**Generative Learning**

In this report we will classify some datasets by using generative learning with a Gaussian discriminant analysis and Naïve Bayes with Bernoulli features and Binomial features. We will see the results and measure the error in each of them. We will try to find which is better for our analysis, considering that we will work in two different datasets, one dataset with continuous features for the Gaussian, and another dataset with text frequency for the Naïve Bayes.

Moreover, this report contains 5 experiments that will analyze. First, in three of the experiments, we will use Gaussian discriminant analysis in the Iris dataset because it contains continuous features and different number of classes. Then, in the other two experiments, we will use Naïve Bayes with Bernoulli features and Binomial Features in the Spam dataset.

In order to make it easy to understand, each experiment has its own source file and dataset file.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Experiment 1 | Experiment 2 | Experiment 3 | Experiment 4 | Experiment 5 |
| Source File | literalA.py | literalB.py | literalC.py | literalD.py | literalE.py |
| Dataset | svar-set1.dat | svar-set2.dat | svar-set3.dat | svar-set4.dat | svar-set5.dat |

To run any of the experiments, it is just required to execute the corresponding python file.

In addition, we created a utilities file that includes some methods that are useful for almost all of the experiments, such as the calculation of the precision, recall, accuracy and f-measure. This file is imported in each of the experiments, so the methods could be used freely.

**Gaussian Discriminant Analysis**

For the experiments 1, 2 and 3, we will use the Iris dataset that contains 3 classes: Iris-Setosa, Iris-Versicolor or Iris-Virginica, and it also has 4 continuous columns that will be used as our features.

This dataset was organized by class and complicating the results with Cross Validation. Due to that, it was required to shuffle the data right after reading it from the file.

**Experiment 1**

In this experiment we take just the first column as a feature, and we remove one class from the dataset, so we could have 2 classes.

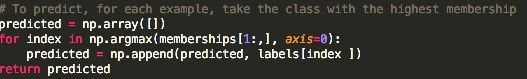
Then we proceed with executing cross validation with k=10. By doing this we evaluate the performance of our model. Basically, the algorithm was three main sections. The first is to fit the data, the second is to calculate the membership values, and the last one is to predict the classes.

To fit the data, first with an indicator we take out the examples that are relevant to a specific class. Then we calculate the mean, variance and prior class for each of the classes.

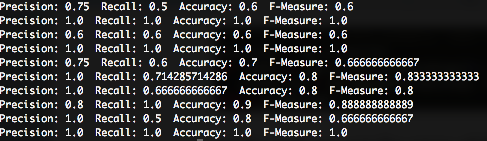
To calculate the membership, we calculate the membership for each of the examples in a specific class, by using the G formula.



Finally, to predict our data, for each example we take the class with the highest membership.



At the end, we get the error of this model over the testing data for each of the Kfolds.



We can realize that the error is too high. This is understandable because we are just taking one feature of the dataset. We did not even considered the most important feature, but we just took one randomly.

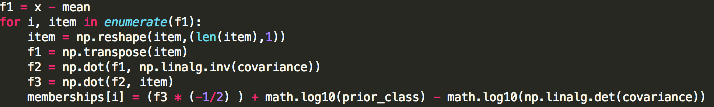
**Experiment 2**

In this experiment we take just the all of the columns as features, and we remove one class from the dataset, so we could have 2 classes.

Then we proceed with executing cross validation with k=10. Likewise the first experiment, we have three main sections: fit the data, calculate the membership values and predict the class.

To fit the data, first with an indicator we take out the examples that are relevant to a specific class. Then we calculate the mean, covariance and prior class for each of the classes.

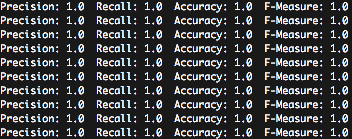
Then, to calculate the membership, we calculate the membership for each of the examples in a specific class, by using the G formula.



Here we are taking each of the examples of one class, and calculating the membership of it according to a Multivariate Gaussian analysis.

Finally, we predict in the same way as we did in the experiment 1. This is by taking the class with the highest membership for each of the data that want to be predicted.

At the end, we calculate the accuracy, precision, recall and f-measure.



We can see that all of our measures are perfect in our training and testing set. This means that there is a big difference between the values of each of them, so it is easily classified as one class or another. Due to this, the area under the curve will be one.

**Experiment 3**

In this experiment we take just the all of the columns as features, and we take all of the classes from the dataset. It means 4 features and 3 classes.

All of the procedure in experiment 3 is exactly the same as the experiment 2. First, we fit our data, then we calculate the membership according to the same formula, and finally we predict the results.

At the end, our result was the same. All of the error measures were one, which means that our classifier found a way to split our data and classify them correctly. In the same way, the area under the curve will be 1.

**Naïve Bayes**

For the experiments 4 and 5, we will use the Spam dataset that contains 2 classes: Spam or Not Spam. It also has 48 columns that will be used as our features. We will omit the lasts 9 columns because they are not relevant in our analysis.

This dataset was organized by class and complicating the results with Cross Validation. Due to that, it was required to shuffle the data right after reading it from the file.

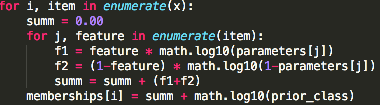
**Experiment 4**

In this experiment we use Naïve Bayes with Bernoulli features to classify the examples. The first step is to convert the frequency document to a binary document. In order to do this, we change any value greater than 0 to 1. Then we proceed to execute the cross validation with k=10. For each of the iterations, we fit our model, calculate the membership and predict the class label.

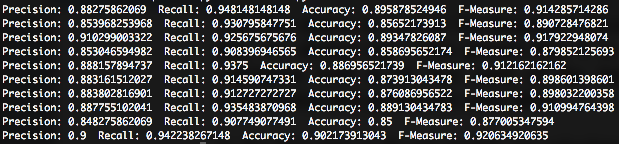
In the first step to fit our model, we calculate the parameters and the values of the prior classes. To calculate the parameters we use the Laplace smoothing of 0.01 to avoid to have a probability of zero. Then we just sum up the values of the features for a specific class and divide it by the number of examples.



Then, to calculate the membership, we use the formula of logarithm so we can sum up instead of multiply probabilities.



Finally, we predict following the same procedure from the previous experiments. At the end, the value of our error measures were:

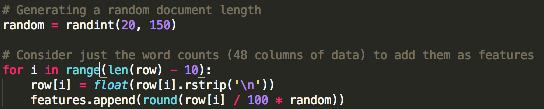


All of the error measures are pretty acceptable and do not change a lot in each fold, so it seems to be that by using binary features is enough for this dataset.

**Experiment 5**

We will use Naïve Bayes with Binomial Features to classify the same dataset from the previous experiment. By using binomial features we need to work with word frequencies, however, the dataset contains the probabilities that the word occurs in a document. Due to that, it was required to transform from a probability to an integer number.

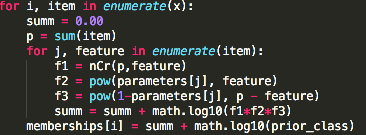
The way to do this transformation was to select a random value between 20 and 150 as the document length for each of the examples. Then we just had to multiply by the probability already in the dataset. This give us a float value, so to convert it into an integer, we just round the value.



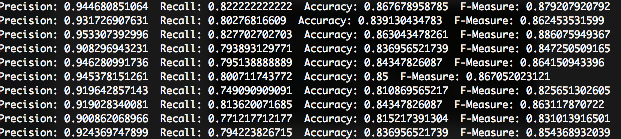
Then we follow the same procedure as before, which is to fit the data, calculate the membership and predict the class labels. Once again, to fit the data we calculate the parameters and the value of the prior class. We find the parameters by adding a Laplace smoothing of 1, and by summing up the total number of words for a specific feature divided by the total number of words of all documents in that specific class.



Then we calculate the membership by using the logarithmic formula of the Binomial features.



Finally we predict the data and print the error measures.



We can see that the recall decreased a lot compared with the experiment 4. Even, almost all of the measures decreased their value for almost 10%, making it worse classifier than the one gotten in experiment 4. We have to consider that not always is a good idea to have binary data, however, could be some words that their frequency is too high and they are not important, for instance the word ‘the’. In this case, by giving a high value to the word ‘the’, we could be skipping the important of a real word that indicate that the document was a spam, such as ‘SPAM’.