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?

   I am assistant professor (in computer science) at University of Warsaw and data scientist in Togetherdata. My main research area is the text algorithms and efficient data structures for texts. During last two years I'm enjoying machine learning. I'm also a big fan of data visualization and data story telling.

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

   First I've divided problem into server problems:
   For each expected recipe mask (1111, 1001, 1100) and for each number of avilable phases (1, 2, 3, 4) create data set that includes only available data (so no data beyond available phases) and train separate model.

   For creating models I've used combination of Keras and LightGBM models (with results blened via linear combination).

   I've used multiple agregate features (like length of each phase, sum/mean turbodity in each phase, mean tank levels, mean/max pressure, etc).

   I've used target mean features that were used for scaling the target values. For computing target mean I've used weighted median and segmentation of data by exepcted recipe mask and object id.
   Just using weighted medians give quite good prediction, and simplify the training.

   For local validation I've used time-based split.

   The final submission is a blend of two solutions generated from two different random seeds.

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

Train/validation/test segmentation:

```
def segment(self, train, validate, test):
    assert test is not None
    for (e, mx), seg_test in test.groupby([
```

```
            test['expected_phase_summary'],
            test['phase_summary'].str.rstrip("?").str.rstrip("0").map(len).clip(1, 4)
        ]):
        f = lambda x: x[(x.expected_phase_summary==e) & (x.max_phase_num>=mx)]
        g = lambda x: self.filter_features(x, mx)
        seg_train = g(f(train)) if train is not None else None
        seg_validate = g(f(validate)) if validate is not None else None
        seg_test = g(seg_test)
        yield ((e, mx), len(seg_test) / len(test)), (seg_train, seg_validate, seg_test)
```

Scaling target values before training:

```
    @staticmethod
    def scale_target(train, res):
        res['target_shift'] = res['f_phase_1_target_median_0.50']
        res['target_scale'] = (
            res['f_phase_1_target_median_0.50'] - res['f_phase_1_target_median_0.01']
        ).clip(10**3, None)
        return res
```

Keras Model:

```
    def gen_model(self, feat, output_size):
        keras_set_random_state(0)
        inputs = keras.layers.Input(shape=(len(feat), ), name='input')
        x = []
        for s in sorted(set(map(lambda x: re.sub(r'^(f_phase_\d)_(.*)$', r'\1', x), feat))):
            if s in feat:
                x.append(keras_slice_layer(inputs, feat, sel_f=[s]))
            else:
                y = keras_slice_layer(inputs, feat, pattern=f'^{s}_.*$')
                y = keras.layers.Dense(self.size_per_block, activation='relu')(y)
                x.append(y)

        if len(x) > 1:
            x = keras.layers.concatenate(x)
        else:
            x = x[0]

        for layer_num in range(self.layers_num):
```

```
        x = keras.layers.Dense(self.network_size, activation='relu')(x)
        x = keras.layers.Dropout(self.dropout, seed=layer_num)(x)

    outputs = keras.layers.Dense(output_size, activation='linear')(x)
    model = keras.models.Model(inputs=inputs, outputs=outputs)
    model.compile(optimizer='adam', loss='mean_absolute_error')
    model.summary()
    return model
```

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

   I've tried using different scaling techniques, but the results were rather disapointing. Also I've tried different segmenting schemes, but models were overfitting (possible due to the small amount of data in some segments).

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

   I've used ad-hoc jupyter notebooks for data-analysis and checking the quality of the submissions.

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

   I've used provided metric implemented as mean absolute error with weights equal to 1/max(threshold, target value).

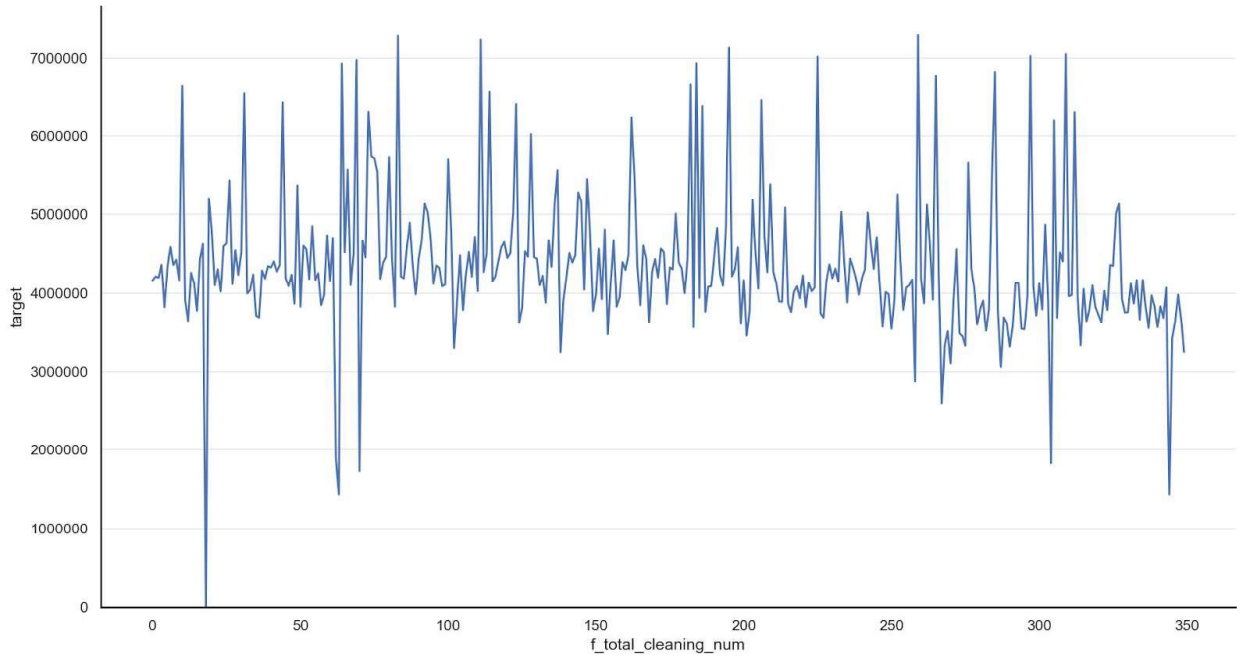
7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

   The solution does not have any special requirements, you can you run it on simple laptop computer.

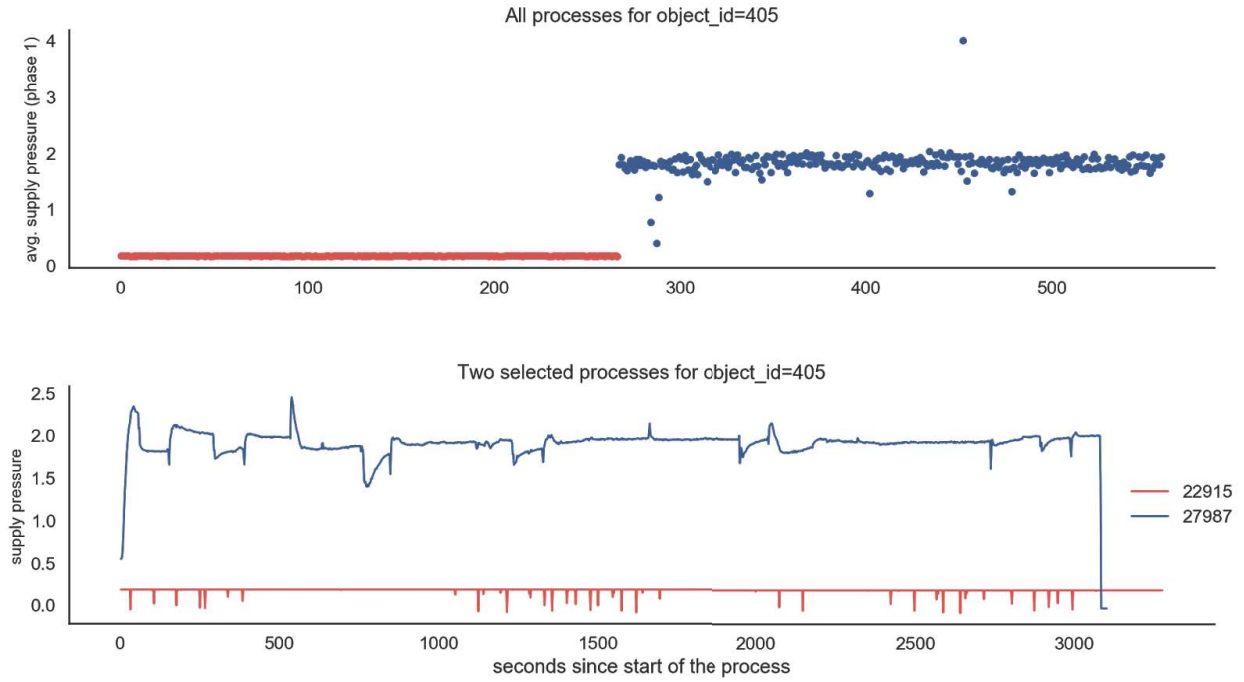
8. Do you have any useful charts, graphs, or visualizations from the process?

Visualization of sample process (27987):

Target values for single object (object_id=405):



Weird pressure values for object 405:



All pressure readings for first part of processes (marked with red color) were limited to 0.17, in the remaining part readings were much higher.

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?

   I've never managed to fix the unreliable data (like values for small flow). I think that without this there is too much noise in the data. Also my local validation was not precise.