COMP3217 Security of Cyber-Physical Systems
21/22 Coursework 2

Detection of Manipulated Pricing in Smart Energy CPS Scheduling

Author: Ahmed Gorelski

Module Leader: Shiyan Hu

Student ID: 31161979

MAY 23, 2022

Contents

1. Introduction

The purposes of this report are the following:

- To explore the linear programming-based energy scheduling for smart home cyber-physical systems.
- To understand the interdependence between the pricing information and the energy load scheduling.
- To develop and discuss several detection techniques for pricing attacks using Python.
- Get familiar with some cyber-physical system security programming skills.

2. Problem

For this project we are considering a small community of 5 users, each of whom uses 10 smart home appliances. The details are given in an excel sheet, consisting of the following information:
Sheet1(**Appendix 1**):

- User & Task ID
- Ready Time
- Deadline
- Maximum scheduled energy per hour
- Energy Demand

Next, we have a set of 10 000 predictive guideline price curves, which is going to be our training data. They are divided into two types. Half of

them are labeled “Normal” and the other as “Abnormal”. Each predictive guideline price curve has 25 numbers, as the first 24 refer to the 24 hours of the day and the last one is either 0 or 1 label, representing a normal predictive price curve if it is 0 and abnormal predictive price curve if it is 1.

We are also given testing data, which consists of 100 predictive guideline price curves with no labels to indicate whether the curve is “Normal” or “Abnormal”, hence there are only 24 numbers in each predictive price curve.

The task is to create a technique to model the training data and compute labels for the 100 predictive guideline price curves from the testing data (either 0 or 1). The computed results have to be clearly specified and saved in a file named “TestingResults.txt” in the same format as the training data set.

After we have computed the labels for the testing data, for each predictive price curve labeled “Abnormal” (labelled with 1) we need to compute the linear programming-based energy scheduling solution according to the that abnormal predictive guideline price curve, and plot the scheduling results showing the hourly energy usage of this community.

Additionally, we need to compute our own linear programming-based scheduling algorithm and as mentioned earlier a technique to compute the labels for the testing data.

3. Proposed Solution

As this is pretty much a classifying data sets problem, the obvious approach would be to use Machine learning algorithms. A trained algorithm has a training set and, in our case, the data set with 10,000 labelled data entries. Then the algorithm is tested on an unlabeled data (testing data) to try and predict the correct label and, in our case, normal or abnormal.

For this project we will be using the machine learning Python library Scikit-learn that has several classification algorithms. A couple of classification models will be tested and the most suitable one will be used.

The original training data set will be divided into two subsets with ratio 7:3. These subsets will represent the new training and testing data respectively. The training subset will consist of 7 000 entries and the testing subset will have 3 000 entries of the predictive price curves. The classifier will be trained on the training data and then used to predict the correct labeling for the new testing data.

As stated in the task description we need to measure the accuracy of the training and the testing data (training and testing error) classification model. Accuracy is a metrics that is used to evaluate a classification model or in other words accuracy is the fraction of predictions our model got right. It looks something like this:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

[1]

4. Technique Used

For this section of the report, I will be using and comparing two different machine learning algorithms on my training and testing data to see which classification model would produce the best performance. The machine learning algorithms tested are Decision Trees and Support Vector Machines (SVM).

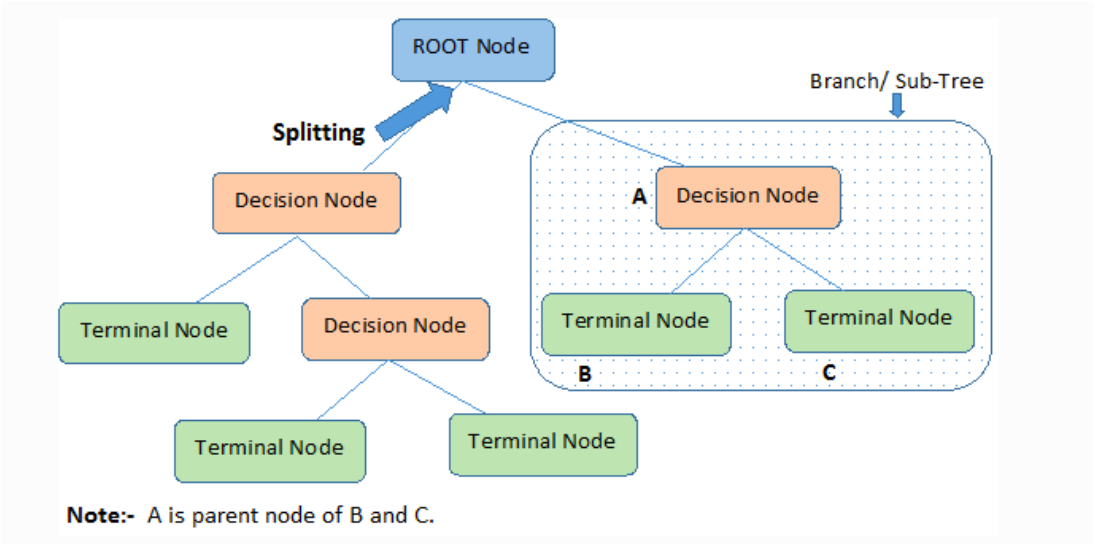
Decision Trees

Decision Tree is an easy and popular classification algorithms to understand and interpret. It is part of the supervised learning algorithms, but unlike some of them the decision tree algorithm can be used for solving regression and classification models. This is perfect for us, as we need a classification model.

The goal of using a Decision Tree is to create a training model that can be used to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).[2]

In Decision Trees, in order to predict a class label for one record we start from the root of the tree. Then the values of the root attribute are compared with the record's attribute. On the basis of comparison, we

follow the branch corresponding to that value and jump to the next node.[2]



A visualization of how Decision tree algorithm works.[2]

Although decision trees are simple to understand and interpret they have some disadvantages like:

- Decision-tree learners can overcomplicate data by overfitting.
- They're sensitive to small variations of data (high variance) and can cause instability.

After using this classification model, Decision Tree, on our data generated based on the 7:3 training and testing data ratio, the evaluated accuracy of both training and testing was:

Training prediction accuracy: 99.8% Test prediction accuracy: 76.8% Predicted number of abnormal price curves: 56

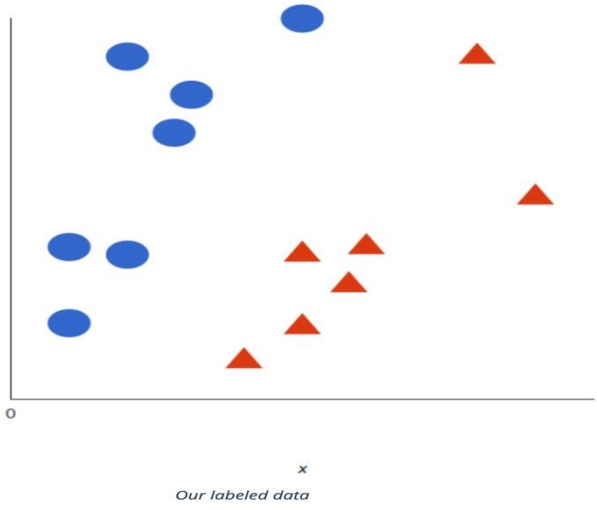
Classifier: Decision Tree

Training prediction accuracy showed amazing results with 99.8% accuracy, but for the testing data the accuracy was not as good with only 76.8%, which led us to use the following machine learning algorithm for our classification:

Support Vector Machines (SVM)

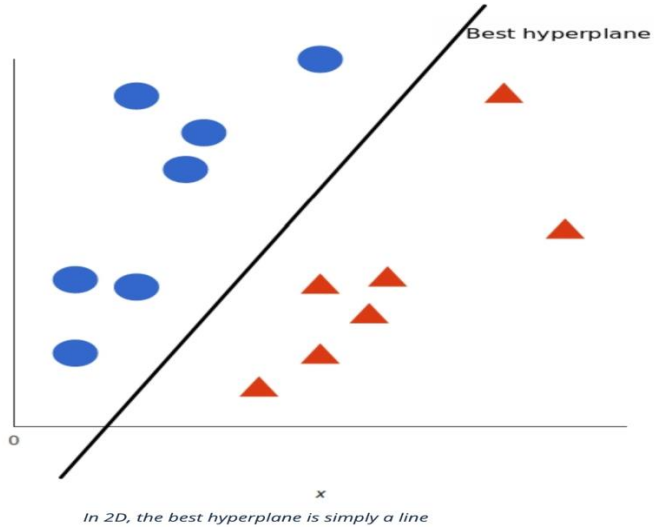
Support vector machine is another simple algorithm that is suitable in our case as it produces significant accuracy with less computation power. Support Vector Machine, abbreviated as SVM can be used for both regression and classification tasks. But, it is widely used in classification objectives.[3]
To understand how SVM works we are going to look at a simple example taken from MonkeyLearn Blog about Support Vector Machines.[4]

Let's imagine we have two tags: *red* and *blue*, and our data has two features: *x* and *y*. We want a classifier that, given a pair of *(x,y)* coordinates, outputs if it's either *red* or *blue*. We plot our already labeled training data on a plane:[4]



Our labeled data *[4]*

A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the tags. This line is the **decision boundary**: anything that falls to one side of it we will classify as *blue*, and anything that falls to the other as *red*.[4]



In 2D, the best hyperplane is simply a line

[4]

But, what exactly is *the best* hyperplane? For SVM, it's the one that maximizes the margins from both tags. In other words: the hyperplane (remember it's a line in this case) whose distance to the nearest element of each tag is the largest.[4]

The evaluation of accuracy for SVM classification model was:

Training prediction accuracy: 95.3% Test prediction accuracy: 94.8% Predicted number of abnormal price curves: 54

Classifier: Support Vector Machine

The results of the classification of labels to the 100 unlabeled predictive guideline price curves using SVM as taken from the produced TestingResults.txt were:

```
1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1,
0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0
```

TestingData produced labels.

5. Implementation

SVM Classification

As we can see from the accuracy evaluation in the previous section, the Support Vector Machine algorithm produced better results than the Decision Tree and so it was the one chosen to compute the missing labels from the testing data.

As mentioned above, in order to compute the labels for each predictive price curve for the testing data, we need to train the model on the training data so that the algorithm can learn the features of the already labeled predictive guideline price curves. For that reason, the training set was divided into 70% training data an 30% testing data.

Initially, the training and testing datasets are loaded into the function and each predictive price curve is extracted to a list. Then using the scikit_learn library the training data is easily divided into train and test subset:

```
# Splitting the training dataset into random 70% train and 30% test subsets
xTrain, xTest, yTrain, yTest = train_test_split(p_curves, labels, test_size=0.3, random_state=15)
```

Splitting the training dataset into train and test subsets.

The Support Vector Classifier is called to fit the training data points and then a prediction is generated to compute the labels for each predictive price curve in the testing dataset.

Once the model is trained and we get the accuracy score for the training and testing subsets, the predict() function is called on the 100 predictive guideline price curves(testing data) to compute the labels.
Finally, the computed labels for the 100 testing curves are put in a list and saved together with the pricing curves of the testing data into the TestingResults.txt file.

That is how the SVM classification model is used to compute the labels for the 100 predictive guideline price curves (testing data).

Scheduling Algorithm

As mentioned above in the **Problem** section the data we are provided with contains 5 users, each of whom has 10 tasks and they need to be scheduled according to the required time period, which is between the values of *Ready time* and *Deadline*. The *Maximum scheduled energy per hour* values represents how many units are allowed for each hour and in our case, it is always 0. The *Energy Demand* is the total number of units needed for the corresponding task to be completed.

The produced algorithm sorts each abnormal pricing curve by the least expensive. Since the maximum scheduled energy per hour is one unit, each user can demand several units, and so it will take the same amount of hours to complete the task. The algorithm compares the time period of each price per hour against the required time period by the user's task, if

the period is between the ready and deadline hours, then the task will be assigned to that time slot. Hourly unit prices are sorted from low to high. A loop is created that assigns the tasks based on the required time period This is repeated for all the tasks 5*10 for every abnormal price curve according to the classification algorithm from earlier.

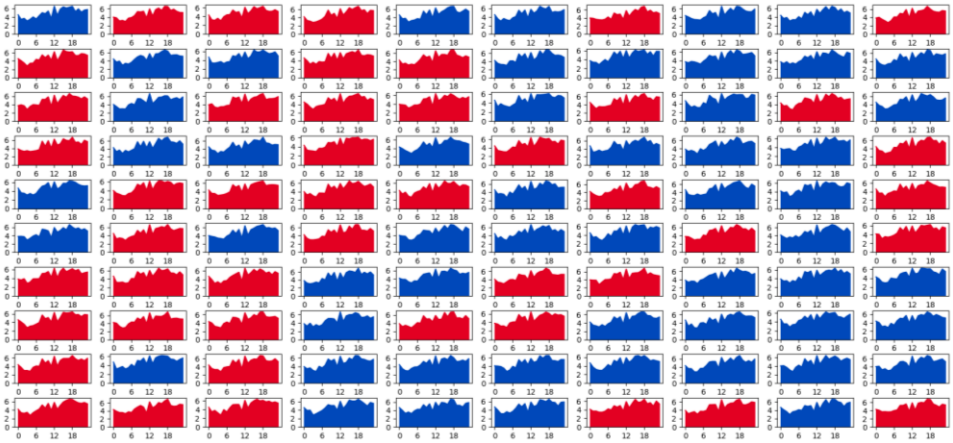
6. Results

The results of the computed labels for the testing data were saved in a TestingResults.txt in the format of the training data with each entry consisting of 25 numbers (24 of them being the hours of the day and 0/1 label representing the curve as either “Normal” or “Abnormal”). The results show that there were 54 abnormal predictive guideline price curves with a high accuracy of 94.8%.

1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0

TestingData produced labels.

The results of the classification algorithm on the testing data were also plotted using the PlotGraph() method that plots graphs for the prediction of each predictive guideline price curve from the testing dataset.
The x-axis represents the hours of the day and the y-axis represents the price of each pricing curve. Graphs that are in blue color represent the “Normal” pricing curves and the red color graphs represent the “Abnormal” pricing curves as computed by the SVC.



Graphs representing Normal and Abnormal pricing curves.

Overall, we can conclude that the data is abnormal due to the larger number of curves being labeled as “Abnormal”.

7. Reference

1. https://developers.google.com/machine-learning/crash-course/classification/accuracy - :~:text=Accuracy = Number of correct predictions,N + F P + F N

2. https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

3. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

4. https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

8. Appendix

Appendix 1:

1	User & Task ID	Ready Time	Deadline	Maximum scheduled energy per hour	Energy Demand
2	user1_task1	20	23	1	1
3	user1_task2	18	23	1	2
4	user1_task3	19	21	1	1
5	user1_task4	12	20	1	3
6	user1_task5	6	12	1	3
7	user1_task6	18	20	1	2
8	user1_task7	4	10	1	2
9	user1_task8	12	18	1	2
10	user1_task9	7	14	1	3
11	user1_task10	8	14	1	3
12	user2_task1	11	22	1	2
13	user2_task2	5	11	1	2
14	user2_task3	5	23	1	1