Statistics for Business Analytics 2: R Laboratory Homework 1

Instructions

In this assignment we are going to look more closely at how we model count data. In particular, in Laboratory 1 we studied the glmnet package and saw an example of how to perform Poisson regression. This is the most basic model to use when modeling count data but as we know, the Poisson distribution is parameterized by a single parameter λ. This is both the mean *and* the standard deviation of the output variable meaning that when using this distribution we are unable to model data in which these two quantities are not related to each other.

Overdispersion

A common phenomenon that we observe with count data is known as **overdispersion**. This basically describes the simple idea that the variance we observe in our output variable is much larger than that predicted by our model. We are going to study this phenomenon to learn a bit more about it. To begin, please install the packages COUNT and AER (note capitals in both cases). We used the AER package in Laboratory 1.

Question 1 (2 points)

By directly using the notes from Laboratory 1 write the sequence of commands that will do the following:

- **Load the AER package onto the workspace**
- **Load the PhDPublications dataset**
- **Train an unregularized (i.e. do not use the LASSO) Poisson regression model that predicts the number of articles in terms of the remaining features in the PhDPublications dataset.**

All of the commands you needed for this first part can be found in the slides themselves. We'll start off nice and easy. One way to test for overdispersion is to examine the ratio of the residual deviance of a model to the number of degrees of freedom. The residual deviance of a model is the constant -2 multiplied by the sum of the log-likelihoods of every single data point under the model. This quantity appears in our model summary, which we can see with the summary() function. We can also compute the residual deviance for a model trained with glm() simply by passing it as an input to the deviance() function. The number of degrees of freedom is essentially the number of variables that we can freely change when calculating a particular statistic. In our glm model, this is just the number of observations in our training data minus the number of model parameters i.e. regression coefficients. It turns out that this number is also available in our model, this time as the attribute df.residual. You can verify this by examining the attributes of the model using the str() function.

Question 2 (2 points)

Using the information in the previous paragraph, create a function that takes in a poisson regression model (trained with glm()) as input and returns the ratio of the residual deviance to the number of degrees of freedom.

We can now use our function to gauge whether our model exhibits overdispersion. As a rule of thumb, when the ratio of residual deviance of a model to the number of degrees of freedom is significantly greater than 1 we are in a situation where we have overdispersion.

Question 3 (2 points)

Using the function you just created check whether the model exhibits overdispersion and comment on the result.

It turns out that there are many more sophisticated tests for overdispersion. One package that offers an overdispersion test for Poisson regression is the qcc package. Install the qcc package on your system before doing the next exercise. We are going to use the qcc.overdispersion.test() function from this package to perform a significance test for overdispersion. This uses the χ^2 distribution. The first parameter of this function is the output column of the data frame we are interested in (so if our data frame is *df* and our output column is *output*, the first parameter we pass to this function would be df$output). After this, we also specify a *type* parameter and set the value of this to the string "poisson".

Question 4 (2 points)

Load the qcc package onto your workspace. Using the information in the previous paragraph, run the significance test for overdispersion for the model we trained earlier.

The result of the previous call provides us with a p-value. The null hypothesis is that there is no overdispersion, similar to the way that the significance tests on our regression coefficients assume a null hypothesis that the output is unrelated to the feature in question.

Question 5 (2 points)

Study the p-value shown. Under a 95% confidence interval, does this tell us that overdispersion is present or absent?

By default, when we train a Poisson model we assume that there is no overdispersion present. To factor in overdispersion we can train a variant of this model, known as a Quasi-Poisson model. This is exactly the same as a Poisson model, and in fact the coefficients of the model that we obtain are the same. What differs is that the interpretation of the coefficients changes as the standard errors are computed in a different way to factor in the effects of overdispersion. To train a Quasi-Poisson model with glm() we must specify the *family* parameter be "quasipoisson".

Question 6 (2 points)

Train a Quasi-Poisson model for the PhDPublications data

If we examine the summary outputs of both our models we will see that both of the regression coefficients are the same. The estimates of the standard errors, however, are different.

Question 7 (2 points)

Look carefully at the results of the significance tests on the coefficients. If we use 95% confidence intervals, do both models suggest that the same input features are significantly related to the output? If not, in which feature(s) do the two models differ in their conclusion?

Overdispersion can be tricky to deal with because the root cause of the problem might be that we are using the wrong type of model. On the other hand, we might be using the right type of model, such as the Poisson regression model, but the model is missing a predictor. We are going to demonstrate this in a simple way with some synthetic data.

Question 8 (8 points)

Write the commands to implement the following sequence of actions.

- **Generate a feature x1 as 300 random samples drawn from the uniform distribution in the interval [0,1]**
- **Generate features x2 and x3 in exactly the same way.**
- **Compute a weighted linear sum of these features ysum, according to the following equation:**

  ysum = 2 * x1 + 5.7 * x2 -7.8 * x3
- **Now compute a vector y as 300 samples drawn from a poisson distribution where lambda = e^{ysum}.**
- **Finally, create a dataframe my_df with columns x1, x2, x3 and y**
- **Using this dataframe and the glm() function, train two Poisson models. In one, use all of the three x features, in the other, use just 2 of them.**
- **Using the function you created in question 2 above, for each model check whether overdispersion is present and comment on the results.**

If we have the wrong type of model, a good alternative model to Poisson Regression is Negative Binomial Regression. This actually has 2 parameters, rather than 1, and consequently allows us to model situations where the variance and the mean are different. This time, the model's output is the mean of a Negative Binomial distribution instead of a Poisson distribution. The negative binomial distributions is parameterized by two parameters, r and p. p is the probability of success of a so-called Bernoulli trial. This is just the same as modeling the outcome of a coin-flip, where the probability of getting heads and therefore "succeeding" is p, whereas the probability of getting tails or "failing" is 1-p. The Negative Binomial distribution models the number of Bernoulli trials made when we want to fail r times (r is the first parameter of this distribution, remember) and the probability of each trial is p.

To train a model with Negative Binomial regression we need the MASS package. Install this package on your system before proceeding.

Question 9 (2 points)

Load the MASS package on to the workspace. Then, use the glm.nb() function to train a model for the PhDPublications data. This takes in the same formula as before as its first input, and its second input is the *data* parameter which takes in the data frame, again as we had before. There is no *type* parameter this time however.

One way to compare whether it is best to use a Poisson model or a Negative Binomial model is to use the Akaike Information Criterion (AIC) to compare the two models.

Question 10 (2 points)

10. Using the AIC() function, compute the AIC of the Poisson model and the Negative Binomial model. Then, based on the results, comment on which model seems to be more appropriate.

Question 11 (2 points)

11. (Bonus Question) Suggest an alternative approach to the AIC for comparing which of the two models (Poisson or Negative Binomial) might be better for our data. Then, implement this in R and discuss the results. This is an open-ended question and there could be many valid answers.