IE590 Midterm

Brian Baller

3/20/2019

Honor Code

I have obeyed all rules for this exam and have not received any unauthorized aid or advice Signed!

Overview

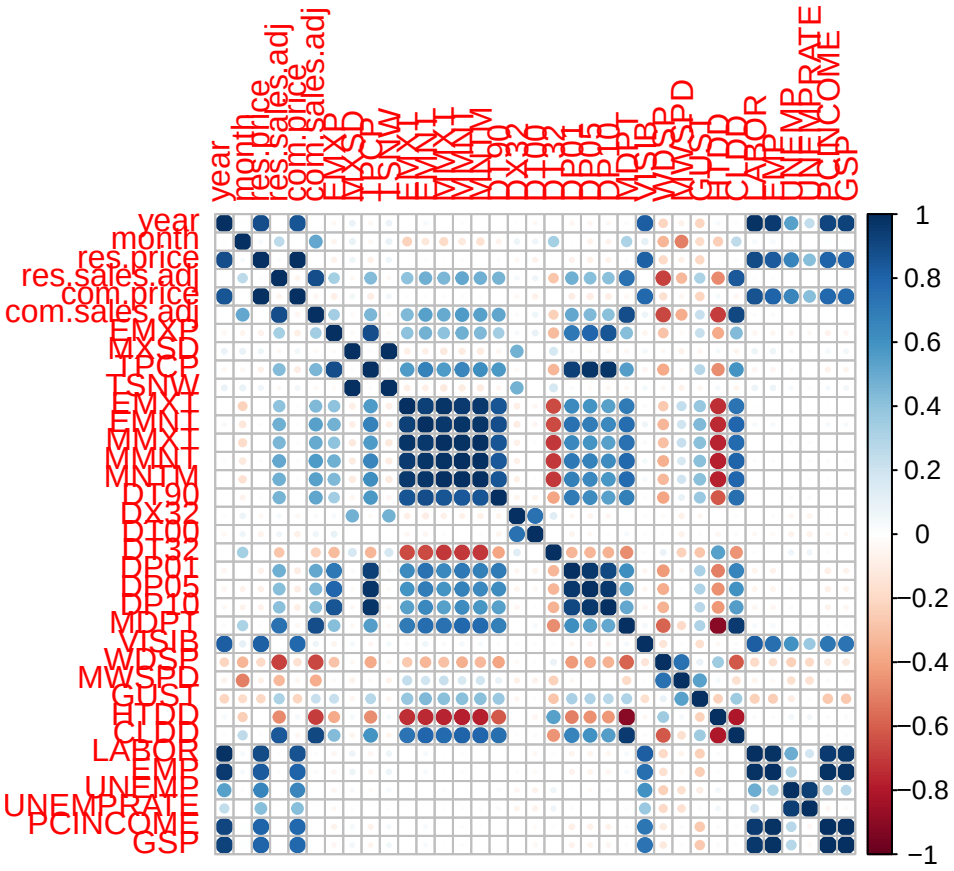
My basic approach to this exam was to run all the applicable models that we have used in class and evaluate their performance. For my analysis, I've chosen *residential sales* (*res.sales.adj*) as the response variable and assumed I'm a utilities employee with a desire to predict demand. As such, a better response variable would likely be a sum of the *commercial sales* and *residential sales*, but I'm not sure how this exam will be graded and didn't wish to create a new response variable. With *res.sales.adj* as the response, the problem becomes a regression problem so I've only chosen those models we've used that are suitable for regression. They are:

- Linear Regression
- Ridge/Lasso Regression
- GAM and all associated non-linear models
- Rpart
- Random Forest/Bagging
- Boosting

Data Import and Cleaning

My procedure for data import and cleaning is well commented in my attached code. Four NAs were found in the data in three columns (MXSD, TSNW, DT00). All three columns have only a few months with non-zero values (3, 2 and 3, respectively). Because this climate is quite warm, these fields are almost always zero. A safe assumption here is to fill the missing values in using the median value for the column, which, in these cases, is zero.

There is some multicollinearity amongst the predictors (see below correlation plot). A few predictors are even perfectly correlated with one another because of low values (like snowfall and freezing cold days) or being algebraically related (labor force, employment, unemployment rate, etc).



Several columns, like our three from above (MXSD, TSNW, and DT00) consist almost entirely of months with zero values. These sparse columns will be problematic. Firstly, months where they aren't zero might have an oversized impact on the model because they deviate so much from the norm. Also, it's likely one of the validation, test or training sets will consist entirely of zeros and perfectly correlate with another sparse column which consists entirely of zeros. To address this problem and the correlation issues from above, I didn't consider the following columns in the models:

- EMP and UNEMPRATE. EMP is simply $LABOR - UNEMPRATE$ while UNEMPRATE is $\frac{UNEMP}{LABOR}$.
- MXSD. It's perfectly correlated with total snowfall as there was only one month with snow in the 15-year period.
- DT00. Like MXSD, the column is mostly zero and was often showing up in my analysis as a column of zeros and reducing the rank of my design matrix.
- *com.sales.adj*. I removed this column as it's highly correlated with the *res.sales.adj* and I've approached this problem as an employee looking to determine demand. If I already know the commercial demand, seems like that's cheating. Again, for my assumptions and scenario a combined *com.sales.adj* and *res.sales.adj* response would work well, but might make grading difficult.

Note that I've removed these columns from consideration in the models, but didn't removed

them from the dataframe. All the models call the full dataframe (all 35 columns).

Models

All analysis was done using a training / test set split of 80/20. To pick parameters for model, I did K-fold CV within the training set. The test set was only used to evaluate the final model for each model type. The GAM model proved the best of all models and is my final model. A few notes on the other models used:

Linear Models

The linear regression model, performed well – the adjusted R^2 is 91%! The test RMSE is 557.2916481. A LASSO model was also fitted – it has a slightly reduced test RMSE (531.364246) and reduces the number of predictors used to 14.

Polynomial Model

A polynomial model was fit in the following manner – a model was run with a degree of five for all predictors. Then and second model was run with all predictors at degree one, save for those predictors which showed significance at some degree of polynomial. The test RMSE was 454.0556233.

Rpart

An rpart model was fit and optimized using *cp*. The optimal model had eight terminal nodes and a test RMSE of 500.5203335.

Random Forest

The random forest model was optimized by altering the number of parameters considered at each decision from 2 to all the parameters (bagging). The optimal value was found to be 16.

The random forest model was optimized by altering the number of parameters considered at each decision from 2 to all the parameters (bagging). A 5-fold cross-validation was performed using only the training data set. The optimal value was found to be 16, a bit over the default $\frac{p}{3}$ (although RMSE improvement from ~9 to 16 wasn't very much). Test RMSE was 484.7959653

Boosting

The Boosting model was optimized by varying the number of trees, interaction depth and shrinkage and checking cross-validation error on the training data set. The number of trees proved to have little effect, but the optimal interaction depth was three and the shrinkage factor was 0.005. Test RMSE is 445.1993306.

Summary of Performance

The below table summarizes the performance of the models discussed.

```
##           Training RMSE Test RMSE
## err.gam         341.9766  440.8045
## err.boost       157.9862  445.1993
## err.poly        332.6585  454.0556
## err.rf          199.8955  484.7960
## err.rpart       464.7568  500.5203
## err.lasso       440.3830  531.3642
## err.lm          418.1220  557.2916
```

GAM

A GAM was fit using the step.Gam optimization function. I used smoothing splines up to df=5. Like the best polynomial function, the best GAM used mostly linear predictors with a few splines at varying degrees. This makes sense: the most of the predictors have a linear relationship with the response, but a few have a non-linear response. GAM allows us to use non-linear basis functions for these while maintaining the simplictity of the linear functions for the other predictors. The step.Gam was able to pick the optimum "wiggliness" of the basis function for each predictor. The general GAM model is:

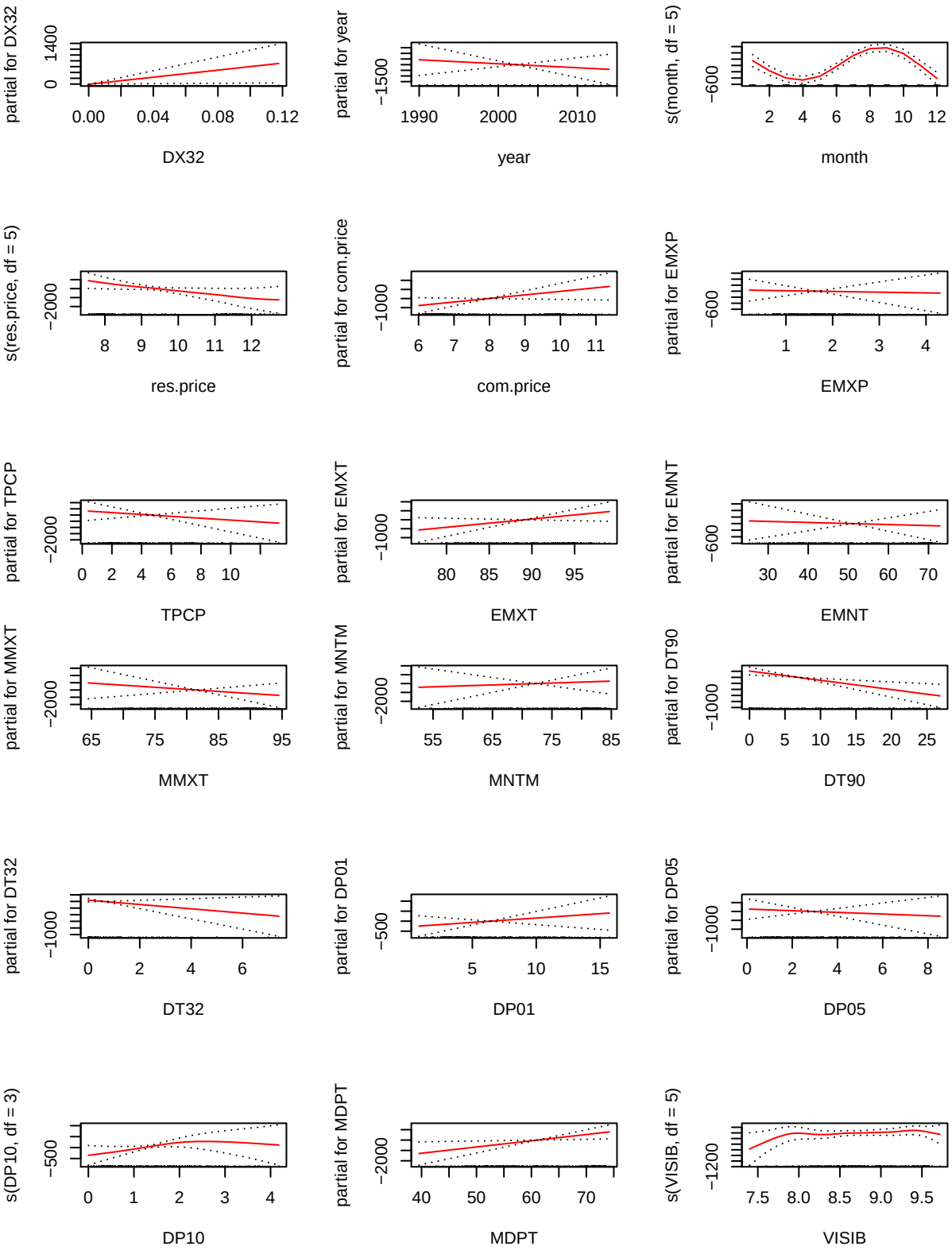
$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i$$

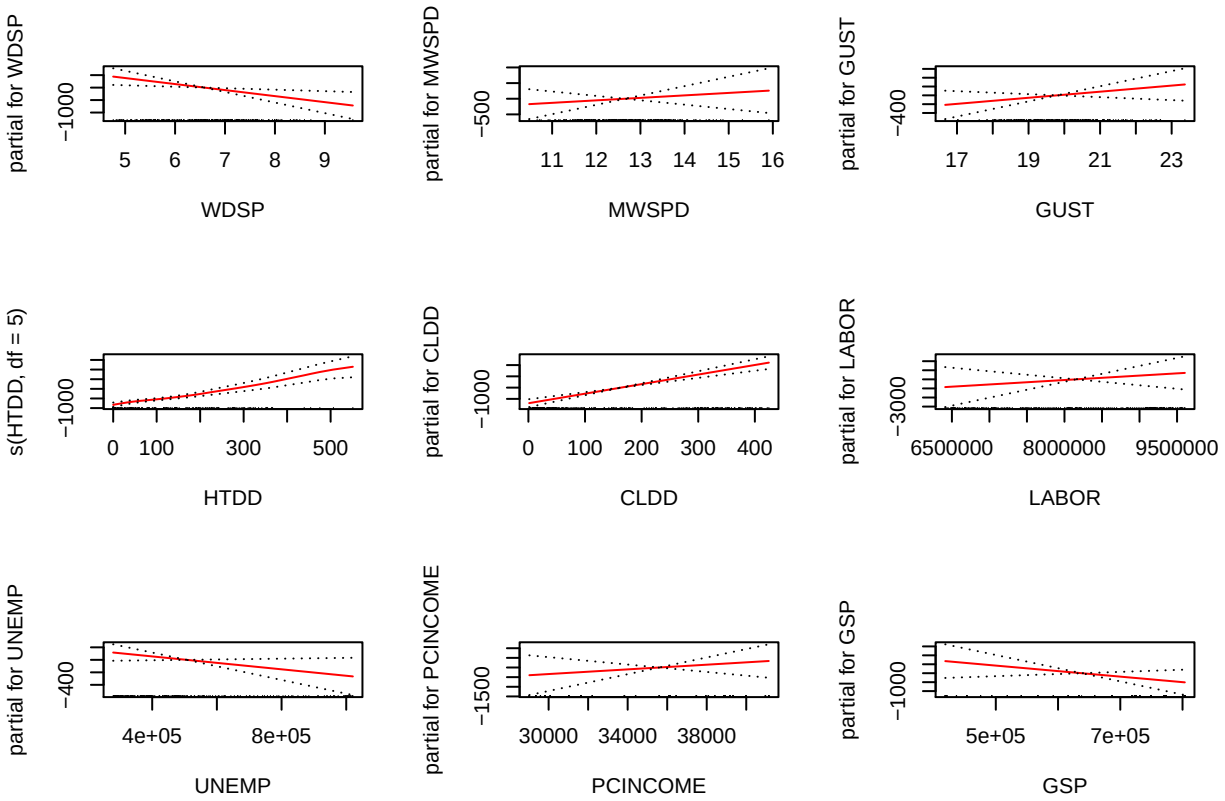
where each function can be linear or non-linear. My final model is : res.sales.adj ~ DX32 + year + s(month, df = 5) + s(res.price, df = 5) + com.price + EMXP + TPCP + EMXT + EMNT + MMXT + MNTM + DT90 + DT32 + DP01 + DP05 + s(DP10, df = 3) + MDPT + s(VISIB, df = 5) + WDSP + MWSPD + GUST + s(HTDD, df = 5) + CLDD + LABOR + UNEMP + PCINCOME + GSP. Thus, all predictors besides *month*, *res.price*, *DP10*, *VISIB*, and *HTDD* use linear basis functions ($f_j((x_{ij}))$.

The non-linear functions used are smoothing splines, which take the form:

$$\underset{g \in S}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 df$$

where λ penalizes the second derivative (or wiggliness) of the function. Note that in the gam function we specify df vice λ. GAM is well illustrated with individual plots:





The predictors *month*, *res.price*, *DP10*, *VISIB*, and *HTDD* all have non-linear functions. Noting the sinusoidal pattern of the *month* function, it's clear why a linear fit for *month* didn't make the best predictions.

Conclusion

I choose the GAM model because it performed the best on the test of all models built (although boosting was a close second). Also, GAM isn't too difficult to interpret as many of the predictors maintain a linear relationship and it provides excellent descriptive plots.