Machine Learning
Programming Assignment I

Ujjwal Sharma and dr. Stevan Rudinac

The following assignments will test your understanding of topics covered in the first two weeks of the course. These assignments **will count towards your grade** and should be submitted through Canvas by **16.11.2018 at 12:59 (CET)**. You can choose to work individually or in pairs. You can get at most 10 points for these assignments, which is 10% of your final grade.

Submission

You can submit either a Jupyter Notebook (*.ipynb) or a python program file (*.py). To test the code we will use Anaconda Python 3.6. Please state the names and student ids of the authors (at most two) at the top of the submitted file.

1 Data

Data for this assignment is available here. In the zip file, you will find:

- A training data file titled `webStats_train.csv`.
- Other files in the zip package (with file-name of the form `webStats_test-*.csv`) contain test data.

The supplied training data consists of 280 *features* and a single positive integer-valued label denoting the number of likes for the page. The last column of the training data is the label and should be extracted before training. The test data has the same structure. You will find the `pandas` library extremely helpful in working with this data.

2 Regression

In the previous weeks, you've been introduced to the *Regression* task which attempts to model relationships between a *dependent* and multiple *independent* variables. For example, the market price of a stock (the dependent variable) can be influenced by the current financial situation, the general public perception of the company and multiple other environmental factors.

In this task, we will apply the same ideas to an interesting problem. We will use features derived from a popular news site to predict the popularity of individual news items. Since popularity is a hard metric to define, we will use regression to predict how many likes a page will get given the content and different metadata. We use a simplifying assumption that likes are a proxy for popularity.

2.1 Regression : Tasks

For this assignment, you will implement the following regressors:

1. An ordinary least-squares linear regression model. Available from `sklearn.linear_model`

✉ u.sharma@uva.nl, s.rudinac@uva.nl

2. A ridge regression model that adds L2 regularization to the ordinary least-squares model. Available from `sklearn.linear_model`

3. A lasso regression model that adds L1 regularization to the ordinary least squares model. Available from `sklearn.linear_model`

For the above models, you will perform the following tasks:

1. Fit the ordinary least-squares model to the data. Once completed, report the *mean squared error* and the R^2 coefficient of determination.

2. Fit the ridge and lasso regression models to the data. To find an optimal value for the α hyperparameter, you can use the scikit-learn grid search functionality in `sklearn.model_selection.GridSearchCV`. You will need to compute the optimal α for both models. Report the best α value from your search.

Hint: For grid search over parameters, it may be a good idea to consult the `sklearn` documentation to check the default value for that parameter and devise a suitable search strategy.

3 Classification

The likes on an article are often indicative of its popularity and by proxy user-engagement. Imagine that the domain expert decided to divide the data into three categories based on the number of likes:

Popularity	Likes
Not-Popular	0
Somewhat-Popular	1
Very-Popular	≥ 2

Table 1: Likes to Popularity Labels

Before starting with the classification task, you will need to convert the numerical like data (i.e. target variable) into class labels.

3.1 Classification : Tasks

In this section, we will focus on classifying articles into these three popularity categories based on their features. For this task, you will implement the following classifiers:

1. A k-nearest neighbor (k-NN) model. The model is available from `sklearn.neighbors`

2. A linear SVM model with default parameters. Available from `sklearn.svm`

For the above models, you will perform the following tasks:

1. Fit the k-NN model to the data. The value of k should be obtained from a grid-search. Once completed, report the *accuracy score* averaged over all test data blocks. Additionally, report the best model from your search and its parameters.

2. Fit the SVM model to the data and report the *accuracy score* averaged over all test data blocks.

Hint: For the SVM task, use the `sklearn.svm.LinearSVC` estimator instead of `sklearn.svm.SVC`. The `LinearSVC` estimator is written on a newer framework and is significantly faster.