2016

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Anomaly Detection API

SOFTWARE ENGINEERING

SANI KIWAN   MATR# 1145786

April 22, 2016

I. Introduction

Anomaly Detection API is a project that clusters given data based on their similarities. This project uses the K-mean algorithm to detect similarity which results on depending on Euclidean distances and thus we can only deal with numeric data. This Project has two main parts:

- Part 1: take samples and cluster them. In this phase the program will learn about the clusters and thus can be considered as the learning phase.
- Part 2: based on the clustering results that we obtained from part 1, the program should detect to which cluster a new given sample belongs if any.

II. Objectives

To fulfill the tasks required and mentioned in the introduction we were asked to implement the following features:

- Import new Data and cluster it.
- Check the sample whether it is an outlier or belongs to a cluster
- Import and append new Data to an existing clustered data. Here only the accepted samples (not outliers) are added and the rest is ignored.
- Calculate Statics about the clusters.
- Find a method to recommend the number of clusters for a given data set

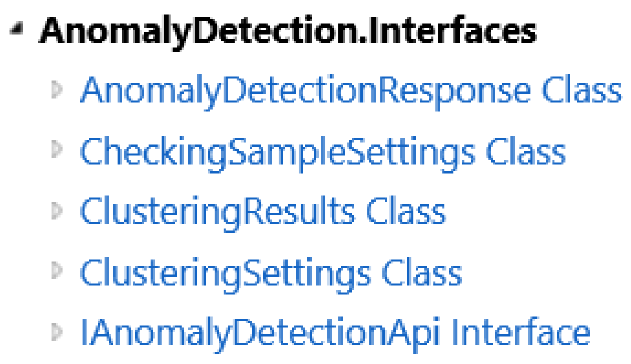
III. Previous Work

From previous semesters, the K-mean Algorithm was implemented. The same version of K-means is currently used.

IV. Work Done

The objectives were fulfilled by creating 2 namespaces (one for the interface), 1 interface and 5 classes.

- Namespaces:
    - AnomalyDetection.Interfaces: this namespace contains the interface (IAnomalyDetectionApi), AnomalyDetectionResponse class, CheckingSampleSettings class, ClusteringResults class, ClusteringSettings class as shown:


◢ AnomalyDetection.Interfaces

- AnomalyDetectionResponse Class
- CheckingSampleSettings Class
- ClusteringResults Class
- ClusteringSettings Class
- IAnomalyDetectionApi Interface

  - AnomalyDetectionApi: this namespace contains the AnomalyDetectionAPI class as shown:

◢ AnomalyDetectionApi

- AnomalyDetectionAPI Class

- Interface: IAnomalyDetectionApi is the interface for AnomalyDetectionAPI and was created so that the other group who are working on the Web application for this project won't be affected by the progress of the API meaning even if the API fails they can progress with their application.
- Classes:
  - AnomalyDetectionResponse: is a class that is used to determine whether the function succeeded or encountered an error. Whenever a function encounters an error, the error will be shown for the users. Also used in case of handled errors.
  - CheckingSampleSettings: is a class that contains the desired settings by the user for checking to which cluster a sample belongs
  - ClusteringResults: is a class that contains the results per cluster of a clustering instance with additional statistics.
  - ClusteringSettings: is a class that contains the desired settings by the user for clustering.
  - AnomalyDetectionAPI: is the main class of this projects. It inherits from IAnomalyDetectionApi. This class is a class containing basic

information about a clustering instance and it contains the major Methods to be called from the web application.

- The major Methods to be called by the web application:
    - ImportNewDataForClustering is a function that start a new clustering instance or add to an existing one. It saves the results automatically. The following function takes the user setting for clustering in a ClusteringSettings object and it returns an AnomalyDetectionResponse Object and not the results. Also this function will automatically call the ClusteringResults Method to create all the needed statistics of the clusters.
    - CheckSample is a function that detects to which cluster the given sample belongs to. This Method takes a CheckingSampleSettings Object as input and returns an AnomalyDetectionResponse Object which, in case no errors were encountered, will state to which cluster the sample belongs to if any
    - GetResults is a function that returns the results of an existing clustering instance. After the clustering process is done the user should call this function to receive the results.
    - GetPreviousSamples is a function that loads samples from a previous clustering instance. This function is used by the user to extract previous samples used in a clustering instance in case those samples were lost or for other purposes.
    - RecommendedNumberOfClusters is a function that returns a recommended number of clusters for the given samples. The recommended number of clusters is calculated using two approaches (both approaches can be used at the same time):
        - Radial method in which the farthest sample of each cluster must be closer to the cluster's centroid than the nearest foreign sample of the other clusters
        - Standard Deviation method in which the standard deviation in each cluster must be less than the desired standard deviation.

V. Suggested Improvements

- Implement a second version of K-means Algorithm as another suggestion for the user to use. The difference between the two algorithms will be the centroid assignment in each iteration. The current implemented version of K-means calculates the mean of the cluster then assign the centroid to the closest sample to this calculated mean (centroid has to be a sample). The second possible implementation of this algorithm is to consider the centroid as the mean (centroid is not necessarily a sample)

- Another improvement to the algorithm will be starting the an initial guess for the centroids instead of randomly assigning the first k samples as the centroids of the clusters
- More testing for different scenarios
- Optimizing the code and make it more efficient

VI. Documentation

The detailed documentation is checked in with the project in the form of a help file (.chm) and in the form of a website (.html). Sandcastle was used to generate the documentation.

VII. Current Error Codes

- ***success: 0-99***

- ***user input errors: 100-199***

  - 100 "RawData is null"
  - 101 "At least one input is null"
  - 102 "RawData is empty"
  - 103 "Means is empty"
  - 104 "Maximum number of clusters must be at least 2"
  - 105 "Unacceptable number of clusters. Clusters more than samples"
  - 106 "Unacceptable number of clusters. Must be at least 2"
  - 107 "Unacceptable number of attributes. Must be at least 1"
  - 108 "Unacceptable number of maximum iterations"
  - 109 "Unacceptable savepath"
  - 110 "Unacceptable tolerance value"
  - 111 "Data sample and number of attributes are inconsistent. First encountered inconsistency in data sample: " + Sample_Index + "."
  - 112 "Mismatch between old and new cluster numbers"
  - 113 "Mismatch between old and new number of atributes"
  - 114 "Mismatch in number of attributes"

- 115 "Inputs have different dimensions"
- 116 "Path provided : " + Path + " has no root"
- 117 "Path provided : " + Path + " contains invalid chars. First invalid char encountered is: " + Invalid_Char + "."
- 118 "Path provided : " + Path + " has no project name specified."
- 119 "Path provided : " + Path + " has a project name containing invalid chars. First invalid char encountered is: " + Invalid_Char + "."
- 120 "Path provided : " + Path + " has wrong extension."
- 121 "Requested load path does not exist"
- 122 "Method must be either 0,1 or 2"
- 123 "Parameter StdDev is needed"

- ***file related errors: 200-299***
  - 200 "File not found"
  - 201 "File already exists"
  - 202 "File cannot be loaded"
  - 203 "File content is corrupted"
  - 204 "Unauthorized access to file. File is readonly."
  - 205 "Unauthorized access to file. File is a system file."
  - 206 "Can't deserialize file"
- ***calculation errors: 300-399***
  - 300 "Division by zero"
- ***unhandled errors: 400***
  - 400 "Function Function_Name: Unhnadled exception: " + Exception