Machine Learning
Exercise 4: Applications of the Naive Bayes classifier

Prof. Dr. Thomas Kopinski

November 23, 2022

Abstract

This week you will get more familiar with two different variants of the *Naive Bayes* classifier as well as the handling and preprocessing of text data. Additionally this exercise focuses on giving you a glimpse into the automated encoding of categorical data.

Task 1:

- Please download the Jupyter notebook for this exercise from here as it contains some useful information, code snippets, help and some directions for the following tasks.
- Work through the examples in the notebook up to the section *Task 2*.
- Additional information about the *Bayes theorem* as well as the *Naive Bayes* classifier can be found in the course material.
- This Youtube video provides a great general overview over the ideas behind this classifier and its (manual) application towards a toy spam dataset.
-

Task 2: Multinomial Naive Bayes

- In this task we will implement a *Multinomial Naive Bayes* classifier to predict spam emails.
- Download the "emails.csv" dataset from here.
- Load the dataset into a dataframe and get an overview about its content.
- Visualize different aspects of the dataset (e.g. class distribution, text length of the different entries)
- Add an extra column called *length* to the dataframe where to store the length of each samples text.
- Display the shortest and longest message(s) in the email dataset.
- Look up the python package *wordcloud* and use it to visualize the data, categorized by spam or non-spam.
- Preprocess the dataframe and use *CountVectorizer* to generate the features we need for the training process.
- Classify the spam dataset with a multinomial Naive Bayes as your underlying model. It can be implemented via *sklearn.naive_bayes.MultinomialNB*. What is the score on your test data? Create a few sample "mails" of your own to feed into the trained classifier and evaluate them. Do you also have to transform these samples by applying the *CountVectorizer*?
- Create a classification report as well as a confusion matrix and display the latter by utilizing the *seaborn* packages' heatmap.

Task 3: Categorical Naive Bayes

- As the name already suggests, this variant is applied on categorical data. You will only have to deal with a very small dataset in this task as the focus lies on the general understanding of the approach to this type of data when using a *Naive Bayes* classifier.
- Download the "flu.csv" dataset from here.
- Load the dataset into a dataframe and take a look at it. Which are the feature columns, which column the target?
- Make yourself familiar with the *sklearn.preprocessing.LabelEncoder()* and use it to encode the feature columns.
- Create a new dataframe from the encoded feature columns (*zip()*) and use it to fit a *Categorical Naive Bayes* classifier (sklearn.naive_bayes.CategoricalNB).
- Generate a few input samples to feed into the classifier. Print the predictions as well as the predicted probabilities for each target class (*model.predict_proba()*).