Project 1 (Report)

Asif Hasan - 301376671

2023-11-26

Introduction: For the prediction problem, we are presented with 21 numeric predictor variables, denoted as X1 to X21. The primary goal is to predict the corresponding numeric response variable, Y.

Correlation Analysis: We computed the correlation matrix for the numeric variables and visualize the correlations to identify potential multicollinearity. We observe that variables X5, X11, X12, X13, X20 and X21 are correlated with each other.

Least Squares Regression: To assess the performance of all other models of our choice compared to a base model in the model selection process, we employed a least squares regression model with all 21 variables as the base model, irrespective of multicollinearity among the variables.

Hybrid Stepwise Regression: We utilized a hybrid stepwise regression on the complete training dataset to identify important variables, considering the minimum AIC value. Our observations reveal that the variables X1, X5, X8, X12, X13, X18, X21, and X20 were selected as the most significant contributors explaining the variance in the response variable Y. This method was applied during the model selection process to assess all our chosen models.

Ridge Regression: We employed ridge regression on the entire training dataset, investigating a range of λ values from 0 to 100 at intervals of 0.05. Through this exploration, we identified the optimal λ value for the regression problem as $\lambda_{\text{min}} = 0.45$. This method was then utilized in the model selection process to evaluate our models.

LASSO Regression: We performed LASSO regression on the predictor variables of the entire training dataset, utilizing both λ_{min} and λ_{1SE} for predicting the response variable Y. Notably, when using $\lambda_{\text{min}} = 0.0003034$, the coefficient values closely resemble those obtained from ridge regression. On the other hand, with $\lambda_{\text{1SE}} = 0.06096$, the variables with non-zero coefficients are X1, X4, X5, X8, X9, X13, X14, X18, X19, X20, and X21. In the model selection process, we employed both the λ_{min} and λ_{1SE} approaches of this method to assess the performance of our models.

Decision Trees: We employed a decision tree model to compare the performance of linear models with that of a basic tree model. Initially, we fitted the model with a complexity parameter value of zero, $cp = 0$, to the entire training dataset. Optimal values were identified as $cp_{\text{min}} = 0.001239$ and $cp_{\text{1SE}} = 0.001415$. Subsequently, we pruned the tree based on the values of cp_{min} and cp_{1SE}. In the model selection process, we utilized both the cp_{min} and cp_{1SE} tree pruning approaches of this method to evaluate the performance of our models.

Random Forest: Given the presence of three tuning parameters—specifically, the number of individual trees to be fitted ($ntree(B)$), the number of variables considered at each split ($mtry(m)$), and the minimum number of observations in each node that can be split ($nodesize(ns)$)—and considering the large dataset, we employed a four-step iterative process to identify optimal values for these tuning parameters.

In the first iteration, we applied the random forest model to the entire dataset with $B = 1,000$ to assess the optimality of fitting $1,000$ trees. We observed that the out-of-bag (OOB) error stabilized at around 800 trees, leading us to use $B = 1,000$ for subsequent iterations.

For the second iteration, we randomly sampled 10% of the training data in three repetitions, exploring values for m ranging from 1 to 10 and ns values of 1, 3, 5, 7, 10, 15, and 20. Based on the OOB error, we identified that m less than 6 and ns greater than 10 were suboptimal. Notably, $m = 10$ and $ns = 3$ constituted the best set of tuning parameters.

In the third iteration, with 15% of the training data randomly sampled in five repetitions, we explored m values from 6 to 10 and ns values of 1, 3, 5, 7, and 10. The analysis revealed that m less than 8 and ns greater than 7 were unfavorable, reiterating that $m = 10$ and $ns = 3$ remained the optimal parameters.

In the final iteration, using 20% of the training data randomly sampled in five repetitions, we explored m values of 8, 9, and 10, as well as ns values from 1 to 6. Based on the OOB error, the best set of parameters was determined as $m = 10$ and $ns = 4$ (RF-1), with the second-best set being $m = 10$ and $ns = 2$ (RF-2).

Upon further examination of variable importance using the best tuning parameters, RF-1, and fitting the model to the entire dataset, nine important variables were identified in order of importance: X8, X21, X18, X12, X20, X1, X11, X13, and X5. In the model selection process, we utilized the two best sets of parameters, RF-1 and RF-2, to evaluate our selected models.

Boosted Trees: We adopted a similar approach to tune parameters for the boosted trees model as employed for the random forest model. However, in the case of boosted trees, only three iterations were necessary to identify optimal values for the tuning parameters—specifically, the number of individual trees to be fitted ($n.trees(B)$), the depth of the tree ($interaction.depth(d)$), and the shrinkage parameter ($shrinkage(\lambda)$).

In the first iteration, we fitted the model to the entire dataset using $B = 10,000$. We observed that the out-of-bag (OOB) error stabilized at around $8,000$ trees, leading us to use $B = 10,000$ for subsequent iterations.

For the second iteration, using 10% of the training data randomly and two repetitions of 2-fold cross-validation, we explored d values from 1 to 5 and λ values of 0.001, 0.005, 0.025, and 0.125. The analysis revealed that d less than 3 and λ greater than or equal to 0.125 were suboptimal. Notably, we found that $d = 5$ and $\lambda = 0.005$ constituted the best set of tuning parameters.

In the final iteration, using 20% of the training data randomly and three repetitions of 2-fold cross-validation, we explored d values from 3 to 8 and λ values of 0.001, 0.005, 0.025, 0.05, 0.075, and 0.01. Based on the OOB error, we identified the best set of values for the tuning parameters as $d = 8$ and $\lambda = 0.001$ (BT-1), with the second-best set being $d = 8$ and $\lambda = 0.025$ (BT-2).

Upon examining the importance of variables using the best tuning parameters (BT-1) and fitting the model to the entire dataset, we identified nine variables deemed important in this method. In order of importance, they are X18, X21, X8, X12, X13, X11, X20, X1, and X5. Notably, these variables align with those identified by the random forest as important, although the order of importance differs. Similar to the random forest, we utilized the best two sets of parameters, BT-1 and BT-2, to evaluate our models in the model selection process.

Model Selection: Our selection comprises 11 models: Least Squares Regression (Base Model), Hybrid Stepwise Regression, Ridge Regression-$\lambda_{\min}$, LASSO Regression-$\lambda_{\min}$, LASSO Regression-$\lambda_{1\text{SE}}$, Decision Trees-$cp_{\min}$, Decision Trees-$cp_{1\text{SE}}$, Best Random Forest (RF-1), Second-best Random Forest (RF-2), Best Boosted Trees (BT-1), and Second-best Boosted Trees (BT-2). Utilizing 5-fold cross-validation, we computed the mean squared prediction errors (MSPEs) for all the mentioned models.

The most effective model, demonstrated by the lowest mean squared prediction error (MSPE) of 1.778, is the Boosted Trees model with the second-best set of tuned parameters ($B = 10,000$, $d = 8$, and $\lambda = 0.025$), referred to as BT-2. Following at a distance is the Random Forest model with the best set of tuned parameters ($B = 1,000$, $m = 10$, and $ns = 4$), labeled as RF-1, showing a mean MSPE of 18.68. The provided code corresponds to the results described here.

```
set.seed(564781)
# set number of folds
V <- 5
# sample the folds
folds <- floor((sample.int(nrow(df)) - 1) * V / nrow(df)) + 1
# create matrix for MSPEs for 11 models
MSPEs.cv <- matrix(NA, nrow = V, ncol = 11)
colnames(MSPEs.cv) <- c("LS", "Step-hybrid", "Ridge", "LASSO-min", "LASSO-1SE",
                        "dTrees-min", "dTrees-1SE", "RandomF-1", "RandomF-2",
                        "BoostedT-1", "BoostedT-2")
# run cross-validation in for-loop
for(v in 1:V) {
  # fit 11 models on fold == !v
  model.ls.cv <- lm(Y ~ ., data=df[folds!=v, ])
  model.step.cv <- step <- step(
    object = lm(data=df[folds!=v, ], formula = Y ~ 1),
    scope = list(upper = lm(data=df[folds!=v, ], formula = Y ~ .)),
    k = log(nrow(df[folds!=v, ])))
  model.ridge.cv <- lm.ridge(Y ~ ., lambda=seq(0, 100, .05), df[folds!=v, ])
  model.lasso.cv <- cv.glmnet(family = "gaussian",
    y = as.matrix(df[folds!=v, 1]), x = as.matrix(df[folds!=v, c(2:22)]), )
  model.tree.cv <- rpart(Y ~ ., method = "anova", data = df[folds!=v, ], cp = 0)
  cpt <- model.tree.cv$cptable
  minrow <- which.min(cpt[, 4])
  cplow.min <- cpt[minrow, 1]; cpup.min <- ifelse(minrow==1, yes=1, no=cpt[minrow-1, 1])
  cp.min <- sqrt(cplow.min * cpup.min)
  se.row <- min(which(cpt[, 4] < cpt[minrow, 4] + cpt[minrow, 5]))
  cplow.1se <- cpt[se.row, 1]; cpup.1se <- ifelse(se.row==1, yes=1, no=cpt[se.row-1, 1])
  cp.1se <- sqrt(cplow.1se * cpup.1se)
  model.tree.cv.prune.min <- prune(model.tree.cv, cp = cp.min)
  model.tree.cv.prune.1se <- prune(model.tree.cv, cp = cp.1se)
  model.rf1.cv <- randomForest(data = df[folds!=v, ], Y ~ .,
    nodesize = 4, ntree = 1000, mtry = 10)
  model.rf2.cv <- randomForest(data = df[folds!=v, ], Y ~ .,
    ntree = 1000, nodesize = 10, mtry = 2)
  model.bt1.cv <- gbm(data = df, Y ~ ., distribution = "gaussian",
    n.trees = 10000, interaction.depth = 8, shrinkage = 0.001)
  model.bt2.cv <- gbm(data = df, Y ~ ., distribution = "gaussian",
    n.trees = 10000, interaction.depth = 8, shrinkage = 0.025)

  # predict Y using the fitted models on fold == v
  pred.ls.cv <- predict(model.ls.cv, newdata=df[folds==v, ])
  pred.step.cv <- predict(model.step.cv, newdata=df[folds==v, ])
  pred.ridge.cv <- as.matrix(cbind(1, df[folds==v, 2:22])) %*%
    coef(model.ridge.cv)[which.min(model.ridge.cv$GCV), ]
  pred.lasso.min.cv <- predict(model.lasso.cv, newx=as.matrix(df[folds==v, c(2:22)]),
                               s=model.lasso.cv$lambda.min)
  pred.lasso.1se.cv <- predict(model.lasso.cv, newx=as.matrix(df[folds==v, c(2:22)]),
```

```
                                s=model.lasso.cv$lambda.1se)
  pred.tree.min.cv <- predict( model.tree.cv.prune.min, newdata=df[folds==v, ])
  pred.tree.1se.cv <- predict( model.tree.cv.prune.1se, newdata=df[folds==v, ])
  pred.rf1.cv <- predict( model.rf1.cv, newdata=df[folds==v, ])
  pred.rf2.cv <- predict( model.rf2.cv, newdata=df[folds==v, ])
  pred.bt1.cv <- predict( model.bt1.cv, newdata=df[folds==v, ])
  pred.bt2.cv <- predict( model.bt2.cv, newdata=df[folds==v, ])

  # calculated MSPEs for 11 models for each v fold
  MSPEs.cv[v, 1] <- mean((df[folds==v, "Y"] - pred.ls.cv)^2)
  MSPEs.cv[v, 2] <- mean((df[folds==v, "Y"] - pred.step.cv)^2)
  MSPEs.cv[v, 3] <- mean((df[folds==v, "Y"] - pred.ridge.cv)^2)
  MSPEs.cv[v, 4] <- mean((df[folds==v, "Y"] - pred.lasso.min.cv)^2)
  MSPEs.cv[v, 5] <- mean((df[folds==v, "Y"] - pred.lasso.1se.cv)^2)
  MSPEs.cv[v, 6] <- mean((df[folds==v, "Y"] - pred.tree.min.cv)^2)
  MSPEs.cv[v, 7] <- mean((df[folds==v, "Y"] - pred.tree.1se.cv)^2)
  MSPEs.cv[v, 8] <- mean((df[folds==v, "Y"] - pred.rf1.cv)^2)
  MSPEs.cv[v, 9] <- mean((df[folds==v, "Y"] - pred.rf2.cv)^2)
  MSPEs.cv[v, 10] <- mean((df[folds==v, "Y"] - pred.bt1.cv)^2)
  MSPEs.cv[v, 11] <- mean((df[folds==v, "Y"] - pred.bt2.cv)^2)
}
# get the MSPEs for each 5 folds
MSPEs.cv
# get the mean MSPEs
(MSPEcv <- apply(X = MSPEs.cv, MARGIN = 2, FUN = mean))
# create boxplots for MSPEs
boxplot(MSPEs.cv, main = "MSPE \n Cross-Validation")
# create boxplots for relative MSPEs
low.cv <- apply(MSPEs.cv, 1, min)
boxplot(MSPEs.cv / low.cv, las = 2, main = "Relative MSPE \n Cross-Validation")
```

Final Model and Predictions: Based on the previous analysis, a substantial difference in the mean MSPE is evident between the two best-performing models. Therefore, we opted for the boosted trees model with tuned parameters: $B = 10,000$, $d = 8$, and $\lambda = 0.025$ (BT-2) to make predictions on the test set. We trained the model on the training dataset and examined the variables considered most important. Notably, the top five variables in order of importance are X18, X21, X8, X12, and X1; it is noteworthy that these five variables are not correlated. Subsequently, we used the model to predict the response variable Y using the test dataset and saved the predictions in a CSV file.

Important Predictor Variables: Employing the methods and metrics, we can be certain that variables X2, X3, X6, X7, X10, X14, X15, X16, and X17 are deemed not important. Taking into account the multicollinearity issue, we can safely estimate that the number of true (uncorrelated) predictors is five.

Conclusion: In summary, we observe that variables X5, X11, X12, X13, X20, and X21 are correlated with each other. Additionally, variables X2, X3, X6, X7, X10, X14, X15, X16, and X17 do not explain the variance in the response variable Y. On the other hand, variables X1, X8, X12, X18, and X21 are identified as significant contributors, explaining a substantial portion of the variance in Y. Through cross-validation, we achieved the best mean MSPE of 1.778 using the boosted trees model with tuned parameters: $B = 10,000$, $d = 8$, and $\lambda = 0.025$. Finally, we utilized the model to predict the response variable Y for the test dataset.

Note: The code not shown is provided in the *rmd* file. If you intend to execute the code within the *rmd* file, modify 'eval=TRUE' in line 9 accordingly.