Title

April 3, 2024

Leo Kraushaar & Ali Hay
STAT 413

1 Introduction

2 Model Building

2.1 Fitting Process

2.2 Model Overview

3 Bootstrapping for Feature Significance

3.1 Resampling Bootstrap

Procedure

Using $B = 10000$, an algorithm was written to perform a resampling bootstrap on the dataset. On each iteration, a sample of size n (where n was the number of rows in the dataset) was taken with replacement. Using this sample, a new model was fit. The parameter estimates were recorded. Relevant code is shown below.

```r
resampBoot <- function(df, B) {
    # Get sample size
    n <- nrow(df)
    # Initialize empty dataframe
    params <- c()
    # Initialize progress bar
    bar <- txtProgressBar(min=0, max=B, style=1)
    # Perform B iterations
    for (b in 1:B) {
        # Select a sample of size n
        indices <- sample(1:n, replace = TRUE)
        samp <- df[indices, ]
        # Fit the model with the sample
        boot_model <- glm.nb(protests ~., data=samp, init.theta =
           10)
        boot_params <- coef(boot_model)
        params <- rbind(params, boot_params)
        setTxtProgressBar(bar, b)
    }
    close(bar)
    return(params)
}
```

Results

	mean	sd	2.5%	50%	97.5%	sig
intercept	3.66	0.53	2.70	3.64	4.77	TRUE
Spr	-0.06	0.08	-0.23	-0.06	0.09	FALSE
Sum	-0.55	0.08	-0.72	-0.55	-0.38	TRUE
Win	-0.24	0.10	-0.42	-0.23	-0.05	TRUE
BC	0.82	0.14	0.55	0.82	1.09	TRUE
M	-1.89	0.93	-3.85	-1.85	-0.18	TRUE
NB	-2.60	1.06	-4.84	-2.55	-0.66	TRUE
NL	-3.04	1.13	-5.43	-2.98	-0.99	TRUE
NT	-5.40	1.37	-8.20	-5.34	-2.94	TRUE
NS	-2.38	1.00	-4.53	-2.34	-0.54	TRUE
N	-4.98	1.28	-7.68	-4.92	-2.61	TRUE
O	5.54	2.37	1.18	5.40	10.58	TRUE
PEI	-4.10	1.23	-6.73	-4.04	-1.82	TRUE
Q	2.17	0.88	0.55	2.12	4.04	TRUE
S	-2.56	0.96	-4.57	-2.52	-0.77	TRUE
Y	-4.11	1.26	-6.77	-4.06	-1.81	TRUE
retail	-1.83	1.04	-4.05	-1.77	0.08	FALSE

3.2 Parametric Bootstrap

Procedure

Using $B = 10000$, an algorithm was written to perform a parametric bootstrap on the dataset.

Using the estimated dispersion parameter $\theta \approx 8.36$, each iteration sampled a random vector from a negative binomial distribution. The distribution had dispersion parameter θ and mean $\hat{y}$, where $\hat{y}$ was the predicted mean value for the corresponding input values.

Using these new estimates, a model was fit on each iteration and the parameter estimates were recorded. Relevant code is shown below.

```
conditionalNegBinom <- function(theta, mu) {
    nb_sample <- rnbinom(size=theta, mu=mu, n=1)
    return(nb_sample)
}

paramBoot <- function(B, X, yhat, theta, func) {

    # Initialize empty vector
    params <- c()
    # Iterate B times
    for (b in 1:B) {
```

```
        # Simulate NB given means
        sim_y <- sapply(yhat, function(y) func(theta, y))
        # Add to the dataframe
        sim_data <- cbind(X, protests=sim_y)
        # Fit the model to the simulated data
        sim_model <- glm.nb(protests ~., data=sim_data, init.theta =
            theta)
        # Access the coefficients and store
        parameters <- coef(sim_model)
        params <- rbind(params, parameters)
    }
    return(params)
}
```

Results

	mean	sd	2.5%	50%	97.5%	sig
intercept	3.68	0.51	2.68	3.68	4.69	TRUE
Spr	-0.07	0.08	-0.23	-0.07	0.10	FALSE
Sum	-0.55	0.09	-0.72	-0.55	-0.38	TRUE
Win	-0.23	0.09	-0.41	-0.23	-0.05	TRUE
BC	0.82	0.16	0.51	0.82	1.14	TRUE
M	-1.92	0.92	-3.75	-1.91	-0.11	TRUE
NB	-2.63	1.05	-4.72	-2.63	-0.57	TRUE
NL	-3.08	1.11	-5.31	-3.06	-0.89	TRUE
NT	-5.41	1.27	-7.92	-5.39	-2.91	TRUE
NS	-2.42	1.00	-4.39	-2.41	-0.46	TRUE
N	-5.01	1.27	-7.52	-5.01	-2.53	TRUE
O	5.61	2.45	0.80	5.59	10.51	TRUE
PEI	-4.13	1.22	-6.54	-4.12	-1.73	TRUE
Q	2.20	0.93	0.36	2.19	4.07	TRUE
S	-2.59	0.94	-4.46	-2.57	-0.75	TRUE
Y	-4.15	1.25	-6.61	-4.14	-1.71	TRUE
retail	-1.86	1.07	-3.98	-1.85	0.23	FALSE

3.3 Smooth Bootstrap

The smooth bootstrap is not an "ideal" method for the given dataset, as only one predictor **(retail)** was continuous and real-valued. However, results were consistent with other methods, as discussed later.

Again using $B = 10000$, an algorithm was written to perform a smooth bootstrap. The **retail** column was found to have a sample variance of 1, due to the fact that it was standardized prior to model building. A reasonable value for the noise term was chosen, that is, $\frac{1}{\sqrt{n}} \approx 0.05783$. On each iteration, some ε_i was added to each row i, where $\varepsilon \sim N(0, 0.05783)$. Using this "new" dataset, a model was fit and the paramter estimates were recorded. Relevant code is shown below.

Procedure

```
addNoise <- function(X) {

    cols <- colnames(X)
    new_X <- X
    for (col in cols) {
        Xi <- X[, col]
        if (class(data[, col]) != "factor") {
            n <- length(Xi)
            S_sq <- var(Xi)
            noise_var <- S_sq / n
            new_X[, col] <- Xi + rnorm(n=n, mean=0, sd=sqrt(noise_
               var))
        } else {
            new_X[, col] <- Xi
        }
    }
    return(new_X)
}

smoothBoot <- function(X, y, B, noisefunc) {

    # Get sample size
    n <- nrow(X)
    # Initialize empty vector
    params <- c()

    # Initialize progress bar
    pb <- txtProgressBar(min = 0, max = B, style = 3)

```

```
    # Perform B iterations
    for (b in 1:B) {
        # Update progress bar
        setTxtProgressBar(pb, b)

        # Get new dataset
        new_X <- noisefunc(X)
        new_data <- data.frame(protests=y, new_X)

        # Fit the model with the simulated data
        smoothboot_model <- glm.nb(protests ~., data=new_data, init.
           theta = 5)
        boot_params <- coef(smoothboot_model)
        params <- rbind(params, boot_params)
    }

    # Close progress bar
    close(pb)

    return(params)
}
```

Results

3.4 Error-Sampling Bootstrap

Another bootstrap method was implemented, in which the error terms from the fitted model were randomly sampled with replacement, and added to the fitted values. Notably, some resulting simulated counts were rounded up to zero in the case where a negative value was produced. This was required both logically; as protests counts cannot be negative, and mathematically; as the negative binomial glm cannot be fit with negative training outputs. As a result, the integrity of the simulated datasets was not assumed to be completely intact. That being said, the results were quite consistent with the previous methods.

Procedure

```
epsilonBoot <- function(X, model, B, errors) {

```

	mean	sd	2.5%	50%	97.5%	sig
intercept	3.00	0.21	2.58	3.01	3.42	TRUE
Spr	-0.04	0.01	-0.07	-0.04	-0.03	TRUE
Sum	-0.55	0.01	-0.56	-0.55	-0.54	TRUE
Win	-0.20	0.01	-0.23	-0.20	-0.18	TRUE
BC	0.67	0.05	0.57	0.67	0.76	TRUE
M	-0.67	0.39	-1.43	-0.68	0.10	FALSE
NB	-1.21	0.44	-2.08	-1.22	-0.33	TRUE
NL	-1.56	0.47	-2.49	-1.57	-0.62	TRUE
NT	-3.68	0.52	-4.70	-3.68	-2.63	TRUE
NS	-1.07	0.42	-1.89	-1.08	-0.23	TRUE
N	-3.30	0.53	-4.33	-3.30	-2.24	TRUE
O	2.24	1.05	0.14	2.25	4.29	TRUE
PEI	-2.47	0.51	-3.48	-2.48	-1.45	TRUE
Q	0.92	0.40	0.13	0.92	1.69	TRUE
S	-1.31	0.40	-2.09	-1.32	-0.52	TRUE
Y	-2.46	0.52	-3.49	-2.47	-1.41	TRUE
retail	-0.39	0.45	-1.29	-0.40	0.52	FALSE

```
    # Get sample size
    n <- nrow(X)
    # Initialize empty vector
    params <- c()
    # Perform B iterations
    for (b in 1:B) {
        # Get errors
        errs <- sample(errors, size=n, replace=TRUE)
        # Get fitted values
        yhat <- fitted(model)
        # Get simulated y
        ystar <- yhat + errs
        # round up negative values
        ystar <- pmax(rep(0, n), ystar)
        # Turn into DataFrame
        sim_data <- data.frame(protests=ystar, X)
        # Fit the model with the simulated data
        paramboot_model <- glm.nb(protests ~., data=sim_data, init.
           theta = 5)
        boot_params <- coef(paramboot_model)
        params <- rbind(params, boot_params)
    }
```

```
    return(params)
}
```

Results

	mean	sd	2.5%	50%	97.5%	sig
intercept	3.66	0.05	3.56	3.66	3.76	TRUE
seasonSpring	-0.07	0.01	-0.09	-0.07	-0.04	TRUE
seasonSummer	-0.56	0.02	-0.59	-0.56	-0.53	TRUE
seasonWinter	-0.24	0.01	-0.27	-0.24	-0.21	TRUE
BC	0.82	0.02	0.78	0.82	0.86	TRUE
M	-1.89	0.09	-2.07	-1.89	-1.71	TRUE
NB	-2.61	0.11	-2.83	-2.61	-2.40	TRUE
NL	-3.07	0.12	-3.30	-3.07	-2.83	TRUE
NT	-5.20	0.28	-5.80	-5.18	-4.70	TRUE
NS	-2.40	0.10	-2.60	-2.40	-2.20	TRUE
N	-4.96	0.25	-5.50	-4.95	-4.51	TRUE
O	5.54	0.24	5.07	5.54	6.02	TRUE
PEI	-4.16	0.18	-4.51	-4.15	-3.83	TRUE
Q	2.17	0.09	1.99	2.17	2.35	TRUE
S	-2.58	0.10	-2.78	-2.57	-2.38	TRUE
Y	-4.18	0.17	-4.54	-4.18	-3.85	TRUE
retail	-1.82	0.10	-2.03	-1.82	-1.62	TRUE

3.5 Method Comparison

4 Monte Carlo Estimation

5 Conclusion