CS3DS19 - Data Science Algorithms and Tools

Major Coursework

2021/2022

Student ID: 27020363

March 23, 2022

Notes to the reviewer

- On folder ’extras’, a version of this document, without figures and proofs may be found. This is to proof that this document is not exceeding 4 pages of text excluding references, title page , Notes to reviewer page and figures/proofs.
- After email correspondence with lecturer, proofs are also excluded from the 4-page limit, just like figures.
- Although written in latex, this document uses standard Microsoft Word margins and paddings.

Task 0: Exploratory Data Analysis

In this section, we will try to familiarise ourselves with the given dataset, with the ultimate goal of designing an appropriate pre-processing process necessary for our future workflows. It is noteworthy that a normalisation process is absent from our EDA; Normalisation will be explained and applied on Task 2.

Missing values detection

The first part of EDA is the detection of possible Missing values. We used the 'statistics' node (output contains 'No of Missing' variable) and sum for all the variables by using 'Group By' Node. The results, along with the node configurations, are given below. Based on the findings, we can safely ignore a missing values pre-prepossessing step.

Workflow and node configurations for missing values detection

Duplicate detection

The second part of EDA detects and removes duplicate rows from our dataset. Our logic is relatively simple, we applied a 'Duplicate Row Filter' into the dataset, and we counted the number of rows of this result. If the number of rows remains unchanged, we do not have duplicates. The results suggest that, indeed, there are no duplicate entries, so it's safe to ignore a duplicate filter as a pre-processing step.

Workflow and node configurations for duplicate detection

Outliers detection

Our third part of our EDA contains the detection of outliers. We need to carefully think about outliers because of one of the limitations of our chosen clustering algorithm of choice. One of K-Means limitations are the sensitivity to outliers. [4].

The process

Assuming an underlying approximate normal distribution on every one of our variables, we will use the famous 68-95-99 empirical rule[1] to detect outliers. Part of the 68-95-99 rule states that 99.7% of the data points in a given approximately normal distribution lies within the range $\mu \pm 3 * \sigma$, where μ is the variable mean and σ is the standard deviation. It is rather simple to detect outliers with the following method. We will use the output of the 'Statistics' node(columns 'max','min','mean','stddev'), and by using a simple mathematical formula, we can determine if, for a given variable, max $> \mu + 3 \cdot \sigma$ or min $< \mu - 3 \cdot \sigma$

Results

The next figure displays the variables that may[1] contain outliers. We can visualize those suspected variables with box plots to visually verify our findings.

[1]because of our assumption of approximately normal distribution on our variables.

Workflow and node configurations for outliers detection

'Class' to string

The last step of our EDA is to transform the 'class' column, from an integer column, into a string column; this is essential to be handled as a categorical variable by the colour manager, PCA, normalization, etc. This can be easily achieved with the following small workflow. and the node configuration is given below

Our Pre-processing strategy

Given our EDA, we now have a clear strategy for the initial data cleansing and processing. The following workflow will be transformed into a meta node and included in every workflow from now on. A noteworthy mention is that the 'numeric outliers' strategy is to remove the outlier rows.

Task 1

Task 1.1 : Generate plot1

The workflow to generate plot1 is given below. And the node configurations with descriptions are given below. Finally, plot1 is given below.

Task 1.2 : Generate plot2

The workflow to generate plot2 is given below. Apart from the clustering algorithm adopted(K-Means), there are no significant changes with the workflow from the previous task; please see task1.1 for an overall workflow review. The Newly introduced nodes configurations, as well as the descriptions, are given below.

Finally, plot2 is given below.

Introduction to K-Means clustering algorithm

K-Means is a heuristic method for grouping similar items in clusters. K is the number of clusters and is pre-determined. By heuristic, we mean that the algorithm does not produce absolute optimal solutions but approximations with a given accuracy. By similarity, on purely numeric data, we usually mean a geometric distance metric. Three of the most widely accepted geometric metrics for determining similarity are Euclidean, Manhattan and Minkowski distances.

K-Means Algorithm

K-Means is a heuristic algorithm that repeats two distinct steps to converge into an acceptable solution. We will explain K-Means with a hands-on example. Let the following fictional 1D Data, $k = 3$. Let initialize $k = 3$ initial random estimated cluster centroids. Now, the repetitive process, First, we should calculate the distances of any given point from each of the $k = 3$ currently estimated cluster centroids. Then, we should assign the given datapoint into the cluster with the closest centroid. Here X_1 will be assigned on the blue cluster(M_1). After the calculation is finished for all the data points, the clustering should look like the following figure. We now calculate the centroids of the clusters. The centroids do not need to be actual data points; they can lie anywhere on the domain space. We repeat steps 2 and 3 until we end up with an acceptable solution. We can quantify the quality of a solution(i.e. quantify if the given solution is acceptable) by using the Sum of Squared errors statistic(SSE). K-Means terminates when SSE converges into a value. SEE will be explained in detail on Task 1.5.

Task 1.3 : Compare plot1 and plot2

The main point of interest in the figures above is the fact that the clustering algorithm(K-Means, k=3) failed to appropriately partition the data and recognize the classes provided by the dataset. This phenomenon can be explained due to the absence of the normalization process, something that will be explained in the next task. The exact scale of the problem cannot be appropriately assessed with only those two plots, though we will need to examine the distribution of the classes in each cluster, something that will be done on the next task (Task 1.4).

Task 1.4 : plot3a,plot3b,plot3c

The workflow to generate the requested plots is given below. The majority of the workflow is identical to task1.1 and task1.2, so please refer to those tasks for detailed explanations of those nodes. The major difference is in the display section. There, we use 3 row filter nodes(one per cluster) to filter the assigned data points on each cluster. Hence in the first plot, we have all the data points assigned to cluster0 and vice versa. The node configurations with the necessary filters are given below. Finally, the generated plots are given in the next figure. It becomes apparent that the clusters generated compose a meaningless clustering solution. Under ideal circumstances, each cluster will have a vast majority of each of the classes, with some to no variation due to outliers or errors; using histograms, we can visualize the extent of the issue This is the result of the lack of a normalization process, something that we will explain on the next task.

Task 1.5 : Cluster Validity Measure

For our cluster validity measure, we will explain and interpret WSS (Within cluster sum of squares) and BSS (Between cluster sum of squares). The workflow for generating the cluster validity measures is given below. The majority of the workflow is identical to workflows of task1.1 to task1.4, so please refer to those tasks for a detailed explanations of those nodes. The Major Difference is on the repeat process and on the intepretation proccess. The newrly introduced nodes configurations are given below, along with a short explanation

WSS and BSS Explained, Prerequisites

Before we attempt explaining WSS and BSS, it is essential to explain SSE (Sum of Square estimate of Errors). SSE is an evaluation metric with a plethora of applications in statistics and predictive analytics[7]. It is used to evaluate a model's performance against a training set. The logic behind SSE is rather simple. Let X a random variable and a model $y = \bar{X}$. We can evaluate our model's performance by calculating the Sum of Errors, where 'error', in this case, is the distance of the observed variable from the response of our model (i.e. the mean $\bar{X}$), as follows Unfortunately, our evaluation metric has a fatal flaw; it allows for error's to 'cancel out' simply because they are placed beneath our model's line. It can be proven that for the case of $y = \bar{X}$ SE is always 0. For other models(in higher dimensions, where the correlation of the involved variables is not 1), it may not be zero, but the fatal flaw remains; by cancelling errors, we severely underestimate the errors involved. A more appropriate approach could be to square the distances, which would give us the Sum of Squared Errors (SSE).

WSS and BSS Inner-Workings

In the world of Clustering models performance evaluation, SSE is translated into two distinct but related ideas, WSS and BSS. WSS or Within Cluster Sum of Squares is a clustering model performance evaluation function. Our 'model' in this case is the cluster centroid, and the error is the distance for each data point from the cluster centroid. WSS can be evaluated with the following formula[7] Where k, the number of clusters involved, C_i the individual cluster and m_i, the given cluster centroid. BSS on the other hand, or Between Cluster Sum of Squares, is a clustering model performance evaluation function, just like WSS. Our 'model' in this case is the overall data mean, and the error is the distance of each cluster's centroid to the overall mean. BSS can be evaluated with the following formula[7] Where k, the number of clusters involved, C_i the individual cluster and m_i, the given cluster centroid.

Interpretation

A good clustering solution would involve well-separated clusters with low WSS to BSS ratio[**?**]. As K-Means is a heuristic-based algorithm, we will not analyse WSS and BSS of a single K-Means run, but rather we will analyse their behaviour after a sequence of 100 runs. It is rather evident that the $\frac{WSS}{BSS}$ Ratio is quite small, something that indicates well-separated clusters(i.e. a good clustering solution). However, a single cluster validity measure is insufficient to provide enough evidence for a meaningful clustering. Considering our observations from Task 1.3 and Task 1.4, this clustering is not meaningful, as it produces wrong clusters. This is something that can be explained due to the absence of a normalisation process, something that we will analyse on Task 2.

Task 2

The importance of Normalization

Let the following figure with some of the 'Proline' and 'Ash' variables statistics. K-Means is a geometric algorithm that uses a distance(proximity) metric to evaluate the similarity between data points. Having one or more variables with a significantly greater range of values will cause this variable to have a more significant influence on the clustering, producing biased results.

How Normalization solves the problem.

Normalization solves the issue as mentioned above by transforming all the variables in question, within the range [0-1], without affecting the scales of the distances. Normalization is described with the following formula.

Changes in the workflows

The only change to apply normalization is on our pre-processing node. Everything else will be the same. Noteworthy is that the 'normalization' node follows the 'min-max' normalization, with min=0 and max=1. Normalization only makes sense on the numeric and not on categorical variables; hence 'class' variable is automatically excluded.

Plots with enabled normalization

Compare and Contrast

As we can see from the newly generated plots, K-Means can now recognise the provided classes with little to no error. The classes per cluster distributions are now dominated by one class, meaning that every cluster contains the majority of one class, plus some errors(something expected, as our model is not perfect). Our clustering validity measure indicates a well-separated solution with a very low WSS and very high BSS. Our cluster validity measure, along with the new plots3a-c, indicates a well separated and meaningful clustering.

References

[1] *Grafarend, E. (2006). Linear and nonlinear models (p. 553). Berlin, New York, N.Y.: Walter de Gruyter*

[2] *Shannon, C., 1948. A Mathematical Theory of Communication. Bell System Technical Journal, 27(3), pp.379-423.*

[3] *Shanker M, Hu MY, Hung MS, Effect of Data Standardization on Neural Network Training* https://www.sciencedirect.com/science/article/pii/0305048396000102

[4] *Sammut, C., n.d. Encyclopedia of machine learning and data mining. Springer.*

[5] *Luke, S., n.d. Essentials of metaheuristics. 2nd ed. lulu.com.*

[6] *KNIME Community Forum, 2022. K-Means initialization. Available at: ¡https://forum.knime.com/t/k-means-initialization/1282¿ Accessed 22 March 2022*

[7] *Grami, A., n.d. Probability, random variables, statistics, and random processes. Kappa Research, LLC (August 24, 2014).*