Preregistration: Machine learning models for lagged predictions of next week alcohol use

Kendra Wyant

2024-04-11

Table of contents

Jump to Section 0.4 for preregistered analyses.

0.1 Background

In a previous manuscript (Wyant et al., in press), we developed three separate models that provide hour-by-hour probabilities of future goal-inconsistent alcohol use (i.e., lapses) with increasing temporal precision: lapses in the next week, next day, and next hour. Model features were engineered from raw scores and longitudinal change in responses to 4X daily ecological momentary assessment (EMA). These features were derived to measure theoretically-implicated risk factors including past use, craving, past pleasant events, past and future risky situations, past and future stressful events, emotional valence and arousal, and self-efficacy. We demonstrated that it was possible to predict future alcohol lapses in the next week, day, and hour with high sensitivity and specificity (area under the receiver operating curves [auROCs] of 0.89, 0.90, and 0.93 respectively).

This study is an extension of our previous work. Our previous models predicted the probability of lapse in the next week, next day, and next hour starting now. While, next hour and next day models are well-positioned to identify and recommend just-in-time interventions to address

these immediate risks, the next week model may not have sufficient temporal specificity to recommend immediate patient action. We believe its clinical utility may improve if we shift this coarser window width into the future. This "time-lagged" model could provide patients with increased lead time to implement multi-modal supports that might not be immediately available to them (e.g., schedule therapy appointment, request support from an AA sponsor). The current study proposes training models to predict the probability of lapse at any point during a week window that is shifted 24 hours (one day), 72 hours (three days), 168 hours (one week), or 336 hours (two weeks) into the future. Our primary goal is to assess how model performance changes as lag increases.

0.2 Modeling Decisions

Our primary performance metric for model selection and evaluation will be auROC.

We will consider four candidate statistical algorithms including elastic net, XGBoost, regularized discriminant analysis (rda), and single layer neural networks. Candidate model configurations will differ across sensible values for key hyperparameters. They will also differ on outcome resampling method (i.e., no resampling and up-sampling and down-sampling of the outcome using majority/no lapse to minority/lapse ratios ranging from 1:1 to 2:1).

We will use participant-grouped, nested cross-validation for model training, selection, and evaluation with auROC. We will use 1 repeat of 10-fold cross-validation for the inner loops (i.e., *validation* sets for model selection) and 3 repeats of 10-fold cross-validation for the outer loop (i.e., *test* sets for model evaluation).

Best model configurations (i.e., the best performing combination of algorithm, hyperparameter values, and resampling method) will be selected using median auROC across the 10 validation sets. Final performance evaluation of those best model configurations used median auROC across the 30 test sets.

0.3 Progress at Time of Preregistration

We have fit the inner nested k-fold cross-validation loop for all XGBoost, glmnet, and RDA model configurations.

Next, we will train our remaining model configurations using the neural network algorithm and evaluate our 30 best model configurations for each model in the outer cross-validation loops. Importantly, we have not fit any models in the 30 outer folds or conducted any planned data analyses at the time of this preregistration.

0.4 Preregistered Analyses

The following sections contain our preregistered analyses. We both describe this process in the text and show sample code using simulated data to be as precise as possible.

0.4.1 Model Evaluation

We will use a Bayesian hierarchical generalized linear model to estimate the posterior probability distributions and 95% Bayesian credible intervals (CIs) for auROC for the five best models (no lag and 24 hour, 72 hour, 168 hour, and 336 hour lagged predictions) on the 30 held-out test sets. We will plot the posterior probability distributions. We will report the median posterior probability and 95% credible intervals (CI) for auROC for our baseline (no lag) model and each lagged model. The median posterior probability for auROC will represent our best estimate for the magnitude of the auROC parameter for each model. If the confidence intervals do not contain .5 (chance performance), this will suggest our model is capturing signal in the data.

0.4.1.1 Sample code with simulated data

Below we demonstrate our model evaluation process for auROC with 30 simulated held-out tests sets for each model. We also show how we will write up our results using the simulated data.

```
# Repeated CV (id = repeat, id2 = fold within repeat)
# with a common variance:  statistic ~ model + (model | id2/id)
set.seed(101)
pp <- sim_auroc |>
  pivot_wider(names_from = lag, values_from = auroc) |>
  rename(id = repeat_num,
         id2 = fold_num) |>
  perf_mod(formula = statistic ~ model + (1 | id2/id),
         transform = tidyposterior::logit_trans,  # for skewed & bounded AUC
         iter = 2000, chains = 4, adapt_delta = .99, # defaults but may increase to fix dive
         family = gaussian,
)
```

```
sim_auroc_perf <- sim_auroc |>
  group_by(lag) |>
  summarize(median = median(auroc), min = min(auroc),
            max = max(auroc), IQR = IQR(auroc))

sim_auroc_perf
```

```
# A tibble: 5 x 5
    lag median   min   max    IQR
  <dbl>  <dbl> <dbl> <dbl>  <dbl>
1     0  0.891 0.808 0.952 0.0402
2    24  0.863 0.796 0.957 0.0479
3    72  0.861 0.779 0.947 0.0488
4   168  0.809 0.739 0.890 0.0526
5   336  0.794 0.731 0.850 0.0456
```

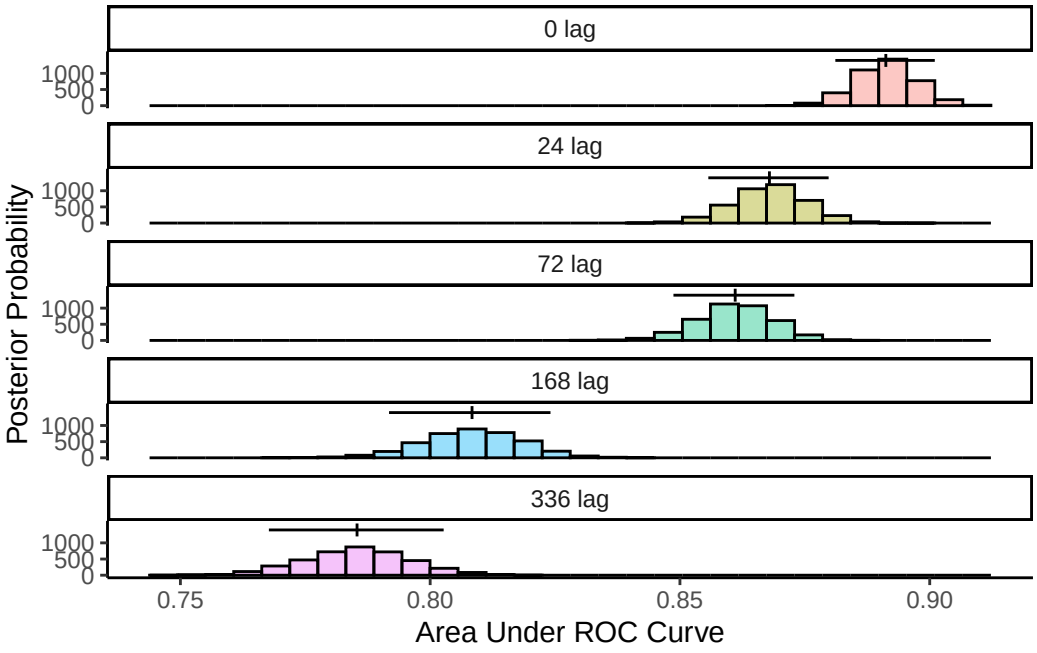
The median auROC across the 30 test sets for our baseline model was high (median=0.891, IQR=0.040, consistent with our previous study. Performance across our lagged models was moderate to high for the 24 hour lag (median=0.863, IQR=0.048, 72 hour lag (median=0.861, IQR=0.049, 168 hour lag (median=0.809, IQR=0.053, and 336 hour lag (median=0.794, IQR=0.046.

```
pp_tidy <- pp |>
  tidy(seed = 123)

q = c(.025, .5, .975)
sim_auroc_pp_perf <- pp_tidy |>
  group_by(model) |>
  summarize(pp_median = quantile(posterior, probs = q[2]),
            pp_lower = quantile(posterior, probs = q[1]),
            pp_upper = quantile(posterior, probs = q[3])) |>
  mutate(model = factor(model, levels = c(0, 24, 72, 168, 336),
                        labels = c("0 lag", "24 lag", "72 lag", "168 lag", "336 lag"))) |>
  arrange(model)

sim_auroc_pp_perf
```

```
# A tibble: 5 x 4
  model   pp_median pp_lower pp_upper
  <fct>       <dbl>    <dbl>    <dbl>
1 0 lag       0.891    0.879    0.903
2 24 lag      0.868    0.853    0.882
3 72 lag      0.861    0.846    0.875
4 168 lag     0.809    0.788    0.827
5 336 lag     0.785    0.765    0.806
```

The posterior probability distributions for auROC for each model are displayed below. The 95% credible intervals (CI) are depicted with a horizontal line. The vertical line represents

the median posterior probability for auROC. The median auROCs from these posterior distributions were 0.891 (baseline), 0.868 (24 hour lag), 0.861 (72 hour lag), 0.809 (168 hour lag), and 0.785 (336 hour lag). These values represent our best estimates for the magnitude of the auROC parameter for each model. The 95% Bayesian CI for the auROCs for these models were relatively narrow and did not contain 0.5: baseline[0.879-0.903], 24 hour lag[0.853-0.882], 72 hour lag[0.846-0.875], 168 hour lag[0.788-0.827], 336 hour lag[0.765-0.806].



0.4.2 Model Comparisons

We will perform two sets of Bayesian model comparisons. Our first set will compare each lagged model to our baseline model (no lag). We have demonstrated previously that this model performs well. Our contrasts will show whether we can predict lapses with similar performance with varying lags (predictions shifted into the future). Our four baseline model comparisons will consist of the following contrasts:

Baseline Contrasts:

1. No lag vs. 24 hour lag.

2. No lag vs. 72 hour lag.

3. No lag vs. 168 hour lag.

4. No lag vs. 336 hour lag.

Our second set of model comparisons will assess for performance differences between adjacent lags. This will allow us to determine the probability that the lagged models systematically degrade as predictions are made further into the future (i.e., longer lag time).

Adjacent Lag Contrasts:

1. 24 hour lag vs. 72 hour lag.

2. 72 hour lag vs. 168 hour lag.

3. 168 hour lag vs. 336 hour lag.

For both sets of model comparisons we will determine the probability that the models' performances differed systematically from each other by regressing the auROCs (logit transformed) from the 30 test sets as a function of the contrast. We will set two random intercepts: one for the repeat, and another for the fold within repeat. We will also report 95% (equal-tailed) Bayesian CIs for the differences in performance associated with the Bayesian comparisons. We will plot the difference in the posterior probability distributions for each model comparison. We will also report the precise posterior probability for the difference in auROCs and the 95% Bayesian CIs. If there is a probability > 0.95 that the more lagged model's performance is worse, we will label the model contrast as significant.

0.4.2.1 Sample code with simulated data

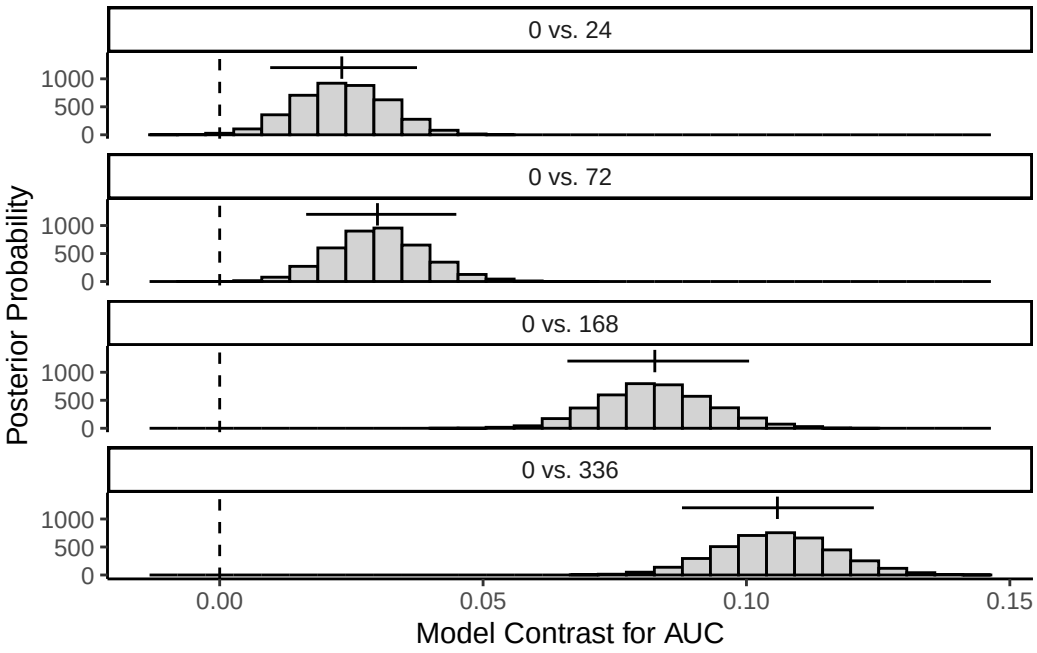
Below we demonstrate our planned model comparisons with simulated data. We also show how we will write up our results using the simulated data.

```
ci_baseline <- pp |>
  contrast_models(list("0", "0", "0", "0"),
                  list("24", "72", "168", "336")) |>
  summary(size = 0) |>
  mutate(contrast = factor(contrast,
                           levels = c("0 vs 24", "0 vs 72", "0 vs 168", "0 vs 336"),
                           labels = c("0 vs. 24", "0 vs. 72", "0 vs. 168", "0 vs. 336")))

ci_median_baseline <- pp |>
  contrast_models(list("0", "0", "0", "0"),
                  list("24", "72", "168", "336")) |>
  group_by(contrast) |>
  summarize(median = quantile(difference, .5)) |>
  mutate(contrast = factor(contrast,
                           levels = c("0 vs. 24", "0 vs. 72", "0 vs. 168", "0 vs. 336")))
```

```r
ci_baseline <- ci_baseline |>
  left_join(ci_median_baseline, by = c("contrast"))

ci_baseline |>
  select(contrast, probability, median, lower, upper) |>
  arrange(contrast)
```

```
# A tibble: 4 x 5
  contrast  probability median   lower  upper
  <fct>           <dbl>  <dbl>   <dbl>  <dbl>
1 0 vs. 24        0.996 0.0232 0.00959 0.0375
2 0 vs. 72        1.00  0.0300 0.0164  0.0449
3 0 vs. 168       1     0.0826 0.0660  0.101
4 0 vs. 336       1     0.106  0.0878  0.124
```

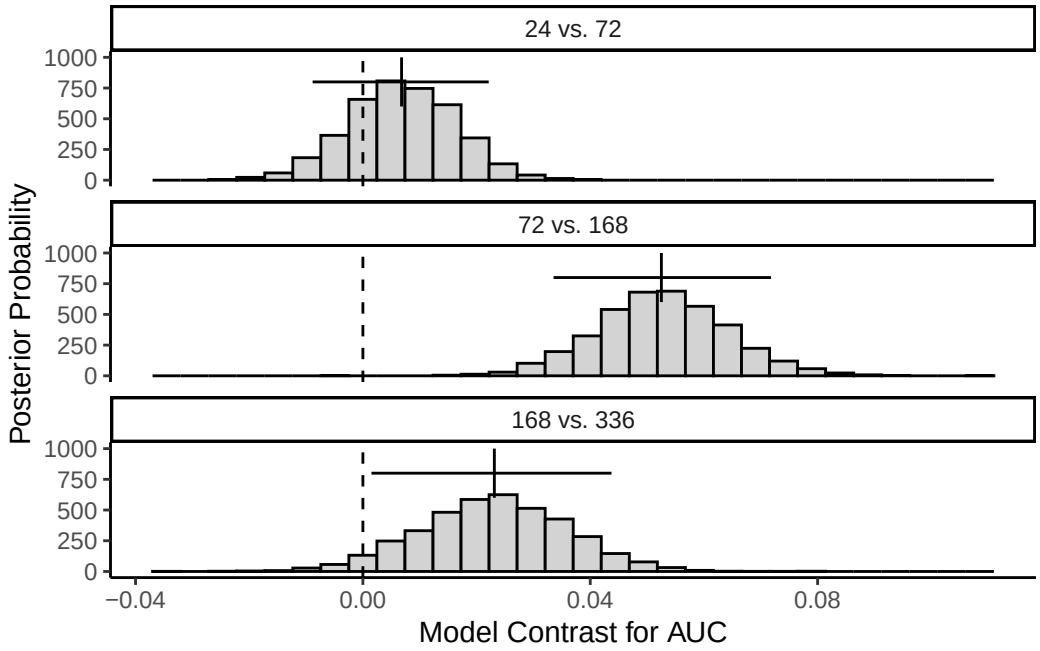
The median decrease in auROC for the baseline vs. 24 hour lag model was 0.023 (95% CI=[0.010-0.037]), yielding a significant probability of 0.996 that the lagged model had worse performance. The median decrease in auROC for the baseline vs. 72 hour model was 0.030 (95% CI=[0.016-0.045]), yielding a significant probability of 1.000 that the lagged model had worse performance. The median increase in auROC for the baseline vs. 168 hour lag model was 0.083 (95% CI=[0.066-0.101]), yielding a significant probability of1.000 that the lagged model had worse performance. The median increase in auROC for the baseline vs. 336 hour lag model was 0.106 (95% CI=[0.088-0.124]), yielding a significant probability of 1.000 that the lagged model had worse performance. The plot below presents histograms of the posterior probability distributions for these model contrasts on auROC.




```
ci_lag <- pp |>
  contrast_models(list("24", "72", "168"),
                  list("72", "168", "336")) |>
  summary(size = 0) |>
  mutate(contrast = factor(contrast,
                           levels = c("24 vs 72", "72 vs 168", "168 vs 336"),
                           labels = c("24 vs. 72", "72 vs. 168", "168 vs. 336")),
         y = 1200)

ci_median_lag <- pp |>
  contrast_models(list("24", "72", "168"),
                  list("72", "168", "336")) |>
  group_by(contrast) |>
  summarize(median = quantile(difference, .5)) |>
  mutate(contrast = factor(contrast,
                           levels = c("24 vs. 72", "72 vs. 168", "168 vs. 336")))

ci_lag <- ci_lag |>
  left_join(ci_median_lag, by = c("contrast"))

ci_lag |>
  select(contrast, probability, median, lower, upper) |>
  arrange(contrast)
```

```
# A tibble: 3 x 5
  contrast    probability  median    lower  upper
  <fct>             <dbl>   <dbl>    <dbl>  <dbl>
1 24 vs. 72         0.766 0.00680 -0.00884 0.0221
2 72 vs. 168        1.00  0.0525   0.0335  0.0718
3 168 vs. 336       0.961 0.0231   0.00150 0.0437
```

The median decrease in auROC for the 24 hour vs. 72 hour lag model was 0.007 (95% CI=[-0.009-0.022]), yielding a non-significant probability of 0.766 that the 72 hour lag model had worse performance than the 24 hour lag model. The median decrease in auROC for the 72 hour vs. 168 hour lag model was 0.053 (95% CI=[0.034-0.072]), yielding a significant probability of 1.000 that the 168 hour lag model had worse performance than the 72 hour lag model. The median decrease in auROC for the 168 hour vs. 336 hour lag model was 0.023 (95% CI=[0.002-0.044]), yielding a significant probability of 0.961 that the 336 hour lag model had worse performance than the 168 hour lag model. The plot below presents histograms of the posterior probability distributions for these model contrasts on auROC.



0.4.3 Feature Importance Indices

We will calculate Shapley values from the 30 test sets to provide a description of the importance of categories of features across our five models in one of two ways.

1. If our best model is an XGBoost model, we will calculate Shapley values as we did in our previous study. We will use the `SHAPforxgboost` package designed specifically for XGBoost models. This method is preferred in that has low computational cost.
2. However, if our best model configuration uses an algorithm other than XGBoost we will use a model-agnostic, more computationally expensive, method for calculating Shapley values. Specifically, we will calculate Shapley values using the `DALEX` package. This package calculates Shapley values by generating permutations of feature values, computing the model's predictions for each permutation, and then aggregating the differences between these predictions and the original prediction. This process, however, is time-intensive and computationally expensive.

Once Shapley values are calculated, we will average the three Shapley values for each observation for each feature (i.e., across the three repeats) to increase their stability. An inherent property of Shapley values is their additivity, allowing us to combine features into feature categories. We will create separate feature categories for each of the nine EMA questions, the rates of past alcohol use and missing surveys, the time of day and day of the week of the start of the prediction window, and the seven demographic variables included in the models.

To calculate the local (i.e., for each observation) importance for each category of features, we will add Shapley values across all features in a category, separately for each observation. To calculate global importance for each feature category, we will average the absolute value of the Shapley values of all features in the category across all observations. These local and global importance scores based on Shapley values allow us to contextualize relative feature importance.

We will plot the relative ordering of global feature importance using a bar plot. We will also use a sina plot to show the distribution of local feature importance for each observation.