<u>Machine Learning Assignment</u>

Chandan Adiga,

2016HT12329,

2016HT12329@wilp.bits-pilani.ac.in

Data set	Iris flower data set (https://en.wikipedia.org/wiki/Iris_flower_data_set)
Approach	Likelihood based
Algoirthm	Naive Bayes
Programming language	Python 2.7.6 or 3.4

(1) In this assignment, we are going anaylise iris-data set. By analysing the iris data set table, one can conclude that there are 3 types(classes, namely: I. setosa, I. versicolor & I. virginica) of species that can be categorized into from given attributes(*4 attributes namely: Sepal length, Sepal width, Petal length & Petal width*). The attributes which play major impact on classifications are petal width and height but not limited to, by observation.

(2) We will categorize this table in to training set and testing set. Training set will be used to build the model and tranin the algorithm to predict the species for any given data. As per quidelines, we will use every 3rd row of this table as training set and rest as testing data. Doing so, will end up in 100 rows of training set and remaining 50 rows for testing.

Copy the table in to some text file. We have choosen python as implementation programming language. First, we develop a parser to parse the copied data in to a appropriate data structure i.e two dimensional array; each row representing a sample; and each column – an attribute; except last row which is a class/species.

Ex:

Sepal length	**Sepal width**	**Petal length**	**Petal width**	**Species**
5.1	3.5	1.4	0.2	I. setosa

Then, iterate through each row, to seperate training and testing data based on row no.(PS: every 3rd row is our training set). Once sets are seperated, make sure attributes are formatted to proper data types(float) and group them in to respective classes. Since we are considering training set, we group them by refering to their species type directly. As data is already in grouped order, we choose respective index to group. As of now, it is hardcoded as training set remains static and pre-known.

i.e:

```
class1 = X[0:34]
class2 = X[34:67]
class3 = X[67:100]
```

(Ref: Classifier_NB.py > model())

(3) We will be using most likey hood approach through Naive Bayes algorithm to classify the species. We use probability density function (whcih makes use of meand and standard deviation) for normal distribution, in order to transfer numercal variables in to categorical counterparts/classes(species). Finally, the class with higher posterior probability is the result of prediction. As we have testing set with pre-known species, we can cross check implementation for corectness of prediction by comparing result class against its origical class form table.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$ Mean

$$\sigma = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2\right]^{0.5}$$ Standard deviation

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$ Normal distribution

(Normal distribution computation: *Ref: Classifier_NB.py > computeProb())*

First, we calculate mean and variance for each of attributes per class (Ref: Classifier_NB.py > model()). So that each attribute now can be cateogorized to one of the class based on likely hood of its value. Now, time to classify!.

```
probAttribute1 = computeProb(mean_and_varience_array[classIndex][0]
[0],mean_and_varience_array[classIndex][0][1],numpy.float(predictItemAttributes[0]))
```

$$\underbrace{P(c \mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x \mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Where,

P(c|x) is the posterior probability of class (target) given predictor (attribute).

P(c) is the prior probability of class.

P(x|c) is the likelihood which is the probability of predictor given class.

P(x) is the prior probability of predictor.

i.e We will compute probablity/likelihood of given attribute to belong to each class and then compute total probability for each class.

```
classProbability = probAttribute1 * probAttribute2 * probAttribute3 * probAttribute4 *
probabilityOfEachClass
```

The class with maximum value is most appropriate class.

```
maxProbValue = max(class_probability_of_item)
 maxProbClassIndex = class_probability_of_item.index(maxProbValue)
```

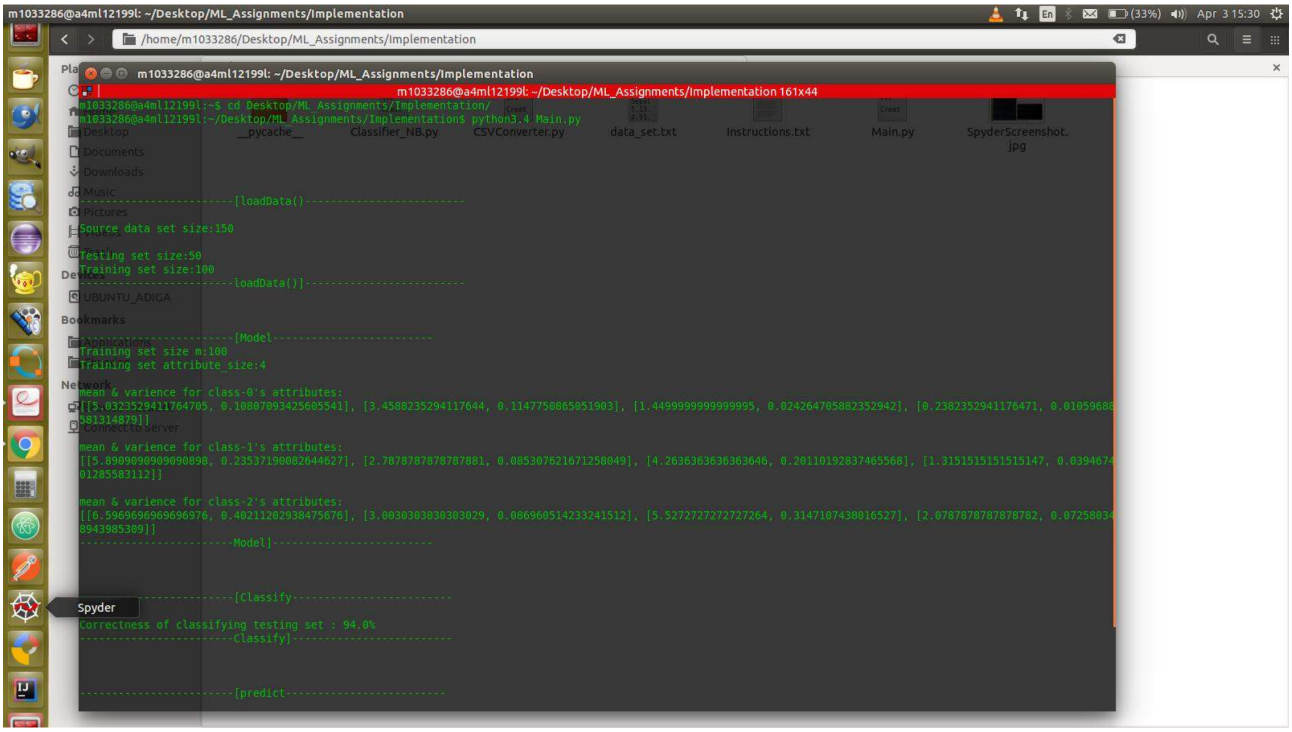
(4) *We can compute the performance, by comparing class of test sample against predicted class. On a set, its ratio of sum of correct predictions to total set size.*

i.e:

```
if(classes[maxProbClassIndex] == testItem[4]) :#4th index is species name
        correct_prediction_count += 1

correctnessPercentage = correct_prediction_count/float(test_set_size) * 100
```

With current implementation on testing set, i.e with 50 samples(Except every 3rd row) percentage of correct classification is **94%**


m1033286@a4ml12199l: ~/Desktop/ML_Assignments/Implementation
/home/m1033286/Desktop/ML_Assignments/Implementation
m1033286@a4ml12199l: ~/Desktop/ML_Assignments/Implementation 161x44
m1033286@a4ml12199l:~$ cd Desktop/ML_Assignments/Implementation/
m1033286@a4ml12199l:~/Desktop/ML_Assignments/Implementation$ python3.4 Main.py
__pycache__
Classifier_NB.py
CSVConverter.py
data_set.txt
Instructions.txt
Main.py
SpyderScreenshot.jpg
-----------------------[loadData()-----------------------
Source data set size:150
Testing set size:50
Training set size:100
-----------------------loadData()]-----------------------
-----------------------[Model-----------------------
Training set size m:100
Training set attribute_size:4
mean & varience for class-0's attributes:
[[5.0323529411764705, 0.10807093425605541], [3.4588235294117644, 0.1147750865051903], [1.4499999999999995, 0.024264705882352942], [0.2382352941176471, 0.01059688
581314879]]
mean & varience for class-1's attributes:
[[5.8909090909090898, 0.23537190082644627], [2.7878787878787881, 0.085307621671258049], [4.2636363636363646, 0.20110192837465568], [1.3151515151515147, 0.0394674
01285583112]]
mean & varience for class-2's attributes:
[[6.5969696969696976, 0.40211202938475676], [3.0030303030303029, 0.086960514233241512], [5.5272727272727264, 0.3147107438016527], [2.0787878787878782, 0.07258034
8943985309]]
-----------------------Model]-----------------------
Spyder
-----------------------[Classify-----------------------
Correctness of classifying testing set : 94.0%
-----------------------Classify]-----------------------
-----------------------[predict-----------------------