STAT 452 Project

Simon Fraser University

STAT 452

Professor Owen Ward

Dilpreet Grewal

301451516

December 5th 2024

Initial Data Exploration And How Models Were Evaluated

When I initially loaded in and viewed the data my goal was to identify whether I was dealing with a regression or classification problem. As this would significantly change the process in which I would fit and choose the best model. As the response column (Y) only contained values “0” or “1” I identified this as a classification problem, edited the data so the Y column would be categorical and moved forward with the modelling. The metric used to compare and evaluate these classification models was the misclassification error.

I had also performed a normality test on each of the X columns and found that the majority did not follow a normal distribution. This will be important in deciding to scale or not for some models.

Models Tried For This Problem

Section 11 is where we started to learn about classification methods, so I took most of the classification methods learned from that point onward to fit the data, while ensuring the data was scaled when necessary for different models. I started with *K Nearest Neighbors* and tuned for the best number of neighbours as the number of neighbours impacts the bias variance tradeoff as described on slide 15 of section 12. I took the tuned value of the best number of neighbours for both the minimum and one standard error. Fitted these tuned models and calculated an error of 0.0504 for the minimum model and 0.0584 on the one standard error model.

Moving on to section 12, the second model I fit was *Logistic Regression with Multinom*. As the data only has 2 responses we are able to use logistic regression as it was “developed specifically for binary responses,” from section 12 slide 7. No tuning was done for this method, the data was rescaled, and the calculated misclassification error was 0.1112. The third and fourth models fit were *Logistic Regression Models with glmnet() and cv.glmnet()*. The difference between these 2 models is that by using cv.glmnet() we were able to find the optimal lasso fit asa described in the class notes. Once again I used the rescaled data and found the following misclassification errors: glmnet: 0.1112, cv.glmnet: 0.1168. Lastly, the last 2 models from section 12 was *Linear Discriminant Analysis* using scaled and unscaled data. LDA relies on the “assumption of normality for X in each class,” so it was important to scale the data. I fit the data

on the unscaled data as well just as a comparison tool knowing that the model will be biased and as a result LDA would not be able to adapt. I calculated the following misclassification errors: LDA (scaled): 0.2312, LDA (unscaled): 0.0448. The low error on the unscaled error is what I expected due to the X columns not being normally distributed leading to a biased model.

Moving on to section 17 onward where we learned random forest and boosting. The next model I fit as a *Random Forest Classifier*. I tuned the number of variables (mtry) and the node sizes (nodesize) as at times random forest "builds very large trees for classification," which can end up being too large, from section 17 slide 16. When running these models it took a long time to run so I kept the mtry between 1 and 6 and the number of nodes under 5 iterations. Based on the output of the RF Tuning Variables and Node Sizes boxplot I determined mtry=6 and nodesize=3 gave the best predictions. This tuned model had a misclassification error of 0.0272.

Moving on to Tutorial 11, we were introduced to newer packages ranger and xgboost. I followed the code provided in the Tutorial 11 solution to implement these 2 packages. First I used *Ranger*, the tuning parameters were mtry and min.node.size. Again I used these tuning parameters to make sure our tree does not become too large. From the relative OOB Boxplot the best parameters were mtry=6 and min.node.size=1 as seen below in figure 1. This tuned model has a misclassification error of 0.0264.

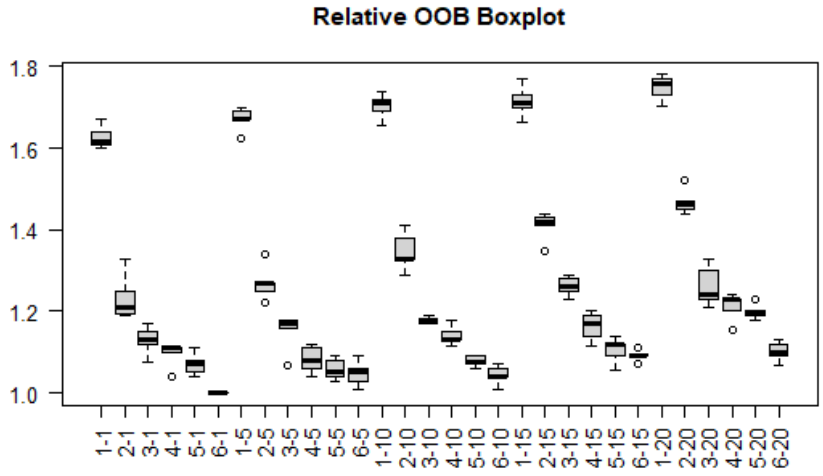


Figure 1

Lastly I fit an *xgboost model* as shown in tutorial 11. I chose an xgboost model over gbm as "it is the next version on top of a gradient boosting model," (GradientBoosting vs AdaBoost vs XGBoost vs CatBoost vs LightGBM, 2023), and has performance enhancements which make it faster (Machine Learning - GBM vs XGBOOST? Key Differences?, 2017). The parameters tuned in this model include eta (Step size shrinkage used in update to prevent overfitting) and depth (maximum depth of the tree) as described in (XGBoost Parameters — Xgboost 1.5.2 Documentation, n.d.). Once more I created a Relative OOB Boxplot shown in figure 2 below and chose the tuning parameters eta=0.25 and depth=3. The tuned model had a misclassification error of 0.0176.

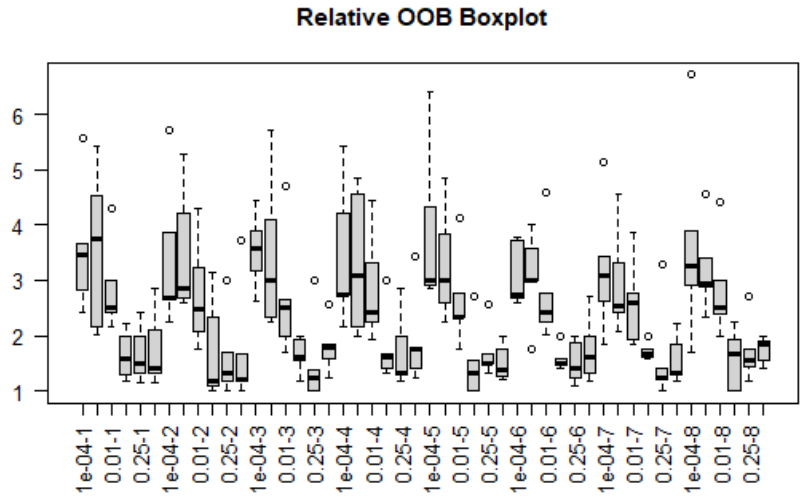


Figure 2

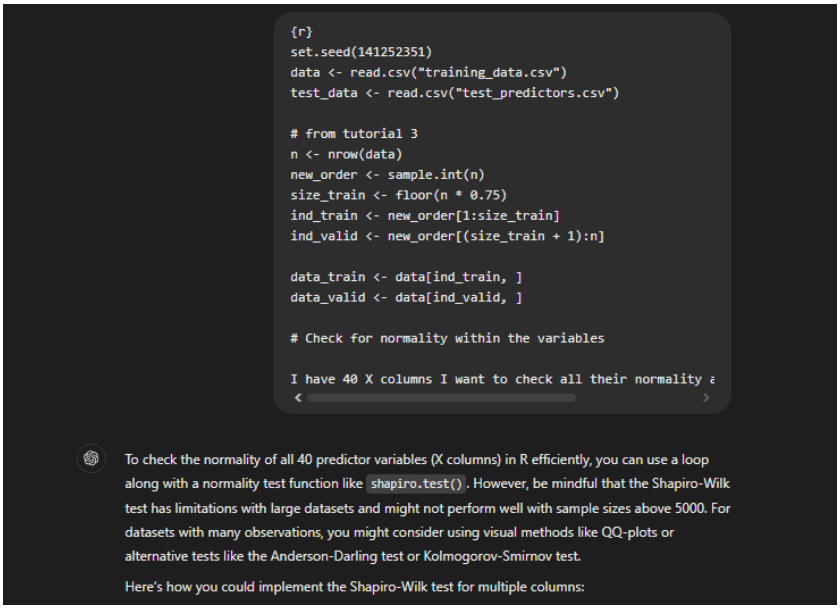
After fitting all the models, I found that the xgboost model had the lowest misclassification error and submitted predictions on the test data using the tuned xgboost model.

Identification Of Unimportant Predictors

I did not identify any predictors that were not important, therefore I do not have an answer for the number/estimate of the true number of predictors. The reasoning behind this was due to the built in variable selection processes each of the classification models have. When I fit logistic regression I had a version in which I used LASSO which shrinks estimates and selects variables. Also, as I discussed with professor Ward in office hours Random Forest also has a variable selection process. As seen in (Raghunandan, 2019) xgboost also is effective in choosing

important variables as well. Overall, I trusted the models I fitted to make the right decisions in choosing the important predictors for the response Y.

Appendix



Here's how you could implement the Shapiro-Wilk test for multiple columns:

r Copy code

```r
# Assuming your X columns are named X1, X2, ..., X40
# Modify the column names as needed based on your actual data frame

results <- list()  # To store results

# Loop over each predictor
for (i in 1:40) {
  # Construct column name
  column_name <- paste("X", i, sep="")
  # Perform the Shapiro-Wilk test on the column
  test_result <- shapiro.test(data_train[[column_name]])
  # Store the result with the column name
  results[[column_name]] <- test_result$p.value
}

# View results
results
```

This code snippet initializes an empty list, `results`, to store the normality test outcomes. It then loops through each of the 40 predictors, assuming they are named `X1, X2, ..., X40` in your data frame. It performs the Shapiro-Wilk test on each of these columns from the training data and stores the p-values in the list under their respective column names.

You can examine the p-values in the `results` list to assess the normality of each variable. Typically, a p-value less than 0.05 would indicate that the data is not normally distributed.

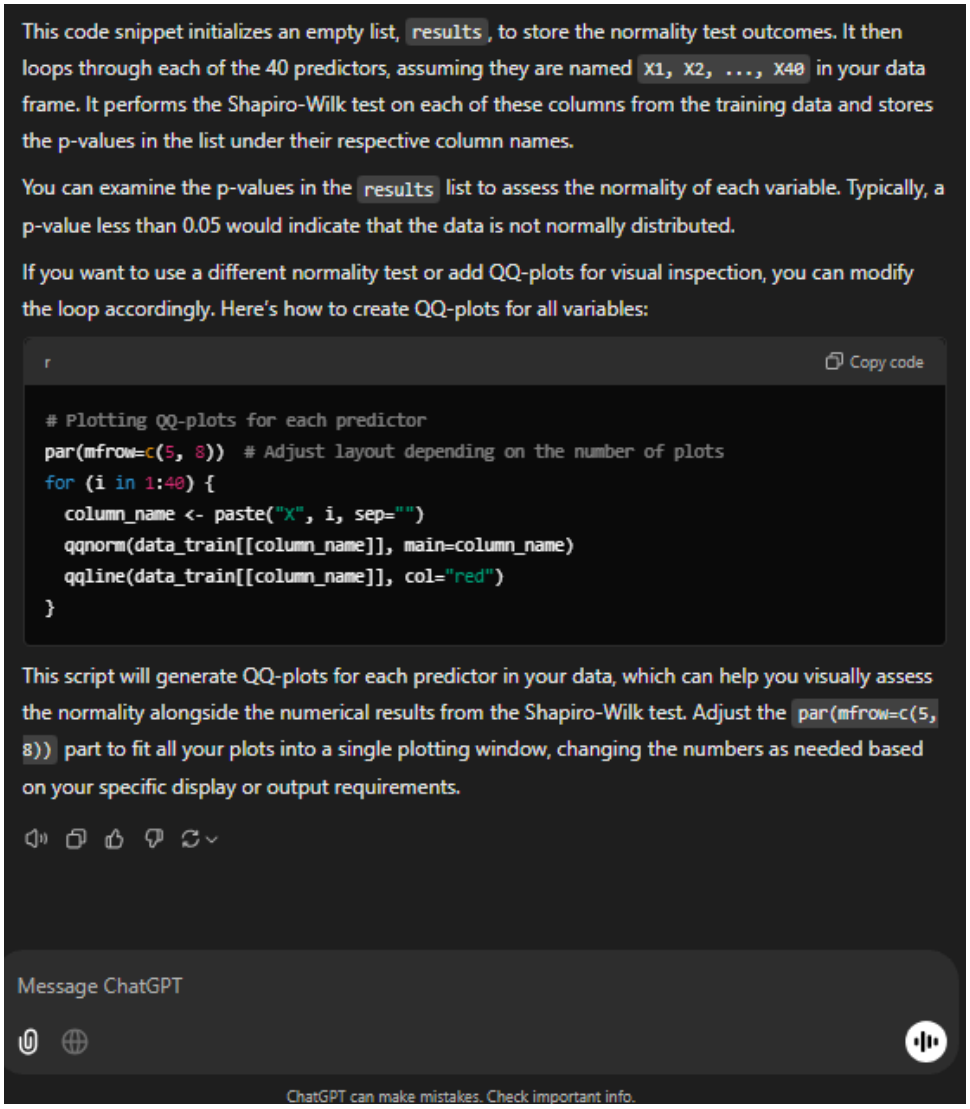
If you want to use a different normality test or add QQ-plots for visual inspection, you can modify the loop accordingly. Here's how to create QQ-plots for all variables:

r Copy code

```r
# Plotting QQ-plots for each predictor
par(mfrow=c(5, 8))  # Adjust layout depending on the number of plots
for (i in 1:40) {
  column_name <- paste("X", i, sep="")
  qqnorm(data_train[[column_name]], main=column_name)
  qqline(data_train[[column_name]], col="red")
}
```

This script will generate QQ-plots for each predictor in your data, which can help you visually assess

Message ChatGPT



References

Raghunandan. (2019, July 3). Is XGBoost effective for variable selection? Stack Overflow. https://stackoverflow.com/questions/56865981/is-xgboost-effective-for-variable-selection

GradientBoosting vs AdaBoost vs XGBoost vs CatBoost vs LightGBM. (2023, January 6). GeeksforGeeks. https://www.geeksforgeeks.org/gradientboosting-vs-adaboost-vs-xgboost-vs-catboost-vs-lightgbm/

XGBoost Parameters — xgboost 1.5.2 documentation. (n.d.). Xgboost.readthedocs.io. https://xgboost.readthedocs.io/en/stable/parameter.html

machine learning - GBM vs XGBOOST? Key differences? (2017, February 11). Data Science Stack Exchange. https://datascience.stackexchange.com/questions/16904/gbm-vs-xgboost-key-differences