Assignment 1

Data Preprocessing (Python)

The objectives of this assignment are:

1) To practice the design of a basic knowledge discovery process.
2) To try different types of data pre-processing.

Preliminary of Language/Software: You can choose Pycharm or Jupiter Notebook (or others that you have been familiar with) as your Python IDE. To install packages, use command `pip install [package name]` in the Windows' or Mac's terminal.

TASK 1: visualising data

1.1) Read the iris_data.csv and iris_labels.csv files via python and preprocess it. These files can be found on Studium. The Iris dataset includes information about three iris species known as Setosa, Versicolor and Virginica. Each instance includes some information about the shape of the flower including *sepal length*, *sepal width*, *petal length* and *petal width*. As a result, our dataset includes 150 data points/record that have 4 features/attribute and each input is associated with a class, i.e., Setosa, Versicolor, Virginica. The first 4 features are in the iris_data.csv file and the label of each record can be found in the iris_labels.csv file

Import `pandas` and `numpy` packages with the `import` command and use `pandas.read_csv()` to read the files.

Check the size of the data tables and print the first 5 rows with `print(data.shape)` and `data.head()`.

1.2) Join the two csv files into a single data frame. Use `pandas.merge(data, labels, on = 'id', how = 'inner')`.

1.3) Filter out the attribute "examiner", that is not needed for the analysis. Use the `data.drop(['id'], axis = 1, inlace = True)` command.

1.4) Order the data frame by the "species" attribute. Use the `data.sort_values()` command.

1.5) You can use the seaboard package to see the scatter plot of the dataset. Use the `pairplot(data, hue = "Species")` command.

TASK 2: data cleaning

During the previous task you should have noticed that some observations have suspicious values, that we should take care of.

2.1) Missing values have been coded using -9999 in the input file. You can filter out the observations containing missing values. Use the `data[data["attribute_name"] == -9999]` command.
2.2) Use again the scatterplots to see if there are outliers. If there are, remove them.

TASK 3: data transformation
The objective of this task is to transform the features describing the objects. We start making the scale of the numerical attributes uniform.
3.1) Perform a min/max normalization on the range [0,1]. You can use the `min` and `max` commands in `numpy`. The `sklearn` package has a specific command for this.

```
from sklearn.preprocessing import MinMaxScalar
scaled = MinMaxScalar().fit_transform(data)
```

3.2) Using the same commands and the `sd` command in `numpy`, perform a standardization (in some references it is called z-transformation). The `sklearn` package has a specific command for this.

```
from sklearn.preprocessing import StandardScalar
scaled = StandardScalar().fit_transform(data)
```

3.3) Apply PCA to the current data and view the scatterplot of the new data set.

```
from sklearn.decomposition import PCA
pca = PCA()
principalComponents = pca.fit_transform(data)
```

3.4) Check the contribution of each attribute on the resulting components.

```
pandas.DataFrame(pca.components_, columns=["Sepal L", "Sepal W", "Petal L",
"Petal W"],index = ['PC 1','PC 2','PC 3','PC 4']).abs().mean(axis = 0)
```

Also, check which original attributes contribute to each component (under Eigenvectors).

3.5) Do the same as in 3.4, but applied to a modified dataset where you rescale the *pl* attribute to the range [0, 100]. Inspect the definition of the new components.

3.6) Do the same as in 3.4, but applied to a modified dataset where you add an outlier (noise). For example, modify the value of attribute *pl* in the first record to 5000 using the *Set data* operator. Inspect the definition of the new components.

Now that you have checked the effect of different value ranges and outliers on PCA, you can continue from the result of 3.4.

TASK 4: sampling
As a last preprocessing step, you will try and compare different types of sampling.

4.1) Sample 150 instances (that is, flowers) uniformly at random. Use the `sample` command in `pandas`.

4.2) Sample 150 instances using bootstrapping. Use the `sample` command in `pandas`.
4.3) Compute a stratified sampling of half of the instances, with the sampling probability proportional to the size of the species. You can use the following command, but try to understand it:

```
data.groupby('Species', group_keys=False).apply(lambda x:
x.sample(frac=0.5))
```

4.4) Compute a stratified sampling of 150 instances, including 50 instances for each species. You can use the following command, but try to understand it:

```
data.groupby('Species', group_keys=False).apply(lambda x: x.sample(50))
```