Machine Learning
Programming Assignment II-a

Ujjwal Sharma and dr. Stevan Rudinac

The following assignments will test your understanding of topics covered in the first three weeks of the course. These assignments **will count towards your grade** and should be submitted through Canvas by **28.11.2019 at 12:59 (CET)**. You can choose to work individually or in pairs. You can get at most 6 points for these assignments, which is 6% of your final grade.

Submission

You can only submit a Jupyter Notebook (*.ipynb) for this homework. To test the code we will use Anaconda Python 3.6. Please state the names and student ID's of the authors (at most two) at the top of the submitted file.

1 Implementation Details

In this assignment, we will be looking at data preprocessing as well as classification tasks. Since they have a fixed sequence of execution, you will be required to use the sklearn `Pipeline` functionality to encapsulate your preprocessing transformations as well as classification models into a single estimator. In the following assignments, you should perform preprocessing, model fitting and prediction operations only with a `Pipeline` estimator.

Any grid search should also be performed on the `Pipeline`, not on standalone estimators or transforms.

2 Data

With this assignment, you will receive two additional files:

- A data file titled `kwb.csv`. You will need to split this file yourself. For this assignment, you can use the default split ratio in `sklearn.model_selection.train_test_split`.
- An PDF document `Toelichting-variabelen-kwb-2016.pdf` containing descriptions for demographic data attributes in the accompanying data. This file is in Dutch. If you're not a native Dutch speaker, you can use the document translation option in Google Translate to translate this document into English.

Each year, the *Centraal Bureau voor de Statistiek*(CBS) or *Statistics Netherlands* collects demographic statistics at the *gemeente* (town), *wijk* (district) and *buurt* (neighbourhood) levels. The accompanying CSV file contains an extract from that data for the year 2016. The PDF file contains a detailed explanation of the features.

3 Data Preprocessing

The CBS data provided has data fields that need to be processed before they can be used. These operations are described in this section. As before, `pandas` is an immensely helpful tool to work with

✉ u.sharma@uva.nl, s.rudinac@uva.nl

tabular data. You will need to perform the following tasks:

1. Load the CSV file. You will observe that it contains statistics at all three levels (Gemeente, Wijk and Buurt). Filter the data to only retain Buurt level statistics and drop all other rows.
2. You can now remove other textual data like area codes that may not be useful as inputs to the model. Remove the following columns

   `'gwb_code', 'gwb_code_8', 'regio', 'gm_naam', 'recs', 'ind_wbi'`
3. CBS marks missing or hidden data with a period (.) and this data will need to be imputed before a model can be used. Make sure you handle this missing data appropriately (You will learn about imputation strategies in Week 4 of the course).
4. The columns `g_ele` and `g_gas` contain the average electricity and gas consumption within a Buurt. Extract them from the dataframe. They will serve as target variables. All other columns are input variables.
5. As the average gas and electricity consumption are real-valued labels, they are better suited to a regression task. Since we're attempting to perform classification, we will need to convert these real-valued numbers into discrete labels.

   We will use quantile-based discretization for this. This discretization strategy takes a set of values and ascertains binning boundaries such that each bin has an equal number of values within it. For example, let's say you were given a set of numbers from 1 to 12

   `x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]`

   We'd like to place them into 4 bins with equal number of samples. That would be:

   `discretized_x = [0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3]`

   Each number in `x` has been allocated a label in `discretized_x`. If you carefully observe, the number of values in each bin is the same. Let's do this again for numbers 1 to 11.

   `x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]`

   `discretized_x = [0, 0, 0, 1, 1, 1, 2, 2, 3, 3, 3]`

   The label counts are not the same anymore but are still close. This binning strategy averts issues caused by imbalanced classes. Within Pandas, you can use `pd.qcut` to perform quantile discretization. For this assignment, split the data into 4 bins (quartiles)

   For this homework, you need to predict discretized labels of `g_ele` and `g_gas`. Make sure you build models for predicting both the targets.
6. Drop all rows where the label is missing. Label values cannot be filled-in and data points without a label should be dropped.

4 Models

In this exercise, you will use three components within your pipelines:

1. Feature Imputation: Real-world data often has missing or hidden values that cannot be found. Since machine learning models require a fixed number of features to predict the target variable, missing values pose serious problems. One way to work around this issue is by *filling in* missing values with a suitable value. These could be the mean/median/mode of that specific feature or some other computed value.

   Sklearn contains an in-built imputer called `SimpleImputer` which can be used within a pipeline to fill in missing values. You can use this to replace the missing values in the CBS data (marked with a period [.]). You can also experiment with the `strategy` argument to specify various imputation strategies and test which works best for this data.

2. Feature Scaling: The range of feature values can vary widely in a dataset. To bring this variation within the same scale, feature scaling is helpful. For this task, you can experiment with the `StandardScaler` and `MinMaxScaler` provided within sklearn to scale your data.

3. Classification: The last component of your pipeline will be an estimator. In this homework, you will be required to use the `LinearSVC` and `LogisticRegression` classifiers. For these classifiers, you must perform the following experiments:

   (a) Create a `LinearSVC` classifier with default parameters. Fit your classifier on scaled as well as unscaled versions of the data and report the estimator scores. Additionally, analyze the effects of data preprocessing on the classification performance.

   In no more than 50 words, explain your observations and your assessment of the underlying situation. You can use a text box (i.e. Markdown Cell) in Jupyter to write down your analysis.

   (b) Create a `LogisticRegression` classifier. Additionally use `GridSearchCV` to find an optimal value for the parameter `C`.

   As before, you will be required to run your estimator on scaled as well as unscaled versions of the data and report your observations for the estimator score. Additionally also analyze the effects of regularization strength on the performance of the estimator [1]. In not more than 50 words, explain your observations and your assessment of the effects of `C` and data scaling on the classification performance.

   For each of the pipelines, you must report classification metrics like accuracy, recall, F1 and micro/macro averaged precision statistics and present your observations on their implications for each of the models. These metrics will be discussed at the beginning of Week 5.

5 Grading

Component	Points
Imputation	1
Scaling	1
Classifiers	1
Hyperparam Optimization	1
Experiments, Observations, Analysis and Code Quality	2

[1] The `LogisticRegression` hyperparameter C represents the *inverse* of regularization strength. So a smaller value of C is, in fact, stronger regularization.