# CPSC 552 – Data Mining – Assignment #2

**Problem #1:** Implement Naïve Bayes Algorithm from scratch for classification of the flower types in the Iris dataset. As explained in the lecture, this dataset contains 150 data items with 4 measurements for each flower. There are three classes of flowers (setosa, versicolor, virginica). The dataset contains 50 rows for each flower type.

Try to program this yourself. If you need help, the solution is given on the next page.

**Problem #2**: Implement the Naïve Bayes for wheat seed dataset. This is a three class dataset with 210 observations with 7 features. The variable names are as follows:

Area.
Perimeter.
Compactness
Length of kernel.
Width of kernel.
Asymmetry coefficient.
Length of kernel groove.

Last column indicates the class (1, 2, 3).

## Solution to Problem #1:

Create a Python Application project called “NaiveBayes”.
Type the following code in NaiveBayes.py

```
import sys
import numpy as np
import pandas as pd
import math


def compute_gaussian_probab(x, mean, var):  # returns product of gaussian
probabs for each feature
    res = 1
    for i in range(0,len(x)):
        exponent = math.exp(-((x[i]-mean[i])**2 / (2 * var[i] )))
        res *= (1 / (math.sqrt(2 * math.pi * var[i]))) * exponent
    return res

def main():
    #---------------Iris Dataset-------------
    df = pd.read_csv("d:/pythonam2/data/iris.csv")

    #---randomize data
    dfrandom = df.sample(frac=1, random_state=1119).reset_index(drop=True)

    # data read from a file is read as a string, so convert the first 4 cols to
float
    df1 = dfrandom.iloc[:,0:4].astype(float)

    #---separate out the last column
    df2 = dfrandom.iloc[:,4]

    #---combine the 4 numerical columns and the ast column that has the flower
category
    dfrandom = pd.concat([df1,df2],axis=1)
    print(dfrandom)

    #---separate the data into training and test parts
    dftrain = dfrandom.iloc[0:100,:]
    print(dftrain)
    dftest = dfrandom.iloc[100:,:]
    print(dftest)

    #---assemble the data by categories i.e., classes
    dfsetosa = dfrandom[dfrandom['species'] == 'setosa']
    print(dfsetosa)
    dfversicolor = dfrandom[dfrandom['species'] == 'versicolor']
    print(dfversicolor)
    dfvirginica = dfrandom[dfrandom['species'] == 'virginica']
    print(dfvirginica)

    #---------find mean of each class---------
    mean_setosa = dfsetosa.iloc[:,0:4].mean(axis=0)
    print('mean setosa\n',mean_setosa)
```

```
    mean_versicolor = dfversicolor.iloc[:,0:4].mean(axis=0)
    print('mean versicolor\n',mean_versicolor)
    mean_virginica = dfvirginica.iloc[:,0:4].mean(axis=0)
    print('mean virginica\n',mean_virginica)

    #---------find variance of each class---------
    var_setosa = dfsetosa.iloc[:,0:4].var(axis=0)
    print('var setosa\n',var_setosa)
    var_versicolor = dfversicolor.iloc[:,0:4].var(axis=0)
    print('var versicolor\n',mean_versicolor)
    var_virginica = dfvirginica.iloc[:,0:4].var(axis=0)
    print('var virginica\n',var_virginica)

    #---do prediction on the test set via Naive Bayes
    count_correct = 0
    print(len(dftest))
    for i in range(0,len(dftest)):
        x = dftest.iloc[i,0:4].values
        probC1 =
compute_gaussian_probab(x,mean_setosa.values,var_setosa.values)
        probC2 =
compute_gaussian_probab(x,mean_versicolor.values,var_versicolor.values)
        probC3 =
compute_gaussian_probab(x,mean_virginica.values,var_virginica.values)
        probs = np.array([probC1,probC2,probC3])
        maxindex = probs.argmax(axis=0)

        if dftest.iloc[i,4] == 'setosa':
            index = 0
        if dftest.iloc[i,4] == 'versicolor':
            index = 1
        if dftest.iloc[i,4] == 'virginica':
            index = 2
        if maxindex == index:
            count_correct = count_correct + 1
        #print(probC1,' ', probC2,' ', probC3,' class=',dftest.iloc[i,4])
    print('classification accuracy =', count_correct/len(dftest)*100)
if __name__ == "__main__":
    sys.exit(int(main() or 0))
```

The output produced in running the program appears as (92% accuracy on the test set):