STATS202 Final Project

AUTHOR
Bue, Alex

PUBLISHED
August 10, 2025

Summary

Selection

I conduct visualizations first. Some initial data exploration shows:

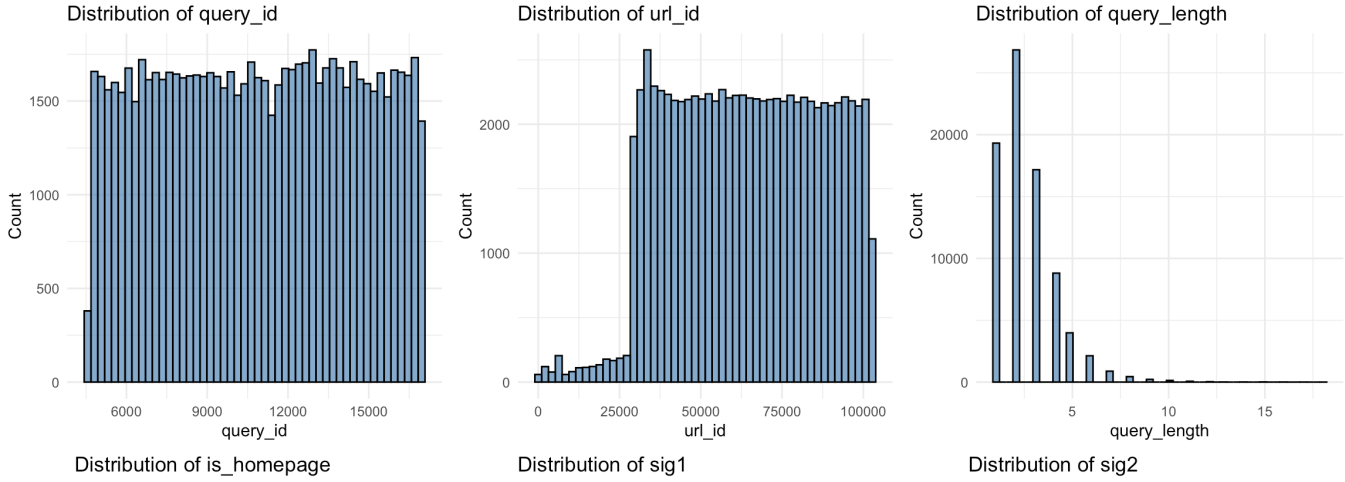
- The dependent variable is not imbalanced ($\hat{p} = .4371$)
- `id` and `query_id` are unique identifiers.
- `url_id` is not clearly a unique identifier because the distribution of histogram buckets is not uniform.
- `is_homepage` is a binary variable.
- `query_length` is a count.
- `sig3`, `sig4`, `sig5`, `sig6` are all highly skewed.

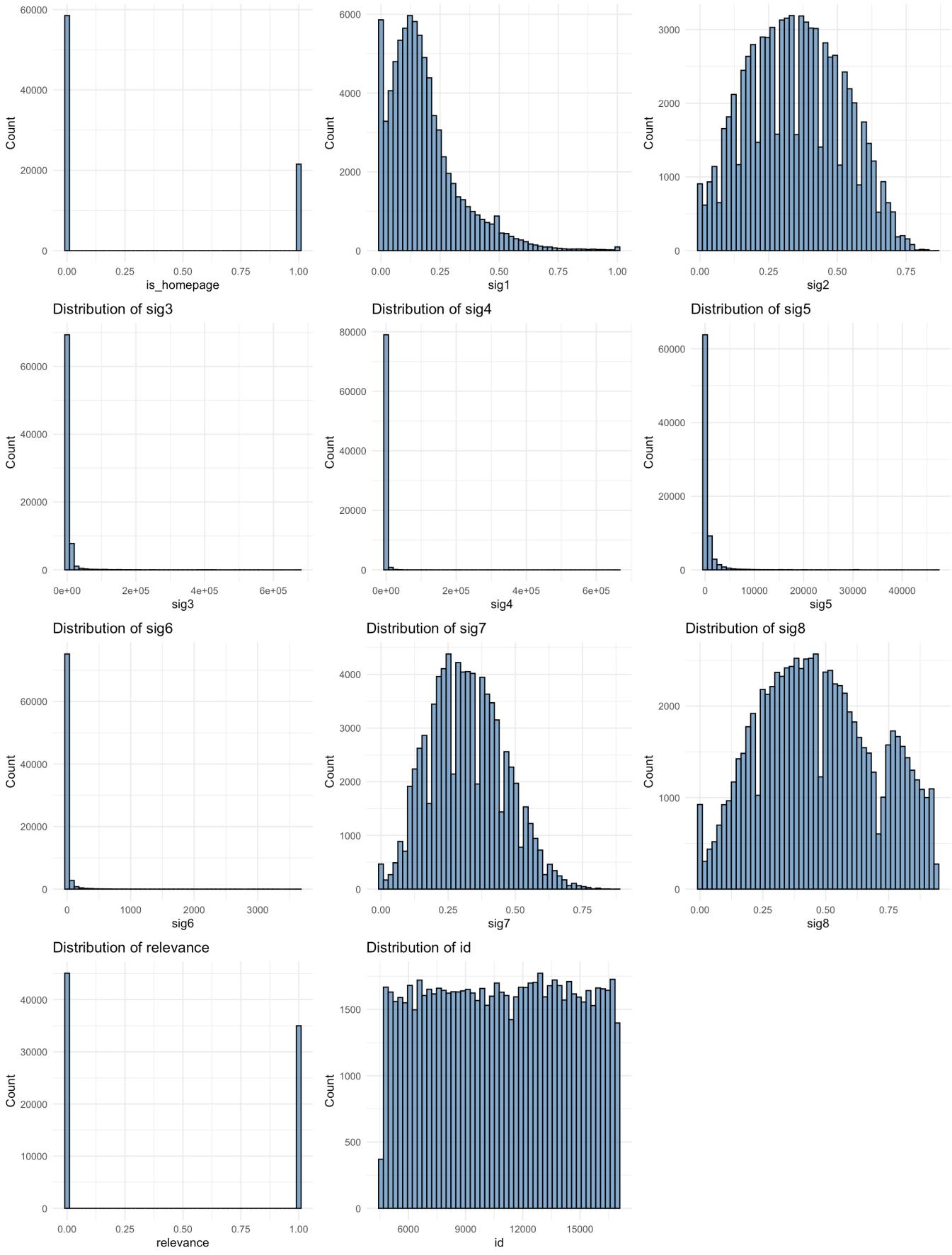
I visualize the density of each predictor grouped by dependent variable. The overlay of the plots helps visualize whether the variable of interest is systematically higher for certain values of the predictors. To make the contrast more conspicuous, I log-transform the skewed variables. I also group predictor values into deciles and condition on the dependent variable. `sig2` appears conspicuously useful as a separator of the dependent variable.

For completeness I investigate multicollinearity but find no significant ($|r| > .8$) correlations. Variance inflation is also more germane to inference than to prediction; the point estimates with multicollinearity do not change even while standard errors increase.

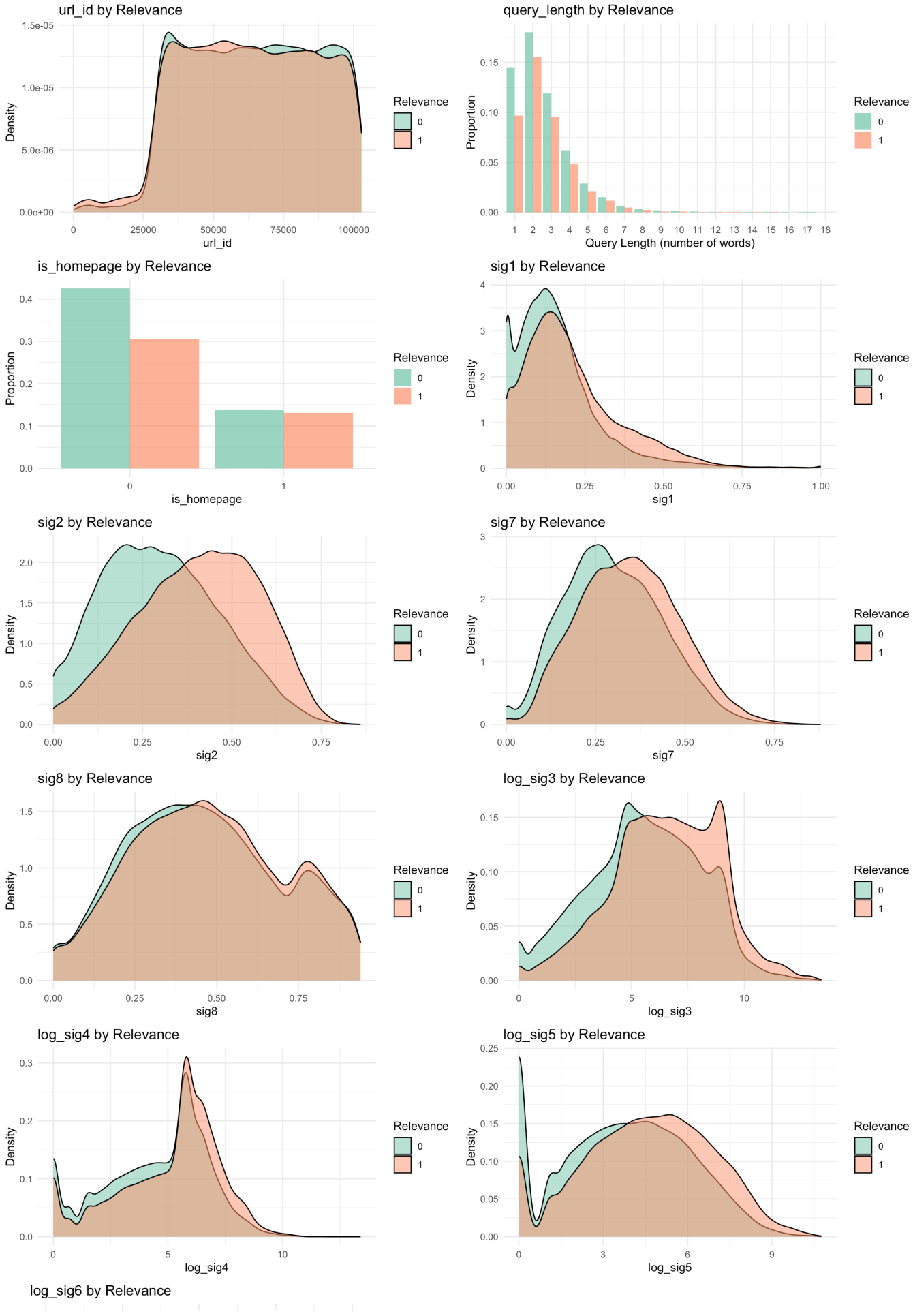
Visualizations

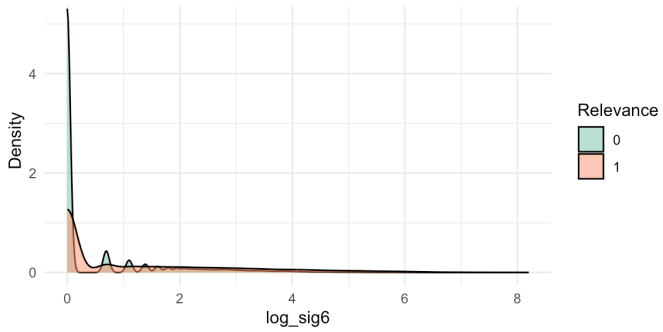
Distributions of Variables



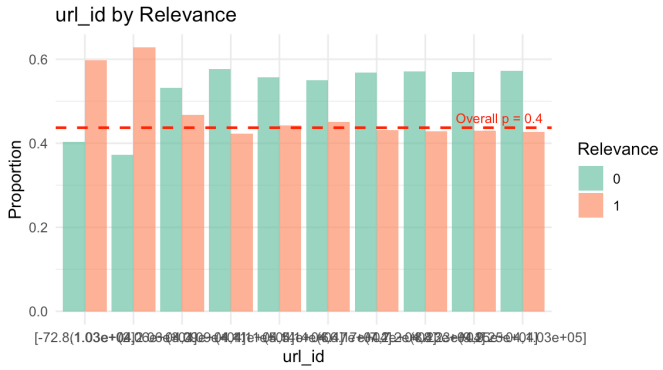


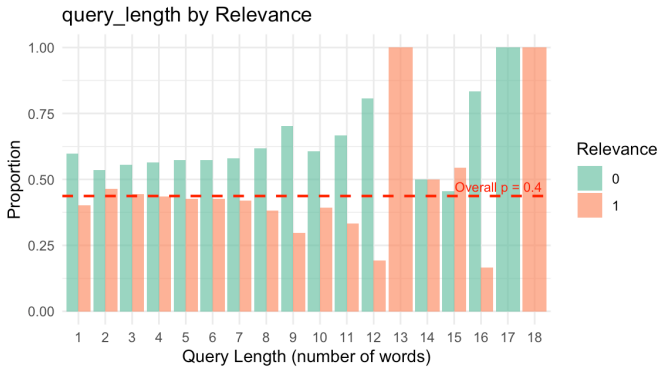
Histogram of Select Predictors Conditioned on Dependent Variable

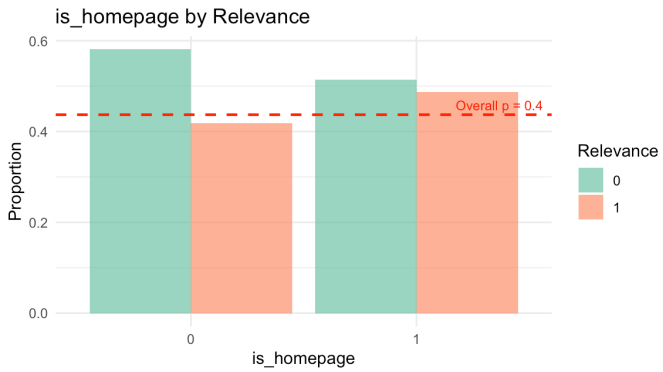

url_id by Relevance
Density
url_id
Relevance
0
1
query_length by Relevance
Proportion
Query Length (number of words)
is_homepage by Relevance
Proportion
is_homepage
sig1 by Relevance
Density
sig1
sig2 by Relevance
Density
sig2
sig7 by Relevance
Density
sig7
sig8 by Relevance
Density
sig8
log_sig3 by Relevance
Density
log_sig3
log_sig4 by Relevance
Density
log_sig4
log_sig5 by Relevance
Density
log_sig5
log_sig6 by Relevance

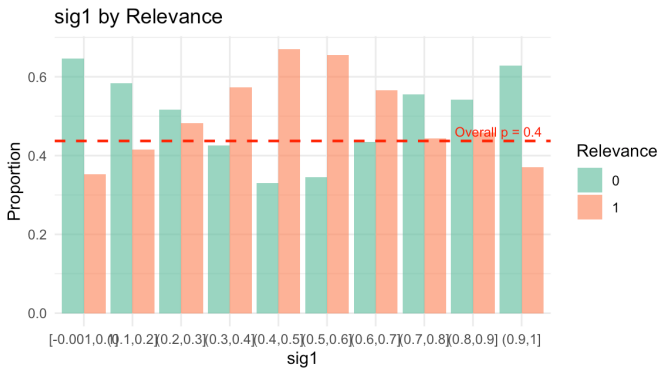


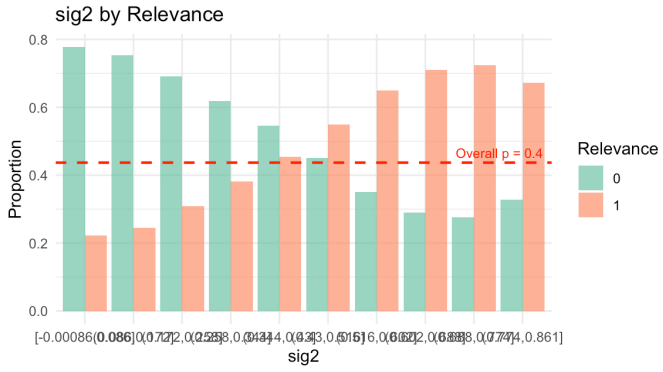
Proportions of Select Predictors Conditioned on Dependent Variables Deciles

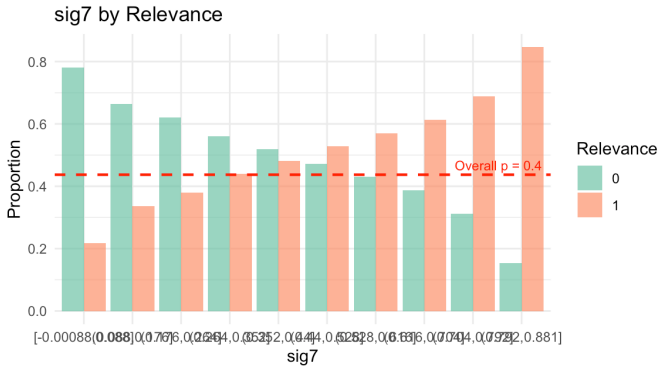


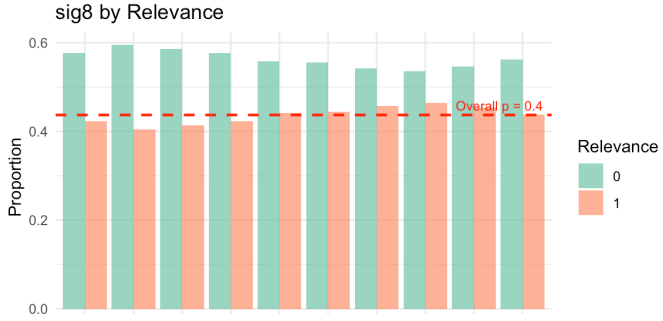


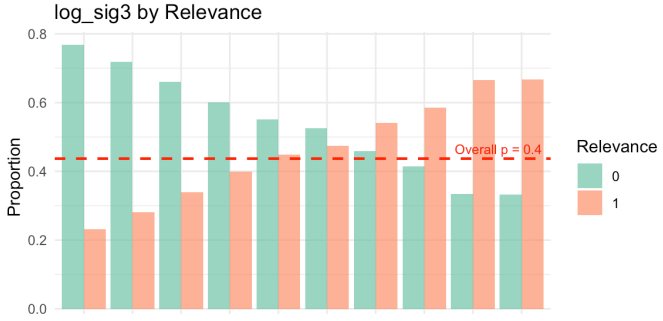


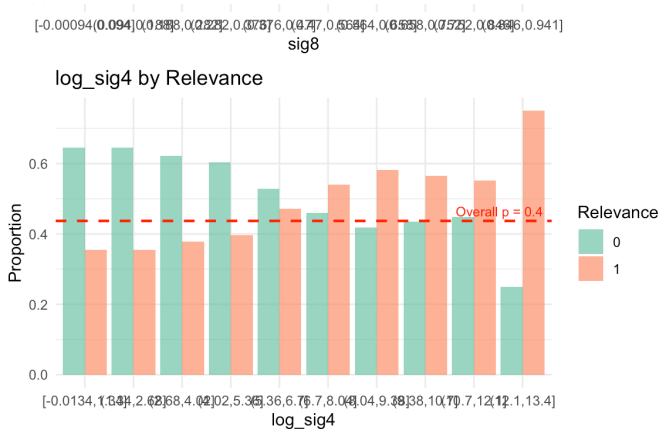


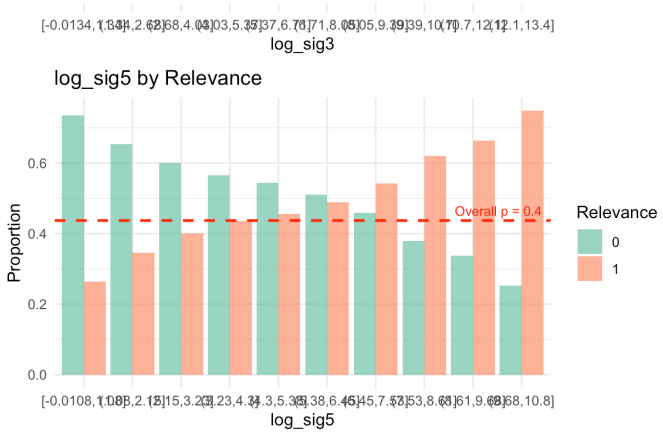


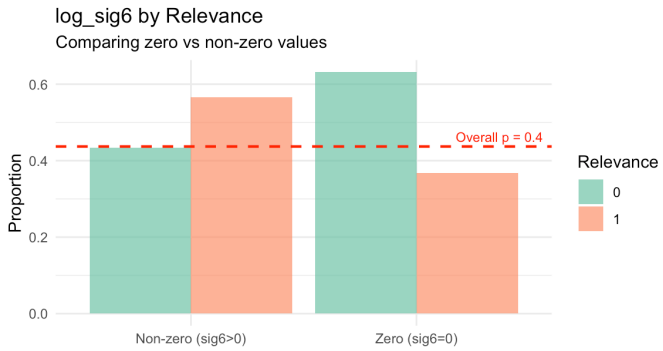




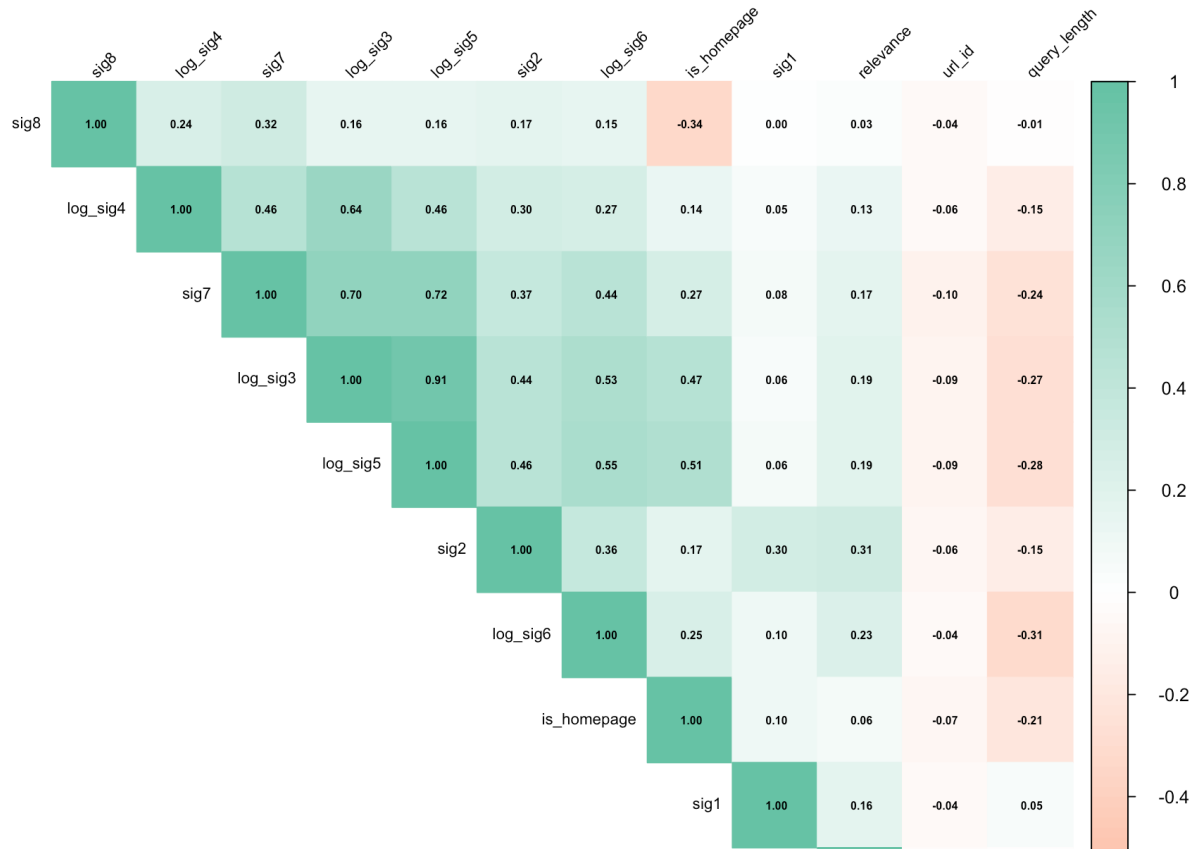


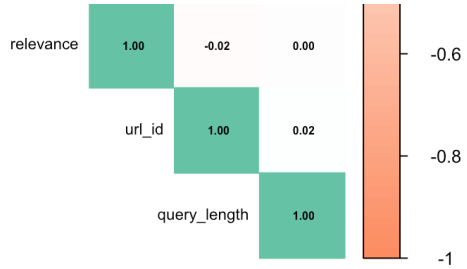






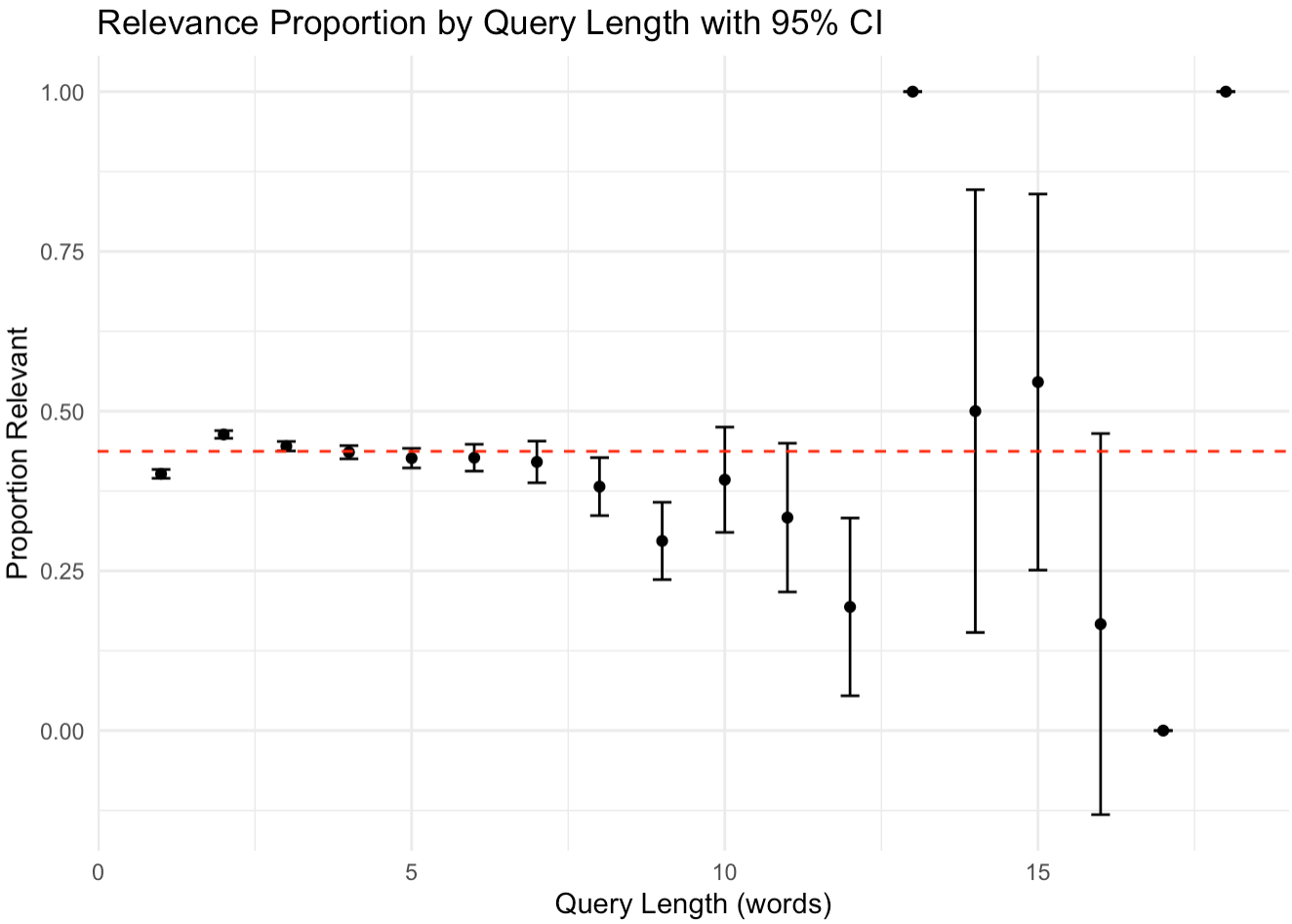
Correlation Plot





Query Length

Longer query lengths have a very high proportion of relevant entries. I calculate standard errors to see whether these trends are statistically significant or may result in over-fitting.

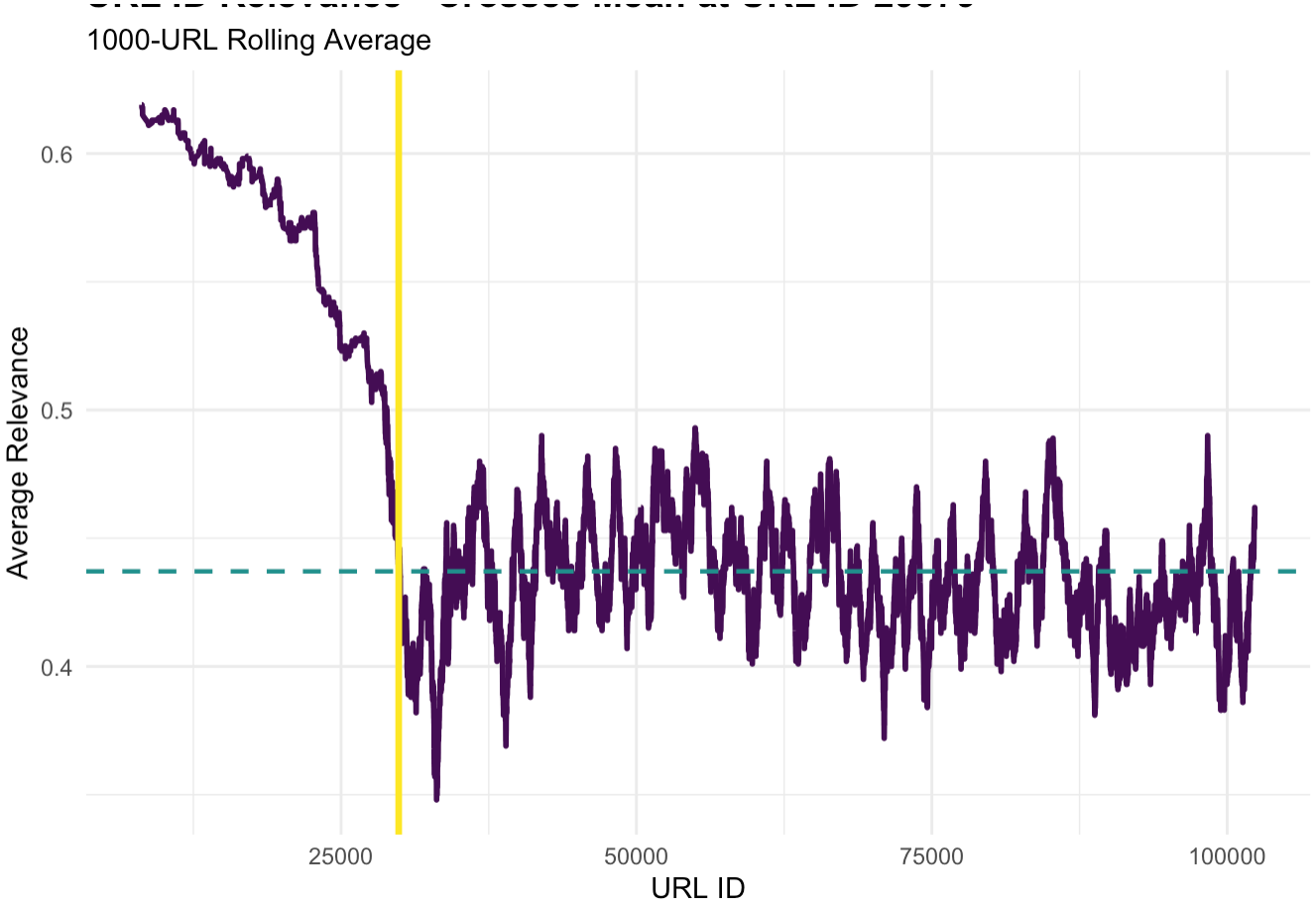


They are not statistically significant.

URL ID

```
Crossing point at URL ID: 29879
```

URL ID Relevance - Crosses Mean at URL ID 29879



Pre-processing

I code `is_homepage` as an unordered factor.

I remove `query_length` because the most predictive lengths are also the most rare. It is sensible to represent `query_length` as a factor variable, because the proportion of relevance does not appear to increase linearly, but factors introduce various challenges with prediction. If some factor is not present in the future data, then our trained model will not work. A compromise is necessary.

`url_id` will need to be handled differently. The numeric scale is not sensible but neither are factors, which would lead to over-fitting. Ultimately, it seems risky to include data that may be idiosyncratic, so I exclude it for conservatism.

`sig2, sig7, and sig8` appear to have regular drop-offs in the histograms. This is an artifact of the measurement scale, which records values in increments of 0.01. Some histogram bins therefore align with the allowed values and receive higher counts, while others fall between these values and receive few or none, producing the visible pattern of regular dips. This is not a problem for analysis.

There are no missing values.

Transformation

I apply the following transformations:

- Log-transform skewed variables (`sig3, sig4, sig5`) for linear models.
  - This linearizes for LASSO, and monotone transformations are unlikely to hurt more flexible modeling strategies.
  - I use `log1p` from base R, which adds a constant of 1 to each value to avoid issues with zeros. The loss of the usual interpretability - where natural logs approximate percentage changes - is not important here.
- Two-part transformation for `sig6`: create a dummy variable indicating zero values, then log-transform the positive values. This preserves the information in zeros (which show a higher proportion of relevance) while making the positive values less skewed.
- I add binary variables for `query_length` 1 and 2, which are statistically significant according to my earlier visualization.
- I add a factor variable for `url_id` below 30000, the cutoff at which mean relevance was highest.
- After transforming, I remove `url_id` and other variables.

Note that I do not standardize at this stage. Since I will later use cross-validation, scaling of the variables should happen within each fold where appropriate.

Data Mining

I will use 10-fold cross-validation to estimate generalization error for each modeling strategy. In all cases, I will evaluate model accuracy, picking the tuning parameters and then the strategy which maximizes accuracy.

My modeling strategies are:

1. LASSO logistic regression: Suitable for correlated predictors and sparse data to reduce variance. Factor variables are one-hot encoded. The tuning variable is λ, imposing a progressively higher penalty that shrinks coefficients towards zero. I use a "kitchen-sink" approach with all interactions.
2. KNN: A simple, intuitive method that makes minimal assumptions about the data's functional form. Factor variables are one-hot encoded; the curse of dimensionality is not salient given the relatively low dimensions. The tuning variable is k, or the number of nearest neighbors.
3. Random Forests: Well-suited to low-dimensional tabular data and capable of capturing complex, nonlinear interactions automatically without needing the earlier transformations; especially attractive given apparent splines in some variables (`sig1`). Factor variables are not one-hot encoded because one-hot encoding increases the number of predictors, resulting in over-representation of factor variables by random forests. The tuning parameters are below:

```
rf_grid <- expand.grid(
  mtry = c(2, 4, 6, default_mtry),
  splitrule = "gini"
```

```
  splitrule = "gini",
  min.node.size = c(1, 5, 10)
)
```

4. I also use boosting with XGBoost. The tuning parameters are below:

```
params <- list(
  objective = "binary:logistic",
  eval_metric = "error",
  max_depth = 6,
  eta = 0.1,
  subsample = 0.8,
  colsample_bytree = 0.8
)
```

Interpretation/Evaluation

Boosting with XGBoost performs the best.

LASSO keeps many variables. The coefficients represent changes in the log odds ratio but analysts are discouraged from making inferences about them; it is a known feature of penalized regressions that correlated covariates may be included in the final model without having an underlying causal effect on the dependent variable.

The random forests model uses a generic 500 trees. `sig2` is the most important feature. When `sig2` is permuted, the model's accuracy drops the most compared to any other predictor, so the forest relies heavily on it for splitting and prediction.

These results generally concur with the earlier visualizations. `sig2` appeared to uniquely partition dependent and independent variables.

Below is a table summarizing results, followed by visualizations of each modeling strategy's variable selection methods and performance.

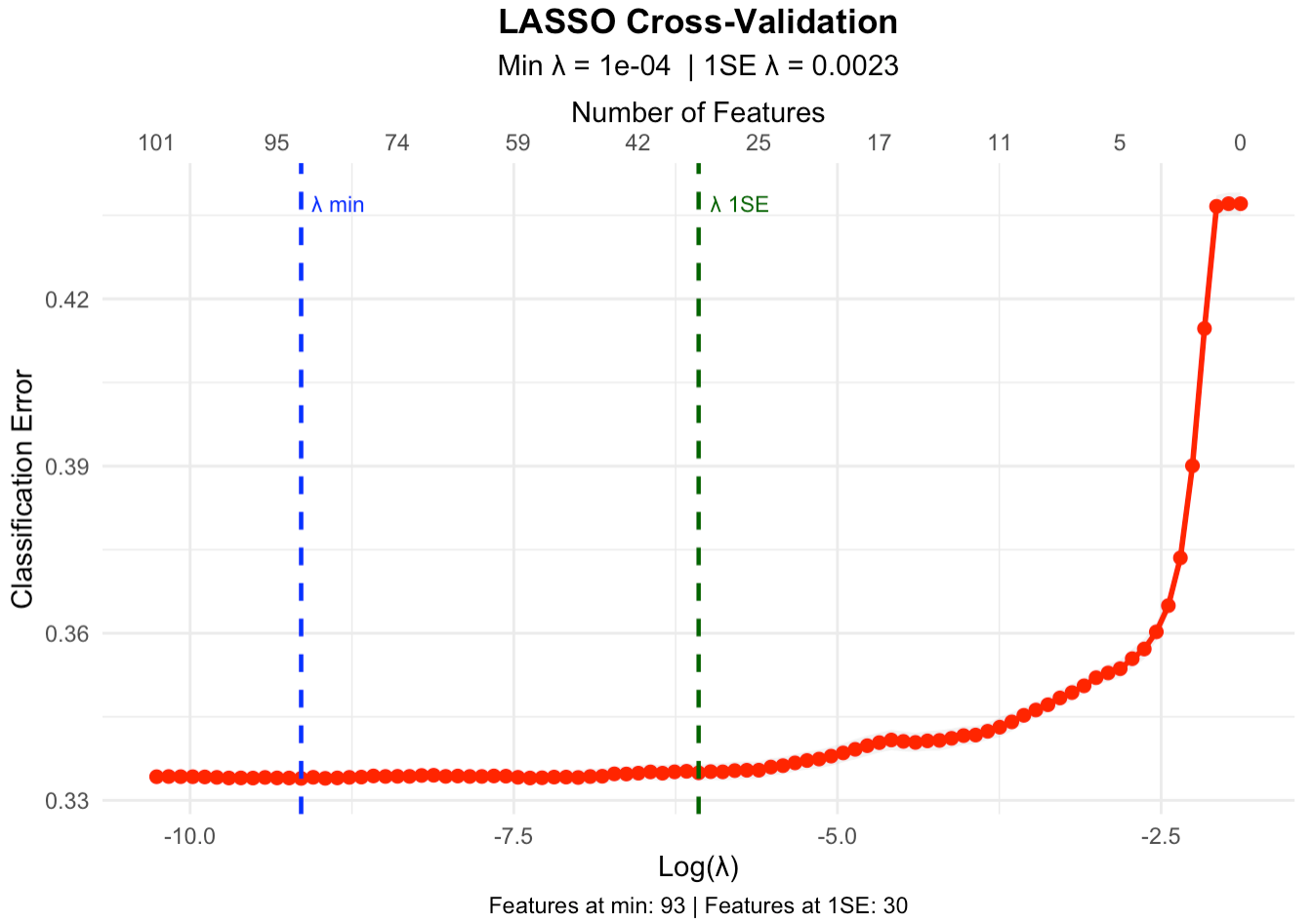
Model Performance Comparison

Model	CV Accuracy	Optimal Parameters
LASSO	0.66605	λ = 1e-04
KNN	0.66600	k = 149
Random Forest	0.66614	mtry = 3
XGBoost	**0.66737**	**iterations = 46**

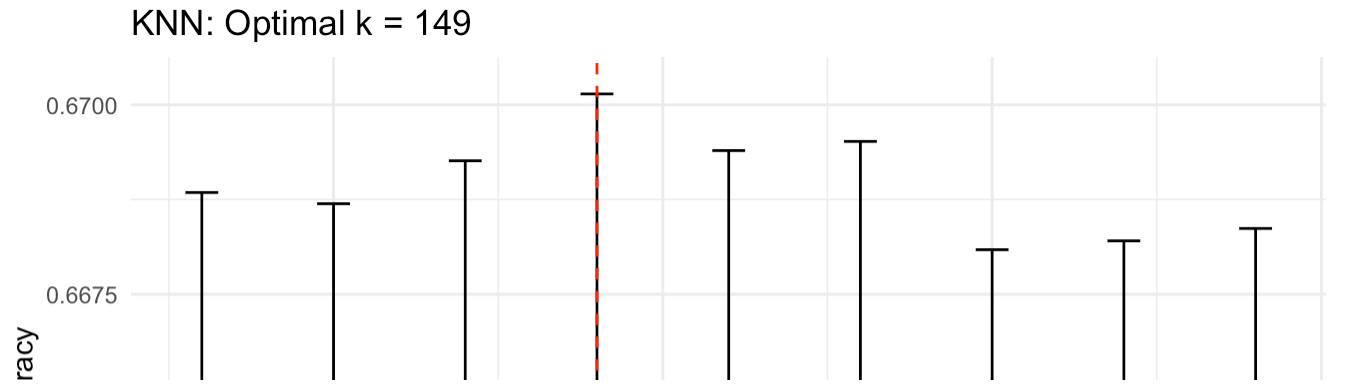
LASSO

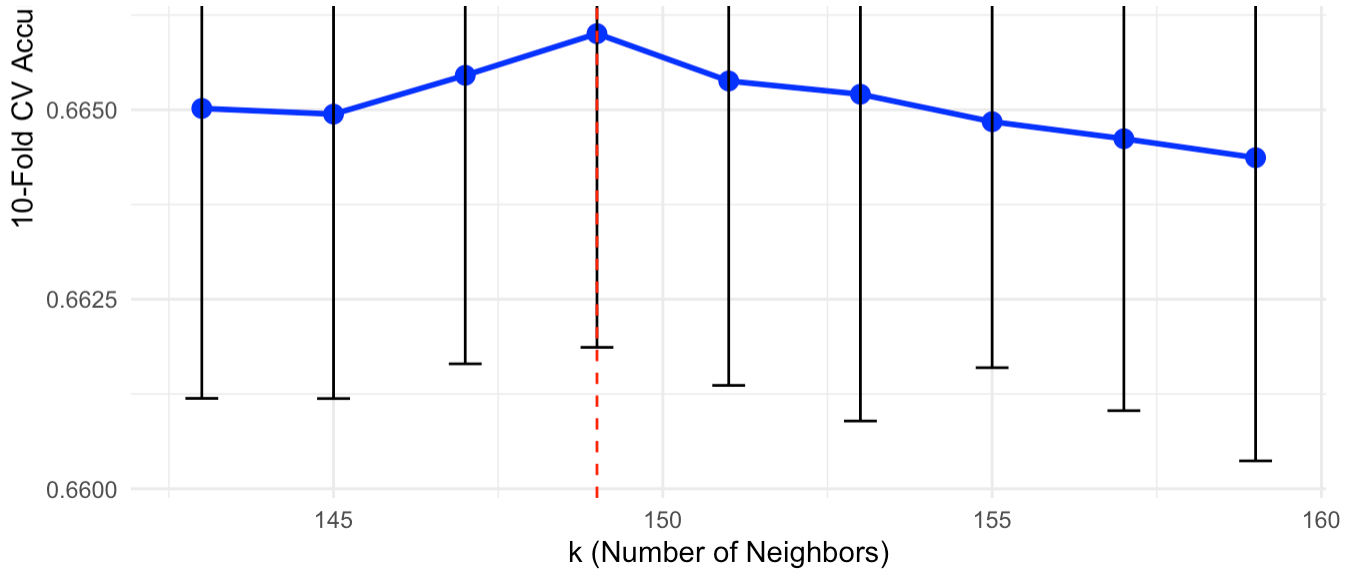
Head of LASSO Selected Features (91 variables + intercept)

Feature	Coefficient
sig2	**2.8803**
is_homepage1:sig7	-1.7308
(Intercept)	-1.5162
sig1:sig7	-1.4295
sig7:sig8	-1.3312
is_homepage1:sig1	1.3215

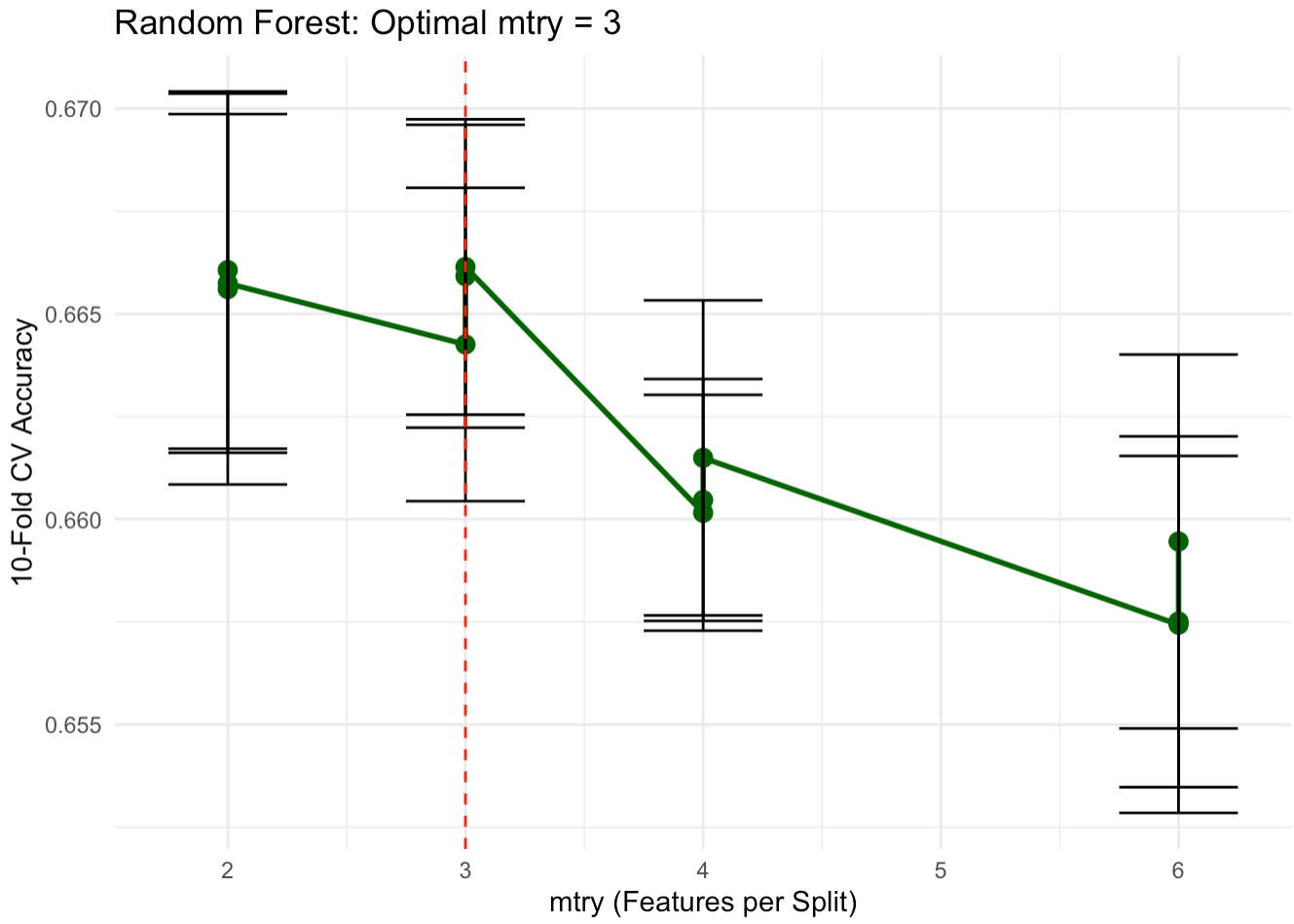


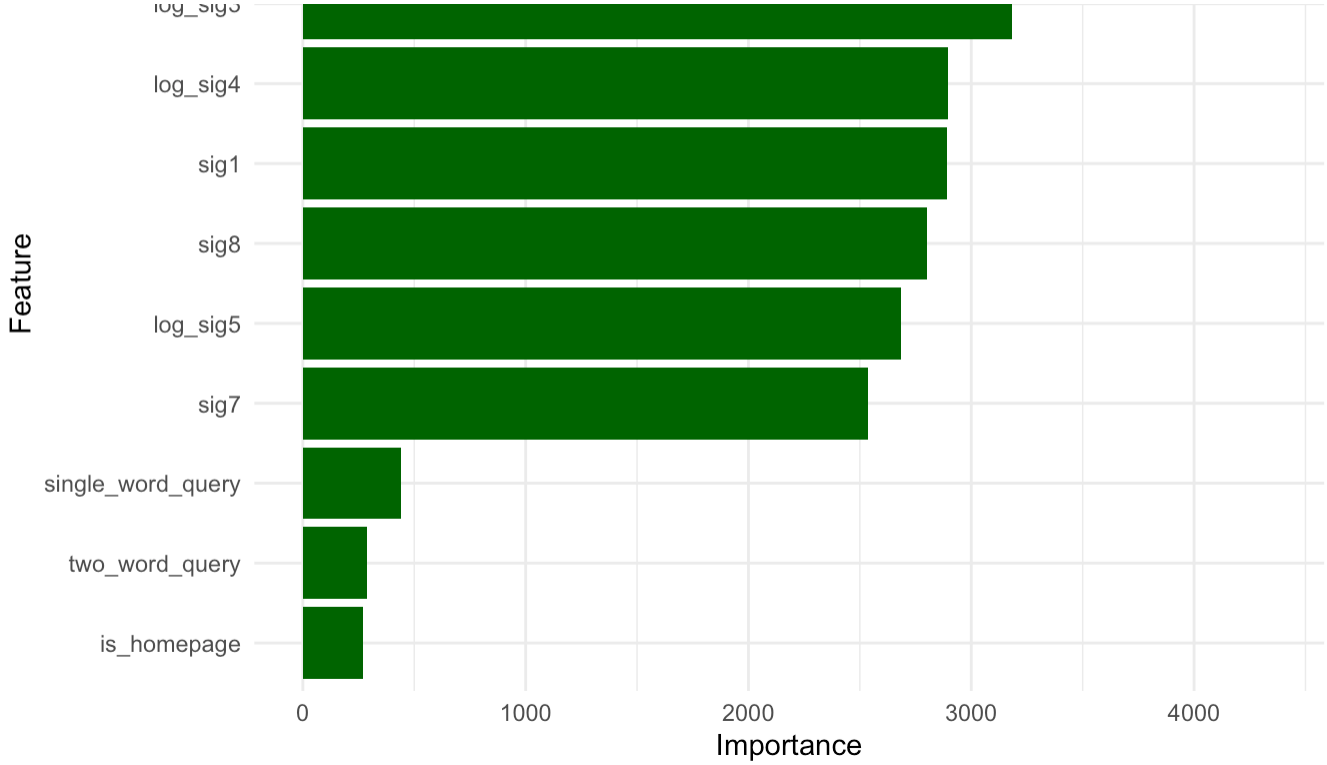
KNN



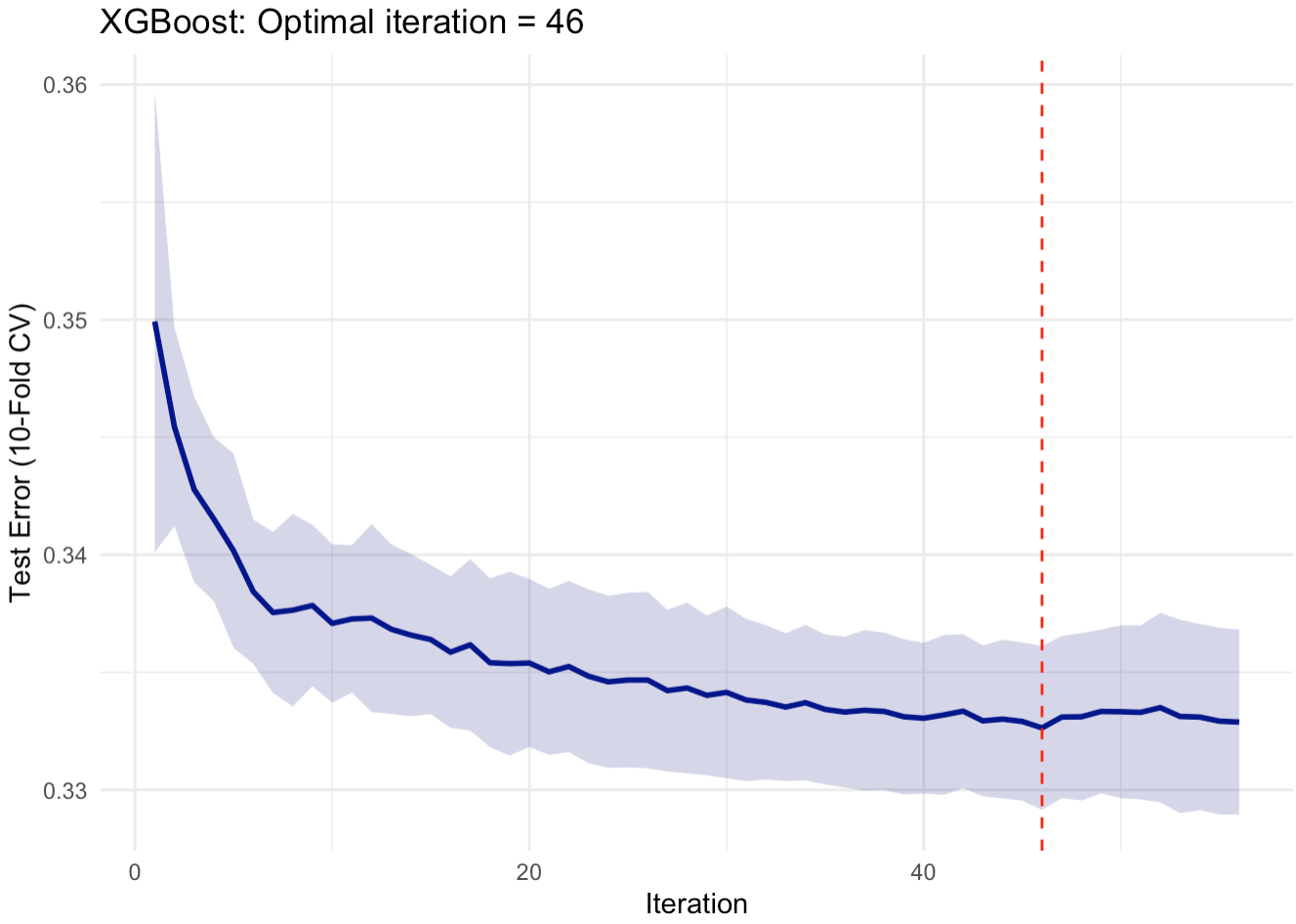


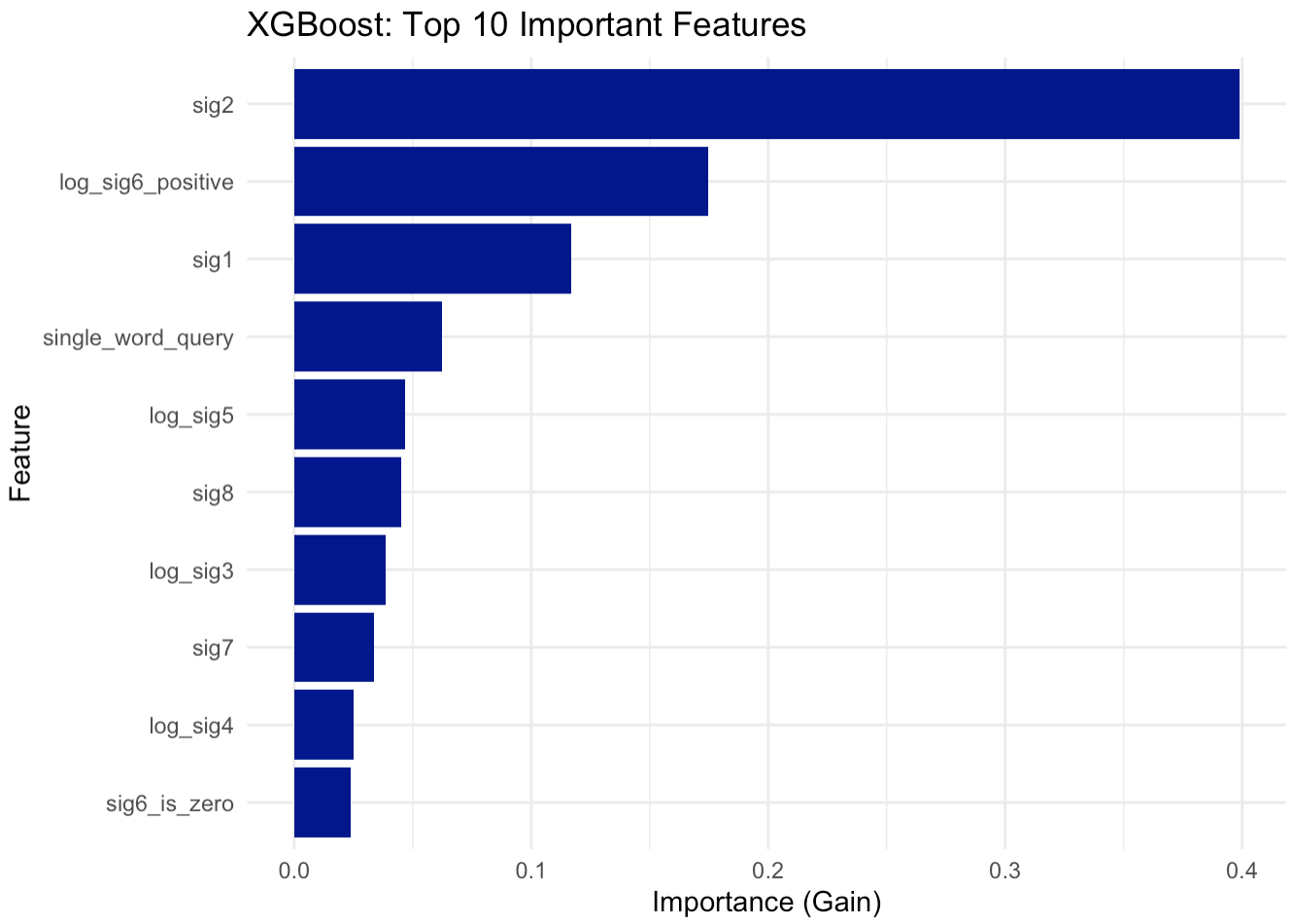
Random Forests




log_sig4
sig1
sig8
log_sig5
sig7
single_word_query
two_word_query
is_homepage
Feature
0
1000
2000
3000
4000
Importance

XGBoost


XGBoost: Optimal iteration = 46
0.36
0.35
0.34
0.33
Test Error (10-Fold CV)
0
20
40
Iteration




```
XGBoost CV Accuracy: 0.6674
```

Testing

A script for creating the test submissions is included in the GitHub Repository: https://github.com/24-bee-supply/stats-202-project.