CS 584 - Machine Learning - Assignment Report

Rakesh Adhikesavan

Master's in Data Science, Illinois Institute of Technology, Chicago

Abstract

In this assignment, generative learning algorithms were implemented. The performance of each algorithm was evaluated and tested on different datasets, using a 10 fold cross validation method [2]. The algorithms were implemented in Python and the implementation details along with the results and conclusions are summarized below.

--*****--

1. Introduction

The purpose of this assignment is to get an idea of how generative learning works and how to evaluate different algorithms.

For the Gaussian Discriminant Analysis the very popular Iris data set was chosen (description of data can be found below).

First, a 2-class dataset with continuous one dimensional features was derived by selecting a subset of the original Iris dataset and GDA was performed on it. The algorithm was evaluated based on precision, recall F-measure and accuracy. The confusion matrix was calculated and plotted.

Secondly, the above procedure was repeated for a 2-class dataset with n-Dimensional (n = 4) continuous features derived by selecting a subset of the original Iris dataset.

Third, GDA was performed on the entire dataset and the results were evaluated.

Next, a classifier was programmed to classify an email as spam or not spam. The features used was count of different words that occurred in each email. First, the dataset was binarized and an assumption was made that the features were independent from each other and that the features followed a Bernoulli distribution. Then, the word count was preserved and an assumption was made that the features followed a Binomial distribution. Both these classifiers were evaluated based on the error, precision and accuracy.

2. Implementation Details

All the algorithms were implemented in Python. Some of the important Python packages used are numpy, sklearn, matplotlib and scipy.

2.1 Data Description

Iris Dataset

This data sets consists of 3 different types of irises': *Setosa, Versicolour*, and *Virginica*. The data is stored in a 150x4 numpy.ndarray

The rows are the samples and the columns are: Sepal Length, Sepal Width, Petal Length and Petal Width.

The training data has 150 sample out of which the number of samples from each type is equal to 50. The scatterplot matrix below

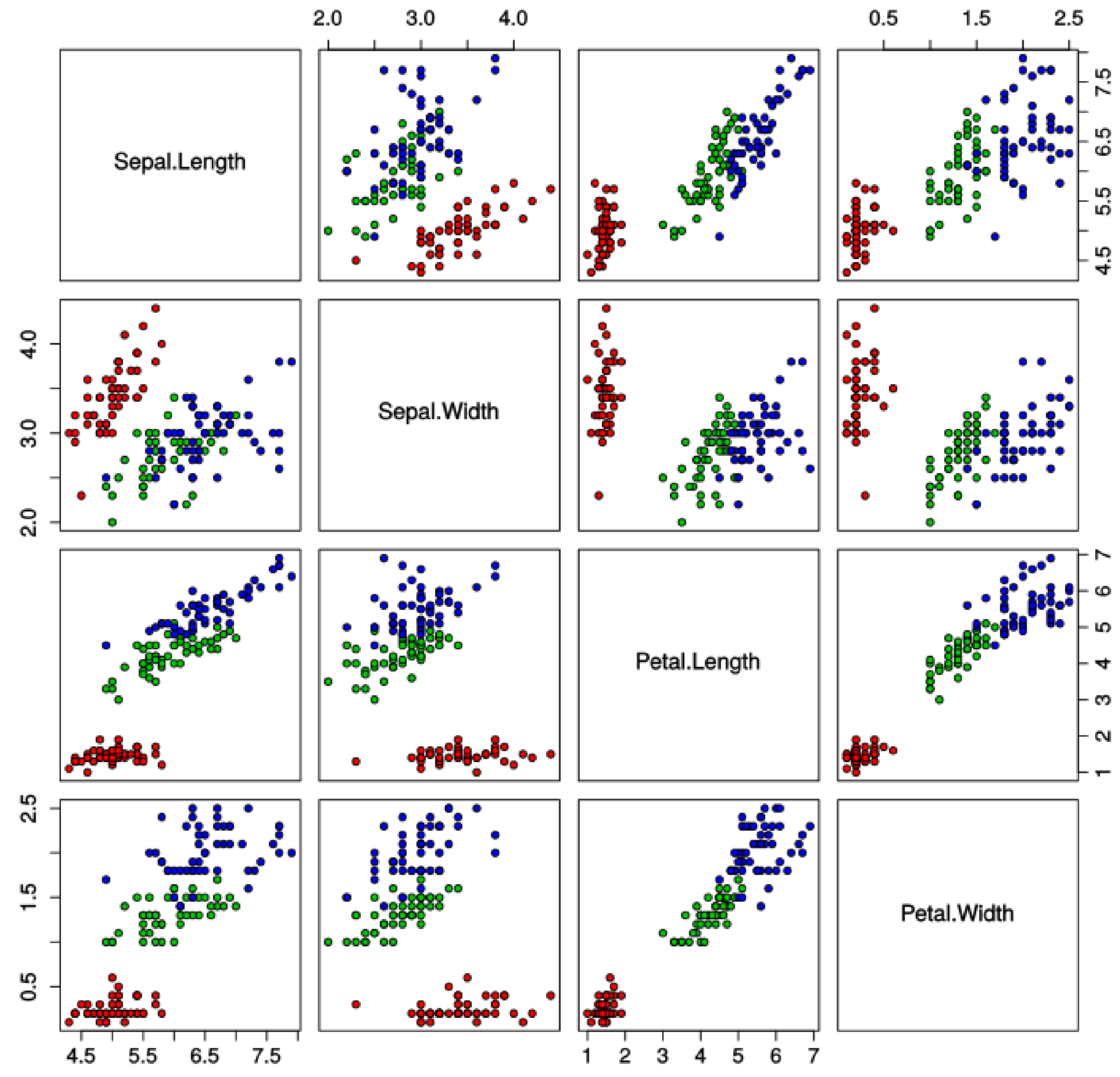


Image: Scatterplot of dataset, red = Setosa, green = Versicolor, blue = Virginica Source: [3]

Spam datasets

The first spam dataset [4] has 48 continuous real features of type word frequency. Which is percentage of words in the e-mail that match WORD, i.e. 100 (number of times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string. This data set was used for the Bernoulli distribution.

For more details on the dataset, visit link in reference [4].

The second spam dataset [5] has word count instead of word frequencies. This dataset was used for binomial distribution Naïve Bayes classifier.

2.2 Algorithm Detail

Functions that are common for all programs are stored in file functions.py, all other programs import functions.py

Program1: 1D2Class.py

This program reads the Iris dataset. A subset of the data is selected by selecting the first 100 rows and the first column. This gives a 2 class dataset with continuous one dimensional features. The data is then shuffled.

Using Kfold Validation, the data is split into train and test datasets.

A function called train takes the training data and the corresponding labels as input and returns two dictionaries, mean and sigma. Both these dictionary variables take the class as key and returns the parameter as the corresponding value.

Function classify takes as input, the test data and the parameters (mean and covariance) that were trained. For each class a membership function calculates a membership value based on formula:

Membership function: $-log(\sigma_j) - \frac{(X-\mu_j)^2}{2\sigma^2}$

The prior probabilities for each class is equal, therefore it is ignored in the membership function.

Classification is done by $\widehat{y} = argmax_j \; g_j(x)$

That is, the class is given by j where the membership function is maximum.

A discriminant function calculates the difference in membership value for class 1 and class 2 for each row of test data. If the difference is positive, the corresponding row is classified as class 1, if negative it is classified as class 2.

The evaluate function calculates the precision, recall, F-Measure and computes the Confusion Matrix. The confusion matrix is plotted and a normalized confusion matrix is plotted as well.

- Precision = TP / (TP+FP)
- Recall = TP / (TP+FN)

Where:

```
       | Declare H1  |  Declare H0 |
|Is H1 |    TP       |   FN        |
|Is H0 |    FP       |   TN        |
```

- TP = true positive (declare H1 when, in truth, H1),
- FN = false negative (declare H0 when, in truth, H1),
- FP = false positive
- TN = true negative

Program 2: nD2Class.py

The algorithm is very similar to program 1. Program 1 was written in a generic manner, such that it can work with any number of classes and any number of dimensions.

The major difference is in the train function. The covariance parameter is matrix of dimension (nxn) where n is the number of features. Other functions are common and imported from functions.py

The same evaluate function is used to calculate precision, recall & confusion matrix is calculated and plotted.

The precision-recall curve was plotted by varying the threshold of the discriminant function.

Program 3: nDkClass.py

The entire Iris dataset is used by this program. The data has 4 features and 3 classes. The same function from the previous programs were used to train, classify the data and to evaluate the classifier.

Program 4: Bernoulli.py

The dataset used by this program is the spam or not spam dataset which has word frequencies as features.

The first step is to binarize the dataset to make it indicate whether a word appeared in a document or not. If the frequency of a word is greater than zero, it indicates that the word appeared in the document, so this will have a value 1 and 0 will indicate that the word does not appear in the document.

The train function calculates the prior probabilities of the classes (spam and not spam) and mean of the features.

The classify function classifies the test data by calculating the membership function for each class and assigning the class with maximum membership function.

The membership function is calculated using the formula:

$$g_i(x) = \sum_{1}^{n} x_j * log(\mu_j^i) + (1 - x_j) * log(1 - \mu_j^i) + log(\alpha_j)$$

X is classified to class i, corresponding to which g(x) is maximum.

The evaluate function calculates the precision, recall, f-measure and the confusion matrix for the classifier.

Program 5: Binomial.py

The dataset used by this program has the count of each word appearing in a document. The data has previously been divided in train and test datasets.

In the training phase the program calculates the parameters required for classification

$$\phi_{k|y=1} = p(x_j = k|y=1) = \frac{\left(\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \text{ and } y^{(i)} = 1\}\right) + 1}{\left(\sum_{i=1}^{m} 1\{y^{(i)} = 1\} n_i\right) + |V|}$$

$$\phi_{k|y=0} = p(x_j = k|y=0) = \frac{\left(\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \text{ and } y^{(i)} = 0\}\right) + 1}{\left(\sum_{i=1}^{m} 1\{y^{(i)} = 1\} n_i\right) + |V|}$$

And prior probabilities:

$$\phi_y = p(y=1) = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{m}$$

Classification is done by:

1. for each document in the test set, calculate

$$\log p(\vec{x}|y=1) + \log p(y=1)$$

2. Similarly, calculate

$$\log p(\vec{x}|y=0) + \log p(y=0)$$

3. comparing the two quantities from (1) and (2) above, a decision is made whether the email is spam or not spam

3. Results and Discussion

3.1 One dimension, 2 class classifier

The results of this classifier are:

accuracy : 0.86

precision : 0.810344827586

recall : 0.94

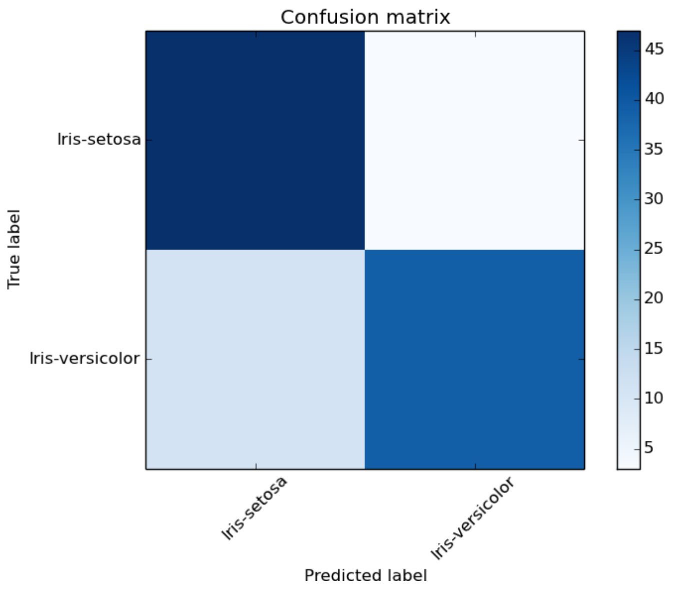
F-Measure: 0.875172413793

Confusion matrix, without normalization

```
[[47  3]
 [11 39]]
```

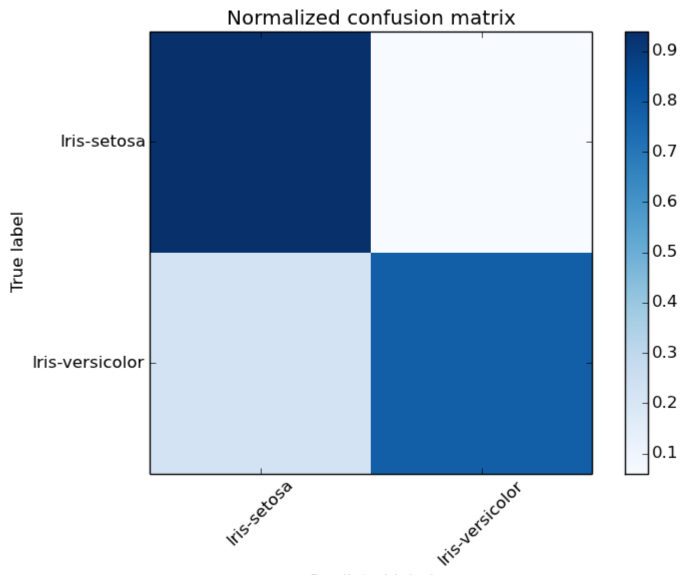
Normalized confusion matrix

```
[[ 0.94  0.06]
 [ 0.22  0.78]]
```



The diagonal of the confusion matrix represents the examples that were correctly classified. It was observed that the classifier was 86 percent accurate. The F-measure gives a better score for the classifier as it takes both precision and recall into account.

From the confusion matrix plot, we can see that, a few data points belonging to '*iris-versicolor*' was incorrectly classified as *'iris-setosa'*



3.2 N dimension, 2 class classifier

accuracy : 1.0

precision : 1.0

recall : 1.0

F-Measure : 1.0

Confusion matrix, without normalization

```
[[50  0]
 [ 0 50]]
```

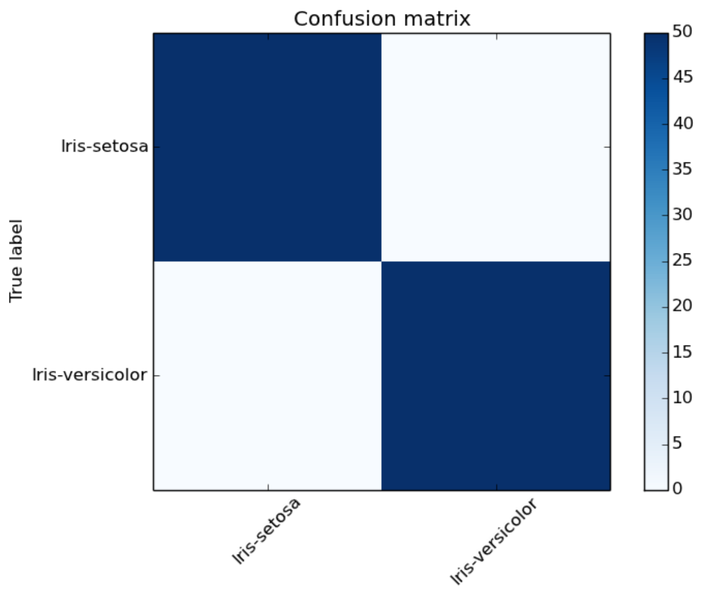
Normalized confusion matrix

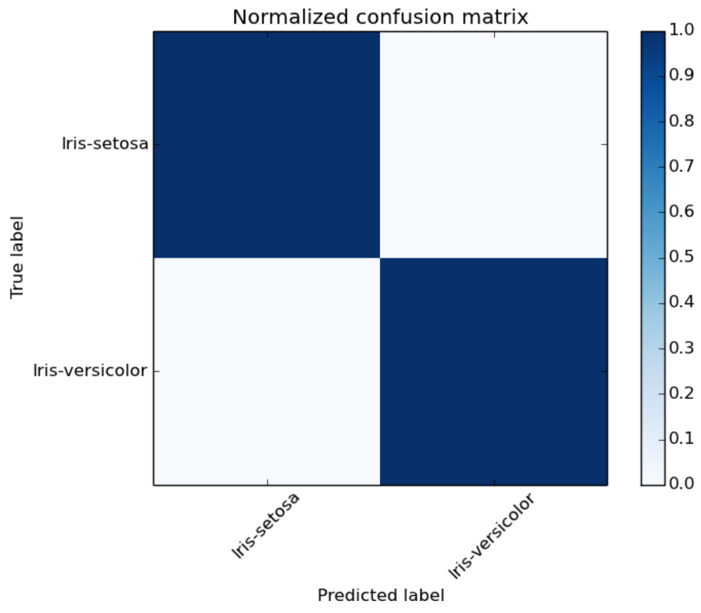
```
[[ 1.  0.]
 [ 0.  1.]]
```

This classifier is 100 percent accurate.

All the test samples were classified correctly.

In the confusion matrix plot, the leading diagonal represents the samples that were classified correctly.





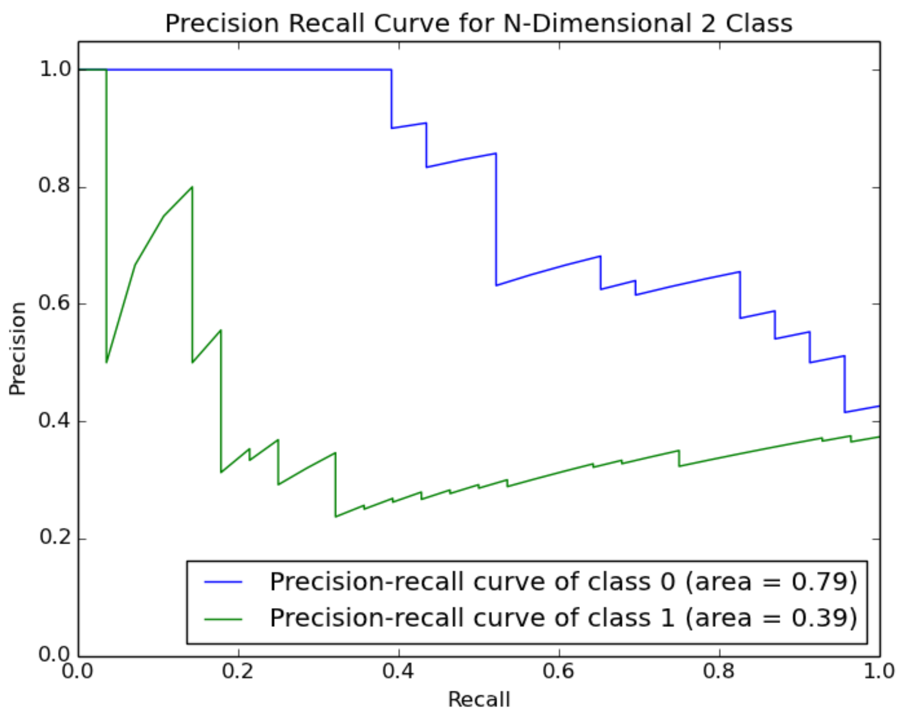
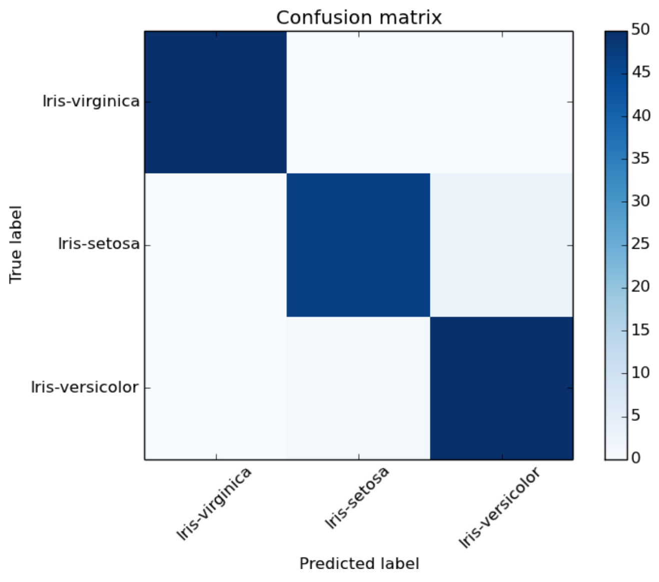


Image: Precision Recall Curve [5]

Precision-recall curves are plotted in binary classification to study the output of a classifier.

The relationship between recall and precision can be observed in the stair-step area of the plot - at the edges of these steps a small change in the threshold considerably reduces precision, with only a minor gain in recall. It is important to observe that the precision may not decrease with recall. The definition of precision $\left(\frac{T_p}{T_p+F_p}\right)$ shows that lowering the threshold of a classifier may increase the denominator, by increasing the number of results returned. If the threshold was previously set too high, the new results may all be true positives, which will increase precision.

3.3 N dimension, k class classifier



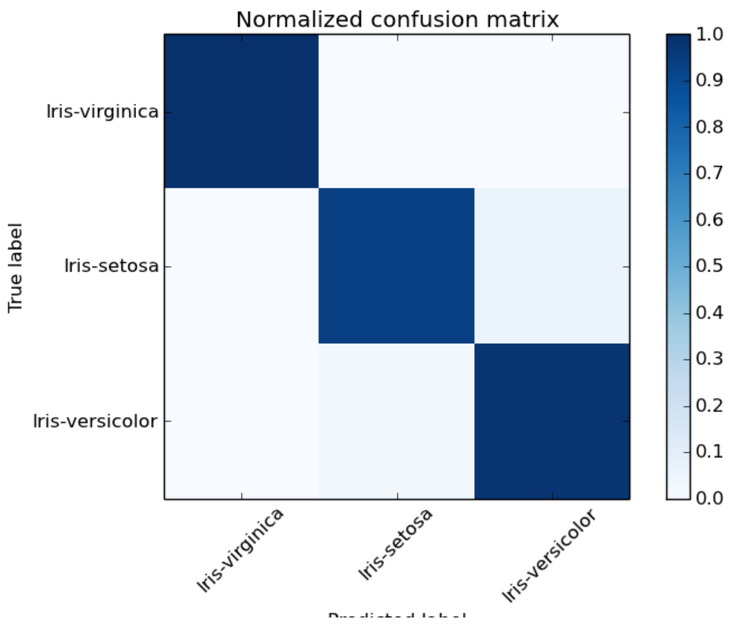
accuracy : 0.973509933775
precision : 0.943396226415
recall : 0.980392156863
F-Measure : 0.961894191639
Confusion matrix, without normalization

```
[[50  0  0]
 [ 0 47  3]
 [ 0  1 50]]
```

Normalized confusion matrix

```
[[ 1.    0.    0.  ]
 [ 0.    0.94  0.06]
 [ 0.    0.02  0.98]]
```

This classifier is 97 percent accurate, The F-measure gives a better measure of how accurate the classifier is because it takes into account both the precision and recall.



3.4 Naïve Bayes with Bernoulli Features

accuracy : 0.671158443817

precision : 0.64829030007

recall : 0.999641319943

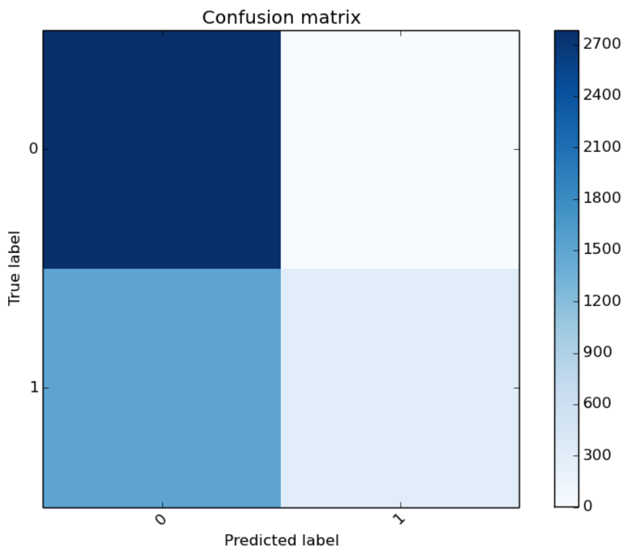
F-Measure 0.823965810006

Confusion matrix, without normalization

```
[[2787    1]
 [1512  301]]
```

Normalized confusion matrix

```
[[  1.00e+00   3.59e-04]
 [  8.34e-01   1.66e-01]]
```



References

[1] Parametric Regression definition: Link http://www.mathworks.com/help/stats/introduction-to-parametric-regression-analysis.html?requestedDomain=www.mathworks.com

[2] k- Fold Cross validation function documentation: http://scikit-learn.org/stable/modules/cross_validation.html

[3] Wikipedia page: Iris Dataset Scatter Plot https://en.wikipedia.org/wiki/Iris_flower_data_set

[4] Dataset: https://archive.ics.uci.edu/ml/datasets/Spambase

[5] Precision Recall Curve http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

$$l(\theta) = \log \prod_{i=1}^{m} P(x^{(i)} / y^{(i)} ; \theta) P(y^{(i)}).$$

$$= \log \prod_{i=1}^{m} \left[\prod_{j=1}^{n} P(x_j^{(i)} | y^{(i)} ; \theta) \right] P(y^{(i)}).$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} P(x_j^{(i)} | y^{(i)} ; \theta) + \sum_{i=1}^{m} \log P(y^{(i)})$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} \log \left[\binom{P(i)}{x_j^{(i)}} \alpha_{j|y=y^{(i)}}^{x_j^{(i)}} (1 - \alpha_{j|y=y^{(i)}})^{P(i) - x_j^{(i)}} \right]$$

$$+ \sum_{i=1}^{m} \log (P(y^{(i)}))$$

$$\theta^* = \underset{\theta}{\arg\max}\, l(\theta).$$

$$\frac{\partial l}{\partial \theta} = 0 \qquad \theta = [\alpha_{1|y=1} \dots \alpha_{n|y=1} \dots, \alpha_{1|y=k} \dots]$$

$$\underbrace{\alpha_1 \dots \alpha_k}_{\text{prior}}$$

$$\frac{\partial l}{\partial \alpha_{i|y=j}} = 0 \quad , \quad \frac{\partial l}{\partial \alpha_j} = 0.$$

$$\boxed{p(y=l) = \alpha_l = \frac{\sum_{i=1}^{m} 1(y^{(i)}=l)}{m}}$$ $l = 1 \cdots k.$

$$\boxed{\alpha_{j/y=l} = \frac{\sum_{i=1}^{m} 1(y^{(i)}=l)\, x_j^{(i)}}{\sum_{i=1}^{m} 1(y^{(i)}=l)\, p^{(i)}}}$$