Project Report: Course CSE-572

Name : Tamasree Sinha Email: tsinha15@asu.edu

***Abstract*—This document contains reports of three projects those were part of CSE-572 Course : 1)Extracting Time series Properties of Glucose Levels in Artificial Pancreas Project, 2) Machine Model Training Project, 3)Cluster Validation Project**

Keywords—CGM, PID, groupby, central tendency, pandas, NumPy, dataframes, dropna, interpolation, linear, polynomial, training dataset, test dataset, random forest classifier, classification, KNN, matrix, pickle, imputer, FFT, KMeans, DBScan, SSE, Entropy, Purity, Clustering, Cluster validation, recall, F1 Score

Project I: Extracting Time series Properties of Glucose Levels in Artificial Pancreas Project

I. Introduction

In this project, I calculated some metrics based on data collected from glucose monitor. The Medtronic 670G Artificial Pancreas system includes a continuous glucose monitor (CGM) and the Guardian Sensor, which measures blood glucose level every 5 minutes and requires replacement after 7 days. The Guardian Link Transmitter sends the CGM data to the MiniMed 670G insulin pump, which uses SmartGuard Technology to modulate insulin delivery. This technology uses a Proportional-Integral-Derivative (PID) controller for precise insulin delivery, or "micro boluses." Users manually enter carbohydrate intake into the Bolus Wizard for meal-related insulin, and the system provides correction boluses if glucose levels exceed a threshold. Insulin delivery can be suspended if glucose levels are low, predicted to drop, or manually by the user.

II. Explanation of solution

Two sets of datasets were provided for this task. One was from Continuous Glucose Sensor and another was from the Insulin pump. Metrics were calculated in both modes of the insulin pump, i.e., auto and manual modes. At first, from the insulin dataset, the timestamp had been collected when the pump turned into auto mode from manual mode. Based on that timestamp, corresponding data from the CGM dataset (data of glucose monitor) had been separated for auto and manual mode. Following metrics had been calculated for three different time frames of day, i.e., daytime (6 am to midnight), overnight (midnight to 6 am) and whole day (12 am to 12 am).

1)Percentage time in hyperglycemia(CGM >180 mg/dl)
2)Percentage time in hyperglycemia critical(CGM > 250mg/dl)
3)Percentage time in range (CGM >= 70 mg/dl and CGM <= 180 mg/dl)
4)Percentage time in range secondary(CGM >=70 mg/dl and CGM <= 150 mg/dl)
5)Percentage time in hypoglycemia level 1(CGM <70 mg/dl)
6)Percentage time in hypoglycemia level 2(CGM <54 mg/dl)

It was assumed that, total number of CGM readings available per day is 288 as collected at every 5 minutes interval. Based on this value all the percentage metrics were calculated.

III. Description of result

All the metrics were calculated as percentage count of total CGM reading count of a whole day (assumed 288) for both auto and manual mode and 18 metrics were calculated for each mode. The metrics were stored in a csv file named Result.csv.

IV. Self Reflection

In this project, I used python as programming language. And libraries used were: pandas, NumPy. I used pandas library to read csv files and make dataframes from those datasets. Python's datetime module was used to calculate timestamp and time intervals i.e., daytime, overnight, whole day. For handling missing data, pandas dropna () method was used on sensor glucose data to drop all missing value rows. Then, I defined three functions to measure the percentage of time from CGM dataframe. First one was for measuring percentage count over a given threshold, second one was to measure percentage count below a given threshold and third one was to measure percentage count within a given range of CGM sensor value. Pandas groupby () method was used to take count of CGM sensor data per day and find the percentage based on 288 data per day and finally take average of that count to find final metrics.

Different methods from Pandas and NumPy libraries were used in this project to perform different operations on data. I learned some methods from datetime module and changing formats of date and time in different required formats for calculation based on timestamp. I also came to learn how to filter pandas dataframe based on different conditions, grouping dataset based on a column value and applying measures of central tendency on grouped dataframe.

I gained knowledge about different strategies of handling missing value in a dataset. This is a very important step in calculating metrics from a dataset and performing other analysis on a dataset.

As a first step of handling missing value, it is sometimes better to drop values where there are missing values of more than a threshold count. One other common strategy to handle missing value is to replace them with some average value (though not used in this project) or by linear or polynomial interpolation method.

Project II: Machine Model Training Project

I. Introduction

In this project, I trained a machine model to assess whether a person had eaten a meal or not eaten a meal based on a training dataset and validated the model on a separate test dataset. This was a classification task and I used random forest classifier to classify meal and no meal data.

II. Explanation of solution

In this project also, two sets of datasets were provided to perform the task. One was from Continuous Glucose Sensor and another was from the Insulin pump. From the insulin dataset the timestamps were collected at which meals were taken. It was assumed that there should be a minimum 2 hrs. gap between two meals. If any meal was taken within 2 hrs. from a meal time, then the previous one was discarded and the latest one was considered. From that corresponding glucose time, a 2 hrs. stretch was considered which was the postprandial period. In addition, a 30 minutes stretch was taken before the start of the meal. This full stretch of glucose sensor readings for these 2 hrs. and 30 minutes (for a meal) was considered as a meal data stretch.

Python pandas library was used to make a dataframe for all meal data stretches. As CGM readings were collected for each 5 minutes, for 2hr and 30 minutes stretch there were 30 readings. So, it was a dataframe with 30 columns.

Next step was to calculate No Meal Data. For this step, from Insulin dataset, I looked for the post-absorptive period i.e., all data after the mealtime+2hrs were no meal data and 2 hrs. stretches were collected for each no meal data. If there was any meal taken in that post-absorptive period then that data was discarded. A separate dataframe were made for no meal data. As CGM readings were collected for each 5 minutes, for 2hr stretch there were 24 readings. So, it was a dataframe with 24 columns.

Some more no meal data were also collected from overnight time period where no meal was taken.

For handling missing values, I followed different strategies in two cases. For meal data, I used KNN (K-nearest neighbor) method to fill missing values with n neighbor count as 5. For no meal data, I used linear interpolation to fill missing values.

As a next step, some features were extracted from meal and no meal data to apply classification method. I collected following features from these two datasets:

- Difference between the Max and Min CGM readings of a meal and no meal data stretch
- Time difference between Max and Min CGM reading times
- Fast Fourier Transform was applied on meal and no meal stretch data and 2nd and 3rd peak magnitudes and frequencies were taken as features
- First order and second order derivatives were calculated of all meal and no meal stretches and averages were taken as features.

With all these features, a separate feature matrix was made. Then the dataset was split into train and validation sets. I used Standard Scaler from Python's Scikitlearn Library to standardize the dataset. Standardization of a dataset is a common requirement for many machine learning estimators. They sometimes behave badly if the individual features do not more or less look like standard normally distributed data (e.g., Gaussian with 0 mean and unit variance).

Then I used random forest classifier to train a model and adjusted some parameters like random_state, n_estimators, max_depth, min_sample_split, criterion to get optimal model performance. The model was saved using pickle method for later use on test dataset.

III. Description of result

The saved scaler and model were used on a separate test dataset to classify meal and no meal data stretch. Output was a dataframe with a single column with two values 0 and 1 as labels for meal and no meal. The dataframe then was stored in a separate csv file named Results.csv. Some accuracy metrics were used to measure the performance of the model on the test dataset such as recall and F1 score.

IV. Self Reflection

In this project I used some Python libraries i.e., Pandas, NumPy, Scikitlearn. I also learned some missing value handling techniques.

KNN Imputer retains most of the data compared to other techniques such as removing rows or columns with missing values [5]. It replaces missing values with imputed values and ensures that entire dataset is used for analysis and modelling. It imputes missing values based on n nearest neighbors and also maintain underlying relationships in the data. It also cares about feature similarities between the data points to estimate the missing values, making it more relevant to the context. KNN Imputer is a non-parametric method, that means it does not make assumptions about the data's distribution. It can handle various types of missing values and this method is suitable for both numerical and categorical data.

Another very useful method of handling missing values is interpolation [4]. Interpolation can be linear or polynomial. Interpolation is useful when there is an order or sequence and the missing values of the sequence need to be estimated. When there is no sequence or order, interpolation should not be used.

In this project, I used FFT (Fast Fourier Transform) on the meal and no meal stretch data as a step in feature extraction [3]. FFT converts a signal or sequence into individual spectral components and thereby provides frequency information about the sequence. In FFT, a signal is sampled over a period of time and divided into its frequency components. These components are single sinusoidal oscillations at distinct frequencies each with their own amplitude and phase.

I used random forest classifier for classification task [2]. It is a supervised machine learning algorithm used for different classification and regression tasks. It works well on complex and high dimensional datasets as well. It creates a set of decision trees from a randomly selected subset from training dataset. It collects votes from different decision trees to decide the final prediction. This classification algorithm uses bagging (Bootstrap Aggregating) to create these diverse subsets. Feature selection is inherently embedded in the construction of individual decision trees and the aggregation process.

Following are some advantages and disadvantages of this classification technique [1]:

<u>Advantages</u>:

- This classification algorithm takes into account the variations associated with individual trees and results in predictions that are more accurate by averaging or voting. As it uses ensemble approach instead of a single decision tree, its accuracy is higher.

- This classification algorithm is robust to noise and outliers in datasets. It can handle missing data as well.
- This classification approach is non-parametric in nature i.e., it does not assume anything about the distribution of dataset or the correlation between the target variable characteristics. For this reason, it works well on various types of datasets.
- This algorithm calculates a feature's importance by taking into account impurity reduction of all the trees in the forest. Features are considered more significant when they regularly result in a larger reduction of impurities.

Disadvantages:

- As this algorithm uses ensemble method and a large number of trees are considered for classification, this method is computationally expensive and memory usage is also high when the sizes of trees are very big and deep rooted.
- For large datasets the prediction time is higher as there are lots of big trees to be considered for decision making.
- For big datasets there is lack of interpretability due to big sizes of trees and this model is considered as black box model.

For measuring model's performance on test dataset following metrics were considered:

Recall: Recall is the ratio of True Positive records and summation of True Positive and False Negative records classified by a model. It computes the fraction of positive examples correctly predicted by the classifier. Higher recall implies very few positive examples are misclassified as the negative class.

F1 Score: F1 score is the harmonic mean of two other accuracy metrics i.e., precision and recall. Harmonic mean tends to favor an attribute that has lower value. For a good model, it is desired that both precision and recall should be higher. So F1 score is a very effective metric to calculate a model's performance and higher value of F1 score is desired for a good model.

Project III: Cluster Validation Project

I. INTRODUCTION

For this project, I performed clustering on a given dataset using Python as programming language. Using the provided training dataset, I performed cluster validation as well to determine the amount of carbohydrates in each meal.

II. EXPLANATION OF SOLUTION

In this project also, two sets of datasets were provided. One was for Insulin Pump and another was for continuous glucose sensor. The first step was similar to the second project (Machine model training project). In this project also, I made a feature matrix from meal and no meal stretch data from Glucose sensor readings.

As next step, for extracting ground truth (for cluster validation), from Insulin dataset, the column for BWZ Carb Input(grams) was considered to get all meal intake data and getting min and max value of meal intake. Then the meal amount was discretized in bins of size 20. In total, there were n= (max-min)/20 bins. Then two metrices (feature matrix and bin matrix) were concatenated.

For the next step, I applied two clustering algorithms to cluster the feature dataset (dropped the ground truth while clustering) i.e., KMeans and DBScan. For KMeans the cluster number was selected 6 as there were 6 bins. For DBScan algorithm, I first applied StandardScaler from Scikitlearn Library to scale the dataset and then applied DBScan algorithm. By adjusting the eps (min threshold distance) and min_sample (min sample within the threshold) values, I got 2 clusters at first. Then I took the bigger cluster and applied Bisecting KMeans to get 5 more clusters from that bigger cluster. In each step, the cluster with highest SSE (Sum squared Error) was chosen to split further.

As a next step in this task, labels from both the algorithms were added to the feature matrix at first. Then two new matrices were made from the main matrix to find the count of each label in each bin (meal amount bins). These were both 6x6 matrices as there were 6 bins and 6 classes from both the algorithms.

For the next step, to validate the clusters those two 6x6 matrices were used. As metrics of cluster validation, SSE, Entropy and purity were calculated from these matrices. Below are the formulas for calculating these metrices:

$$c_i = 1/m_i \sum_{x \in C_i} x$$

$$SSE = \sum_{i=1}^{n} \sum_{x \in C_i} dist\,(c_i, x)^2$$

Where c_i is the centroid of each cluster and n is the number of all clusters and x is the datapoints in a cluster.

$$Entropy(t) = -\sum_{j} p\,(j|t) \log p(j|t)$$

p (j|t) is the relative frequency of class j. After calculating entropy of each cluster, weighted average was calculated to find the final value of entropy.

$$Purity = 1/N \sum_{i=1}^{k} \max_j |c_i \cap t_j|$$

where N= number of data points, k= number of clusters, c_i is one of the clusters in K clusters, t_j is one of the ground truth classes. After calculating purity of each cluster, weighted average was calculated to find final value of purity.

III. DESCRIPTION OF RESULT

Three metrices (SSE, Entropy and Purity) were calculated to validate clusters obtained by using two algorithms i.e., KMeans and DBScan. Six metrices from two algorithms were stored in a csv file named Result.csv. Algorithm that resulted in lower entropy and higher purity, performed better clustering on this dataset.

IV. Self Reflection

In this project, I used python libraries Pandas, NumPy, Scikitlearn for clustering and cluster validation. I came to learn about different clustering algorithms.

KMeans Clustering

KMeans clustering is a centroid based partitional clustering algorithm. This is a very simple algorithm though has some advantages and disadvantages. This clustering algorithm shows different behavior based on initial points selection, so it is a very important step. At first, all the datapoints are assigned to one of the K clusters depending on their distance from the initial K centroids. After that centroids are adjusted iteratively and data points are reallocated unless and until centroids are not moving or moving below a threshold value. KMeans algorithm generally converges after some iteration. As distance metric, I used Euclidian distance.

If the number of clusters (K) is higher, then the chance of selecting one centroid from each cluster is small. There are some methods to over come this initial K value selection problem and one very common useful method is to use Bi-secting KMeans which is another variant of KMeans algorithm.

In Bisecting KMeans all the datapoints are divided in two clusters at each step and cluster with higher size and SSE is selected for further split at each step. In this way the process is continued until desired number of clusters are obtained.

Following are some advantages and disadvantages of KMeans algorithm.

Advantages:

- It is simple and can be used for a wide variety of data types.
- Quite efficient, even though multiple runs are often performed.
- Some variations like Bisecting KMeans are less susceptible to initialization problems

Disadvantages

- This algorithm does not perform well for non-globular clusters or clusters of different sizes and densities.
- It is sensitive to noise and outliers of dataset
- It is sensitive to initial choice of K centroids.

DBScan Clustering

DBScan algorithm is a centroid based partitional clustering algorithm. It locates regions of high density that are separated from one another by regions of low density. Here density is defined using a center-based approach where density of a particular point is estimated by counting the number of points within a specified radius (Eps) of that point including the point itself.

In this center-based approach, all data points at first are classified as being one of the following three categories.

- Core Points: A point is a core point if the number of points within a given neighborhood (as determined by the distance function and a user specified parameter (Eps)) around the point exceeds a certain threshold Min-points which is also a user specified parameter. These points lie in the interior of a cluster.
- Border Points: A border point is not a core point, but falls in the neighborhood of a core point and a border point can fall in the neighborhood of several core points.
- Noise Points: A noise point is any point that is neither a border point, nor a core point.

Any two core points that are close enough (within a distance Eps of one another) are put in the same cluster. Any border point that is close enough to a core point is put in the same cluster as the core point. Noise points are discarded.

Following are some advantages and disadvantages of this algorithm.

Advantages:

- This algorithm is resistant to the noise of dataset.
- It can handle cluster of different shape and sizes

Disadvantages:

- It has trouble when the clusters have widely varying densities
- Has trouble with high dimensional data, as density(distance) is more difficult to define for each data
- Can be expensive when the computation of nearest neighbors requires computing all pairwise proximities, as is usually the case for high dimensional data.

I also learned different measures of cluster validation. For this project I used SSE, Entropy and Purity for cluster validation. SSE measures intra-cluster distance, so lower value of SSE is better for having good clusters. But SSE does not really capture inter cluster distance. Entropy and Purity are other measures for supervised cluster validation approach where ground truth is known. Lower value of Entropy and Higher value of Purity is desired for good clusters. These metrices measure the degree to which each cluster consists of objects of a single class. For each cluster, the class distribution is calculated at first.

References

[1] GeeksforGeeks. (n.d.). What are the Advantages and Disadvantages of Random Forest, GeeksforGeeks, Retrieved February 15th,2024, from https://www.geeksforgeeks.org/what-are-the-advantages-and-disadvantages-of-random-forest/

[2] GeeksforGeeks. (n.d.). Random Forest Classifier using Scikit-learn, GeeksforGeeks, Retrieved January 31st ,2024, from https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/

[3] nti-audio.com: Fast Fourier Transformation FFT-Basics , https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft

[4] machinelearningplus.com: Interpolation in Python-How to interpolate missing data, formula and approaches , https://www.machinelearningplus.com/machine-learning/interpolation-in-python-how-to-interpolate-missing-data-formula-approaches/

[5] Bhanupsingh. Medium: Handling Missing Data with KNN Imputer, Retrieved August 1st ,2023, from , https://medium.com/@bhanupsingh484/handling-missing-data-with-knn-imputer-927d49b09015