'].unique())

print(dframe.groupby('Class').size())
```

```
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
Class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

```
dframe['Class'] = dframe.Class.astype('category')
dframe['Class'] = dframe['Class'].cat.codes
dframe.sample(n=10, random_state=2)
```

	Sepal Length (in cm)	Sepal Width in (cm)	Petal length (in cm)	Petal width (in cm)	Cl
6	4.6	3.4	1.4	0.3	
3	4.6	3.1	1.5	0.2	
113	5.7	2.5	5.0	2.0	
12	4.8	3.0	1.4	0.1	
24	4.8	3.4	1.9	0.2	
129	7.2	3.0	5.8	1.6	
25	5.0	3.0	1.6	0.2	

Question 4

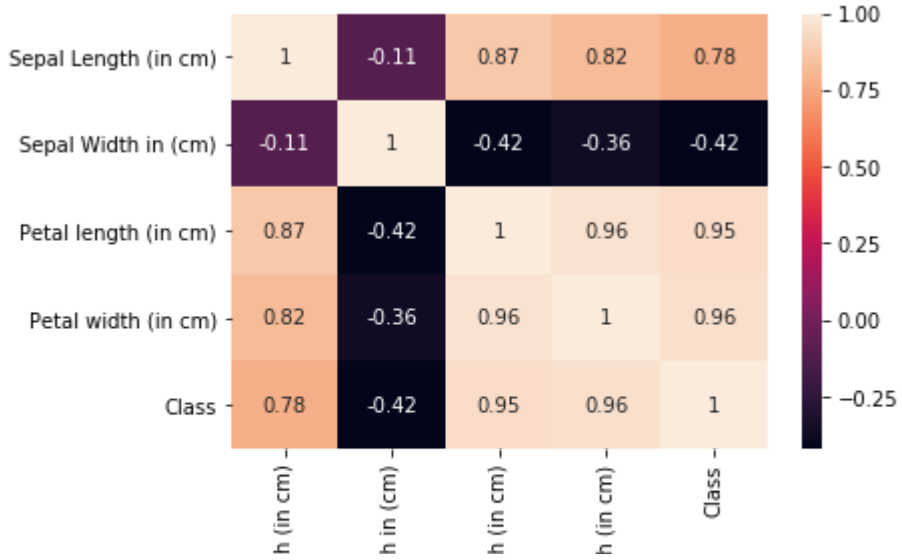
Observe the association of each independent variable with target variable and drop variables from feature set having correlation in range -0.1 to 0.1 with target variable.

```
import seaborn as sns
corr = dframe.corr()
corr
```

	Sepal Length (in cm)	Sepal Width in (cm)	Petal length (in cm)	Petal width (in cm)	Cl
Sepal Length (in cm)	1.000000	-0.109369	0.871754	0.817954	0.782
Sepal Width in (cm)	-0.109369	1.000000	-0.420516	-0.356544	-0.419
Petal length (in cm)	0.871754	-0.420516	1.000000	0.962757	0.949

```
sns.heatmap(corr,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1b30bca20>
```



There is no such variable such that has corelation with target variable in the given range of -0.1 to 0.1

Question 5

Observe the independent variables variance and drop such variables having no variance or almost zero variance(variar 0.1). They will be having almost no influence on the classification.

```
dframe.var()
print(dframe.var())
low_var = np.where((dframe.var()<0.1))
low_var
```

```
Sepal Length (in cm)    0.685694
Sepal Width in (cm)     0.188004
Petal length (in cm)    3.113179
Petal width (in cm)     0.582414
Class                   0.671141
dtype: float64
(array([], dtype=int64),)
```

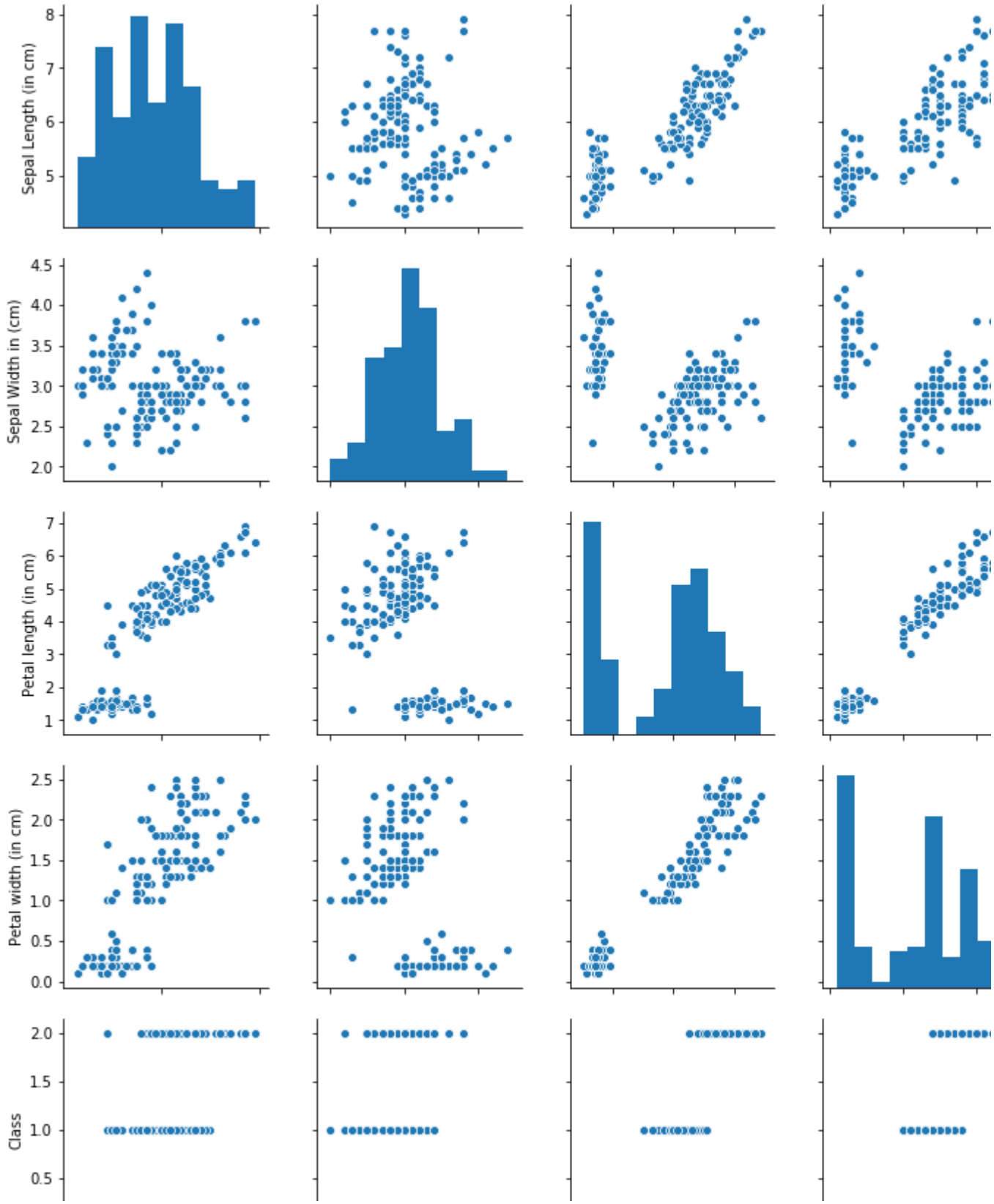
There are no variables that are having no variance or almost zero variance i.e (variance<0.1),. No need to drop any variable

Question 6

Plot the scatter matrix for all the variables.

```
sns.pairplot(dframe)
```

```
<seaborn.axisgrid.PairGrid at 0x7fa1af6e1390>
```



Split the dataset into training and test sets

Question 7

Split the dataset into training and test sets with 80-20 ratio.

```
from sklearn.model_selection import train_test_split
Y = dframe['Class']
X = dframe.drop('Class',axis=1)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
print (X_train.shape, Y_train.shape)
print (X_test.shape, Y_test.shape)
```

```
(120, 4) (120,)
(30, 4) (30,)
```

Question 8 - Model

*Build the model and train and test on training and test sets respectively using **scikit-learn**. Print the Accuracy of the m with different values of **k=3,5,9**.*

Hint: For accuracy you can check **accuracy_score()** in scikit-learn

```
counter = [3,5,9]
for k in counter:
    NNH = KNeighborsClassifier(n_neighbors = k , weights = 'uniform', metric = 'euclidean')
    NNH.fit(X_train, Y_train)
    Y_pred = NNH.predict(X_test)
    print(accuracy_score(Y_test, Y_pred))
```

```
0.9666666666666667
0.9666666666666667
1.0
```

Question 9 - Cross Validation

Run the KNN with no of neighbours to be 1,3,5..19 and *Find the* *optimal number of neighbours** from the above list using the Mis classification error

Hint:

Misclassification error (MSE) = 1 - Test accuracy score. Calculated MSE for each model with neighbours = 1,3,5...19 find the model with lowest MSE

```
counter = np.arange(1,20,2)
MSE_arr=[]
for k in counter:
    NNH = KNeighborsClassifier(n_neighbors = k , weights = 'uniform', metric = 'euclidean')
    NNH.fit(X_train, Y_train)
    Y_pred = NNH.predict(X_test)
    MSE = 1-accuracy_score(Y_test, Y_pred)
    MSE_arr.append(MSE)
    print(k, MSE)

optimal_kvalue = counter[MSE_arr.index(min(MSE_arr))]
print(optimal_kvalue)
```

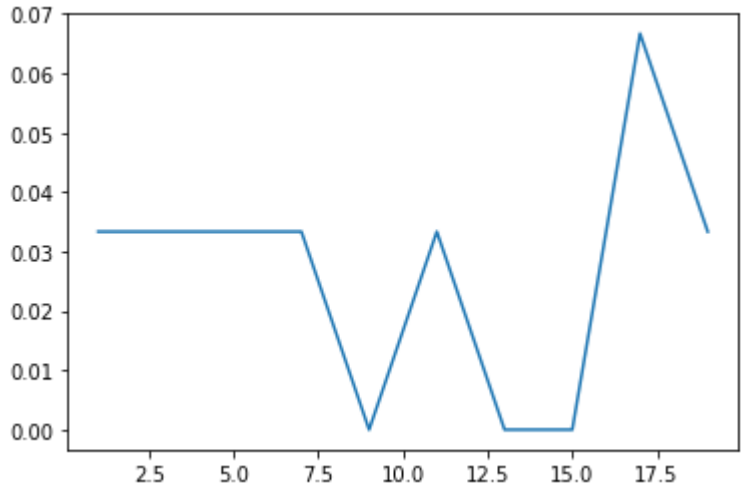
```
1 0.033333333333333326
3 0.033333333333333326
5 0.033333333333333326
7 0.033333333333333326
9 0.0
11 0.033333333333333326
13 0.0
15 0.0
17 0.06666666666666665
19 0.033333333333333326
```

Question 10

Plot misclassification error vs k (with k value on X-axis) using matplotlib.

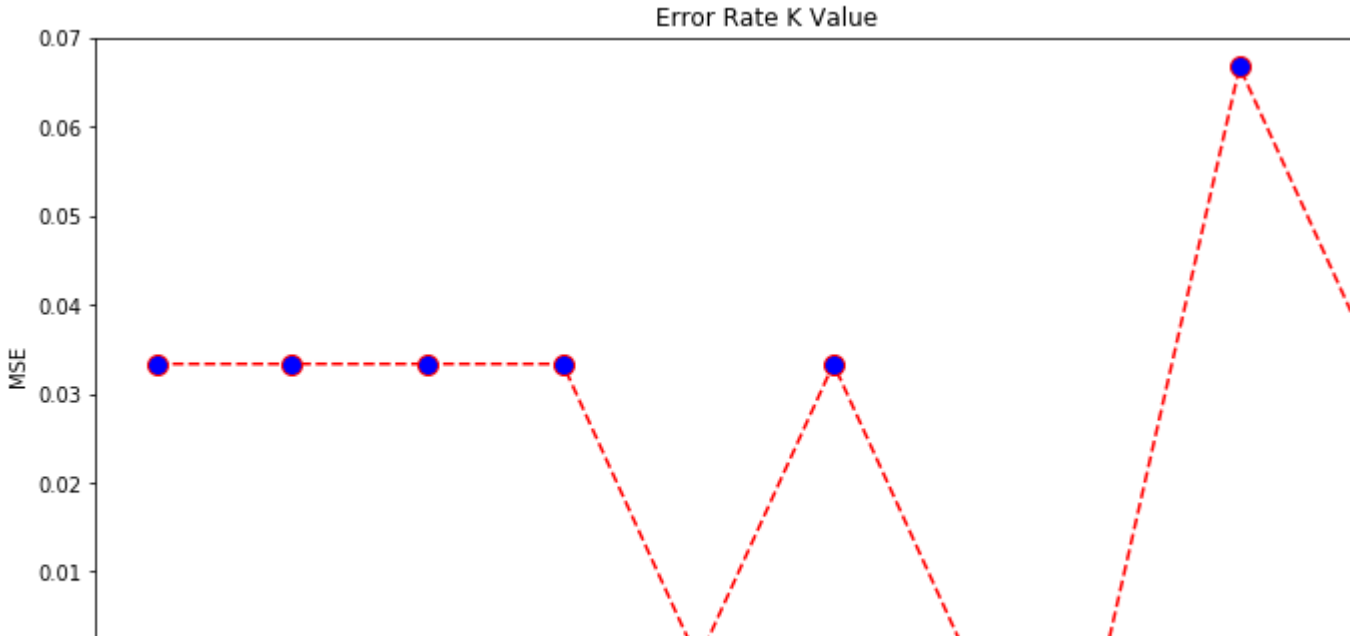
```
import matplotlib.pyplot as plt
MSE_arr=[]
counter = np.arange(1,20,2)
for k in counter:
    NNH = KNeighborsClassifier(n_neighbors = k , weights = 'uniform', metric = 'euclidean')
    NNH.fit(X_train, Y_train)
    Y_pred = NNH.predict(X_test)
    MSE = 1-accuracy_score(Y_test, Y_pred)
    MSE_arr.append(MSE)
print(MSE_arr)
plt.plot(counter,MSE_arr)
```

```
[0.033333333333333326, 0.033333333333333326, 0.033333333333333326, 0.033333333333333326,
[<matplotlib.lines.Line2D at 0x7fa1acf070f0>]
```




```
plt.figure(figsize=(12, 6))
plt.plot(range(1,20,2), MSE_arr, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('MSE')
```

```
Text(0, 0.5, 'MSE')
```



Naive Bayes with Iris Data

```
#Load all required library
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
from sklearn import datasets
from sklearn.decomposition import PCA
```

Slice Iris data set for Independent variables and dependent variables

Please note 'Species' is my dependent variables, name it y and independent set data as X

```
file = '/content/drive/My Drive/PGAIML/26-28 July 2019/Lab/External/Iris.csv'
file
df = pd.read_csv(file)
df['Class']=df.Class.astype('category')
df['Class'] = df['Class'].cat.codes
df
```

	Sepal Length (in cm)	Sepal Width in (cm)	Petal length (in cm)	Petal width (in cm)	Cl
0	5.1	3.5	1.4	0.2	
1	4.9	3.0	1.4	0.2	
2	4.7	3.2	1.3	0.2	
3	4.6	3.1	1.5	0.2	
4	5.0	3.6	1.4	0.2	
5	5.4	3.9	1.7	0.4	
6	4.6	3.4	1.4	0.3	
7	5.0	3.4	1.5	0.2	
8	4.4	2.9	1.4	0.2	
9	4.9	3.1	1.5	0.1	
10	5.4	3.7	1.5	0.2	
11	4.8	3.4	1.6	0.2	
12	4.8	3.0	1.4	0.1	
13	4.3	3.0	1.1	0.1	
14	5.8	4.0	1.2	0.2	
15	5.7	4.4	1.5	0.4	
16	5.4	3.9	1.3	0.4	
17	5.1	3.5	1.4	0.3	
18	5.7	3.8	1.7	0.3	
19	5.1	3.8	1.5	0.3	
20	5.4	3.4	1.7	0.2	
21	5.1	3.7	1.5	0.4	
22	4.6	3.6	1.0	0.2	
23	5.1	3.3	1.7	0.5	
24	4.8	3.4	1.9	0.2	
25	5.0	3.0	1.6	0.2	
26	5.0	3.4	1.6	0.4	
27	5.2	3.5	1.5	0.2	
28	5.2	3.4	1.4	0.2	
29	4.7	3.2	1.6	0.2	

...	...	...	...	...
120	6.9	3.2	5.7	2.3
121	5.6	2.8	4.9	2.0
122	7.7	2.8	6.7	2.0
123	6.3	2.7	4.9	1.8
124	6.7	3.3	5.7	2.1
125	7.2	3.2	6.0	1.8
126	6.2	2.8	4.8	1.8
127	6.1	3.0	4.9	1.8
128	6.4	2.8	5.6	2.1
129	7.2	3.0	5.8	1.6
130	7.4	2.8	6.1	1.9
131	7.9	3.8	6.4	2.0
132	6.4	2.8	5.6	2.2
133	6.3	2.8	5.1	1.5
134	6.1	2.6	5.6	1.4
135	7.7	3.0	6.1	2.3
136	6.3	3.4	5.6	2.4
137	6.4	3.1	5.5	1.8
138	6.0	3.0	4.8	1.8
139	6.9	3.1	5.4	2.1
140	6.7	3.1	5.6	2.4
141	6.9	3.1	5.1	2.3
142	5.8	2.7	5.1	1.9
143	6.8	3.2	5.9	2.3
144	6.7	3.3	5.7	2.5
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 5 columns

```
#Check the dataset

y = df['Class']
X = df.drop('Class',axis=1)
print(y.head(5))
print(X.head())
```

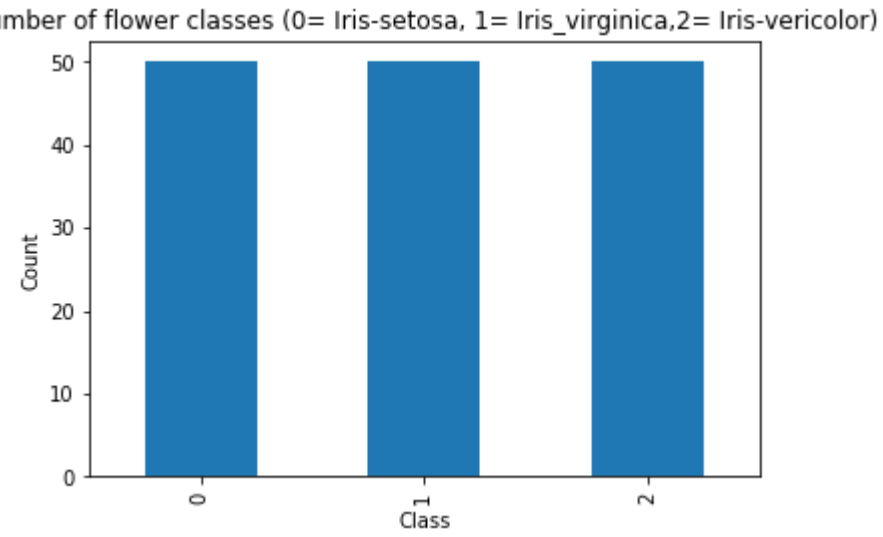
```
0    0
1    0
2    0
3    0
4    0
Name: Class, dtype: int8
   Sepal Length (in cm)  ...  Petal width (in cm)
0                   5.1  ...                  0.2
1                   4.9  ...                  0.2
2                   4.7  ...                  0.2
3                   4.6  ...                  0.2
4                   5.0  ...                  0.2

[5 rows x 4 columns]
```

Question 11

Find the distribution of target variable (Class)

And, Plot the distribution of target variable using histogram

```
plt.figure()
pd.Series(df['Class']).value_counts().sort_index().plot(kind = 'bar')
plt.ylabel("Count")
plt.xlabel("Class")
plt.title('Number of flower classes (0= Iris-setosa, 1= Iris_virginica,2= Iris-vericolor)');
```




```
#Drop Id variable from data
```

Question 12

Find Correlation among all variables and give your insights

```
#Please note, it's Require to remove correlated features because they are voted twice in the model a
## it can lead to over inflating importance.We will ignore it here
```

```
corr_new = df.corr()
corr_new
```

	Sepal Length (in cm)	Sepal Width in (cm)	Petal length (in cm)	Petal width (in cm)	Cl
Sepal Length (in cm)	1.000000	-0.109369	0.871754	0.817954	0.782
Sepal Width in (cm)	-0.109369	1.000000	-0.420516	-0.356544	-0.419
Petal length (in cm)	0.871754	-0.420516	1.000000	0.962757	0.949

Split data in Training and test set in 80:20.

```
from sklearn.model_selection import train_test_split
X=df.drop(['Class'], axis=1)
Y=df['Class']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20)
```

Question 13

Do Feature Scaling

```
# Use StandardScaler or similar methods
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Question 14

Train and Fit NaiveBayes Model

```
#Fit the model
#train
from sklearn.naive_bayes import GaussianNB

clf_GNB = GaussianNB()
clf_GNB = clf_GNB.fit(X_train, Y_train)
```

```
#Predict
Y_pred_GNB=clf_GNB.predict(X_test)
Y_pred_GNB
```

```
array([1, 2, 0, 0, 1, 1, 2, 1, 0, 2, 0, 0, 2, 2, 1, 1, 2, 1, 0, 0, 0, 1,
       1, 2, 1, 2, 1, 2, 1, 0], dtype=int8)
```

Question 15

Print Accuracy and Confusion Matrix and Conclude your findings

```
# show Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(Y_test, Y_pred_GNB)
print(confusion_matrix)
```

```
[[ 9  0  0]
 [ 0 12  0]
 [ 0  0  9]]
```

```
# show accuracy
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, Y_pred_GNB)
```

```
1.0
```

```
#Show precision and Recall metrics
from sklearn.metrics import classification_report
print(classification_report(Y_test, Y_pred_GNB))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      1.00      1.00        12
           2       1.00      1.00      1.00         9

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```