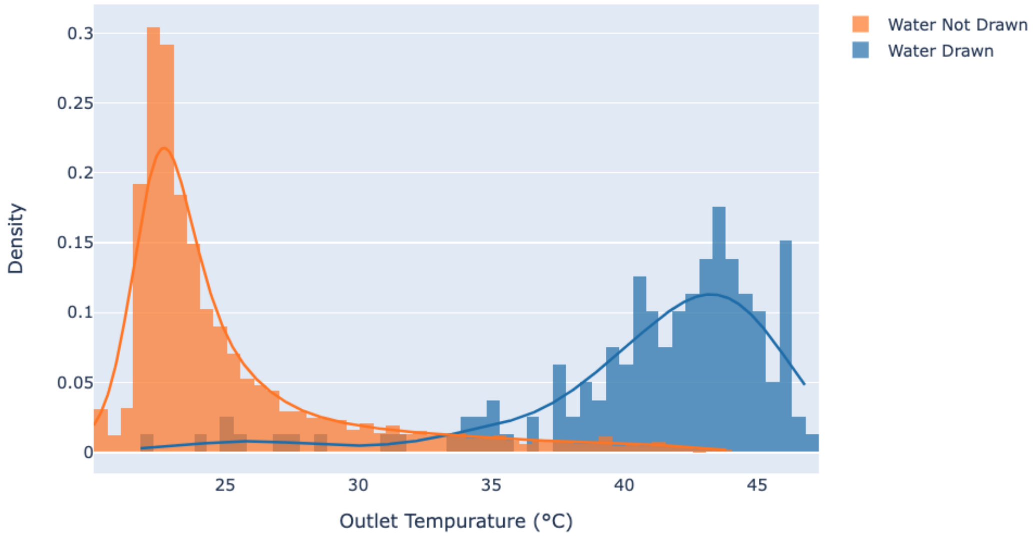
Identifying Water Draws Report

By Luke Neuendorf

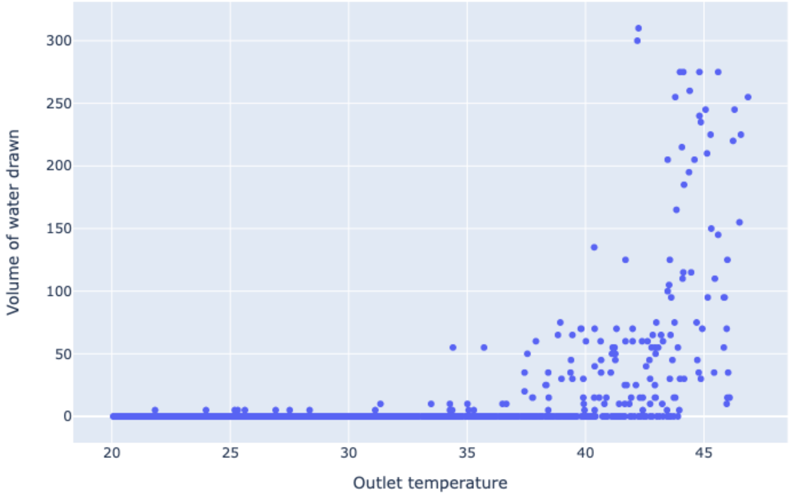
Github: https://github.com/lneuendorf/Plentify_Project (Note: the Jupyter Notebook Plotly plots are not rendering on Github so I saved them as images in the "images" folder.)

a) Describe how your flow identification method works.

I started by plotting the data to see what features would provide useful information for classifying whether or not there was a water draw. As seen in the distribution plot below, there is a clear difference in outlet temperature when water is drawn verse when it is not drawn.

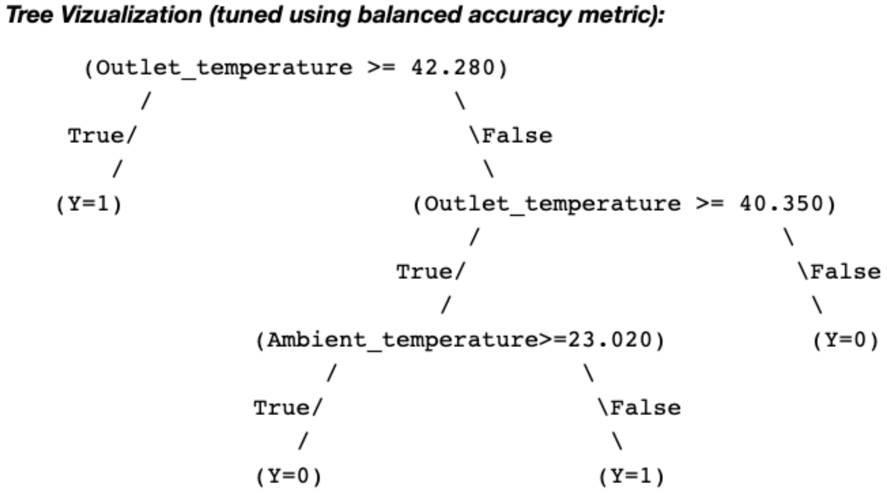


All of the other features have similar "water drawn" verse "water not drawn" distributions, although the ambient temperature distributions have a slight offset. Looking at a scatter plot of outlet temperature vs volume of water drawn (shown below), it is evident that a simple one-split decision tree with a split at 35-43°C should be decent at classifying whether or not water was drawn.



I decided to implement a decision tree classifier because of its easily describable logic. I used information gain to decide what feature/value combination to split on.

I encountered an overfitting problem, where the algorithm was fitting exactly to the training data. Because of this, I added the regularization parameters of max_depth, min_samples_split, and min_samples_leaf. The code for my decision tree implementation can be found in “DTree.py”. Below is the resulting decision tree, where Y=1 indicates water was drawn, and Y=0 indicates water was not drawn.

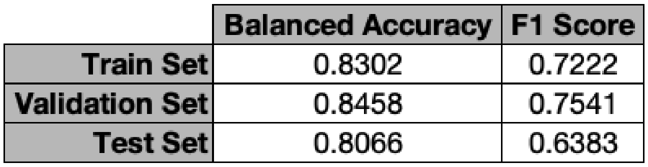


Tree Vizualization (tuned using balanced accuracy metric):

```
     (Outlet_temperature >= 42.280)
        /                    \
   True/                      \False
      /                        \
  (Y=1)                  (Outlet_temperature >= 40.350)
                              /                    \
                         True/                      \False
                            /                        \
            (Ambient_temperature>=23.020)          (Y=0)
                  /                    \
             True/                      \False
                /                        \
             (Y=0)                      (Y=1)
```

b) **Explain why you chose the evaluation metric(s) you did for this problem and the results you achieved on the training set.**

Since this is an unbalanced classification problem (only 159 “Water Draw” occurrences out of 4,653), metrics such as accuracy do not suffice. So, analyzed my model using both balanced accuracy and F1 score. In terms of confusion matrices, balanced accuracy puts equal importance on negatives and positives, whereas the F1 score focuses on the positives. The F1 score is a function of precision and recall. Balanced accuracy is the average of specificity and sensitivity. The results of the model evaluation are below.



	Balanced Accuracy	F1 Score
Train Set	0.8302	0.7222
Validation Set	0.8458	0.7541
Test Set	0.8066	0.6383

c) **Report on the cases where your method performs poorly.**

False positive predictions most often occur when the outlet temperature is above 40°C but water was not drawn. The average outlet temperature of false positive predictions is 42.3°C with a standard deviation of 1.005°C.

False negatives most often occur when the outlet temperature is below 40°C but water was drawn. The average outlet temperature of false negative predictions is 36.1°C with a standard deviation of 5.10°C.

d) Comment on how you believe your methodology could be improved.

My decision tree implementation is slow and could be sped up by using Numpy arrays instead of Pandas dataframes. I could also try other interpretable machine-learning algorithms such as logistic regression or clustering.

e) Briefly describe how you would go about tackling this project if given more time and scope to implement any methodology (ML or otherwise).

Machine learning modeling accuracy is limited by the quality of the data. In this instance, there is only one useful feature, outlet temperature. I would discuss with my supervisor the idea of adding new sensors that may be useful for predicting water draws. Furthermore, I would try feature engineering to extract more information from the data, such as exponential scaling the ambient temperature feature to create a bigger divide between the ambient temperature distribution where water was and was not drawn. I would also consider applying a deep learning algorithm, although it would be hard to keep it from overfitting on the small dataset and deep learning is somewhat of a black-box approach.