COM1011: Fundamentals of Machine Learning

Lecturer: Dr Chico Camargo [f.camargo@exeter.ac.uk]

Practical Session 5: *Naïve Bayes*

Part A – Gaussian Naïve Bayes

1. Import the `numpy`, `pandas`, and `matplotlib.pyplot` libraries.

2. Load the `iris` dataset into `X` and `y` variables.
   This dataset consists of 3 different types of *Iris* flowers (*setosa*, *versicolour*, and *virginica*) petal and sepal length, stored in a 150x4 numpy.ndarray, with the rows being the sample, and the columns being: `Sepal Length`, `Sepal Width`, `Petal Length` and `Petal Width`.
   See here for more information: https://en.wikipedia.org/wiki/Iris_flower_data_set
   Use this:

   ```
   from sklearn.datasets import load_iris

   X, y = load_iris(return_X_y=True)
   ```

3. Plot four histograms, one for the variable represented by each column. Write the name of each column on the `xlabel` of the corresponding histogram.
   Hints:
   - https://www.w3schools.com/python/matplotlib_labels.asp
   - https://stackabuse.com/matplotlib-histogram-plot-tutorial-and-examples/

Note that above we did not separate the histograms by flower type: we plotted them all together.

The information of which row corresponds to which flower is in the `y` variable, which is an array of 0s, 1s, and 2s, corresponding to the three types of flowers.

So for example, by writing `y==0`, you select all rows of `y` with the value `0`,

and by writing `X[y==0]`, you select all rows of `X` corresponding to rows where `y==0`.

4. In this question, make four figures, again one for each column in `X`, but this time plot three histograms per figure: one for each class of *Iris* flower, and each histogram with a different colour (such as red, green, blue).

Extra bits:

- To make all plots with the same bins, you can define the bins in advance, with something like `mybins = np.arange(0, 8, 0.25)`, and then adding `bins=mybins` within your `plt.hist()` function.
- To modify plot aesthetics, try adding `histtype='stepfilled'` or `alpha=0.75` within your `plt.hist()` function.

5. We are now going to train a Gaussian Naive Bayes classifier. First, import the `train_test_split` function, as well as the `GaussianNB` classifier.

Hint: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

6. Now use the `train_test_split` to split the data, with a test set size of 25%.

7. Create a `GaussianNB` classifier, fit it on the training set, and create predictions for `X_test`. Save those predictions to a variable `y_pred`.

8. The number of mislabeled points is the fraction of entries where `y_test` an `y_pred` differ. Print that fraction.

9. Import the `plot_confusion_matrix` function and run it on the Gaussian Naive Bayes classifier you've created, with the X and y variables of the test set.

10. Using the `.score()` method, print the mean accuracy of this classifier model.

(more next page)

Part B – Spam detection with Multinomial Naïve Bayes

For this part, we will be using a set of SMS tagged messages that have been collected for SMS spam research. It contains one set of SMS messages in English of 5,574 messages, tagged spam or non-spam.

For more information: https://www.kaggle.com/uciml/sms-spam-collection-dataset/version/1

1. Read the 'spam.csv' file using the pandas function `read_csv()`, and store it in a variable called `data`. Print its first lines using the method `.head()`.

The next step is to go over all SMS messages, put them all in lowercase, count the occurrence of all words, and produce a `N x K` matrix, where `N` is the number of messages, and `K` is the number of words (except for *stopwords*, like "the", "of", and "and"). In this `N x K` matrix, the `ij` entry should be the number of times the word `j` appears in message `i`. Most entries will be zero.

The code below uses the `CountVectorizer` function to produce that matrix, save it in a *sparse* format, which is a way for python to save space by only storing the non-zero entries, and then save it in a *dense* format, which is a usual matrix, with all the zero and non-zero entries.

```
from sklearn.feature_extraction.text import CountVectorizer

# set up the vectorizer
count_vectorizer = CountVectorizer(stop_words="english")

# apply the vectorizer to our text data in x
sparse_matrix = count_vectorizer.fit_transform( data['text'] )
dense_matrix  = sparse_matrix.todense()

# print the number of words / features
all_words = count_vectorizer.get_feature_names()
print("Number of features for the Naive Bayes classifier:", len(all_words))

X = sparse_matrix
y = data['spam']
```

This matrix will be the `X` input for the Naive Bayes classifier, while the spam / non-spam column of the dataset will be the `y`.

2. You have two options: either run the code and produce the matrix, or try to construct that matrix "from scratch", without using `CountVectorizer`.

3. Use the `train_test_split` function to split `X` and `y` into train and test sets. Print their shape with `print(x_train.shape)` and `print(x_test.shape)`

4. Import the Multinomial Naive Bayes classifier, create a `MultinomialNB` classifier, fit it on the training set, and create predictions for `X_test`. Save those predictions to a variable `y_pred`.

   Hint: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

5. Run the `plot_confusion_matrix` function on the Multinomial Naive Bayes classifier, with the X and y variables of the test set. Then, using the `.score()` method, print the mean accuracy of this classifier model.

Part C – Bernoulli Naïve Bayes

Instead of the Multinomial Naive Bayes, another classifier you can use is the `Bernoulli Naive Bayes`, which is designed for binary features (a word being present/absent) rather than count data (the number of times a word appears).

In that case, you need to make the counts binary variables, such that if a word exists (it's count is greater than 0), it is represented by a 1.

1. Change your `X` input variable above from integer to binary values, import the `BernoulliNB` classifier, and do the same pipeline as above:
   train, test, plot confusion matrix, print score. Which classifier performs better?

   Hints:
   - https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Binarizer.html
   - https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html