# CSE 572-Data Mining Portfolio

SAILENDRI GUNNIA RAVIKUMAR
ID:1229372289
*sgunniar@asu.edu*

***Abstract*—This portfolio includes three projects, extracting implicit knowledge and potentially useful information as features from the sensor data of "Artificial Pancreas Medical Control System" dataset as provided. The extracted features are then used for training models and for applying clustering techniques.**

***Keywords—Metrics, Features, Timestamp, Meal data, No Meal Data, Classification, Clustering.***

## I. INTRODUCTION

The first project is about extracting metrics from the Continuous Glucose Monitoring(CGM) sensor and Insulin pump data. For both Auto and Manual mode, we need to find the metrics as mentioned in Table 1. The six metrics are calculated for three time intervals namely day-time(6:00AM-11:59PM),overnight(12:00PM-5:59AM) and whole-day(12:00AM-12:PM).

The second project deals with the extraction of Meal and No Meal features from two pairs of CGM and Insulin datasets and passing them as input to Decision Tree Classifier Model for the purpose of training. After training, model is stored in pickle file and we use pickle file for classification of test data. and obtain values/output as either '0' (No Meal Data) or '1' (Meal Data).

The third project deals with clustering of the CGM data and validation of those resulting clusters with metrics like SSE, Purity and Entropy. Both K-Means and DB-SCAN Clustering are implemented and results are compared. Only Meal Data Features are extracted. Clustering is done based on the carbohydrates amount in the meal.

## II. EXPLANATION OF SOLUTION

### A. *Project 1*

In this project, we are provided with two csv files. One file has CGM Data and the other file has Insulin Data. These data are taken from the sensors and pump of the '*Artificial Pancreas Medical Control System*'. We need to find the six metrics mentioned in Table 1 like Percentage time in hyperglycemia, percentage time in range, etc., for Manual and Auto modes and also for three time intervals as categorized above (daytime, overnight, whole-day). For achieving this, I started with importing necessary packages and the data is ordered in the increasing order of time [4]. To find the start of the auto mode, we search for the first "AUTO MODE ACTIVE PLGM OFF" tag in 'Q' column of Insulin data. Same Timestamp is used for segregating the manual and auto mode of the CGM dataset. But when we compare both timestamps, it is asynchronous. So, we are finding the equal or the next slightly higher timestamp. Then we are separating manual and auto dataframes into overnight, day-time, whole-day. Once this is done, we get six dataframes.

From all the six dataframes, we take the required metrics by using the value of 'Sensor Glucose (mg/dL)' column and compare it with fixed values like the Table 1 given below.

| No | METRICS | CONDITION |
|---|---|---|
| 1 | Hyperglycemia | CGM > 180mg/dL |
| 2 | Hyperglycemia Critical | CGM > 250mg/dL |
| 3 | Range | 180mg/dL>=CGM>=70mg/dL |
| 4 | Range Secondary | 150mg/dL>=CGM>=70mg/dL |
| 5 | Hypoglycemia level 1 | CGM < 70mg/dL |
| 6 | Hypoglycemia level 2 | CGM < 54mg/dL |

Table 1

We then take the count of rows satisfying the conditions and find percentage for unique dates and for all six metrics. Finally, we find the average and so the result dataframe has 2 rows and 18 columns. Finally, it is written on a csv file.

### B. *Project 2*

In this project, we extract Meal and No Meal data features from training dataset, pass those features to train Decision Tree Classifier[5] and use the model to classify the test dataset as Meal and No Meal data. The first step to achieve this is to train the model. I imported all the necessary packages and the next step is to read all the four given csv files and the records are sorted in ascending order of time. Next, to extract Meal data, we are filtering the rows in insulin data where carbohydrate input is greater than zero. If the difference between two consecutive timestamps is greater than or equal to 2 hours, we consider it as a valid meal timestamp. Now we have a set of meal timestamps. For each valid meal timestamp in the set, start of meal timestamp would be a valid timestamp – 30 mins and end would be a valid timestamp + 120 mins. We need to check for these intervals of timestamps in CGM Data and take the appropriate sensor glucose values. These values form a row in Meal dataframe. After fetching glucose values for all meal timestamps, we clean the data by removing rows with more Nan values. This will be our final Meal dataframe. Similarly, we are extracting No meal data, by filtering rows of Insulin data with carbohydrate values greater than zero. We take valid no meal timestamps, if difference between two consequent timestamps is greater than or equal to 2 hours. After getting a set of valid No meal timestamps, for each value, we take start of No Meal timestamp as value (a valid no meal timestamp) +120 mins and end as value + 240 mins. We take sensor glucose values from the start to end timestamps from CGM data. We clean the data and keep it as final No Meal Dataframe. The next step would be creating feature matrices for both Meal and No Meal final data. For that, we interpolate the data with linear method. Then for each row of values, we compute 1-D n-point Discrete Fourier Transformation and take features like power at second max frequency and its index, power at third max frequency and its index. As fifth feature, we take 'Tau time' which is time

difference between start of meal timestamp and timestamp where CGM reaches its maximum for meal data and difference between start of no meal timestamp and 24 for No Meal data. Then sixth feature would be normalized difference between start of meal/no meal CGM from max CGM for meal/no meal data respectively. So, in the case of normalized difference, after subtraction, we divide the value by meal/no meal CGM depending on the type of data (meal/no meal). There is an increase in slope after meal/no meal time. We take double differential of that and have it as our seventh feature. The eight feature would be standard deviation of data along the rows. Then, we add label as 1 for Meal data and label as 0 for No Meal data. Finally, we get our Meal and No meal feature matrices. Next, we concatenate meal and no meal feature matrices data, shuffle them. We do k-fold validation (10 folds) and split into train and validation sets. Then we fit the model [5], train it and dump it as pickle file. For testing the model, we read the test data, extract the above-mentioned features to form feature matrix, but we do not include labels here as we use the model, to find it out. After creating feature matrix for test data, we load the pickle file and predict values for test set, store in a single column as a dataframe and then convert to excel. As a result, we give 0 (No Meal) or 1 (Meal) for given test data.

### C. *Project 3*

In this project, we perform K-Means and DBSCAN clustering [3] on the meal data and validate the results based on the metrics which are SSE, Entropy, Purity. In order to achieve that, we start extracting Meal data and its features as mentioned in the Project 2. For binning the carbohydrate values, we take the meal data and find minimum and maximum of carbohydrate values and we figure out the number of bins by dividing the difference between max and min carbohydrate value by 20. Each bin has different range of carbohydrate values starting from 'min_value' to min_value+20 and so on till max value, having interval difference of 20. Following that, we find bin values for all the carbohydrate inputs of the meal data and store in a separate dataframe. As a next step we implement K-Means Algorithm [3] by scaling the feature matrix, fitting the scaled features. We get K-Means labels for all the Meal data. Using '.inertia_' we calculate SSE. For purity and Entropy, we create Cluster-Bin Matrix. For each meal data, we refer bin number from bin dataframe, for columns and K-Means label for rows, of Cluster-Bin matrix and increment the value at that position by 1. Initially we create a matrix filled with zeros of size, length of K-Means label x Number of rows in bin dataframe. We find purity values by finding max value along the rows of Cluster-Bin matrix, and add them and divide by no of meal data labels. For total entropy, we find entropy of each cluster, multiply it with the number of points in the respective cluster, add them up and divide by count of meal data labels. Next part of the project focusses on DBSCAN Clustering. The scaled features are fit to DBSCAN with eps of value 2.5 and min_samples of value 20. With '.labels_', we will get the DBSCAN labels and then we apply Bisecting K-Means algorithm. With the finally obtained labels, we calculate SSE, purity and entropy similar to K-Means. We store the obtained metric values to the excel file.

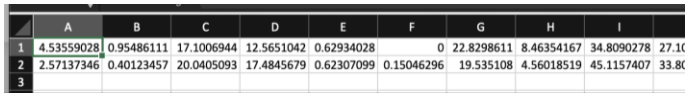
## III. RESULTS

### A. *Project 1*

We add the extracted metrics for manual (0) and auto (1) in first and second rows respectively. The columns are arranged, in the below order for Overnight followed by Day-time and Whole-day respectively.

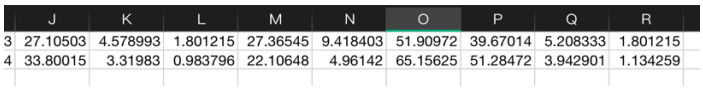
1. Hyperglycemia
2. Hyperglycemia Critical
3. In Range
4. In Range Secondary
5. Hypoglycemia level 1
6. Hypoglycemia level 2

The below figures 1.1 and 1.2 shows the stored results



| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4.53559028 | 0.95486111 | 17.1006944 | 12.5651042 | 0.62934028 | 0 | 22.8298611 | 8.46354167 | 34.8090278 |
| 2 | 2.57137346 | 0.40123457 | 20.0405093 | 17.4845679 | 0.62307099 | 0.15046296 | 19.535108 | 4.56018519 | 45.1157407 |
| 3 | | | | | | | | | |

Fig 1.1



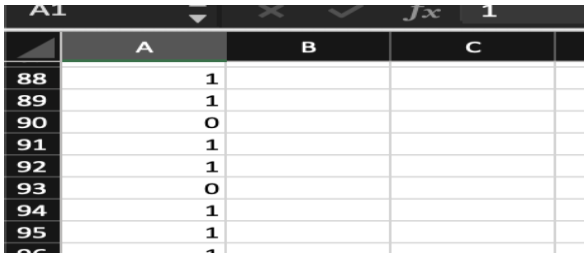
| | J | K | L | M | N | O | P | Q | R |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 27.10503 | 4.578993 | 1.801215 | 27.36545 | 9.418403 | 51.90972 | 39.67014 | 5.208333 | 1.801215 |
| 4 | 33.80015 | 3.31983 | 0.983796 | 22.10648 | 4.96142 | 65.15625 | 51.28472 | 3.942901 | 1.134259 |

Fig 1.2

From Fig 1.2, we find that Auto mode Whole-day In Range metric(2nd row, O column) has the highest average value among all metrics. As the auto grader does not accept headers, I have not included them.

### B. *Project 2*

The Excel stores directly the results of the classifier, 0 for No Meal and 1 for Meal.



| | A | B | C |
|---|---|---|---|
| 88 | 1 | | |
| 89 | 1 | | |
| 90 | 0 | | |
| 91 | 1 | | |
| 92 | 1 | | |
| 93 | 0 | | |
| 94 | 1 | | |
| 95 | 1 | | |

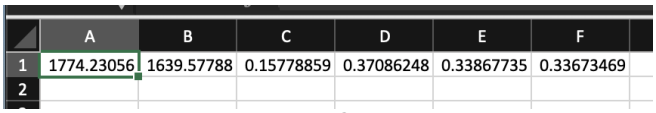
Fig 2

From Fig 2, we can observe that we are getting more data points for meal data (160) and number of No Meal data is less (67). These results completely depends on the given test data.The F1 and accuracy score came out higher than the values mentioned in the overview document[1]. This shows the trained model has less generalisation error and classifies results more accurately.

### C. *Project 3*

This project result consists of 6 values (columns) in the following order respectively.

1. K-Means SSE
2. DBSCAN SSE
3. K-Means Entropy
4. DBSCAN Entropy
5. K-Means Purity
6. DBSCAN Purity



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | 1774.23056 | 1639.57788 | 0.15778859 | 0.37086248 | 0.33867735 | 0.33673469 |
| 2 | | | | | | |

Fig 3

Getting a lower entropy shows that clustering is done properly by grouping datapoints of higher similarity. By higher purity also, we get more similarity. But the obtained results shows lower entropy(C and D column) but purity(E and F column) should be enhanced further to attain its maximum value.

## IV. CONTRIBUTIONS

These are individual projects, and I started the project with the course videos [2] for the projects and the documentation [1] helped me to pursue the projects in right manner. I have set the technology requirements to the versions as mentioned. For example, Python-3.10.9, scikit-learn-1.1.1, pandas-1.4.3, numpy-1.23.1, scipy-1.8.1. For this, I created a new environment in Anaconda with all the above-mentioned packages and versions. Initially for the first project, I was taking time to understand the Artificial Pancreas Medical Control System. But the dataset holds all necessary information in headers. Cleaning the data always plays a vital role and it has consequences on results. So, for data preprocessing, I removed most of the rows with Nan data, which did not satisfy the threshold and for second and third projects, I also used linear interpolation method to handle the missing data. Specifically in the second project, in order to store model/machine in the pickle file, I used 'joblib' package. With that package, we are able to dump(store) and load the pickle files. The format for dates were quite different in the given two CGM files. So I compared dates from Insulin to CGM with their respective formats. With the change of threshold for Nan, I was able to get different results, but with the threshold of 2, I was able to get more accurate classification. For third project, exploring the eight features was slightly challenging when compared to the other projects. But the importance of each feature was mentioned in the videos [2] which helped me to complete the project successfully.

## V. LESSONS LEARNED

First, I learned the setup of new environment in Anaconda, as we usually work on the base or root environment. In this new environment, I had to upgrade and few packages, I degraded few packages.

As we have variety of datatypes in different columns in the dataset, I got a 'DtypeWarning' describing me to specify 'dtype' option on import or to set 'low_memory' parameter to false. I did the second option, to fix the warning. For creating Timestamp, instead of concat function, I tried using apply function [4], to join them with spaces in between date and time, along the rows of the dataframe. I used 'sort_values' function [4] to sort the timestamp in ascending order.

With the where function from the NumPy library, we can filter rows for a necessary condition. Other ways to filter rows, given a condition is to use '.loc' or use '.iloc' for filtering rows given indices[4]. To take the desired columns, we can just use double square brackets and write the required column name(s) inside. Axis parameter helps us to do the actions across rows and columns by specifying 1 and 0.

To separate the data as overnight, daytime and whole-day, I used 'between_time' function specifying the start and end time as parameters. But if we do that directly, we get a 'type error' indicating that, the index should be set as 'DateTime' value.

I used 'groupby' function to group rows according to a column [4]. We can even use it group rows with more than one columns. In that case, for all the possible combinations of values in the selected columns, the rows will be grouped. If we do not have data with particular combination, it is ignored.

To get more accurate results for mean in the first project, I used 'nunique' function to count the number of unique dates (used as denominator in finding average) and found the average for each metric, instead of usual mean function ().

In project 2, for shuffling the data I used 'shuffle' from 'sklearn.utils' package. I gave different set of rows from meal and no meal data it performs random shuffling. We can also use 'sample' function to shuffle the rows.

The 'scipy.fftpack' has different types of transformation functions, differentiation functions which are very much used for finding the values of the features. I have performed 'rfft-Discrete Fourier Transformation' for finding the power at maximum frequencies and its indices. Also, to find the second differential values, I used 'diff' function twice.

For DBSCAN and K-Means clustering [3], objects from 'sklearn.cluster' package are used. In DBSCAN clustering in order to find 'Eps' distance and Minimum points, a graph is drawn with nearest neighbor's indices on X-label and distances on Y-label. At one point in the graph we find, a curve. That corresponding distance value is taken as Eps and it is optimal. Similarly for min points, we take the optimal value, as greater than or equal to number of features multiplied by two. For different combinations of eps and min distance, the datapoints are clustered differently. Even though DBSCAN object does not give desired number of clusters, we can use those labels for removing noise points and pass it to the Bisecting KMeans, specifying number of clusters and by fitting noise removed and scaled data.

We should build our models such that it does not overfit on the data causing high generalization error and also it should not underfit the data causing high training error.

I learnt switching between different environments of anaconda. The csv files used as the dataset, should be in the same folder as the 'ipynb' file. The pickle file and the excel file containing results are also stored on the same folder. I got a good hands-on training on the basic dataframe operations and to modify accordingly for extracting useful information.

By end of these projects, I learnt to do data preprocessing, extracting the necessary features, classification and clustering of datasets, and validation of classification and clustering thoroughly.

## REFERENCES


[1] https://www.coursera.org/learn/cse572/supplement/YSm4h/course-projects-overview-downloadable-printable

[2] https://www.coursera.org/learn/cse572/home/week/6

[3] https://realpython.com/k-means-clustering-python/

[4] https://devdocs.io/pandas~0.25/.

[5] https://towardsdatascience.com/implementing-a-decision-tree-from-scratch-f5358ff9c4bb