Data Pre-Processing Exercises

In the first exercises you work on a dataset, related to direct marketing campaigns of a Portuguese banking institution. The marketing campaigns involved phone calls to clients to try and get them to subscribe to a particular product. The dataset contains the details of each client contacted, and whether they subscribed to the product.

<u>The data set needs some cleaning before being useful for analysis and prediction.</u>

1. Load the Banking data from the following location into a data frame.

   https://github.com/TrainingByPackt/Data-Science-with-Python/blob/master/Chapter01/Data/Banking_Marketing.csv

   Use pandas' function read_csv()

2. Explore the data frame, discovering its shape, data types and values. Use pandas variables `shape`, `columns`, `dtypes` and functions `info()`, `head()`, `sample()`.

   The dataset includes age, job, marital status, education level, credit default status, average yearly balance, and information on housing and personal loans of the client. Data related to the last contact of the current campaign includes contact type, month, day, and duration of the call. Other attributes cover the number of contacts during the current and previous campaigns, days passed since the last previous contact, and the outcome of the previous campaign. It also includes general social and economic context attributes like employment variation rate, consumer price index, consumer confidence index, Euribor 3-month rate, and number of employees. The target variable, y, indicates if the client subscribed to a term deposit or not.

Data Cleaning

3. Find out if there are missing values in the data. Use the functions `isna()` and `sum()`.

4. There are several rows with missing values in different columns. We can process them in several different ways:

   Method A: Remove all rows with missing values, use the function dropna()

   Method B: Replace the missing values with

   - the average of their column (for the numeric data)
   - the mode in their column (for the categorical data)

   Hint: First calculate the average `mean(col)` or the `mode(col)`

   Example of finding `mode`:

   ```
   cmode=mode(contact)[0]
   df.contact.fillna(cmode, inplace=True)
   ```

   Alternatively, the missing values can be replaced by the average of their neighbors of group.

Data Transformation

Categorical to Numeric

5. Most analytical methods need the data in numeric format. Therefore, we need to
   a. find out which data needs transformation
   b. transform categorical data into numeric
   c. transform continuous numeric data into discrete

   Again, we have multiple options to identify and transform categories into numbers. The simplest is to just replace one category with one integer.

   To select the categorical columns only, we apply another pandas function, `select_dtypes()`

   ```
   df_categorical = df.select_dtypes(exclude=[np.number])
   ```

   Then, we find the unique values in each categorical column:

   ```
   df_categorical['Risk'].unique()
   ```

   and count the frequency of appearance of these unique values:

   ```
   df_categorical.Risk.value_counts()
   ```

   Finally, replace the discovered categorical values with numbers:

   ```
   df_categorical.Risk.replace({"Low":0,"High":1}, inplace= True)
   ```

   Alternatively, we could have replaced all categorical values at once by a method of `LabelEncoder` class, like in Titanic example.

6. In many cases, the best encoding method is the so called one-hot encoding. According to this method, one categorical column is replaced by as many other columns, as the unique values in it are. For example, the column `Risk` will be replaced by two columns: `Risk_Low` and `Risk_High`. Depending on the original Risk category, one of the replacing columns will get a value of `1`, while the rest will be `0`.

   Here is how to do it:

   ```
   df_onehot = pd.get_dummies(df['Risk'])
   new_df = pd.concat([df, df_onehot], axis = 1)
   new_df.drop('Risk')
   ```

Continuous to Discrete

7. In some tasks is helpful to replace continuous data with discrete by dividing all continuous values in groups (bins) and encode each bin with one value. We use the function `cut()`:

   ```
   labels = ['Child','Teen','Adult']
   bins = [1, 12, 19, 100]
   df['age_group'] = pd.cut(df['age'], bins=bins, labels=labels
   ```

Feature Engineering

Normalisation and Scaling

Scaling and normalisation are operations, which change individual numeric features in a way, so they share similar measurement scales and ranges. It ensures that each feature has proper effect on the model in comparison with the other features.

In this exercise, you will load the `Wholesale customer's data.csv` dataset, which contains data about clients of a wholesale distributor, with their annual expenses, spent on various product categories:

https://github.com/TrainingByPackt/Data-Science-with-Python/blob/master/Chapter01/Data/Wholesale%20customers%20data.csv

8. Load the data in `pandas` DataFrame, as you loaded data in previous exercises.

   ```
   df =
   ```

   Explore `df` to see the difference in scales of, for example, the spending on `Fresh` products and `Frozen` products. Just because one feature contains smaller numbers than the other does not mean it is less significant for the analysis.

   The feature scaling, standardization, and normalization are three similar ways of solving the difference issue.

9. Apply `MinMaxScaler` to transform all individual features, so that they all fit into the same range `[0,1]`.

   The method achieves this by converting the smallest value in the column (`min`) with 0 and biggest value (`max`) with 1, then all values in between are transformed in proper proportion of the interval (`max-min`).

   The method `fit_transform(df)` is in the class `MinMaxScaler` from `sklearn.preprocessing` library.

   Import the class, create an instance and call the method to perform the normalization.

10. See the result and compare it to the initial data frame.

    Observe and explain the new values.