**kaggle report**

Introduction This report encapsulates the learning and experiences from the Kaggle competition, where the challenge was to predict used car prices. The dataset comprised 40,000 used cars, described by various features and conditions. The primary objective was to construct an accurate predictive model, balancing the intricacies of data preparation, feature selection, and model tuning.

Data Exploration and Understanding The initial phase involved a deep dive into the dataset. Each car was described by a plethora of features, which necessitated a careful examination to understand their nature and distribution. Key steps included categorizing variables (numeric vs. categorical), summarizing their characteristics, and visually exploring them for patterns and anomalies. This phase revealed the complexities of the dataset, such as missing values and the varied nature of variables. For instance, numeric variables like mileage and horsepower varied significantly across entries, while categorical variables like body type had multiple levels, some of which were rare.

Data Preparation and Transformation Given the data's complexity, significant effort was directed towards cleaning and preparing the dataset for analysis. Missing values were a major concern. A systematic approach was taken to address this: for numeric variables, I applied the mutate\_at function from the dplyr package to replace missing values with the mean of the column, ensuring that the central tendency of the data was maintained.<code>

data <- data %>%

mutate\_at(vars(names(data)[numeric\_vars]), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .))

When it came to categorical variables, I opted for a mode-based imputation strategy. This was especially evident in the treatment of exterior\_color, where missing values were replaced with the most frequent color. The code snippet below encapsulates this process. <code>

mode\_color <- data %>%

group\_by(exterior\_color) %>%

summarize(count = n()) %>%

arrange(desc(count)) %>%

top\_n(1, count) %>%

pull(exterior\_color)

data$exterior\_color[is.na(data$exterior\_color)] <- mode\_color

Variable transformation was another critical aspect. This involved normalizing numeric variables to a common scale and reducing the levels of categorical variables, like consolidating less common car models into an 'Other' category. These transformations were pivotal in reducing noise and enhancing the model's ability to learn from the data. <code>

reduce\_model\_name <- function(df) {

top\_models <- names(model\_name\_sorted)[1:50]

df$model\_name <- ifelse(df$model\_name %in% top\_models, df$model\_name, "Other")

return(df)

}

data <- reduce\_model\_name(data)

The impact of these transformations was multifold: they not only simplified the variable space but also improved the model's ability to discern patterns without being misled by outliers or skewed distributions. The end result was a dataset that was more homogenized and amenable to the complex algorithms that would soon be at play.

Feature Selection and Model Building With a large number of predictors, selecting the right features was crucial to avoid overfitting. I utilized Lasso regression, known for its efficacy in both variable selection and regularization. By imposing a penalty on the absolute size of the coefficients, Lasso effectively reduces the less important variables' coefficients to zero, thus performing feature selection while fitting the model. The process began with the creation of a subset of the data, randomly sampling 10,000 observations to ensure computational efficiency and manageability:

sample\_indices <- sample(1:nrow(data), 10000)

data\_sub <- data\_dummies[sample\_indices, ]

y\_sub <- y[sample\_indices]

The Lasso regression was then applied to this subset using the **glmnet** package in R, which conveniently combines model fitting and feature selection in a cross-validated framework:

lasso\_fit\_sub <- glmnet(data\_sub, y\_sub, alpha = 1)

cv\_fit\_sub <- cv.glmnet(data\_sub, y\_sub, alpha = 1)

The true value of this method was realized upon extracting the non-zero coefficients, which correspond to the variables that the Lasso model identified as significant:

coef\_vals <- coef(cv\_fit\_sub, s = cv\_fit\_sub$lambda.min)

non\_zero\_coef\_names <- rownames(coef\_vals)[coef\_vals[, 1] != 0]

By filtering out the variables with zero coefficients and focusing only on those with a substantial impact, I was able to ascertain which features were most predictive. Moreover, by cross-referencing these variables with the original categorical variables, a clear picture emerged of which factors were most crucial in predicting used car prices:

important\_original\_vars <- sapply(original\_vars, function(x) {

any(grepl(paste0("^", x, ".\*"), non\_zero\_coef\_names))

})

Next step, these variables were then carried forward into the Random Forest model.

The chosen model was a Random Forest, known for its robustness and ability to handle a mix of variable types. The model was fine-tuned through several iterations, adjusting parameters like the number of trees and depth of each tree to optimize performance.

# Random Forest model construction

rf\_model <- randomForest(price ~ ., data=sub\_data, ntree=1000, mtry=13, importance=TRUE)

（sub\_data是我通过遴选的可用于预测的最佳的变量集）

Model Evaluation and Iteration Post-modeling, the focus shifted to evaluating its predictive power and understanding the importance of different variables. This step was crucial in identifying areas for improvement. For instance, certain variables that were initially deemed important did not significantly impact the model, leading to their removal in subsequent iterations.

# Assessing variable importance

importance\_values <- importance(rf\_model)

sorted\_importance <- importance\_values[order(importance\_values[,"%IncMSE"], decreasing = TRUE), ]

Unfortunately, another very important piece of knowledge about how to find the best mtry comes after the deadline for submitting a model. But I still want to discuss the feasibility of this approach here.

To find the best mtry value for the Random Forest model using the caret package, you can define a tuning grid that includes a range of mtry values you want to test. The caret package will then use cross-validation to evaluate the performance of the model at each mtry value and determine which one results in the best model performance.

library(caret)

library(randomForest)

# Define the control using a cross-validation approach

train\_control <- trainControl(method="cv", number=5)

# Create a grid to search for the optimal mtry value

tuning\_grid <- expand.grid(mtry = 8:20)

# Set a seed for reproducibility

set.seed(1031)

# Train the Random Forest model

tuned\_model <- train(

price ~ ., # Use all other variables to predict price

data = sub\_data, # Dataset for training

method = "rf", # Use the 'randomForest' package

trControl = train\_control, # Defined train control with cross-validation

tuneGrid = tuning\_grid, # Grid of tuning parameters

ntree = 1000 # Number of trees to grow in the forest

)

# Extract the best mtry value

best\_mtry <- tuned\_model$bestTune$mtry

Furthermore, the process of model tuning was iterative. Initial results often led back to the data preparation stage, tweaking variable transformations and trying different feature subsets to improve accuracy.

Conclusion This project was a comprehensive exercise in predictive modeling, emphasizing the importance of a thorough understanding of the data, careful preparation, judicious feature selection, and iterative model tuning. The journey from raw data to a refined predictive model highlighted the nuanced and often non-linear path of data science projects. Each step, from data exploration to final model evaluation, contributed to a deeper understanding and more accurate prediction of used car prices.