Module 7 HW

Alanna Hazlett

2024-07-03

Module 7

> You can download the R Markdown file (https://gedeck.github.io/DS-6030/homework/Module-7.Rmd) and use it to answer the following questions.
>
> If not otherwise stated, use Tidyverse and Tidymodels for the assignments.

1. Predicting Prices of Used Cars (Regression Trees)

The dataset contains the data on used cars (Toyota Corolla) on sale during late summer of 2004 in the Netherlands. It has 1436 records containing details on 38 variables, including Price, Age, Kilometers, HP, and other specifications. The goal is to predict the price of a used Toyota Corolla based on its specifications.gzc

Load the data from `https://gedeck.github.io/DS-6030/datasets/homework/ToyotaCorolla.csv.gz`.

(a.) Load and preprocess the data. Convert all relevant variables to factors.

```
corolla <- read_csv("https://gedeck.github.io/DS-6030/datasets/homework/ToyotaCorolla.csv.gz")
```

```
## Rows: 1436 Columns: 39
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): Model, Fuel_Type, Color
## dbl (36): Id, Price, Age_08_04, Mfg_Month, Mfg_Year, KM, HP, Met_Color, Auto...
##
## 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.
```

```
corolla<-corolla %>%
   mutate(Model =factor(Model),
          Mfg_Year = factor(Mfg_Year),
          Fuel_Type = factor(Fuel_Type),
          Met_Color = factor(Met_Color),
          Color = factor(Color),
          Automatic = factor(Automatic),
          CC = factor(CC),
          Doors = factor(Doors),
          Cylinders = factor(Cylinders),
          Gears = factor(Gears),
          Mfr_Guarantee = factor(Mfr_Guarantee),
          BOVAG_Guarantee = factor(BOVAG_Guarantee),
          ABS = factor(ABS),
          Airbag_1 = factor(Airbag_1),
          Airbag_2 = factor(Airbag_2),
          Airco = factor(Airco),
```

```
          Automatic_airco = factor(Automatic_airco),
          Boardcomputer = factor(Boardcomputer),
          CD_Player = factor(CD_Player),
          Central_Lock = factor(Central_Lock),
          Powered_Windows = factor(Powered_Windows),
          Powered_Steering = factor(Power_Steering),
          Radio = factor(Radio),
          Mistlamps = factor(Mistlamps),
          Sport_Model = factor(Sport_Model),
          Backseat_Divider = factor(Backseat_Divider),
          Metallic_Rim = factor(Metallic_Rim),
          Radio_cassette = factor(Radio_cassette),
          Parking_Assistant = factor(Parking_Assistant),
          Tow_Bar = factor(Tow_Bar)) %>%
  dplyr::select(-c(Id))
```

(b.) Split the data into training (60%), and test (40%) datasets.

```
set.seed(123)
corolla_split <- initial_split(corolla, prop=0.6, strata=Price)
corolla_train <- training(corolla_split)
corolla_holdout <- testing(corolla_split)
dim(corolla_train)
```

```
## [1] 860  39
```

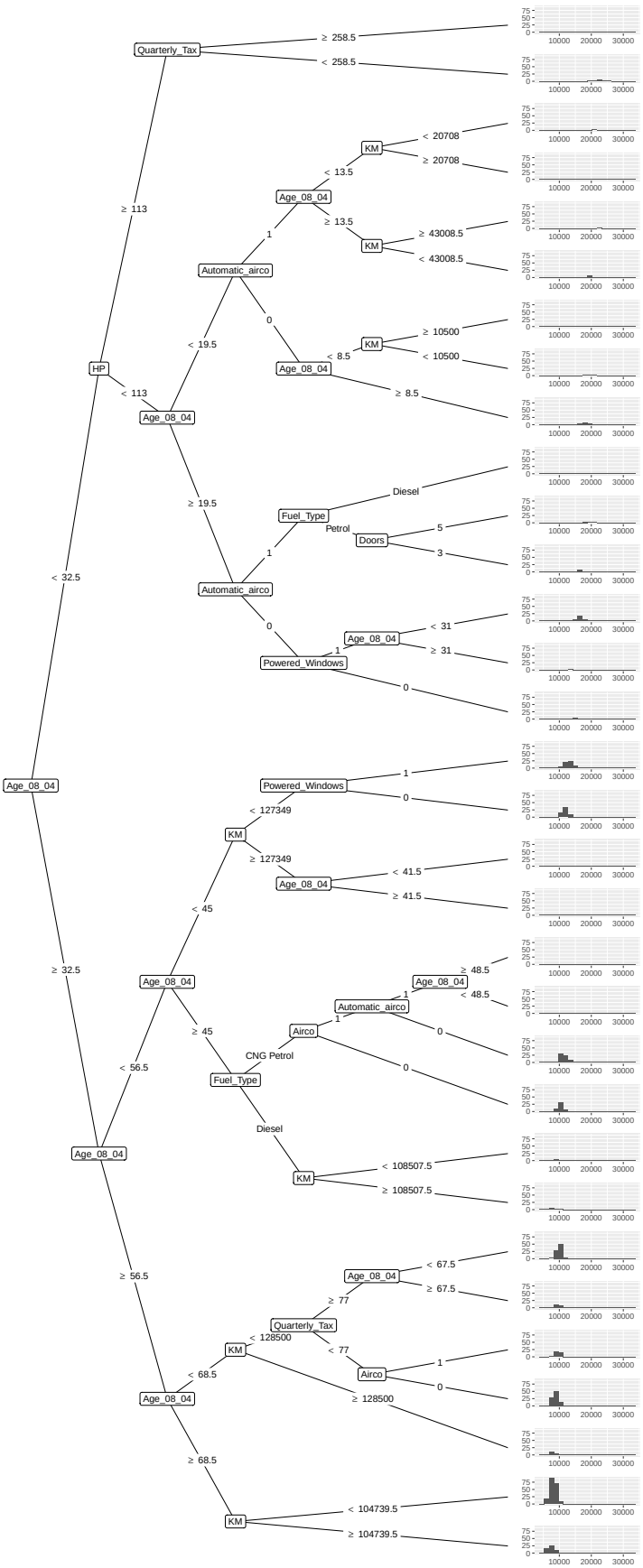
(c.) Define a workflow for a model to predict the outcome variable `Price` using the following predictors: `Age_08_04`, `KM`, `Fuel_Type`, `HP`, `Automatic`, `Doors`, `Quarterly_Tax`, `Mfr_Guarantee`, `Guarantee_Period`, `Airco`, `Automatic_airco`, `CD_Player`, `Powered_Windows`, `Sport_Model`, and `Tow_Bar`. Keep the minimum number of records in a terminal node to 1 (`min_n = 2`), maximum number of tree levels to 30 (`tree_depth`), and *cost_complexity* = 0.001 (`cost_complexity`), to make the run least restrictive.

```
corolla_formula<- Price ~ Age_08_04 + KM + Fuel_Type + HP + Automatic + Doors +
                          Quarterly_Tax + Mfr_Guarantee + Guarantee_Period + Airco +
                          Automatic_airco + CD_Player + Powered_Windows + Sport_Model +
                          Tow_Bar

corolla_rec <- recipe(corolla_formula, data=corolla_train)

corolla_spec <- decision_tree(min_n = 2,tree_depth = 30, cost_complexity = 0.001) %>%
                set_engine("rpart") %>%
                set_mode("regression")

corolla_wf <- workflow() %>%
              add_recipe(corolla_rec) %>%
              add_model(corolla_spec)
```

(i.) Fit a model using the full training dataset and visualize the resulting tree. Which appear to be the three or four most important car specifications for predicting the car's price?

```
# Fit model, because ggparty requires decision_tree and not workflow
corolla_model <- decision_tree(mode="regression", engine="rpart", min_n = 2,
                               tree_depth = 30, cost_complexity = 0.001) %>%
                               fit(corolla_formula, data=corolla_train)
# Visualize tree
#autoplot(partykit::as.party(corolla_model$fit))
```

```
ggparty::ggparty(partykit::as.party(corolla_model$fit), horizontal=TRUE) +
    ggparty::geom_edge() +
    ggparty::geom_edge_label() +
    ggparty::geom_node_label(aes(label=splitvar), ids="inner") +
    ggparty::geom_node_plot(gglist=list(geom_histogram(aes(x=Price), binwidth=1500),
                                        theme(axis.title.x = element_blank(),
                                              axis.title.y = element_blank())))
```


Age_08_04
< 32.5
≥ 32.5
HP
< 113
≥ 113
Quarterly_Tax
≥ 258.5
< 258.5
Age_08_04
< 19.5
≥ 19.5
Automatic_airco
1
0
Age_08_04
< 13.5
≥ 13.5
KM
< 20708
≥ 20708
KM
≥ 43008.5
< 43008.5
KM
≥ 10500
< 10500
Age_08_04
< 8.5
≥ 8.5
Fuel_Type
Diesel
Petrol
Doors
5
3
Automatic_airco
1
0
Age_08_04
< 31
≥ 31
Powered_Windows
1
0
Age_08_04
< 45
≥ 45
Powered_Windows
1
0
KM
< 127349
≥ 127349
Age_08_04
< 41.5
≥ 41.5
Age_08_04
< 56.5
≥ 56.5
Age_08_04
≥ 48.5
< 48.5
Automatic_airco
1
0
Airco
1
0
Fuel_Type
CNG Petrol
Diesel
KM
< 108507.5
≥ 108507.5
Age_08_04
< 68.5
≥ 68.5
Age_08_04
< 67.5
≥ 67.5
Quarterly_Tax
≥ 77
< 77
KM
< 128500
≥ 128500
Airco
1
0
KM
< 104739.5
≥ 104739.5
75
50
25
0
10000
20000
30000

From the decision tree diagram we can see that the first split is made based on the variable Age_08-04, so this appears to be the most important factor. This variable is also used for several other splits later on in the decision tree. We see following splits of HP and quarterly tax as well as Age_08_04 in the second and third splits indicating they are also in the top three most important factors. Determining the fourth most important predictor is more difficult as there are several other variables used at this level for determining splits.

(ii.) Determine the prediction errors of the training and test sets by examining their RMS error. How does the predictive performance of the test set compare to the training set? Why does this occur?

```
#Fitting best model and calculating metrics
custom_metric<-metric_set(rmse)
corolla_model_metrics<-bind_rows(
                              bind_cols(Dataset="DT Train",custom_metric(augment(corolla_model,corolla
                              bind_cols(Dataset="DT Test",custom_metric(augment(corolla_model,corolla_
corolla_model_metrics%>%
    knitr::kable(digits=3, caption="Metrics")
```

Table 1: Metrics

Dataset	.metric	.estimator	.estimate
DT Train	rmse	standard	994.103
DT Test	rmse	standard	1111.892

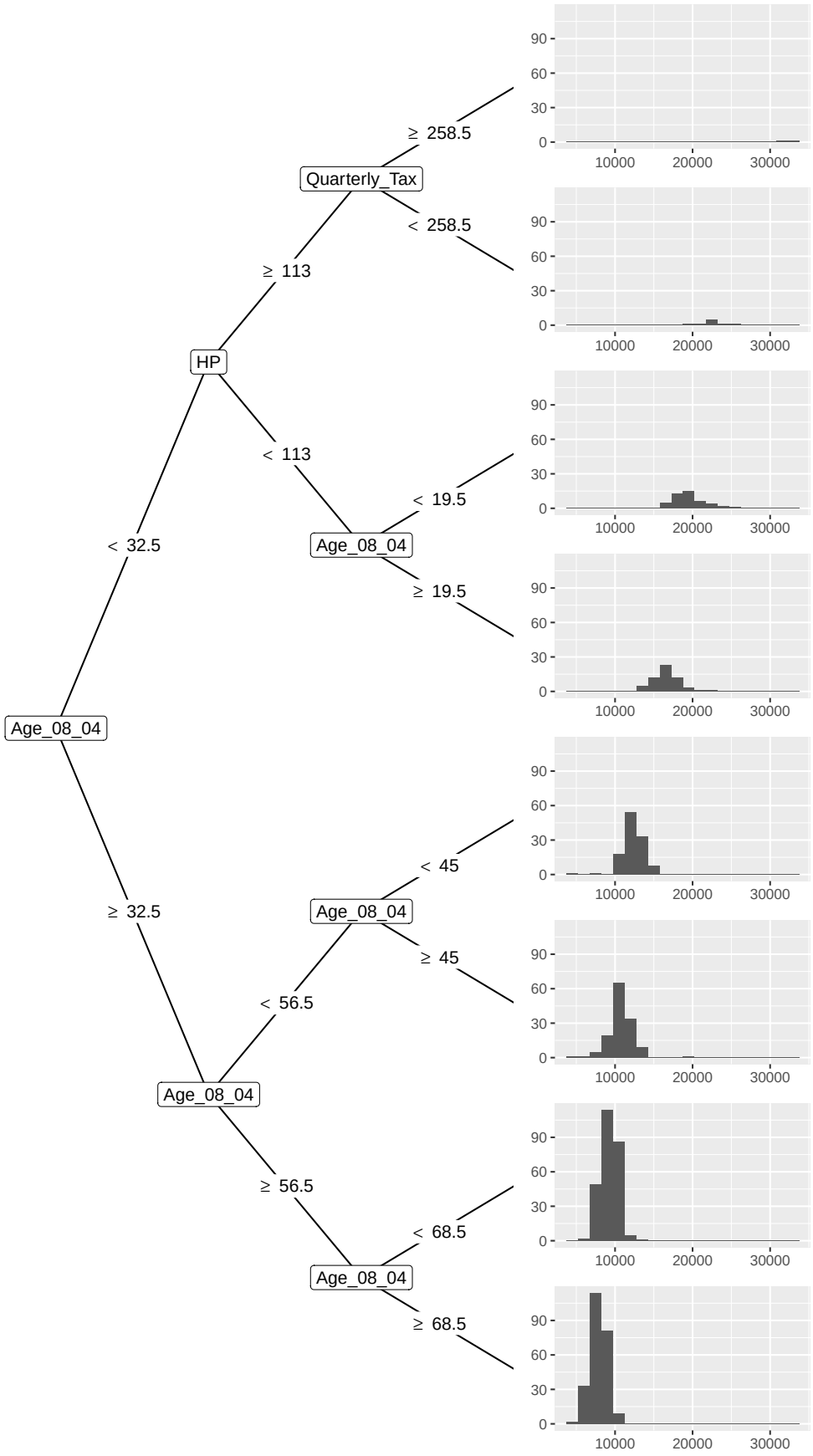
For Table 1: Based on the rmse we see a siginificantly worse value for the test set than we do for the training set. This is a sign of overfitting by the model. The decision tree we made in this problem is highly specified, meaning by allowing it to have a large value for max depth, low minimum number of observations at a terminal node, and the lower value of cost-complexity it made a more complex model. This is the equivalent to a very flexible model fitting training data very well, but not being as good with new test data.

(iv.) How might we achieve better test predictive performance at the expense of training performance?

We could acheive better test results by pruning this model. Having fewer splits could lead to lower variance. We could prune by utilizing cross-validation to find the best sub tree based on the smallest test error, although realistically this is too computationally/memory difficult. A more functional approach is to utilize cost-complexity pruning, where a value of 0 results in a full tree and a large value results in a small sub tree. Choosing the best value for cost-complexity can be found by using cross-validation. We then apply that value when we define or finalize the model.

(d.) Create a smaller tree by leaving the arguments at their default values; `cost_complexity=0.01`, `min_n=2`, and `tree_depth=30` at their defaults. Compared to the deeper tree, what is the predictive performance on the test set?

```
# Fit model
short_corolla_model <- decision_tree(mode="regression", engine="rpart", min_n = 2, tree_depth = 30, cos
                              fit(corolla_formula, data=corolla_train)
# Visualize tree
#autoplot(partykit::as.party(corolla_model$fit))
ggparty::ggparty(partykit::as.party(short_corolla_model$fit), horizontal=TRUE) +
    ggparty::geom_edge() +
    ggparty::geom_edge_label() +
    ggparty::geom_node_label(aes(label=splitvar), ids="inner") +
    ggparty::geom_node_plot(gglist=list(geom_histogram(aes(x=Price), binwidth=1500),
                                        theme(axis.title.x = element_blank(),
                                              axis.title.y = element_blank())))
```


Age_08_04
< 32.5
≥ 32.5
HP
≥ 113
< 113
Quarterly_Tax
≥ 258.5
< 258.5
Age_08_04
< 19.5
≥ 19.5
Age_08_04
< 56.5
≥ 56.5
Age_08_04
< 45
≥ 45
Age_08_04
< 68.5
≥ 68.5
90
60
30
0
10000
20000
30000

```
short_corolla_model_metrics<-bind_rows(
  bind_cols(Dataset="Short DT Train",custom_metric(augment(short_corolla_model,corolla_train),
                                                   truth=Price,estimate=.pred)),
  bind_cols(Dataset="Short DT Test",custom_metric(augment(short_corolla_model,corolla_holdout),
                                                   truth=Price,estimate=.pred)),
  bind_cols(Dataset="Large DT Train",custom_metric(augment(corolla_model,corolla_train),
                                                   truth=Price,estimate=.pred)),
  bind_cols(Dataset="Large DT Test",custom_metric(augment(corolla_model,corolla_holdout),
                                                   truth=Price,estimate=.pred)))
short_corolla_model_metrics%>%
    knitr::kable(digits=3, caption="Metrics")
```

Table 2: Metrics

Dataset	.metric	.estimator	.estimate
Short DT Train	rmse	standard	1344.487
Short DT Test	rmse	standard	1352.010
Large DT Train	rmse	standard	994.103
Large DT Test	rmse	standard	1111.892

For Table 2: While we saw that the difference in the rmse for training and test set for the large decision tree was large, we see that the difference for the training and test sets for the short tree is very small. We do however see that the large decision tree resulted in lower rmse values for both training and test compared to the short decision tree. This indicates that the short decision tree may not be a good model for this data.

(e.) Now define a workflow that tunes `cost_complexity` and leaves all other arguments at their default values. Define a suitable range for the `cost_complexity` parameter and use a tuning strategy of your choice. Make sure that the resulting best parameter is within the given range.

```
#Tuning decision tree for cost complexity
tune_spec <- decision_tree(cost_complexity=tune()) %>%
    set_engine("rpart") %>%
    set_mode("regression")

# Use recipe defined before based on previously defined formula
tune_wf <- workflow() %>%
    add_recipe(corolla_rec) %>%
    add_model(tune_spec)

#We can see the current range by calling parameters$object
corolla_parameters <- extract_parameter_set_dials(tune_wf)
corolla_parameters <- corolla_parameters %>%
    update(cost_complexity = cost_complexity(range=c(-5, -1)))
```

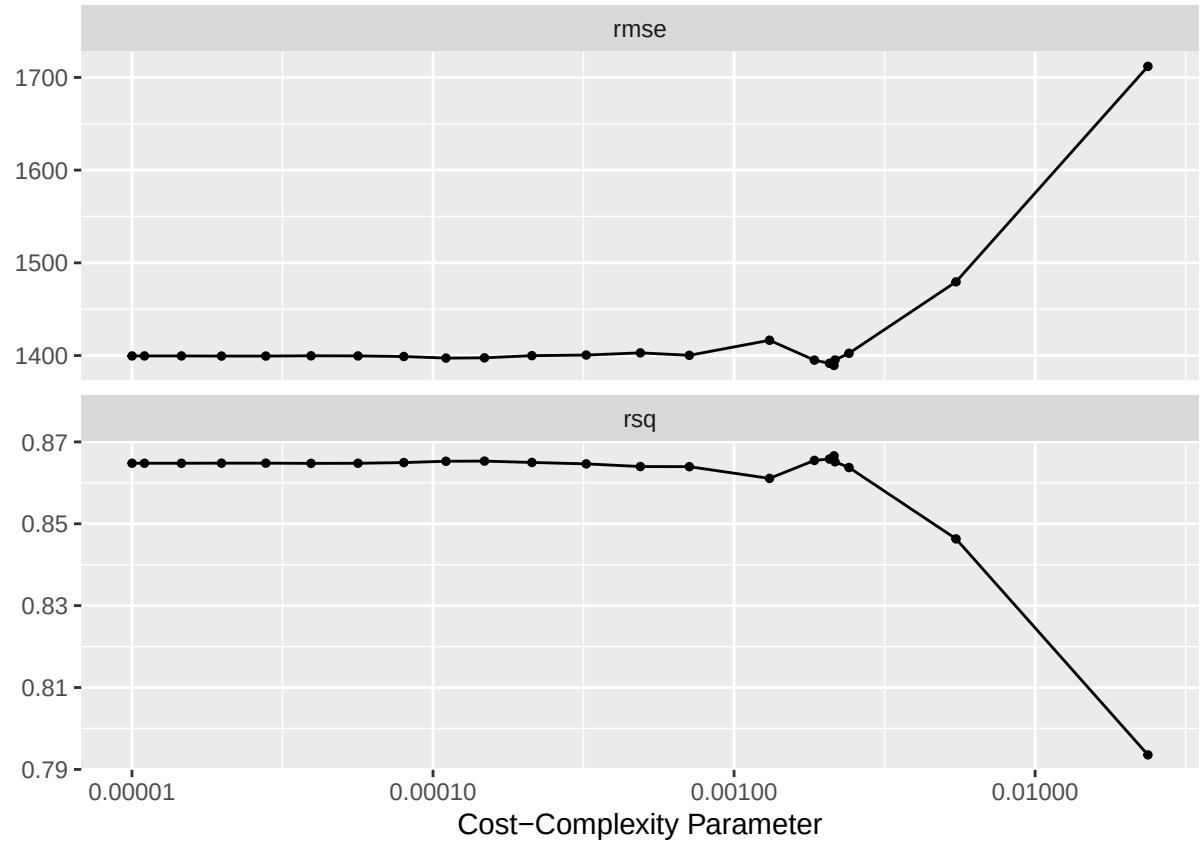
```
set.seed(123)
corolla_resamples <- vfold_cv(corolla_train, v=10, strata=Price)
cv_control <- control_resamples(save_pred=TRUE)
```

Cross-validation combined with tuning

```
#Get rid of scientific notation
options(scipen=999)
set.seed(123)
corolla_tree_tune <- tune_bayes(tune_wf, resamples=corolla_resamples, param_info=corolla_parameters, it
```

```
## ! No improvement for 10 iterations; returning current results.
```

```
autoplot(corolla_tree_tune)
```



(i.) What is the best value for `cost_complexity`? What is the predictive performance of the resulting model on the test set?

```
best_corolla_parameter <- select_best(corolla_tree_tune, metric="rmse")
best_corolla_parameter$cost_complexity
```

```
## [1] 0.002147668
```

```
tuned_corolla_model <- decision_tree(mode="regression", engine="rpart", min_n = 2, tree_depth = 30,
                                     cost_complexity = best_corolla_parameter$cost_complexity) %>%
                               fit(corolla_formula, data=corolla_train)

tuned_corolla_model_metrics<-bind_rows(
  bind_cols(Dataset="Tuned DT Train",custom_metric(augment(tuned_corolla_model,corolla_train),
                                                   truth=Price,estimate=.pred)),
  bind_cols(Dataset="Tuned DT Test",custom_metric(augment(tuned_corolla_model,corolla_holdout),
                                                  truth=Price,estimate=.pred)))
tuned_corolla_model_metrics%>%
    knitr::kable(digits=3, caption="Performance of Tune Model")
```

Table 3: Performance of Tune Model

Dataset	.metric	.estimator	.estimate
Tuned DT Train	rmse	standard	1119.199
Tuned DT Test	rmse	standard	1166.748

Dataset	.metric	.estimator	.estimate

(ii.) How does the predictive performance of the tuned model compare to the models from (c) and (d)? What do you observe?

```
all_model_metrics<-bind_rows(
  bind_cols(Dataset="Tuned DT Train",custom_metric(augment(tuned_corolla_model,corolla_train),
                                                  truth=Price,estimate=.pred)),
  bind_cols(Dataset="Tuned DT Test",custom_metric(augment(tuned_corolla_model,corolla_holdout),
                                                  truth=Price,estimate=.pred)),
  bind_cols(Dataset="Short DT Train",custom_metric(augment(short_corolla_model,corolla_train),
                                                  truth=Price,estimate=.pred)),
  bind_cols(Dataset="Short DT Test",custom_metric(augment(short_corolla_model,corolla_holdout),
                                                  truth=Price,estimate=.pred)),
  bind_cols(Dataset="Large DT Train",custom_metric(augment(corolla_model,corolla_train),
                                                  truth=Price,estimate=.pred)),
  bind_cols(Dataset="Large DT Test",custom_metric(augment(corolla_model,corolla_holdout),
                                                  truth=Price,estimate=.pred)))
all_model_metrics%>%
    knitr::kable(digits=3, caption="Metrics")
```

Table 4: Metrics

Dataset	.metric	.estimator	.estimate
Tuned DT Train	rmse	standard	1119.199
Tuned DT Test	rmse	standard	1166.748
Short DT Train	rmse	standard	1344.487
Short DT Test	rmse	standard	1352.010
Large DT Train	rmse	standard	994.103
Large DT Test	rmse	standard	1111.892

Table 4: The tuned model had a similar rmse for the test data to the large decision tree model. However, we know that the large tree was overfitted, making it a non-optimal model. The short decision tree resulted in worse rmse values all around.

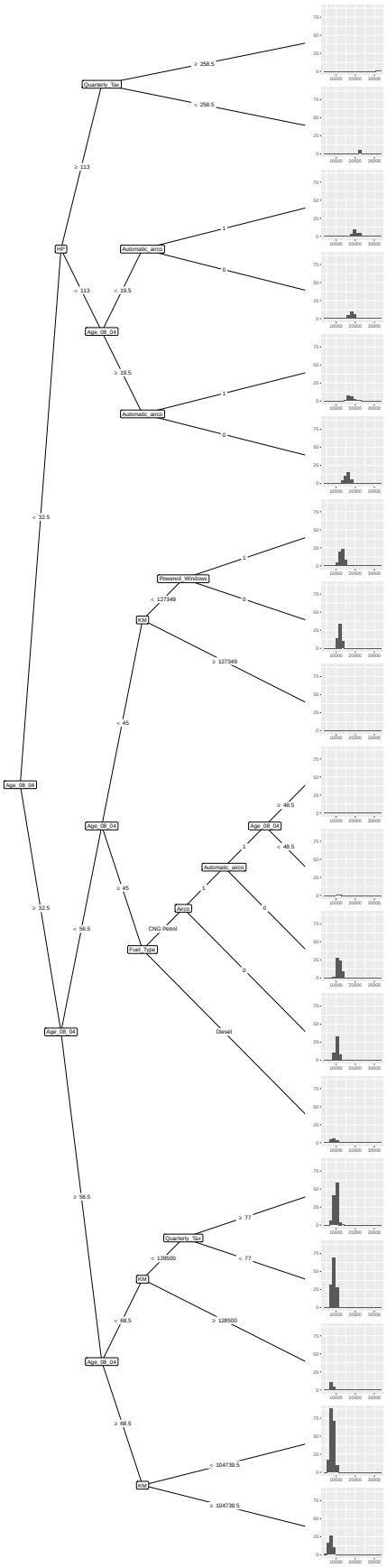
(iii.) How does the predictive performance of the tuned model compare to the model with the deeper tree?

The tuned model returns a similar rmse for the test data as the deeper tree. The difference is that with the tuned model we would expect relatively the same performance with other new data, whereas the deeper tree is less predictable or consintent in performance since it is overfitted; potentially if the deep tree model was provided a set of data similar to that of our training data set it would outperform compared to our tuned model, but the chances of this can be slim.

(iv.) Train a final model and visualize the resulting tree.

Creating visual for model with best tuning parameter

```
#Visual - Utilizing tuned_corolla_model previously defined
ggparty::ggparty(partykit::as.party(tuned_corolla_model$fit), horizontal=TRUE) +
    ggparty::geom_edge() +
    ggparty::geom_edge_label() +
    ggparty::geom_node_label(aes(label=splitvar), ids="inner") +
    ggparty::geom_node_plot(gglist=list(geom_histogram(aes(x=Price), binwidth=1500),
                                        theme(axis.title.x = element_blank(),
                                              axis.title.y = element_blank())))
```


Quarterly_Tax
≥ 258.5
< 258.5
≥ 113
HP
< 113
Automatic_airco
1
0
< 19.5
Age_08_04
≥ 19.5
Automatic_airco
1
0
< 32.5
Powered_Windows
1
0
< 127349
KM
≥ 127349
< 45
Age_08_04
≥ 48.5
Age_08_04
1
< 48.5
Automatic_airco
1
0
≥ 45
Airco
1
0
≥ 32.5
< 58.5
CNG Petrol
Fuel_type
Diesel
Age_08_04
≥ 58.5
≥ 77
Quarterly_Tax
< 128500
< 77
KM
< 68.5
≥ 128500
Age_08_04
≥ 68.5
< 104739.5
KM
≥ 104739.5

(f.) Given the various models, what is the predicted price for a car with the following characteristics (make sure to handle the categorical variables correctly):

> `Age_08_04=77`, `KM=117000`, `Fuel_Type=Petrol`, `HP=110`, `Automatic=No`, `Doors=5`, `Quarterly_Tax=100`, `Mfr_Guarantee=No`, `Guarantee_Period=3`, `Airco=Yes`, `Automatic_airco=No`, `CD_Player=No`, `Powered_Windows=No`, `Sport_Model=No`, `Tow_Bar=Yes`

Create new dataframe with variable values we want to base our prediction on:

```
prediction_values<-data.frame(Age_08_04<-c(77),
                              KM<-c(117000),
                              Fuel_Type<-c('Petrol'),
                              HP<-c(110),
                              Automatic<-c(0),
                              Doors<-c(5),
                              Quarterly_Tax<-c(100),
                              Mfr_Guarantee<-c(0),
                              Guarantee_Period<-c(3),
                              Airco<-c(1),
                              Automatic_airco<-c(0),
                              CD_Player<-c(0),
                              Powered_Windows<-c(0),
                              Sport_Model<-c(0),
                              Tow_Bar<-c(1),
                              stringsAsFactors=TRUE)
prediction_values<-prediction_values %>%
   mutate(Fuel_Type = factor(Fuel_Type),
          Automatic = factor(Automatic),
          Doors = factor(Doors),
          Mfr_Guarantee = factor(Mfr_Guarantee),
          Airco = factor(Airco),
          Automatic_airco = factor(Automatic_airco),
          CD_Player = factor(CD_Player),
          Powered_Windows = factor(Powered_Windows),
          Sport_Model = factor(Sport_Model),
          Tow_Bar = factor(Tow_Bar))
```

```
full_preds<-predict(corolla_model, prediction_values)
short_preds<-predict(short_corolla_model, prediction_values)
tuned_preds<-predict(tuned_corolla_model, prediction_values)
```

```
rbind(bind_cols(Model="Large Decision Tree", full_preds),
      bind_cols(Model="Short Decision Tree", short_preds),
      bind_cols(Model="Tuned Decision Tree", tuned_preds))%>%
    knitr::kable(digits=2, caption="Predictions")
```

Table 5: Predictions

Model	.pred
Large Decision Tree	7245.56
Short Decision Tree	7932.26
Tuned Decision Tree	7245.56