CS 5310 Data Mining

Lab 2 – Chapter 6 Forecasting Numeric Data – Regression Methods

Strain of Bacteria Identification with Mass Spectrometer Data

In this lab, we will use regression tree method to analyze protein and peptides data from a mass spectrometer system and build a model to analyze the relationship among proteins. We give some background information about the project as follows.

The project is about mining of big data on molecules making up and produced by beneficial microbes found in human. Currently, scientists are interested in studying microbiota that produce molecules involving in mind control (see <https://www.youtube.com/watch?v=mioR_WrkRaU> ). There is a state-of-the-art mass spectrometer system, Bruker's Tims TOF Pro, which can identify over 3,500 proteins, 20,000 peptides in about 15 minutes. It can also identify metabolites (molecules produced by living organisms including microbiota in human’s gut) that are important to health, including our mind, mood and mental health (more than gut feelings).

While the instrument is powerful in producing huge volume of data (about 3-5 G in a 20-minute run), data mining becomes a bottleneck, which is needed to improve the instrument performance, answer biological questions, and direct the development of analytical methods. The instrument comes with a third-party software (Peaks studio) that converts raw data into molecular structures. However, we need to extract much more information from the raw data by using machine learning means.

The data we will use for this lab are saved in a xlsx file named “Data\_SubjectStripTearSamples-Dec2020\_Report.xlsx”, which contains experimental data from 22 subjects, in columns named “[*X*] Strip-No*X*\_Slot*X*-*XX*\_*XXXX*.d.PG.Quantity”, where italicized *X*’s are numbers. Data in each row are identity information about one protein and its mass spectrometer experimental data got from those 22 subjects. In this lab, we only use mass spectrometer experimental data from 22 subjects, so, all protein identity data should be removed, except for the column named “PG.ProteinGroups”, which we will use as names of the proteins. In other words, we should remove all columns except those 22 with a name in pattern “[*X*] Strip-No*X*\_Slot*X*-*XX*\_*XXXX*.d.PG.Quantity”, and the column named “PG.ProteinGroups”.

Please note that the dataset we use in this lab is different from the one used in the first lab of this chapter.

Firstly, we need to fill-in entries where data are “filtered”. These are the entries that have a string “filtered” instead of a numeric value. Because of this string, the data read into the data frame could be typed as “string” instead of a numeric type. We need to replace those “filtered” strings by value 0, and convert the type of columns to floating point type, if necessary.

We are interested in analyzing the relationship among proteins. So, transpose the matrix so that data for each protein becomes a column, and the name of protein in “PG.ProteinGroups” becomes the column label. Note that after the transposition, the matrix should have 22 rows.

Identify proteins that have high correlations (>0.8 or <-0.8), and remove redundant proteins (i.e., for columns that have high correlation, retain only one of them). You can adjust the threshold of high correlation to a lower or higher value to control the number of columns you want to retain for modeling.

Normalize every remaining column using min-max normalization.

Use the data in the first column (the first protein), as the dependent variable, and all other columns as independent variables, and build a linear regression model to analyze the relation among the proteins.

Note that we do not split the dataset for training and testing in this lab. Instead, we train the model using the entire dataset and predict the results for the entire dataset.

Perform the following activities in Python.

1. Load the data in the xlsx file into a data frame.
2. Remove columns except those 22 with a name in pattern “[*X*] Strip-No*X*\_Slot*X*-*XX*\_*XXXX*.d.PG.Quantity”, and the column named “PG.ProteinGroups”.
3. Zero fill-in entries with a string “filtered” and convert the type of columns to floating point type, if necessary.
4. Transpose the data frame so that data for each protein becomes a column, and the name of protein in variable “PG.ProteinGroups” becomes the column label. Please note that after this step, there should not be a row corresponding to column “PG.ProteinGroups” in the data frame before transposition.
5. Remove collinearity among the columns (i.e., proteins): For columns that have a correlation coefficient that is higher than 0.8 or less than -0.8, only one of them is kept in the data frame, and all others are removed. Adjust the threshold of high correlation to a lower or higher value to control the number of columns you want to retain for modeling.
6. Normalize every remaining column using min-max normalization.
7. Create a line plot that displays the experimental data of the first protein (or first column of the data frame), and the average of the experimental data of all the other proteins, using different colors.
8. Create a correlation coefficient matrix of all the columns.
9. Train a DecisionTreeRegressor model in the “sklearn.tree” package using the entire dataset. The dependent variable is the first column of the dataset, and all others are independent variables.
10. Create a visualization of the tree.
11. Use the model to make predictions for the entire dataset.
12. Compute the correlation coefficient and mean absolute error (MAE) between the true values of the first column and the predicted values.

Submit your codes (.py file) and results, including the line plot (Step 7), correlation coefficient matrix (Step 8), the visualization of the DecisionTreeRegressor model (Step 10), and the correlation coefficient and mean absolute error (MAE) between the true values of the first column and the predicted values (Step 12), in a Word document, by the due date.