University of Glasgow | School of Computing Science

CSC3005 Laboratory/Tutorial 2: Data Preprocessing and Classification Analysis I

1. **Aggregation** is a preprocessing task where the values of two or more objects are combined into a single object. The motivation for aggregation includes:
    a) reducing the size of data to be processed,
    b) changing the granularity of analysis (from fine to coarse scale)
    c) improving the stability of the data.

    The raw daily precipitation time series data for a weather station was obtained at https://www.ncdc.noaa.gov/cdo-web/datasets. It contains the daily precipitation time series data from year 2001 to 2017

    Load the raw T2Q1_precipitation.csv into a Python Pandas Data Frame object and perform data visualization with line plot for the daily time series data. Obtain the variance of the data using dataframe.var() method. What have you observed?

    Perform aggregation and data visualization for the following two cases.
    a. Aggregate the time series data into monthly using dataframe.groupby () method and obtain the variance
    b. Aggregate the time series data into yearly a dataframe.groupby () method and obtain the variance.

    What can you conclude on the result of aggregation?

2. **Sampling**

    Sampling is an approach commonly used to facilitate
    a) data reduction for exploratory data analysis and scaling up algorithms to big data applications
    b) quantifying uncertainties due to varying data distributions.

    There are various methods available for data sampling and we will cover
    a) sampling without replacement, where each selected instance is removed from the dataset
    b) sampling with replacement, where each selected instance is not removed, thus allowing it to be selected more than once in the sample.

    Load clinical T2Q2_breast_cancer.data (same file as T1Q2_breast_cancer.data) into a Python Pandas DataFrame object and show first five records for simplicity

a. Sampling without replacement

I. Use the dataframe.sample() method to obtain a sample of size 7 from the original set of 699 dataset. Display the sample data and run the method few times to observe the randomicity.

II. Use the dataframe.sample() method with *frac* argument to randomly select 1% of the data and display the sample data. *random_state* argument set to 1 to ensure the reproducibility of the examples.

b. Sampling with replacement

I. Use the dataframe.sample() method with *replace* argument to randomly select 1% of the data with replacement and display the sample data. Increase the sample size to observe the duplicability of the sample data.

3. Discretization

Discretization is a data preprocessing step that is often used to transform a continuous-valued attribute to a categorical attribute. There are two simple but widely-used unsupervised discretization methods namely

a) equal width
b) equal depth

a. Equal Width

Use dataframe.hist() method with bin size of 5 and 10 for attribute 'Single Epithelial Cell size' and plot the histogram. Count the value into each bin using dataframe.value_counts() method.

b. Equal depth (frequency)

Use pandas.qcut() method partition the values into 3 bins such that each bin has nearly the same number of instances for attribute 'Single Epithelial Cell size' and plot the histogram. Count the value into each bin using dataframe.value_counts() method.

4. Principle Component Analysis (PCA)

Principal component analysis (PCA) is a classical method for reducing the number of attributes in the data by projecting the data from its original high-dimensional space into a lower-dimensional space. The new attributes (also known as components) created by PCA have the following properties:

a) they are linear combinations of the original attributes
b) they are orthogonal (perpendicular) to each other
c) they capture the maximum amount of variation in the data.

Apply PCA on image files

Step 1: Load and display the images

I. Unzip clinical T2Q4_pics.zip. There are 16 RGB files in T2Q4_pics folder, each of which has a JPEG size of 111 x 111 pixels.
II. Import the *mpimg* method from *matplotlib.image* library and use *mpimg.imread()* to read and then use *imshow()* method from *matplotlib.pyplot* to display all the 16 images

Step 2: Create an array to store the 16 pictures

I. Use *numpy* array to create a 16x(111x111x3)=16x36963 dimension array
II. Flatten and reshape the 16 images into 16 1-dimensional images and stores into the *numpy* array created in part (I) using *flatten()* and *reshape()* method from *mpimg*

Step 3: Find the principle component (PCs) of the numpy array data that stores the 16 1-dimensional images

I. Import PCA method from sklearn.decomposition module https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA.fit
II. Use the *fit()* and *transform ()* method from PCA to mean normalize and find the PCA projection of the array data
III. Display up to 5 PC components

Step 4: Determine the number of PC components required to represent the whole data

I. Find the explained variance through *pca.explain_variance_*. It represents the eigenvalues of the associated eigenvector of the array data in descending variance power
II. Find cumulative variance and hence determine what is total number of principle components (eigenvectors) to represent the data.

Step 5: Use two PC components to display the data sufficiency and determine its error if any

5. Theory on PCA

Given a data matrix $\mathbf{x} = \begin{bmatrix} 4 & 2 \\ 1 & 8 \end{bmatrix}$ where column represents feature and row represents number of data record. Derive out the PC components of this data matrix $\mathbf{x}$. Obtain the projection data using just 1 PC component. (No need to do coding)

6. Decision Tree Classifier

We use a sample of the vertebrate data to demonstrate Decision Tree Classifier. Each vertebrate is classified into one of 5 category class:

1)mammals
2)reptiles
3)birds
4)fishes
5)amphibians,

based on a set of qualitative attributes (predictor variables). Except for the attribute "*name*", the rest of the attributes have been converted into a one hot encoding binary representation.

Explore the data with visual inspection

Step 1: Load the vertebrate.csv and display the content using pandas.read.csv() method

Step 2: Convert the 5 classes to either mammal and non-mammals namely reptile, birds, fish and amphibian will be classified as non-mammals. Use the *dataframe.replace ()* method.

Step3: Examine the relationship among warm blooded, give birth attributes with respect to the class using *pandas.crosstab()* method. Observe any conclusion

Explore the data with Decision Tree Classifier

Train the dataset

Step 1: Import *tree* module from *sklearn.tree* module. Use *tree.decisiontreeclassifer() class. Set the entropy as the impurity measure with max depth of the tree as 3 to prepare the data training.*
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Step 2: Train the data using fit() method. You can drop the attributes "Name" since during training using *dataframe.drop() function.*

Step 3: Display the tree structure of the training set by using the *tree.plot_tree()*.

Test the trained decision tree model

Step 1: Load the testing dataset given as follows:

Name	Warm - blooded	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiberna tes	Class
Gila monster	0	0	0	0	1	1	non-mammals
platypus	1	0	0	0	1	1	mammals
Owl	1	0	0	1	1	0	non-mammals
Dolphin	1	1	1	0	0	0	mammals

Step 2 : Predict the class of the testing datatset with the trained decision tree model obtained earlier on. Use the *predict()* function. Hence use the *pandas.concat ()* method to display the two columns, namely, the testing data attributes *"Name"* and the predicted class to match the table follow to assess how accurate is the decision tree model on unseen data.

Step 3: You can check the accuracy of the decision tree model on the true class versus predicted class by using sklearn.metrics accuracy score module.
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score

Print out the accuracy between the true and predicted class.

7. Theory on Decision Tree Classifier

Without coding, work out the decision tree classifier using the same dataset in Question 6 that comprises of 15 data based on the theory learned in the lecture. Verify your decision tree model and the entropy impurity measure match the solution with *tree.decisiontreeclassifer() class used in* Question 6.