III. Model documentation and write-up

You can respond to these questions either in an e-mail or as an attached file (any common document format is acceptable such as plain text, PDF, DOCX, etc.) **Please number your responses.**

1. Who are you (mini-bio) and what do you do professionally?

My passion about Mathematics came from my results on Math competitions in Brazil, where I was born. Thanks to the gold medal I won in 2006 on the toughest math competition in Brazil (OBMU - Brazilian Math Olympiad for University Students), I was invited to participate in the Ecole Polytechnique's engineering program. I came to France in January 2009, since then, I became a husband, a father of a little boy (Edouard) and an entrepreneur (founder at Wintics).

I finished my studies at Ecole Polytechnique in September 2012. I chose then to start my career at a consulting firm in order to develop some business skills that my academic studies didn't provide me with. I worked for 4.5 years at Accuracy (consulting firm) where most of the time I worked on highly quantitative projects (Financial modelling, Derivatives Pricing and Data Science).

Last year I quitted that comfortable job to start a professional adventure : Wintics, a start-up that develops taylor-made AI solutions for corporates.

If you are on a team, please complete this block for each member of the team.

2. High level summary of your approach: what did you do and why?

I combine prediction-based and rule-based approaches to detect abnormal energy consumption. The main idea is to fit a machine learning model that predicts the energy consumption for the next timestamp, then we measure kind of the level of "surprise" of the model based on the gap between the prediction and the true consumption. If we find a gap, then either ***(i)*** the model actually made a mistake, or ***(ii)*** an abnormal energy usage has happened. When detecting overconsumption, we want to avoid the cases where the model just made a mistake. Therefore, we complete our diagnostic with a rule-based approach which basically filters the anomalies found by the predictive model with respect to some statistical criteria. The main metric for the rule-based approach is the percentile of the energy consumption for the timestamp given the ***(i)*** hour, ***(ii)*** outside temperature and ***(iii)*** type of day (working day vs holiday).

3. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

First part: Determining the consumption intervals for the predictive model. I thought that fitting a regression model would be very hard given the noise and the number of abnormal consumptions. Then, I fitted a classification model that predicts the range in which the energy consumption would be for the next timestamp.

```
def consumption_label(x):
    if x >= np.percentile(train['Values'], 99):
        return 'VeryHigh'
    elif x >= np.percentile(train['Values'], 95):
        return 'High'
    elif x >= np.percentile(train['Values'], 80):
        return 'MediumHigh'
    elif x >= np.percentile(train['Values'], 60):
        return 'Medium'
    elif x >= np.percentile(train['Values'], 40):
        return 'MediumLow'
    elif x >= np.percentile(train['Values'], 20):
        return 'Low'
    elif x >= np.percentile(train['Values'], 5):
        return 'VeryLow'
    elif x >= np.percentile(train['Values'], 1):
        return 'StandBy'
    else:
        return 'Off'
```

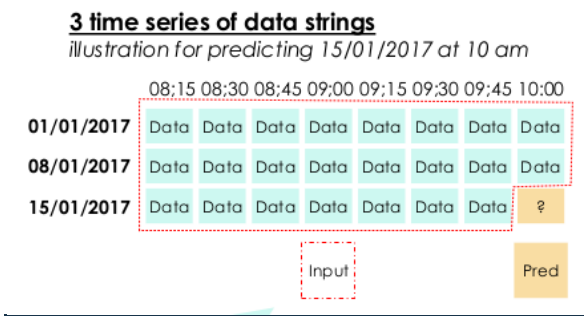
```
train['Values'] = train['Values'].fillna(-1)
```

```
train['target'] = train['Values'].apply(lambda row: consumption_label(row), 1)
```

Second part: Building the training dataset. The main idea was to catch intra-day patterns to help fitting the predictive model.

```
def generate_df(df, sequence_length):

    target_labels = ['Off', 'StandBy', 'VeryLow', 'Low', 'MediumLow', 'Medium', 'MediumHigh', 'High','VeryHigh']
    target_labels = np.array(target_labels)

    X = []
    y = []

    for i in range(96*7*2, len(df) - sequence_length - 1):
        for j in range(3):
            if j == 0:
                tmp = feature_scaler.transform(df[features].iloc[i:(i+sequence_length)])
            else:
                ts = feature_scaler.transform(df[features].iloc[(i-j*96*7):(i+sequence_length-j*96*7+1)])
                tmp = np.concatenate((tmp, ts), axis=0)

        X.append(tmp)
        y.append(np.where(target_labels == df['target'].iloc[i+sequence_length + 1])[0][0])

    return np.array(X), np.array(y)
```

Here is an illustration of the code below :



Third part: Calculating the entropy of predictions. This step was critical since it takes into account the level of confidence of the prediction.

```
def entropy(model_dict, sequences, labels):
    model = WinticsRNN(hidden_size=hidden_size,
                       features_size=len(features))

    model.load_state_dict(model_dict)
    model.cuda()
    model.eval()

    sequences = torch.FloatTensor(sequences)
    sequences = autograd.Variable(sequences).cuda()

    labels_true = torch.LongTensor(labels)
    labels_true = autograd.Variable(labels_true).cuda()

    labels_predicted = model(sequences)

    entropy = [exp_logits/torch.sum(exp_logits) for exp_logits in torch.exp(labels_predicted)]
    entropy = [-torch.log(probs).dot(probs) for probs in entropy]
    entropy = np.array([e.data.cpu().numpy() for e in entropy])

    _, labels_predicted = labels_predicted.max(1)
    labels_predicted = labels_predicted.data.cpu().numpy()

    print("Dataset accuracy: %.3f" % accuracy_score(labels_predicted, labels))
    cm = confusion_matrix(labels_predicted, labels)
    extended_diagonal = cm.trace()+cm[1:, :-1].trace()+cm[:-1, 1:].trace()
    extended_diagonal_2 = extended_diagonal+cm[2:, :-2].trace()+cm[:-2, 2:].trace()

    print("Dataset extended accuracy (max gap 1): %.3f" % (extended_diagonal/np.sum(cm)))
    print("Dataset extended accuracy (max gap 2): %.3f" % (extended_diagonal_2/np.sum(cm)))

    return labels_predicted, entropy
```

```
overconsumption_train, entropy_train = entropy(best_model, rnn_sequences, rnn_labels)
```

```
Dataset accuracy: 0.789
Dataset extended accuracy (max gap 1): 0.983
Dataset extended accuracy (max gap 2): 0.995
```

```
overconsumption_val, entropy_val = entropy(best_model, val_sequences, val_labels)
```

```
Dataset accuracy: 0.779
Dataset extended accuracy (max gap 1): 0.981
Dataset extended accuracy (max gap 2): 0.994
```

4. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

I tried to detect anomalies via Fourier transform. If I had more time, I would definitely go further and explore that idea. The main idea would be to build features in the frequency domain and then apply some machine learning model.

5. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

No, the Jupyter notebook I prepared takes the raw data as input.

6. How did you evaluate performance of the model other than the provided metric, if at all?

Since we were dealing with an unsupervised learning problem, I didn't have a relevant metric other than the score on the public leaderboard. I am not even sure that it is possible to build a machine learning model that actually reaches a very high score (i.e., more than 85%) with the information we were given.

7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

Nothing to report.

8. Do you have any useful charts, graphs, or visualizations from the process?

Illustration for the Site 334 : the correlation between consumption and outside temperature. It gives a clear clue about what consumption level to expect given the outside temperature.

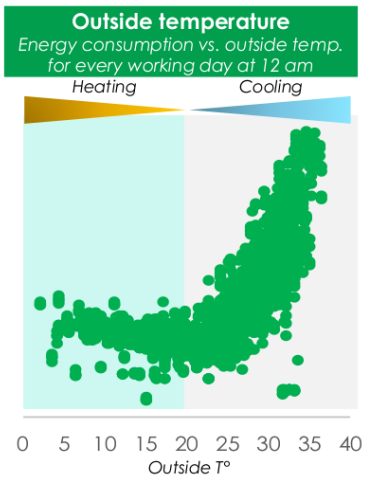
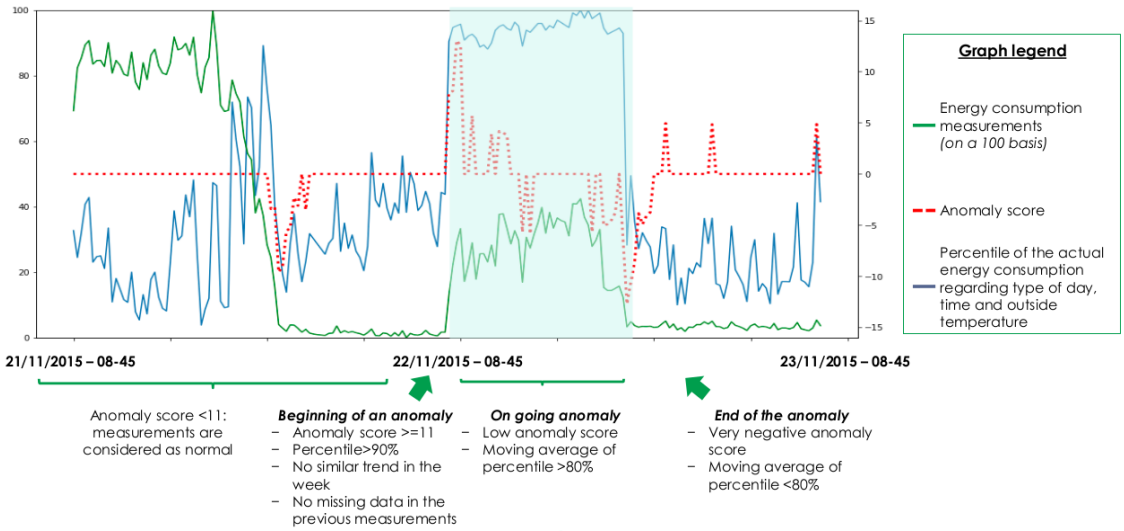


Illustration for the Site 334 : overlapping time series ***(i)*** the energy consumption, ***(ii)*** the anomaly score and ***(iii)*** the percentile of the energy consumption.



9. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

It is surely easier to work on supervised or semi-supervised learning problems. I think that if we were given more information about anomalies (examples, descriptions, causes, etc) then it would be easier to get better models and even discover new types of anomalies. Moreover, some qualitative information would be very useful like : is this a normal working week ? Or is there any punctual event going on (seminar, conference, etc) ?

Just as I said, in the answer of the question 4: I would explore frequency domain features. Moreover, I would definitely work on my predictive model. The actual architecture is quite simple, and a better predictive model would surely highlight more abnormal energy consumptions.