Problem_1 - Riya

Code

Hide

```
set.seed(15072)

library(dplyr)
library(caret)
library(rpart)
library(randomForest)
library(tidyverse)
library(readr)
library(ggplot2)
library(olsrr)
library(rpart.plot)
#install.packages("gbm")
library(gbm)
#install.packages("Metrics")
library(Metrics)
```

```


Attaching package: ‘Metrics’

The following objects are masked from ‘package:caret’:

    precision, recall
```

Hide

```
df <- read_csv("./ames.csv")
```

```
Rows: 2818 Columns: 73── Column specification ──────────────────────────────────────
──────────────
Delimiter: ","
chr (40): MSZoning, Street, Alley, LotShape, LandContour, LotConfig, LandSlope,...
dbl (33): SalePrice, LotFrontage, LotArea, YearBuilt, YearRemodAdd, MasVnrArea,...
ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Hide

```
head(df)
```

SalePrice <dbl>	MSZon... <chr>	LotFrontage <dbl>	LotArea <dbl>	Street <chr>	Alley <chr>	LotSha... <chr>	LandContour <chr>	LotConfig <chr>
215000	RL	141	31770	Pave	No Alley	IR1	Lvl	Corner
105000	Other	80	11622	Pave	No Alley	Reg	Lvl	Inside

SalePrice <dbl>	MSZon… <chr>	LotFrontage <dbl>	LotArea <dbl>	Street <chr>	Alley <chr>	LotSha… <chr>	LandContour <chr>	LotConfig <chr>
172000	RL	81	14267	Pave	No Alley	IR1	Lvl	Corner
244000	RL	93	11160	Pave	No Alley	Reg	Lvl	Corner
189900	RL	74	13830	Pave	No Alley	IR1	Lvl	Inside
195500	RL	78	9978	Pave	No Alley	IR1	Lvl	Inside

Hide

```
# Split the data into test and train (70% training, 30% testing)
set.seed(15072)
nrow = nrow(df)
df[sapply(df, is.character)] = lapply(df[sapply(df, is.character)], as.factor)
train_index = sample(1:nrow, size = 0.7*nrow)
train_ames = df[train_index, ]
test_ames = df[-train_index, ]
```

a) List the four out-of-sample R-squared values of the models you built from last time: linear regression; default-parameter CART; cp-optimized tree; random forest.

- For mod_linear_final: 0.8254482.
- Default parameter CART: 0.691.
- CP-optimized tree: 0.7894255.
- Random forest: 0.8733.

b) Fit an initial GBM model to the training data with the parameters listed below. Paste the code you used, the summary() of that model, and both its in-sample and out-of-sample R-squared.

Hide

```
set.seed(15072)
# initial gb model
initial_model_gbm <- gbm(SalePrice ~ .,
                 data = train_ames,
                 distribution = "gaussian", # For regression
                 n.trees = 100,             # Number of boosting iterations
                 interaction.depth = 6,     # Maximum depth of each tree
                 shrinkage = 0.1,           # Learning rate
                 cv.folds = 10)             # Number of cross-validation folds
summary(initial_model_gbm)

pred_train = predict(initial_model_gbm, newdata=train_ames)
sse_train = sum((train_ames$SalePrice - pred_train)^2)
sst_train = sum((train_ames$SalePrice - mean(train_ames$SalePrice))^2)
insampler2 = 1-(sse_train/sst_train)

pred_test = predict(initial_model_gbm, newdata=test_ames)
sse_test = sum((test_ames$SalePrice - pred_test)^2)
sst_test = sum((test_ames$SalePrice - mean(test_ames$SalePrice))^2)
outofsampler2 = 1-(sse_test/sst_test)

cat("In sample r squared:", round(insampler2, 4), "\n")
cat("Out of sample r squared:", round(outofsampler2, 4), "\n")
```

- In sample r squared: 0.9516.
- Out of sample r squared: 0.8813.

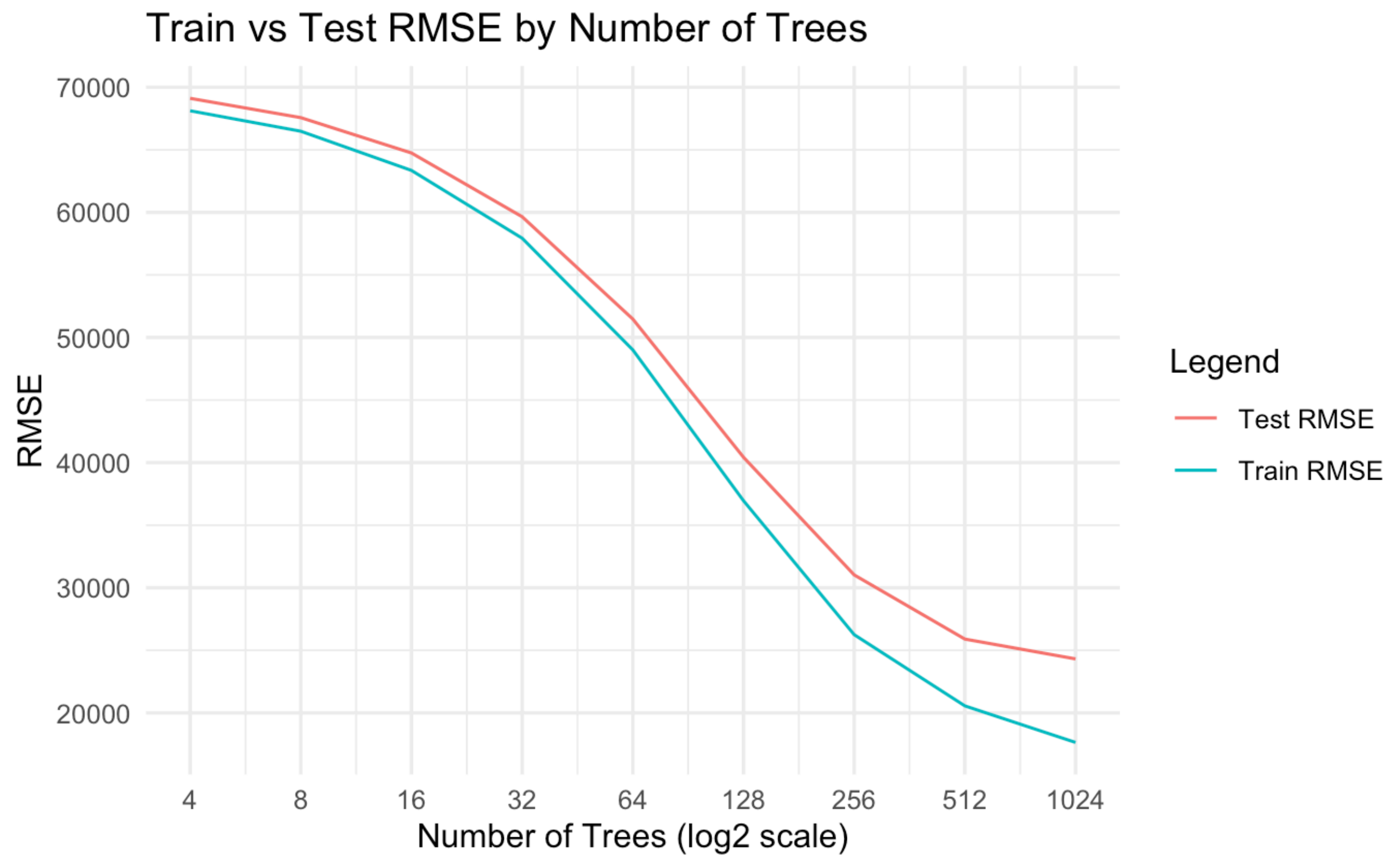
c) For each of the following hyperparameters explain how these values affect the in-sample and out-of-sample R-squared values.

- Shrinkage is the learning rate, so if this number is high it means each tree has a strong influence, quickly fitting the training data and potentially over fitting. A smaller shrinkage dampens the effect of each tree, requiring more trees to reach the same level of fit. When shrinkage is large, in sample r squared is high and out of sample r squared is lower. When shrinkage is small, we get a lower r squared on the training data but a significantly better and more honest r squared on unseen data (out-of-sample R-squared).
- Interaction.depth is the maximum depth per tree, so if this number is too big we risk over fitting and if it's too small we risk under fitting. Generally having too big a number for interaction.depth when we train our model is bad because it'll over fit to the test set resulting in a very high in sample r squared but then it'll perform poorly on the new data in the test set resulting in a low out of sample r squared.
- N.trees gives us the number of boosting iterations, so the more iterations we have, generally speaking, the more our trees can learn and pinpoint improvement spots (high residual areas) from past residuals they have encountered. Having a higher n.trees hyper parameter improves both in and out of sample r squared.

d) Suppose instead that you set interaction.depth to 3 and shrinkage to 0.01 (and left all other hyperparameters as they were in part b). For n.trees = 4, 8, …, 1024, compute the train RMSE (root mean square error) and the test RMSE. Plot the two RMSE lists on the same graph. Also include all code used for this part.

Hide

```r
set.seed(15072)

# tree counts
exponents <- 2:10
n.trees_list <- 2^exponents

# initialize RMSE storage lists
train_rmse <- numeric(length(n.trees_list))
test_rmse <- numeric(length(n.trees_list))

# loop over tree counts
for (i in seq_along(n.trees_list)) {
  n_trees <- n.trees_list[i]

  model <- gbm(SalePrice ~ .,
               data = train_ames,
               distribution = "gaussian",
               n.trees = n_trees,
               interaction.depth = 3,
               shrinkage = 0.01,
               cv.folds = 10,
               verbose = FALSE)

  # predictions
  train_pred <- predict(model, newdata = train_ames, n.trees = n_trees)
  test_pred <- predict(model, newdata = test_ames, n.trees = n_trees)

  # RMSE calculations
  train_rmse[i] <- rmse(train_ames$SalePrice, train_pred)
  test_rmse[i] <- rmse(test_ames$SalePrice, test_pred)
}

rmse_df <- data.frame(
  Trees = n.trees_list,
  Train_RMSE = train_rmse,
  Test_RMSE = test_rmse
)

ggplot(rmse_df, aes(x = Trees)) +
  geom_line(aes(y = Train_RMSE, color = "Train RMSE")) +
  geom_line(aes(y = Test_RMSE, color = "Test RMSE")) +
  scale_x_continuous(trans = "log2", breaks = n.trees_list) +
  labs(title = "Train vs Test RMSE by Number of Trees",
       x = "Number of Trees (log2 scale)",
       y = "RMSE",
       color = "Legend") +
  theme_minimal()
```



e) Which value of n.trees minimizes train set RMSE? test set RMSE? Are those two values the same or different? Which one is more important for model performance, and why?

The value of n.trees that minimizes both train and test set RMSE is 1024. The one that is more important for model performance is test set RMSE because we want to know how our model performs on data it hasn't seen before. A high train set RMSE could just be the result of over fitting. If we really wanted to be be more robust in our techniques, we would have split our data into not just train/test sets, but train/test/validation sets to prevent information leakage when we go back and forth testing different hyper parameters. We would only use the test check to test the validity and accuracy of our final tuned model the final model.

f) Identify the patterns the graphs of train set RMSE and test set RMSE follow, and explain why.

Both the train and test RMSE decrease sharply at first because as the number of trees increases, the model can fit the data better and better – each additional tree corrects residuals from the previous ensemble. The test RMSE starts to flatten out a bit around 512-1024 trees, telling us that it has reached an approximate minimum in its RMSE improvement, suggesting minimal over fitting. This pattern makes sense because GBM builds trees based on the residuals of those coming before it. Having more trees means having more capacity to fit the training data, so we see a lower training RMSE. Also, because our shrinkage is set to 0.01 (pretty small), each tree contributes only a small adjustment. This slows learning and helps prevent over fitting, but requires many trees to reach optimal performance. All in all, we see that initially, test RAMSE drops as the model captures additional signal.

However, after a certain point (512-1024 trees), those gains diminish because each of the later trees adds very little information in terms of residuals. The flattening of the RMSE curve here signals that the model is no longer improving generalization to unseen data points despite the fact that training RMSE continues to drop.

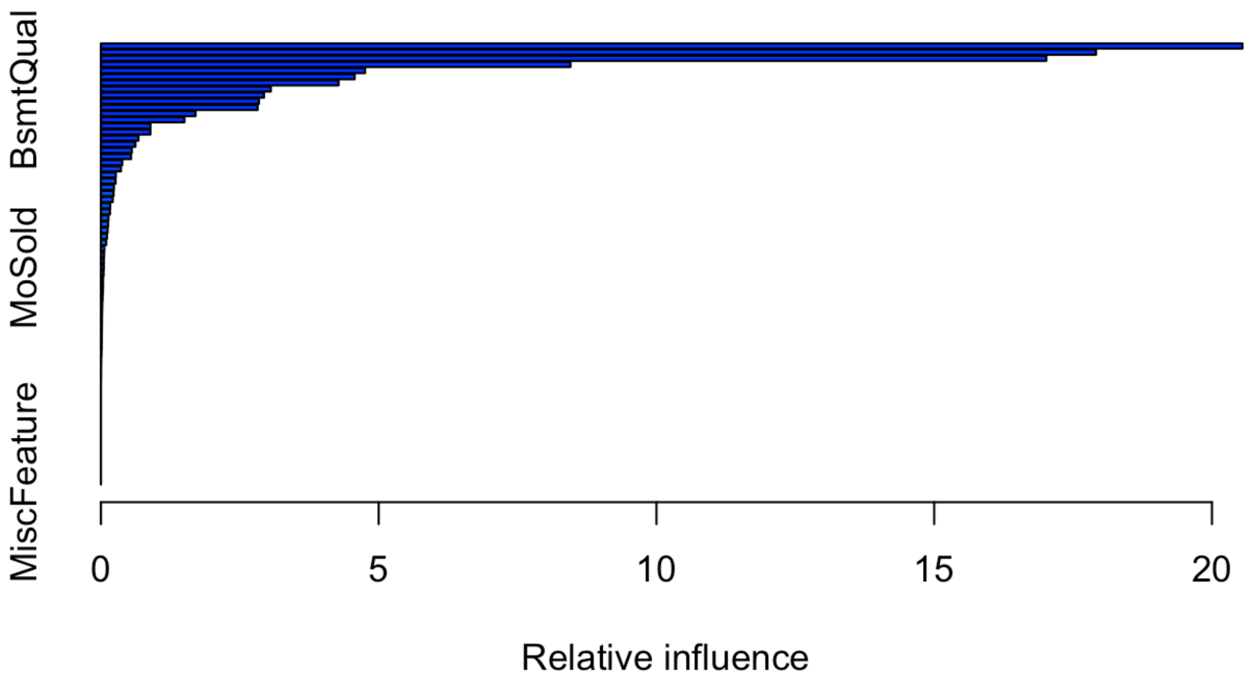
g) Consider the 6 different models you now have: the 4 from last deliverable, the initial GBM from part b, and your new optimal GBM from part e. Which one has the largest R-squared?

Hide

```
set.seed(15072)
#fitting the final model with 1024 trees as this was the best one based off the plot (lo
west rmse test)
best_model <- gbm(SalePrice ~ .,
                  data = train_ames,
                  distribution = "gaussian",
                  n.trees = 1024,
                  interaction.depth = 3,
                  shrinkage = 0.01,
                  cv.folds = 10,
                  verbose = FALSE)

summary(best_model)
```

	var <chr>	rel.inf <dbl>
Neighborhood	Neighborhood	20.548534391
GrLivArea	GrLivArea	17.913012441
ExterQual	ExterQual	17.019096096
TotalBsmtSF	TotalBsmtSF	8.452098387
GarageCars	GarageCars	4.754806480
YearBuilt	YearBuilt	4.566606769
KitchenQual	KitchenQual	4.277585102
BsmtQual	BsmtQual	3.058224316
GarageArea	GarageArea	2.934236978
X1stFlrSF	X1stFlrSF	2.843893173

1-10 of 72 rows   Previous **1** 2 3 4 5 6 ... 8 Next



Hide

```
test_pred <- predict(best_model, newdata = test_ames, n.trees = 1024)
actual <- test_ames$SalePrice
r2_out_sample <- 1 - sum((actual - test_pred)^2) / sum((actual - mean(actual))^2)
print(paste("Out-of-sample R^2:", round(r2_out_sample, 4)))
```

```
[1] "Out-of-sample R^2: 0.8811"
```

- For mod_linear_final: 0.8254482.
- Default parameter CART: 0.691.
- CP-optimized tree: 0.7894255.
- Random forest: 0.8733.
- Initial GBM from part b: 0.8813.
- New optimal GBM from part e: 0.8811.
- The model with the largest out of sample r squared is the initial GBM from part b with oos r squared of 0.8813. While one may think this is not ordinary as the model from part e was built more optimally, there are key hyper parameters that differ in the construction of both models. In part b, the GBM was built with a deeper interaction depth (6 vs 3) and a higher learning rate (shrinkage of 0.1 vs 0.01). These choices allow the model to capture more complex interactions and adapt more aggressively to the training data. This can lead to better performance when the underlying signal is strong and the model is not overfitting. In contrast, part e's model—though more conservative—uses a smaller interaction depth and lower shrinkage, which slows learning and enforces stronger regularization. This setup is often preferred for stability and generalization, but in this case, the added regularization may have slightly limited the model's ability to fit

subtle patterns in the data.Ultimately, both models perform very similarly, and the marginal difference in r squared reflects a delicate tradeoff between model complexity and regularization.