Model Comparison Via Expected Predictive Performance

Ben Goodrich

April 04, 2022

What Is Supervised Learning?

- Frequentism – Probability = Basic Supervised Learning
- More influenced by tech companies. Within academics, more influenced by computer science, engineering, and operations research than statistics.
- Supervised learning does not require or use probability for estimation, so no standard errors, p-values, confidence intervals, or hypothesis tests and no need for datasets to be random samples from any well-defined population
- Supervised learning retains the Frequentist approach for point estimating $\boldsymbol{\theta}$ by solving an optimization problem and ignores all uncertainty about $\boldsymbol{\theta}$
- But if you are not going to do a hypothesis test, there is no need to maximize a (log)-likelihood function. Supervised learning instead maximizes a *penalized* (log)-likelihood function over $\boldsymbol{\theta}$, given a value of the tuning parameter(s), λ.

Example of Supervised Learning: Rain Prediction

- Scientists predict weather using physics and chemistry with barometric pressure, temperature, humidity and other data
- Google can predict weather better than scientists up to 6 hours into the future by dividing the USA up into one square kilometer blocks & using a neural net to learn that when it rains in one block it tends to rain in adjacent blocks
- Advantages of Google's approach:
  - "Google says its system can produce results in less than 10 minutes", as opposed to 1 – 3 hours while "computational demands limit the spatial resolution to about 5 kilometers" for science-based models
  - "you'd be able to look up a minute-by-minute rainfall forecast for your specific [bike] route. Today's conventional weather forecast, by contrast, might just tell you that there's a 30-percent chance of precipitation in your town over the next couple of hours."
- But is Google's approach scientific or useful for (social) science?

Penalization vs. Priors

- These penalty functions are often some prior log-kernel conditional on a tuning parameter, λ, and possibly auxiliary parameters like σ

Shrinkage prior	Conditional prior density $p(\beta_j \mid \lambda, \ldots)$	Reference
Ridge	$p(\beta_j \mid \sigma^2, \lambda) = \sqrt{\frac{\lambda}{2\pi\sigma^2}} \exp\left\{-\frac{\lambda\beta_j^2}{2\sigma^2}\right\}$	Hsiang (1975)
Local Student's t	$p(\beta_j \mid \sigma^2, \lambda) = \frac{\sigma^2}{\pi\lambda}\left(1 + (\frac{\sigma^2}{\lambda\beta_j})^2\right)$	Griffin and Brown (2005); Meuwissen et al. (2001)
Lasso	$p(\beta_j \mid \sigma^2, \lambda) = \frac{\lambda}{2\sqrt{\sigma^2}} \exp\left\{\frac{-\lambda\lvert\beta_j\rvert}{\sqrt{\sigma^2}}\right\}$	Park and Casella (2008)
Elastic net	$p(\beta_j \mid \sigma^2, \lambda_1, \lambda_2) = C \exp\left\{-\frac{1}{2\sigma^2}(\lambda_1\lvert\beta_j\rvert + \lambda_2\beta_j^2)\right\}$	Li and Lin (2010)
Group lasso	$p(\beta_j \mid \sigma, \lambda) = C \exp\left\{-\frac{\lambda}{\sigma}\sum_{g=1}^{G} \lVert\boldsymbol{\beta}_g\rVert\right\}$	Kyung et al. (2010)
Hyperlasso	$p(\beta_j \mid \lambda) = \lambda(2\pi)^{\frac{1}{2}}\left[1 - \frac{(\lambda\lvert\beta_j\rvert)\{1-\Phi(\lambda\lvert\beta_j\rvert)\}}{\phi(\lambda\lvert\beta_j\rvert)}\right]$	Griffin and Brown (2011)
Horseshoe	Not analytically tractable	Carvalho et al. (2010)
Discrete normal mixture	$p(\beta_j \mid \gamma_j, \phi_j^2) = (1-\gamma_j)\left(\frac{1}{\sqrt{2\pi\phi_j^2}} \exp\left\{-\frac{\beta_j^2}{2\phi_j^2}\right\}\right) + \gamma_j\left(\frac{1}{\pi(1+\beta_j^2)}\right)$	George and McCulloch (1993); Mitchell and Beauchamp (1988)

Note. C denotes a normalization constant. $\Phi()$ and $\phi()$ in the hyperlasso are the cumulative density function and the probability density function of the standard normal distribution.

- Penalty functions used in supervised learning make for poor priors because they are intended to shift the mode rather than reflect beliefs

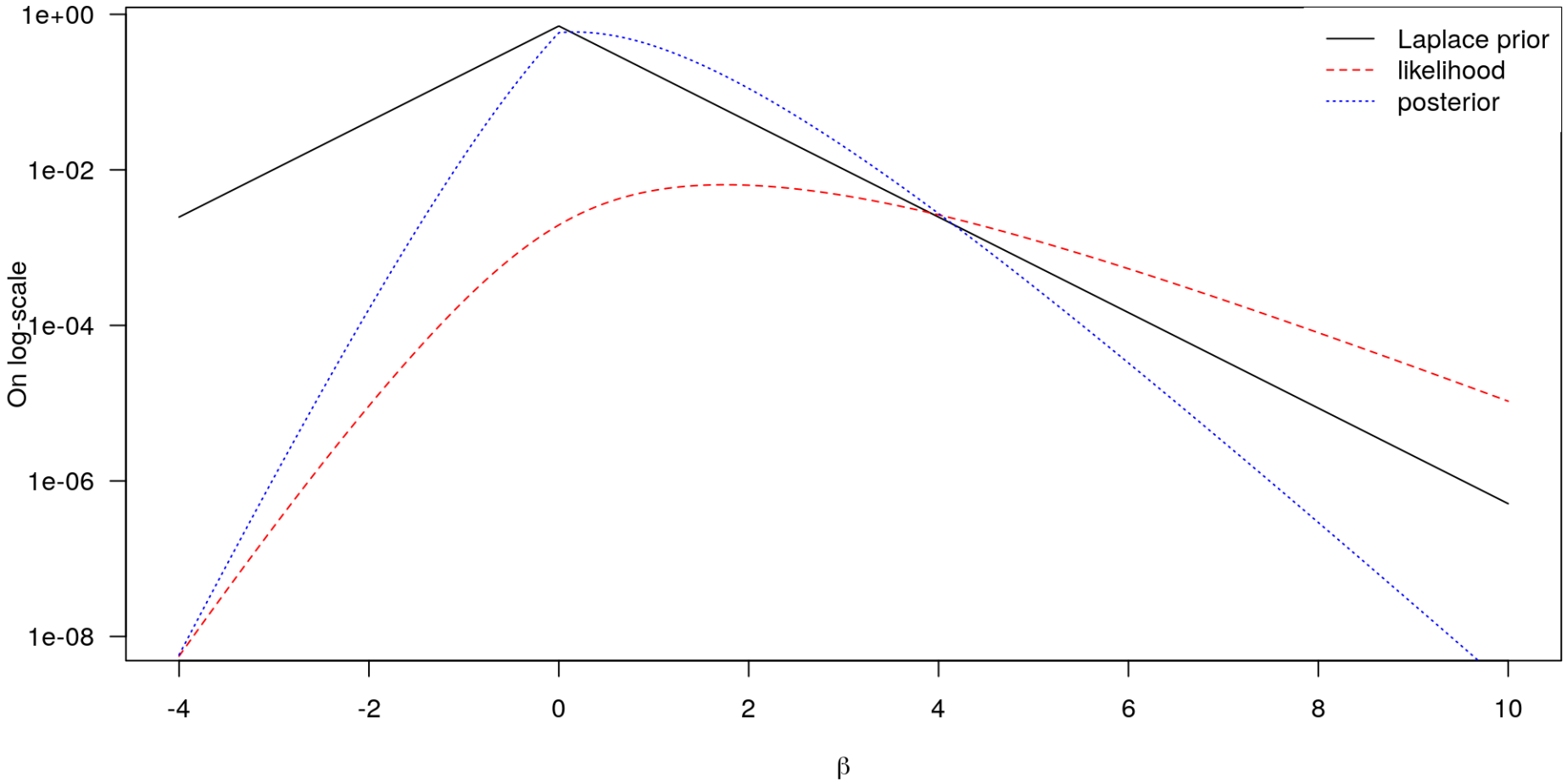
Comparison to Lasso

- Lasso chooses coefficients to minimize the penalized objective function

$$\sum_{n=1}^{N}\left(y_n - \alpha - \sum_{k=1}^{K}\beta_k x_{nk}\right)^2 + \lambda\sum_{k=1}^{K}|\beta_k|$$

- That is proportional to the negative posterior kernel with $\sigma = 1$, a Gaussian likelihood, and independent Laplace priors on the coefficients with location zero and FIXED inverse scale λ
- The Laplace distribution is the maximum entropy distribution for a real random variable under the constraint $\mathbb{E}\,|\beta| = \frac{1}{\lambda}$
- Posterior MODE corresponds to the Lasso solution for a given λ
- Only recently did some of the Tibshiranis (and co-authors) derive a sampling distribution for the Lasso point estimator (for a given sparsity) across datasets

Logit Model, No Intercept, Single Coefficient



Training and Testing Data

- How do supervised learners choose λ?
- Split the dataset randomly into about 80% training and 20% testing (but ignore how this randomization affects all your conclusions)
- Solve a penalized optimization problem using the training data and a guess of λ to get a $\widehat{\boldsymbol{\theta}}$, often with K-fold or bootstrapping within training
- Do not look at $\widehat{\boldsymbol{\theta}}$, but predict the outcome in the testing data with it
- Average the prediction metric over the observations in the testing set
- Repeat on the same training and testing datasets but with other models
- Primary focus is on the decision of which model (among many) to use
- But there is no probability in $\widehat{\boldsymbol{\theta}}$ so there are no EXPECTATIONS
- Choose the model that actually works best in that testing set in hopes that it will work best in future datasets

What Is Bayesian Machine Learning?

- No consensus definition, but there are several non-exclusive possibilities:

1. Specifies $f(\boldsymbol{\theta} \mid \mathbf{y})$ but does not care about it except insofar as it affects the posterior predictive density of a future $\tilde{y}$, whose PDF is

$$f(\tilde{y} \mid \mathbf{y}) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tilde{y} \mid \boldsymbol{\theta}) f(\boldsymbol{\theta} \mid \mathbf{y}) \, d\theta_1 d\theta_2 \ldots d\theta_K$$

2. Specifies $f(\boldsymbol{\theta} \mid \mathbf{y})$ but only finds the mode, rather than doing MCMC
3. Finds a tractable (multivariate normal) distribution that is closest to $f(\boldsymbol{\theta} \mid \mathbf{y})$
4. If $y = \mu(\boldsymbol{\theta}, \mathbf{x}) + \epsilon$, put a prior on the unknown but continuous function $\mu(\boldsymbol{\theta}, \mathbf{x})$, rather than assuming it is linear, polynomial, etc.
5. Bayesian but intended to scale well with data rather than model complexity

Overfitting

- Maximum likelihood maximizes overfitting: A MLE is the value of $\boldsymbol{\theta}$ such that the most likely dataset of size N to be drawn from the population is the dataset at hand. Every other potential dataset will exhibit worse fit.
- A fundamental principle of supervised learning is that you evaluate models on testing data that were not used when solving the optimization problem for $\widehat{\boldsymbol{\theta}}$
- Overfitting is primarily due to the choice to use optimization to find a single $\widehat{\boldsymbol{\theta}}$
- If you choose to use Bayesian methods and average the fit measure over your uncertainty in the posterior distribution, you shouldn't overfit the observed $\mathbf{y}$
- If overfitting is still a problem when doing MCMC, it is often due to improper or excessively weak priors (or likelihoods) that no one believes anyway
- If Bayesians are going to split into training and testing data, there should be $N-1$ observations in the training data and 1 in the testing data

Diamonds Data

```
data("diamonds", package = "ggplot2")
diamonds <- diamonds[diamonds$z > 0, ]
glimpse(diamonds)
```

```
## Rows: 53,920
## Columns: 10
## $ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.…
## $ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Ver…
## $ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I,…
## $ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, …
## $ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64…
## $ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58…
## $ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 34…
## $ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4.…
## $ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4.…
## $ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.…
```

Overfitting Diamonds?

```
post <- # note interaction terms
  stan_lm(log(price) ~ carat * (log(x) + log(y) + log(z)) +
          cut + color + clarity, data = diamonds,
          adapt_delta = 0.9, seed = 12345,
          prior = R2(location = 0.8, what = "mode"))


PPD <- posterior_predict(post, draws = 1000); dim(PPD)


## [1]  1000 53920


lower <- apply(PPD, MARGIN = 2, quantile, probs = 0.25)
upper <- apply(PPD, MARGIN = 2, quantile, probs = 0.75)
with(diamonds, c(too_low = mean(log(price) < lower), too_high = mean(log(price) > upper),
     just_right = mean((log(price) > lower) & (log(price) < upper)))) # not too bad


##    too_low   too_high just_right
##  0.2370734  0.2327522  0.5301743
```

Utility Function for Predictions of Future Data

- For Bayesians, the log predictive PDF is the most appropriate utility function
- Choose the model that maximizes the expectation of this over FUTURE data

$$\text{ELPD} = \mathbb{E}_Y \ln f\left(y_{N+1}, y_{N+2}, \ldots, y_{2N} \mid y_1, y_2, \ldots, y_N\right) =$$

$$\ln \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f\left(y_{N+1}, y_{N+2}, \ldots, y_{2N} \mid \mathbf{y}\right) dy_{N+1} dy_{N+2} \ldots dy_{2N} \approx$$

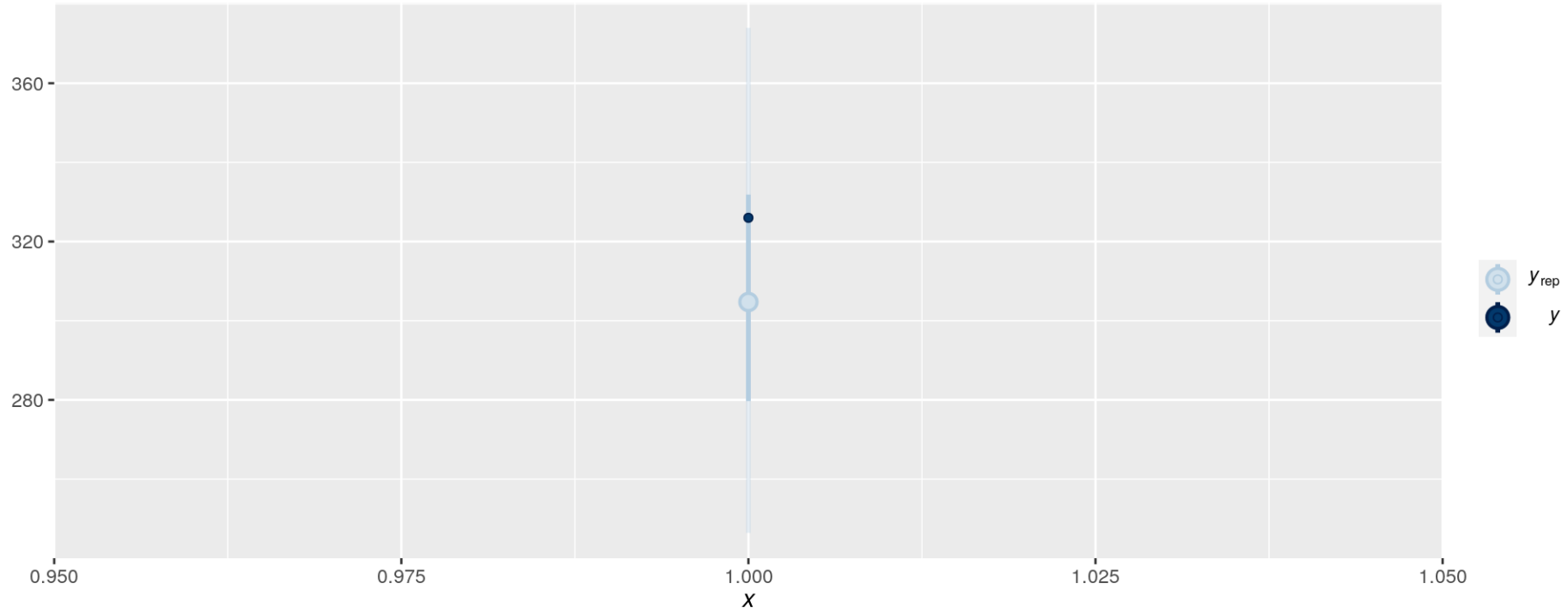
$$\sum_{n=1}^{N} \ln f\left(y_n \mid \mathbf{y}_{-n}\right) = \sum_{n=1}^{N} \ln \int_{\Theta} f\left(y_n \mid \boldsymbol{\theta}\right) f\left(\boldsymbol{\theta} \mid \mathbf{y}_{-n}\right) d\theta_1 d\theta_2 \ldots d\theta_K$$

- $f\left(y_n \mid \boldsymbol{\theta}\right)$ is just the n-th likelihood contribution, but can we somehow obtain $f\left(\boldsymbol{\theta} \mid \mathbf{y}_{-n}\right)$ from $f\left(\boldsymbol{\theta} \mid \mathbf{y}\right)$?
- Yes, assuming y_n does not have an outsized influence on the posterior

Literal Leave-One-Out Cross-Validation

```
post_1 <- update(post, data = diamonds[-1, ]) # tedious for many observations

bayesplot::ppc_intervals(y = diamonds$price[1], # in dollars rather than log-dollars
                         yrep = exp(posterior_predict(post_1, newdata = diamonds[1, ])))
```



PSISLOOCV with Logit Models

```
load("nes.rda") # creates data.frame called nes
nes$income <- as.factor(nes$income) # do not treat income categories as continuous
nes$age <- nes$age / 10 # put respondent age into decades rather than years
nes2000 <- nes[!is.na(nes$rvote) & nes$year == 2000, ] # George W. Bush vs. Al Gore


post_flat <- stan_glm(rvote ~ age + I(age ^ 2) + income + white, data = nes2000,
                      family = binomial(link = "logit"), QR = TRUE)


(loo_post_flat <- loo(post_flat)) # p_loo is an estimate of the effective parameters


##
## Computed from 4000 by 861 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -574.8  7.4
## p_loo         8.1  0.3
## looic      1149.5 14.8
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

Smooth Models

- What if instead of saying that the log-odds of voting Republican are quadratic in the person's age, we allow the log-odds to be any smooth function of age?
- This smooth function is estimated as the summation of many basis functions that have coefficients with normal priors whose standard deviation is *unknown* and estimated along with everything else by Stan

```
post_smooth <- stan_gamm4(rvote ~ s(age) + income + white, data = nes2000, seed = 33,
                          family = binomial(), QR = TRUE, adapt_delta = 0.99)
```

```
(loo_post_smooth <- loo(post_smooth)) # p_loo is actually smaller now
```

```
##
## Computed from 4000 by 861 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -573.2  7.4
## p_loo         7.8  0.2
## looic      1146.3 14.7
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
```

Comparison

```
loo_compare(loo_post_flat, loo_post_smooth)
```

```
##             elpd_diff se_diff
## post_smooth  0.0       0.0
## post_flat   -1.6       0.4
```

```
loo_model_weights(stanreg_list(post_flat, post_smooth))
```

```
## Computing approximate LOO-CV (models do not already have 'loo' components).
```

```
## Method: stacking
## ------
##             weight
## post_flat   0.000
## post_smooth 1.000
```

- The second model is expected to predict future data better than the first

PSISLOOR^2 for Previous Examples

```
summary(loo_R2(post)) # uses leave-one-out concepts
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9832  0.9844  0.9846  0.9846  0.9847  0.9852
```

```
summary(as.data.frame(post)$R2) # does not use leave-one-out concepts
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9843  0.9846  0.9846  0.9846  0.9847  0.9849
```

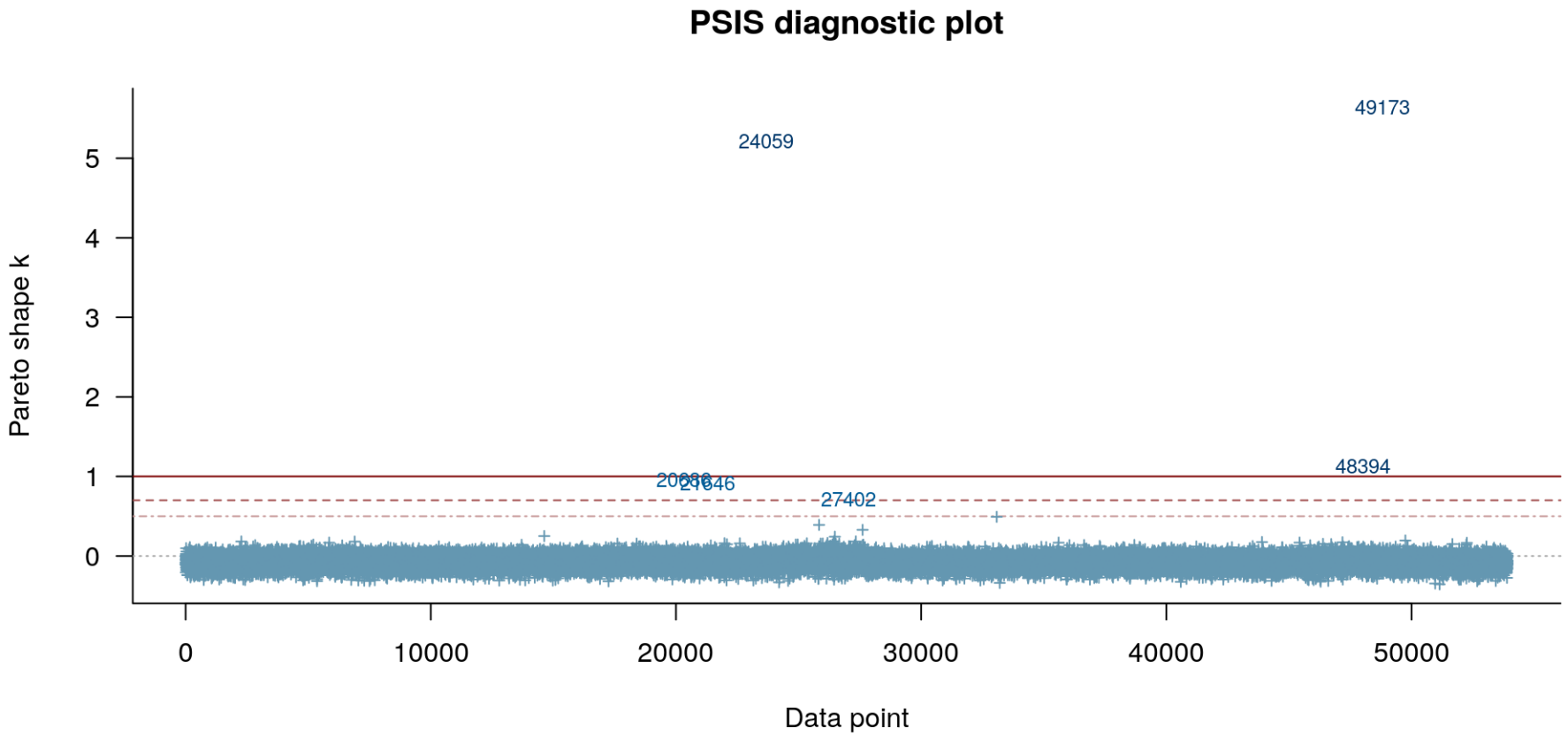
```
# works with stan_glm and stan_gamm4 too
rbind(loo_R2    = summary(loo_R2(post_smooth)),
      bayes_R2  = summary(bayes_R2(post_smooth)))
```

```
##                Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## loo_R2   -0.01768030 0.03568136 0.04624250 0.04588812 0.05629680 0.09853044
## bayes_R2  0.02584816 0.05733333 0.06754566 0.06822696 0.07857924 0.12752773
```

This Is Not Good (& likely caused slight overfitting)

```
plot(loo(post), label_points = TRUE) # from diamonds example
```

```
## Warning: Found 6 observation(s) with a pareto_k > 0.7. We recommend calling 'loo' again wit
```



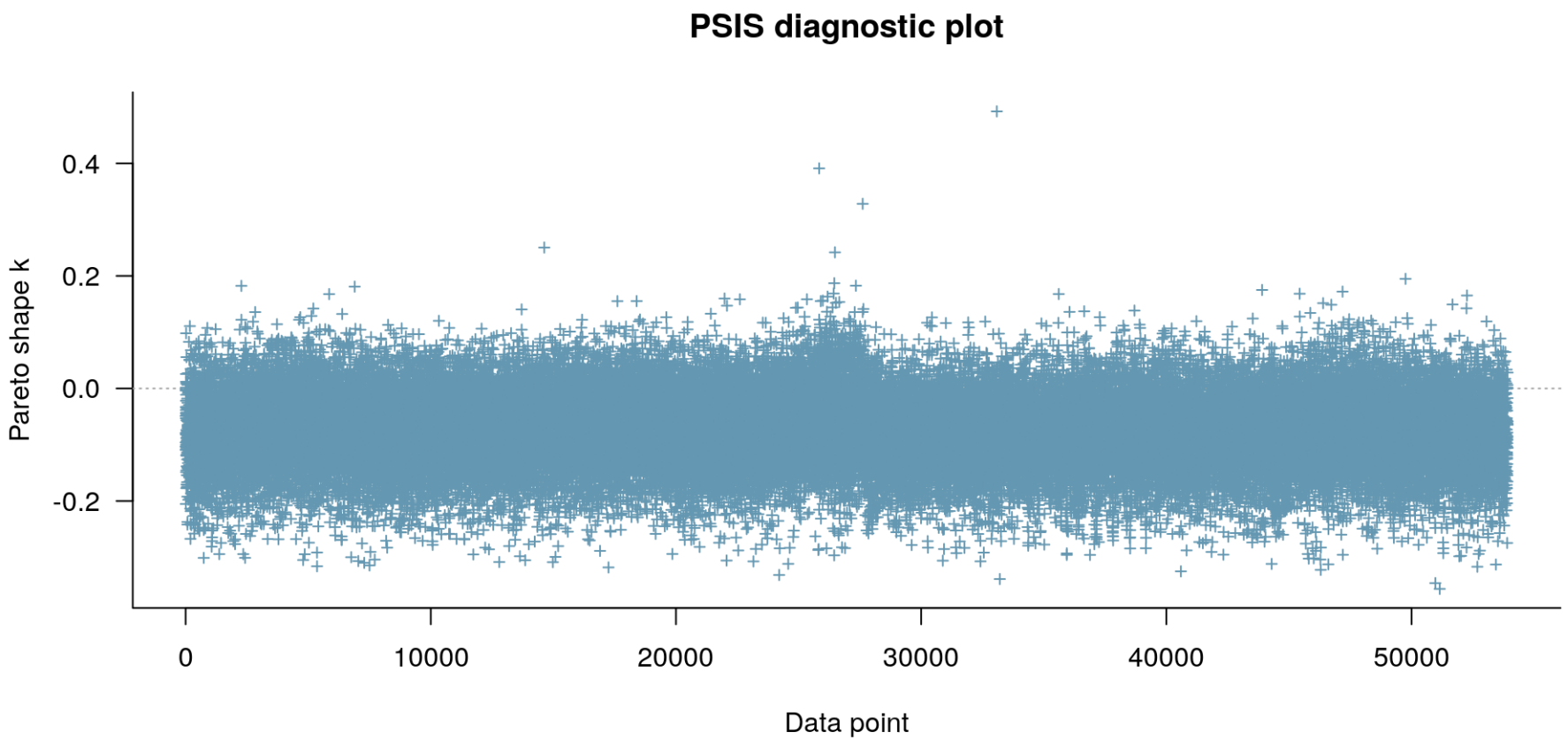
What were those data points?

```
glimpse(diamonds[c(20686, 21646, 24059, 27402, 48394, 49173), ]) # these are messed up data
```

```
## Rows: 6
## Columns: 10
## $ carat   <dbl> 1.53, 1.41, 2.00, 5.01, 0.51, 0.51
## $ cut     <ord> Ideal, Ideal, Premium, Fair, Very Good, Ideal
## $ color   <ord> I, H, H, J, E, E
## $ clarity <ord> SI1, VS1, SI2, I1, VS1, VS1
## $ depth   <dbl> 61.9, 60.7, 58.9, 65.5, 61.8, 61.8
## $ table   <dbl> 54.0, 56.0, 57.0, 59.0, 54.7, 55.0
## $ price   <int> 8971, 9752, 12210, 18018, 1970, 2075
## $ x       <dbl> 7.43, 7.31, 8.09, 10.74, 5.12, 5.15
## $ y       <dbl> 7.50, 7.22, 58.90, 10.54, 5.15, 31.80
## $ z       <dbl> 1.53, 1.41, 8.06, 6.98, 31.80, 5.12
```

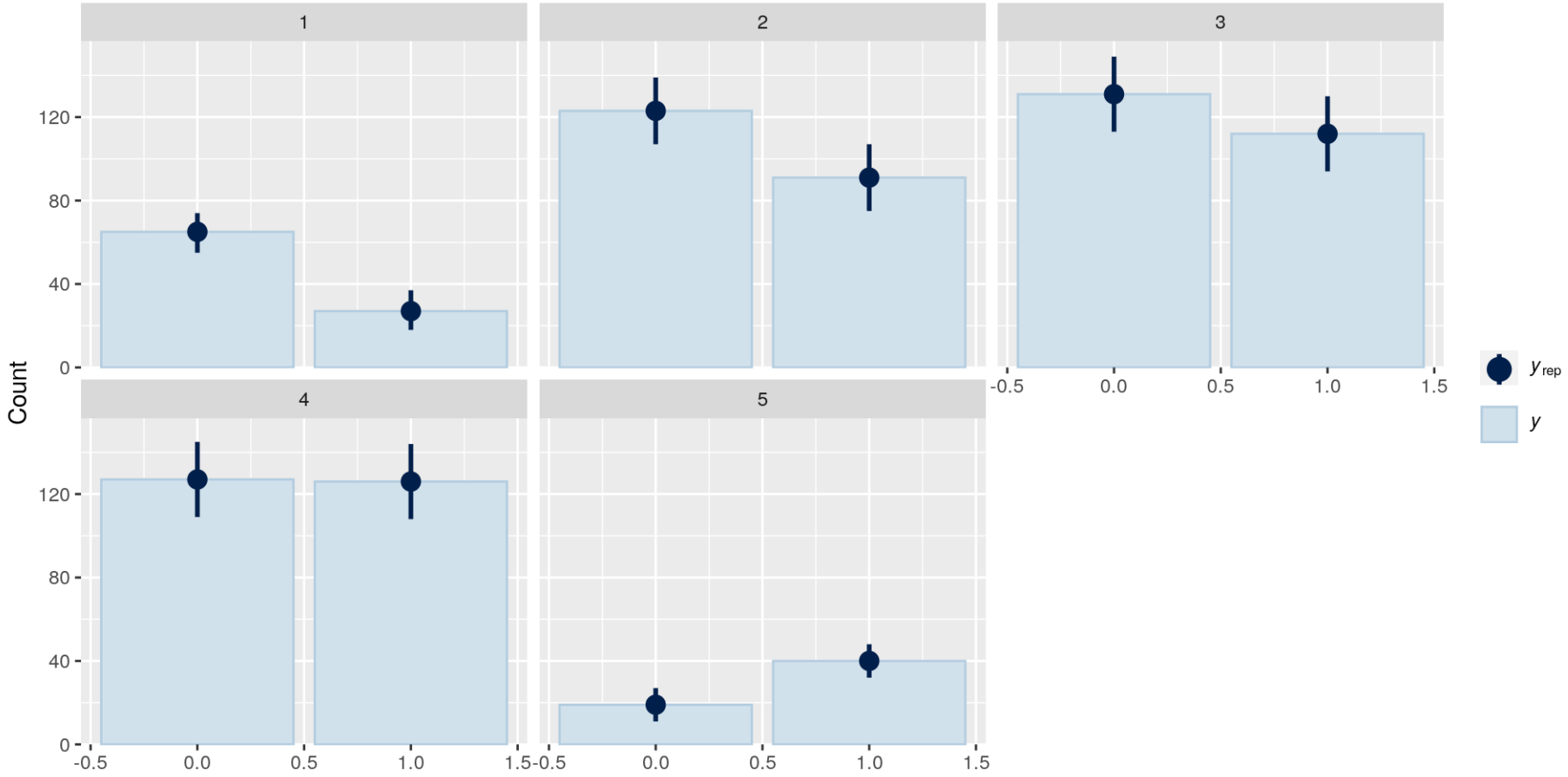
Refitting the Diamonds Model 6 Times

```
plot(loo(post, k_threshold = 0.7), label_points = TRUE)
```



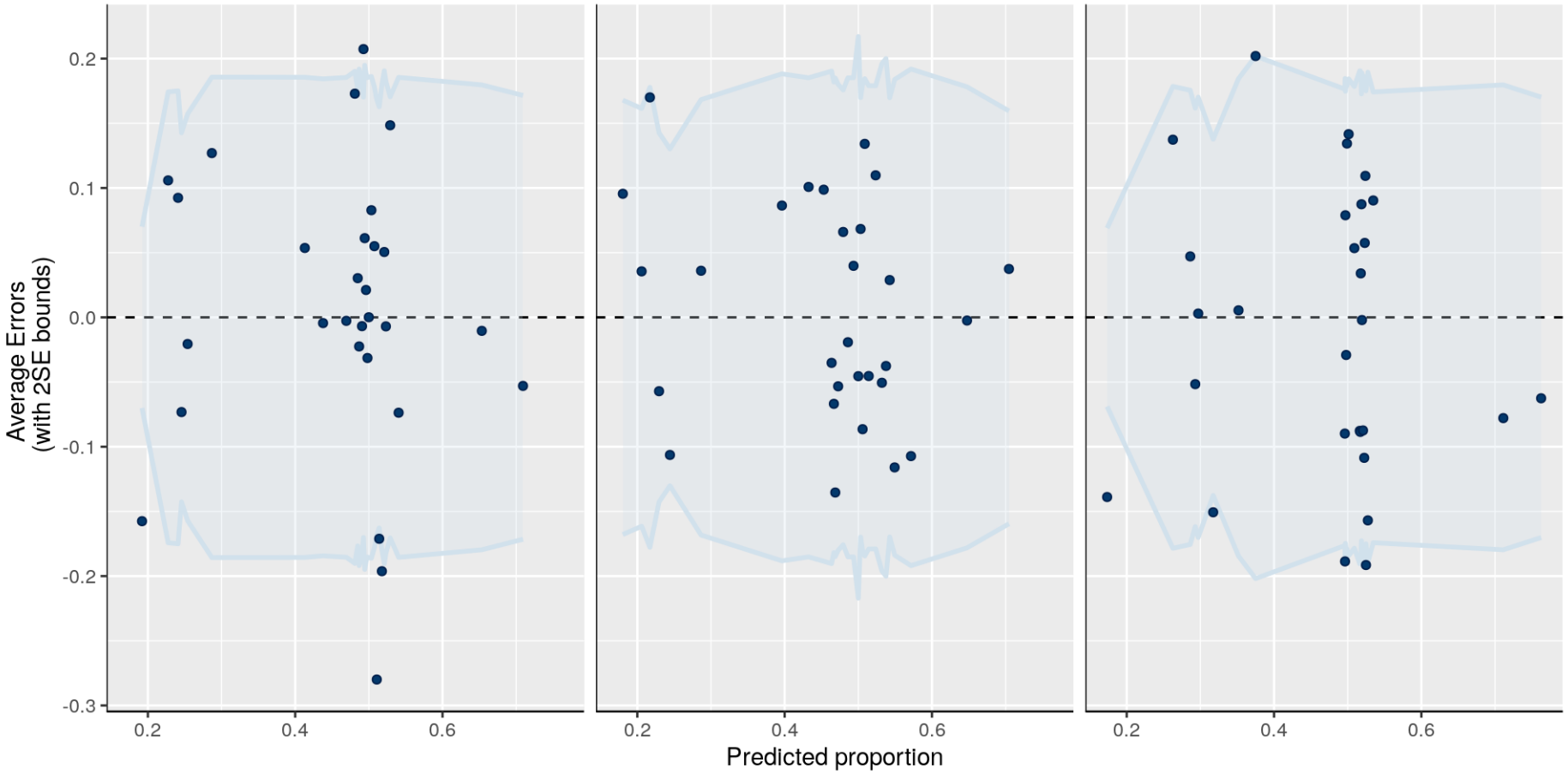
Posterior Predictive Checking

```
pp_check(post_smooth, plotfun = "bars_grouped", group = nes2000$income)
```



More Posterior Predictive Checking

```
pp_check(post_smooth, plotfun = "error_binned")
```



Classification

- When the outcome is binary, supervised learners will often obtain classifications of the outcome in the testing data

```
MLE <- glm(rvote ~ age + I(age ^ 2) + income + white, data = nes2000, family = binomial)
nes1996 <- nes[!is.na(nes$rvote) & nes$year == 1996, ] # Bill Clinton vs. Bob Dole
classifications <- predict(MLE, newdata = nes1996, type = "response") > 0.5
round(prop.table(table(truth = nes1996$rvote, classifications)), digits = 2)
```

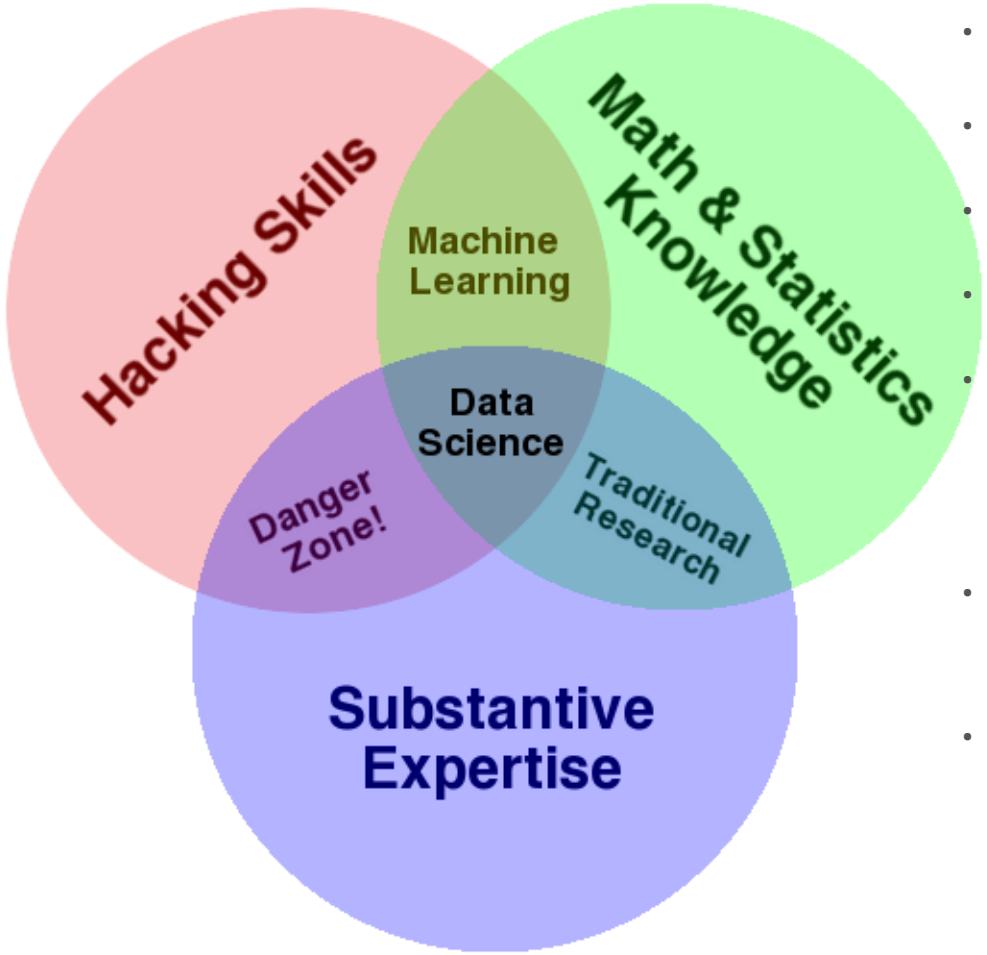
```
##      classifications
## truth FALSE TRUE
##     0  0.33 0.28
##     1  0.13 0.26
```

- This is known as a confusion matrix and you can calculate many functions of it

Scoring Rules

- The proportion of correct classifications is a classic example of an “improper” scoring rule due to the discontinuity at 0.5
- Bayesians (should, and typically do) use “strictly proper” scoring rules https://en.wikipedia.org/wiki/Scoring_rule , such as the ELPD, which are defined by the property that they are maximized / minimized by the truth
- If the model is correct, the posterior predictive distribution is expected to yield the best-calibrated predictions with respect to any strictly proper scoring rule
- Supervised learning presumes there is no truth, in which case, the justification for using a strictly proper scoring rule is less compelling. However, the choice of which improper scoring rule to use can determine which modeling procedure is best in the testing data

Data Science Venn Diagram



- What do you get if you equate . . .
- Probability
- Priors, DAGs, and Model Building
- Markov Chain Monte Carlo
- Since at least 1990, Bayesian estimation has been what data science purported to be but isn't
- Data science came along later but largely ignored Bayesian approaches
- In reality, data science programs are asymmetric with more emphasis on hacking skills and less on the others