Ira A. Fulton Schools of Engineering
Arizona State University

CSE 572: Data Mining

Machine Model Training Project

Purpose

In this project, you will use a training dataset to train and test a machine model. The purpose is to distinguish between meal and no meal time series data.

Objectives

Learners will be able to:

- Develop code to train a machine model.
- Assess the accuracy of a machine model.

Technology Requirements

- python 3.13.5
- scikit-learn 1.7.0
- pandas 2.2.3
- numpy 2.3.1
- scipy 1.16.0

Project Description

In this project, you will train a machine model to assess whether a person has eaten a meal or not eaten a meal. A training data set is provided.

Watch the **three (3) Machine Model Training Project introductory videos** before beginning. These are located in Ed Lessons before the project's code challenge.

Note: Project details in the Overview Document were updated since the recording of the videos, so some directions or items may not match. Follow the Overview Document directions to complete your project correctly.

Directions

Accessing Ed Lessons

You will complete and submit your work through Ed Lessons. Follow the directions to correctly access the provided workspace:

1. Go to the Canvas Assignment, "**Submission: Machine Model Training Project**"
2. Click the "**Load Submission…in new window**" button.
3. Once in Ed Lesson, select the assignment titled "**Submission: Machine Model Training Project**".
4. In the code challenge, first review the directions and resources provided in the description.
5. When ready, start working in the Python files titled "**train.py**" and "**test.py**"

Project Directions

Step 1: Extracting Meal and No Meal Data

Extraction: Meal data

1. From the InsulinData.csv, search column Y(**BWZ Carb Input(grams)**) for a non-NAN non-zero value. This time indicates the start of the meal consumption time **tm**.
2. From CGMData.csv, get the glucose time series data corresponding to the meal (tm).
3. Meal data comprises a 2hr 30min stretch of CGM Data. So once you get the corresponding glucose time look for a 2hr stretch (tm+2hrs) which is the postprandial period. In addition, take a 30 min stretch (tm-30min) before the start of the meal (tm).
4. To extract the meal data there can be three conditions:
    a. There is no meal from time tm to time tm+2hrs. Then use this stretch as meal data
    b. There is a meal at some time tp in between tp>tm and tp< tm+2hrs. Ignore the meal data at time tm and consider the meal at time tp instead. For example, suppose the start of the meal (tm) is at 9:15 and you find a meal at 10:00 (tp) where tp > tm and tp < tm+2hrs, consider the meal at tp.
    c. There is a meal at time tm+2hrs, then consider the stretch from tm+1hr 30min to tm+4hrs as meal data.

5. Each stretch will be one row and will have 30 columns i.e. 2hr 30 min stretch for every 5 minutes in CGMData.csv. After extracting the all meal data, you need to create a Meal Data Matrix (P x 30) where P is the total number of rows for the meal data time series.

Extraction: No Meal data

1. From InsulinData.csv, you need to look for the post-absorptive period i.e. all the data after the tm+2hrs will be no meal data but you need to only find 2hr stretches.
2. To extract no meal data there can be two conditions:
    a. There is no meal found in the post-absorptive period. If found, you can't consider this stretch as no meal data.
    b. If the meal value found in the post-absorptive period is 0, then you can ignore this and consider the stretch
3. From the start of no meal data, take corresponding data from CGMData.csv and take a 2hr stretch.
4. Each stretch will be one row and will have 24 columns i.e 2hr stretch for every 5 minutes in CGMData.csv. After extracting all the no meal data, you need to create a No Meal Data Matrix (Q x 24) where Q is the total number of no meal data time series.

Handling missing data:

You have to carefully handle missing data. This is an important data mining step that is required for many applications. Here there are several approaches:

1. Ignore the meal or no meal data stretch if the number of missing data points in that stretch is greater than a certain threshold.
2. Use linear interpolation (not a good idea for meal data but maybe for no meal data)
3. Use polynomial regression to fill up missing data (untested in this domain).
4. Also, consider KNN to handle missing data

Choose wisely.

Step 2: Feature Extraction and Selection:

1. From step 1, you will have obtained the Meal and No Meal Data Matrix this will be the raw data.

2. This raw data may contain a lot of noise, so you have to carefully select features from the meal time series that are discriminatory between meal and no meal classes.
3. You need to pass the meal and no meal data matrix to the feature extractor. This will convert the meal and no meal data into a Feature Matrix P x F_L and Q x F_L respectively, where F_L is the features length obtained from feature selection.

Step 3: Train a Machine:

1. You need to concatenate the meal feature matrix P x F_L with no meal feature matrix Q x F_L
2. We will be using supervised classification. So we need a label vector where 1 will be meal and 0 will be no meal.
3. You can train a machine either using SVM or a Decision Tree based on your choice.

Note: For meal data, you are asked to obtain a 2 hr 30 min time series data, while for no meal you are taking 2 hr. However, a machine will not take data with different lengths. Hence, in the feature extraction step, you have to ensure that features extracted from both meal and no meal data should be same.

Test Data:

The test data will be a matrix of size N×24, where N is the total number of tests and 24 is the size of the CGM time series. N will have some distribution of meal and no meal data.

Output format:

You have to output an N×1 vector of 1s and 0s, where if a row is determined to be meal data, then the corresponding entry will be 1, and if determined to be no meal, the corresponding entry will be 0.

- This vector should be saved in a "**Result.csv**" file.

Given:

- Meal Data and No Meal Data of subjects 1 and 2
- Ground truth labels of Meal and No Meal for subjects 1 and 2

Using Python, train a machine model to recognize whether a sample in the training data set represents a person who has eaten (Meal), or not eaten (No Meal). The training data set contains ground truth labels of Meal and No Meal for 5 subjects.

You will need to perform the following tasks:

1. Extract features from the Meal and No Meal training data set.
2. Make sure that the features are discriminatory.
3. Train a machine to recognize Meal or No Meal data.
4. Use k-fold cross-validation on the training data to evaluate your recognition system.
5. Write a function that takes a single test sample as input, and outputs 1 if it predicts the test sample as a meal or 0 if it predicts the test sample as No meal.

Note:

1. The train.py reads CGMData.csv, CGM_patient2.csv, InsulinData.csv, and Insulin_patient2.csv, extracts meal and no meal data, extracts features, trains your machine to recognize meal and no meal classes and stores the machine in a pickle file (Python API pickle)
2. The test.py reads test.csv which has the N x 24 matrix and outputs a Result.csv file which has N x 1 vector of 1s and 0s, where 1 denotes meal, and 0 denotes no meal.
3. Assume that CGMData.csv, CGM_patient2.csv, InsulinData.csv, Insulin_patient2.csv, and test.csv files are all in your compilation and execution folder.

Submission Directions for Project Deliverables

You are given a(n) unlimited number of attempts to submit your best work. The number of attempts is given to anticipate any submission errors you may have in regards to properly submitting your best work within the deadline (e.g., accidentally submitting the wrong paper). It is not meant for you to receive multiple rounds of feedback and then one (1) final submission. Only your most recent submission will be assessed.

You must submit your Machine Model Training Project deliverable through Ed Lessons. Carefully review submission directions outlined in this overview document in order to correctly earn credit for your work. Learners may not email or use other means to submit any assignment or project for review, including feedback, and grading.

The Machine Model Training Project includes two (2) deliverables:

- **train.py**: Use the file provided in your workspace which also includes all necessary datasets. Execute your code by running the “python3 train.py” command in the terminal to test your work. Submit your work when finished.
- **test.py**: Use the file provided in your workspace which also includes all necessary datasets. Execute your code by running the “python3 test.py” command in the terminal to test your work. Submit your work when finished.

Submitting to Ed Lessons

This project will be auto-graded. You must complete and submit your work through Ed Lesson’s code challenges to receive credit for the course:

1. To get started, use the '**train.py**' and '**test.py**' files provided in your workspace.
2. All necessary datasets are already loaded into the workspace.
3. Execute your code by running “**python3 train.py**” and “**python3 test.py**” commands in the terminal to test your work and catch any potential runtime errors.
4. When you are ready to submit your completed work, click on “**Test**” at the bottom right of the screen.
5. You will know you have completed the assignment when feedback appears for each test case with a score.
6. If needed: to resubmit the assignment in Ed Lesson
    a. Edit your work in the provided workspace
    b. Execute your code again by running the commands in the terminal
    c. Click “**Test**” at the bottom of the screen
7. Once you have finished working on the project, please submit it by clicking the ***"Submit"*** button at the top right corner of your submission space.

Your submission will be reviewed by the course team and then, after the due date has passed, your score will be populated from Ed Lesson into your Canvas grade.

Note:

1. Do not change the code file name; it must remain ‘**train.py**' and '**test.py**' for the auto-grader to recognize your submission.

2. When the auto-grader runs your Python file, it should generate a '**Result.csv**' file with the specified format. The **'Result.csv'** file should not include any headers and should only contain the metrics in a Nx1 matrix.
3. Avoid using absolute paths when accessing other files.
4. Before submitting to the grader, it is recommended to run your code file to catch any potential runtime errors.

Evaluation

The auto-grader will evaluate your code as well as the accuracy of your results based on a set of Meal and No Meal data that is not included in the training set.

- 100 points: For developing a code in Python that takes the given dataset, extracts Meal and No Meal data, and trains a machine model
- 40 points: For developing in Python that implements a function to take a test input and run the trained machine to provide the class label as output
- 60 points: Based on the accuracy, F1 score, and results obtained by your machine.