Lab 10: Regression Tress, Random Forest (Problem 37)

AUTHOR
Aaron Younger

Business Understanding

Jerry Stevenson is the manager of a travel agency. He wants to build a model that can predict customers' annual spending on travel products. A Regression Tree and the Ensemble Method random forest will be used for predicting the customer's annual household spending on travel products.

Data Understanding

R Version

```
options(scipen=999)
suppressWarnings(RNGversion("3.5.3"))
```

Libraries

```
library(readxl)
library(DataExplorer)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(dlookr)
library(e1071)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(pdp)
library(gridExtra)
```

```
library(readxl)
travel_data <- read_excel("jaggia_ba_2e_ch13_data.xlsx",
    sheet = "Travel_Data")
View(travel_data)

library(readxl)
travel_score <- read_excel("jaggia_ba_2e_ch13_data.xlsx",
    sheet = "Travel_Score")
View(travel_score)
```

Data Set Key:

- College: A binary variable representing if the individual went to College.
- Credit Card: A binary variable representing if the individual has a credit card.
- Food Spend: Annual household spending on Food.
- Income: The individuals annual income.
- Travel Spend: The annual household spending on travel products. The dependent variable.

Dataset Exploration

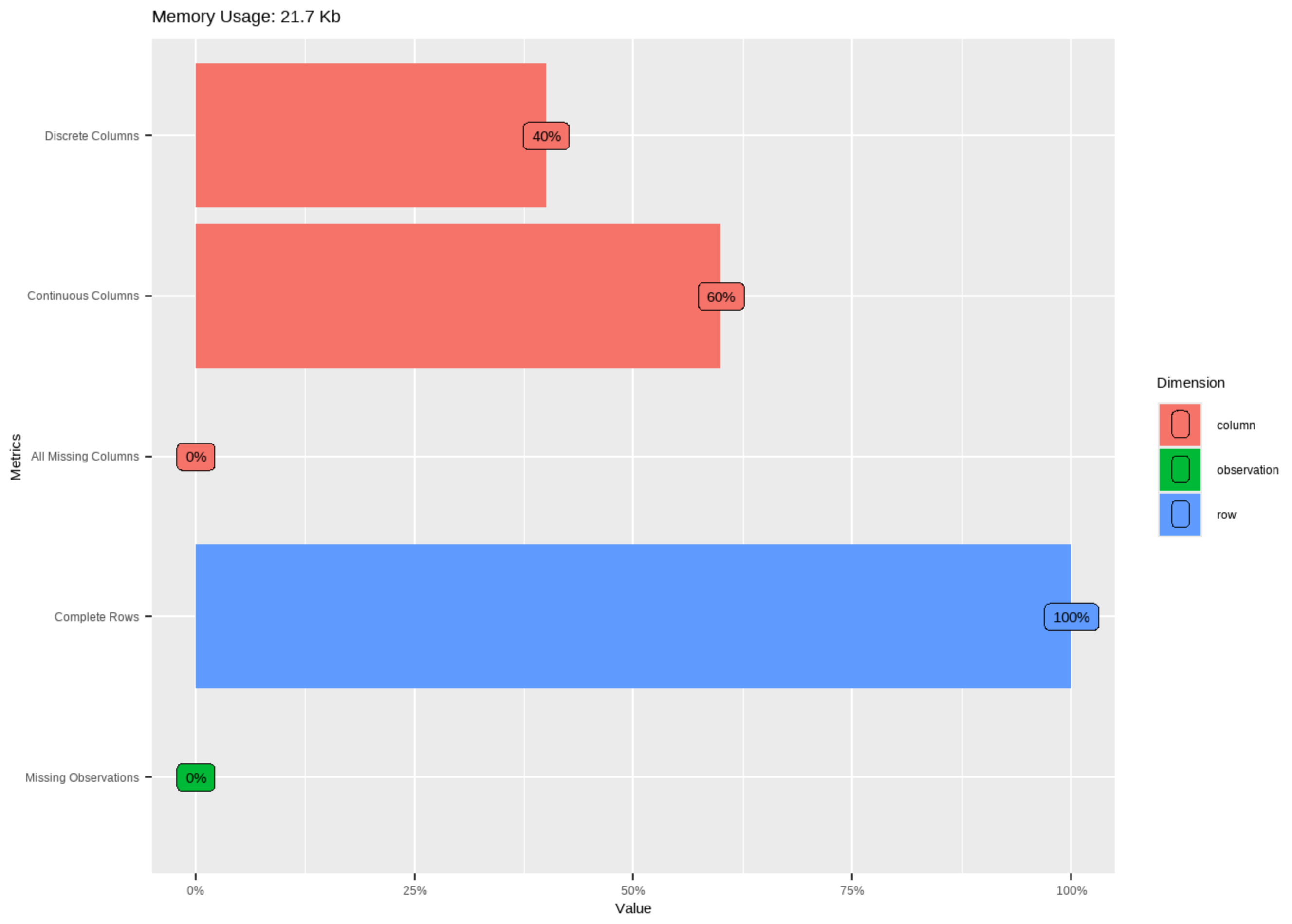
```r
travel_data %>% head()
```

```
# A tibble: 6 × 5
  College CreditCard FoodSpend Income TravelSpend
  <chr>   <chr>          <dbl>  <dbl>       <dbl>
1 Yes     Yes            5472.  49150        827.
2 No      Yes            9131.  47806        864.
3 No      Yes            4451.  46050       1180.
4 No      Yes            5021.  42600        756.
5 No      Yes            3408.  52092        851.
6 No      No             2850.  60248       1752.
```

```r
travel_data %>% tail()
```

```
# A tibble: 6 × 5
  College CreditCard FoodSpend Income TravelSpend
  <chr>   <chr>          <dbl>  <dbl>       <dbl>
1 No      No             4711.  57274       3448.
2 Yes     No             3787.  65982       2059.
3 No      No             6461.  57274       3853.
4 No      No             3586.  42250       2252.
5 Yes     No             1834.  60248       2000.
6 No      No             6206.  58520       3668.
```

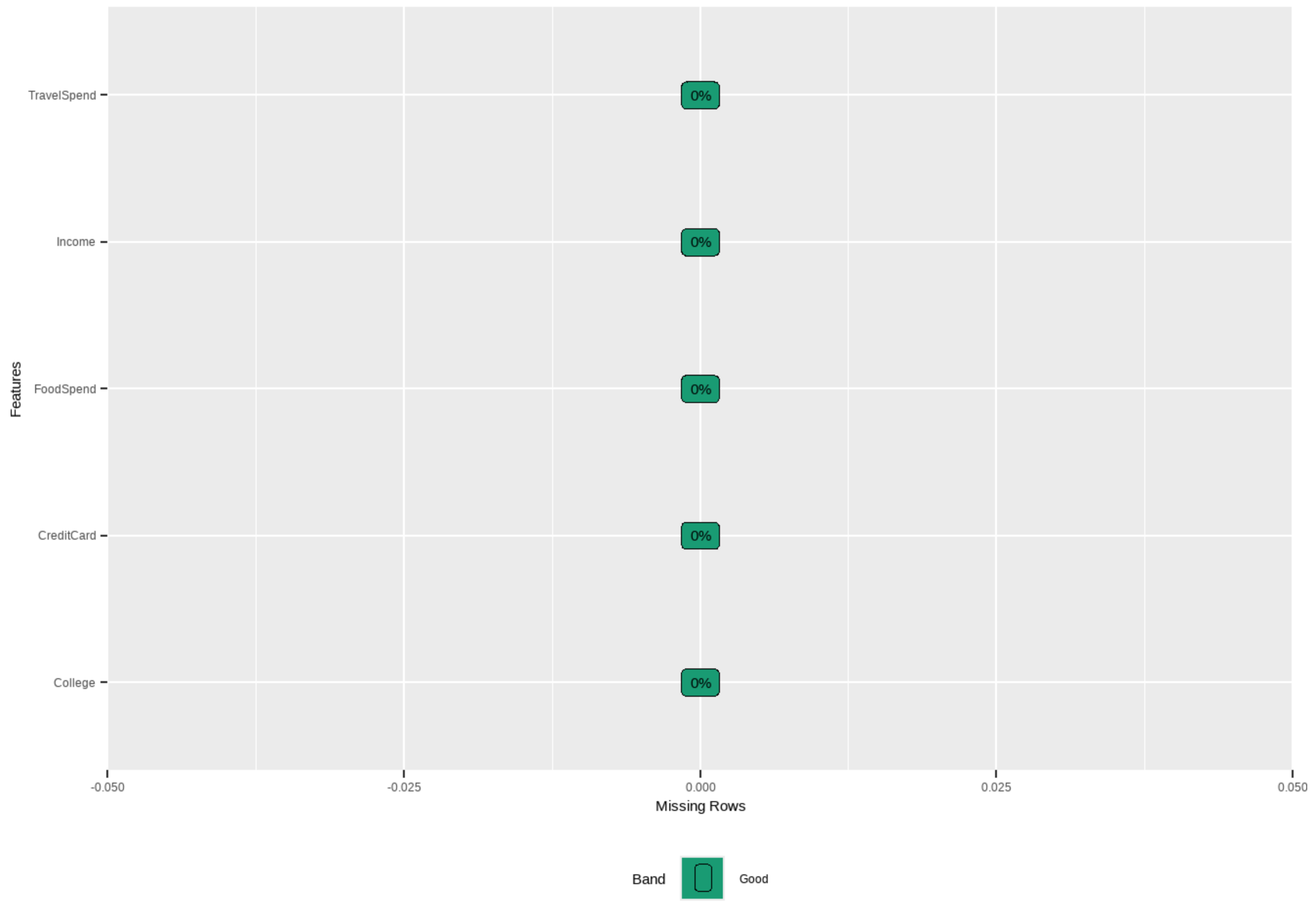
```r
travel_data %>% plot_intro()
```



```r
travel_data %>% glimpse()
```

```
Rows: 500
Columns: 5
$ College     <chr> "Yes", "No", "No", "No", "No", "No", "No", "No", "No", "No…
$ CreditCard  <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "No", "No", "Yes", "No"…
$ FoodSpend   <dbl> 5472.43, 9130.73, 4450.67, 5020.72, 3408.11, 2850.33, 3680…
$ Income      <dbl> 49150, 47806, 46050, 42600, 52092, 60248, 49412, 58520, 57…
$ TravelSpend <dbl> 827.40, 863.55, 1180.05, 755.70, 851.40, 1751.70, 4557.30,…
```

```r
travel_data %>% plot_missing()
```

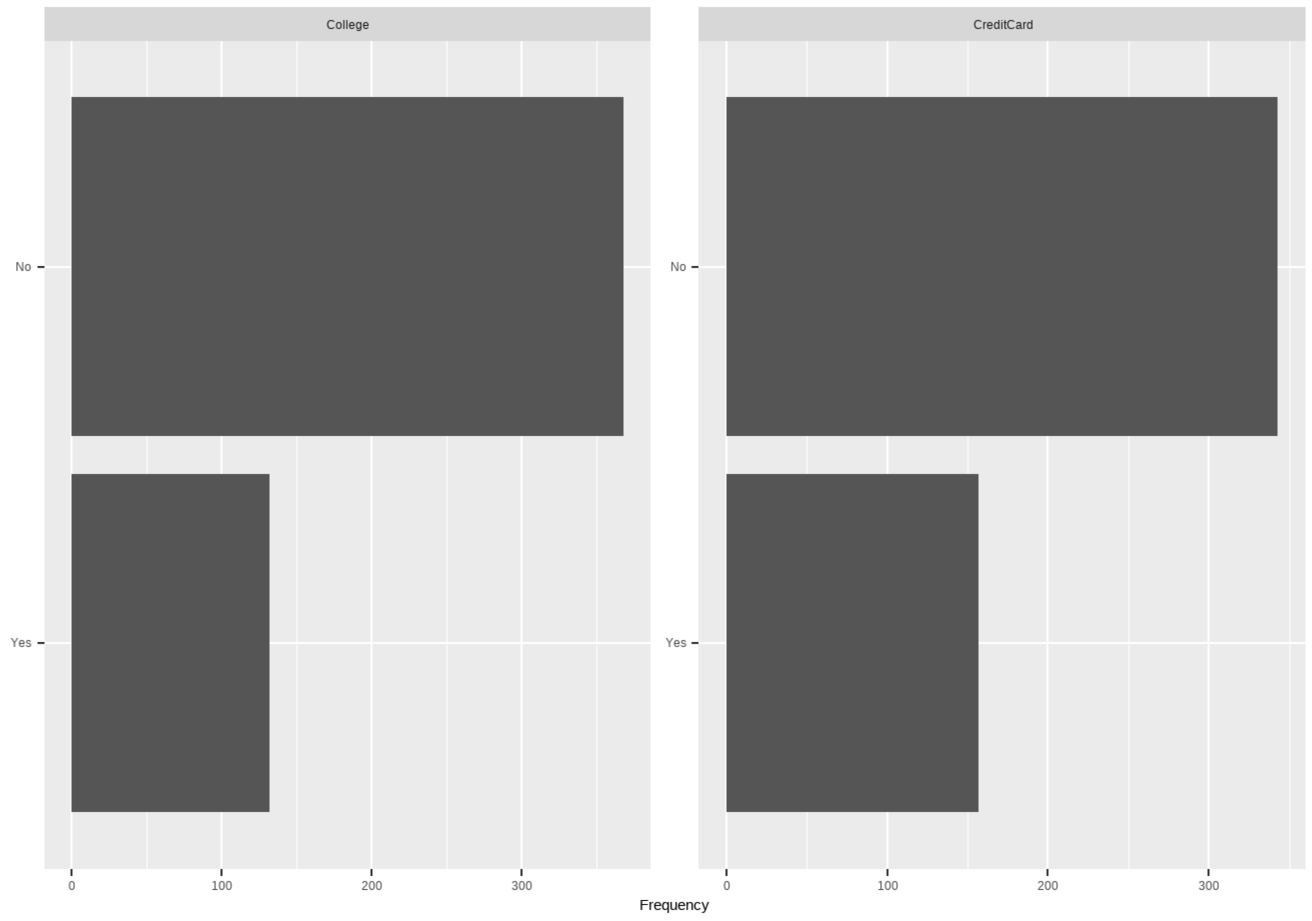


```
travel_data %>% str()
```

```
tibble [500 × 5] (S3: tbl_df/tbl/data.frame)
 $ College    : chr [1:500] "Yes" "No" "No" "No" ...
 $ CreditCard : chr [1:500] "Yes" "Yes" "Yes" "Yes" ...
 $ FoodSpend  : num [1:500] 5472 9131 4451 5021 3408 ...
 $ Income     : num [1:500] 49150 47806 46050 42600 52092 ...
 $ TravelSpend: num [1:500] 827 864 1180 756 851 ...
```

Variable EDA

```
travel_data %>% plot_bar()
```



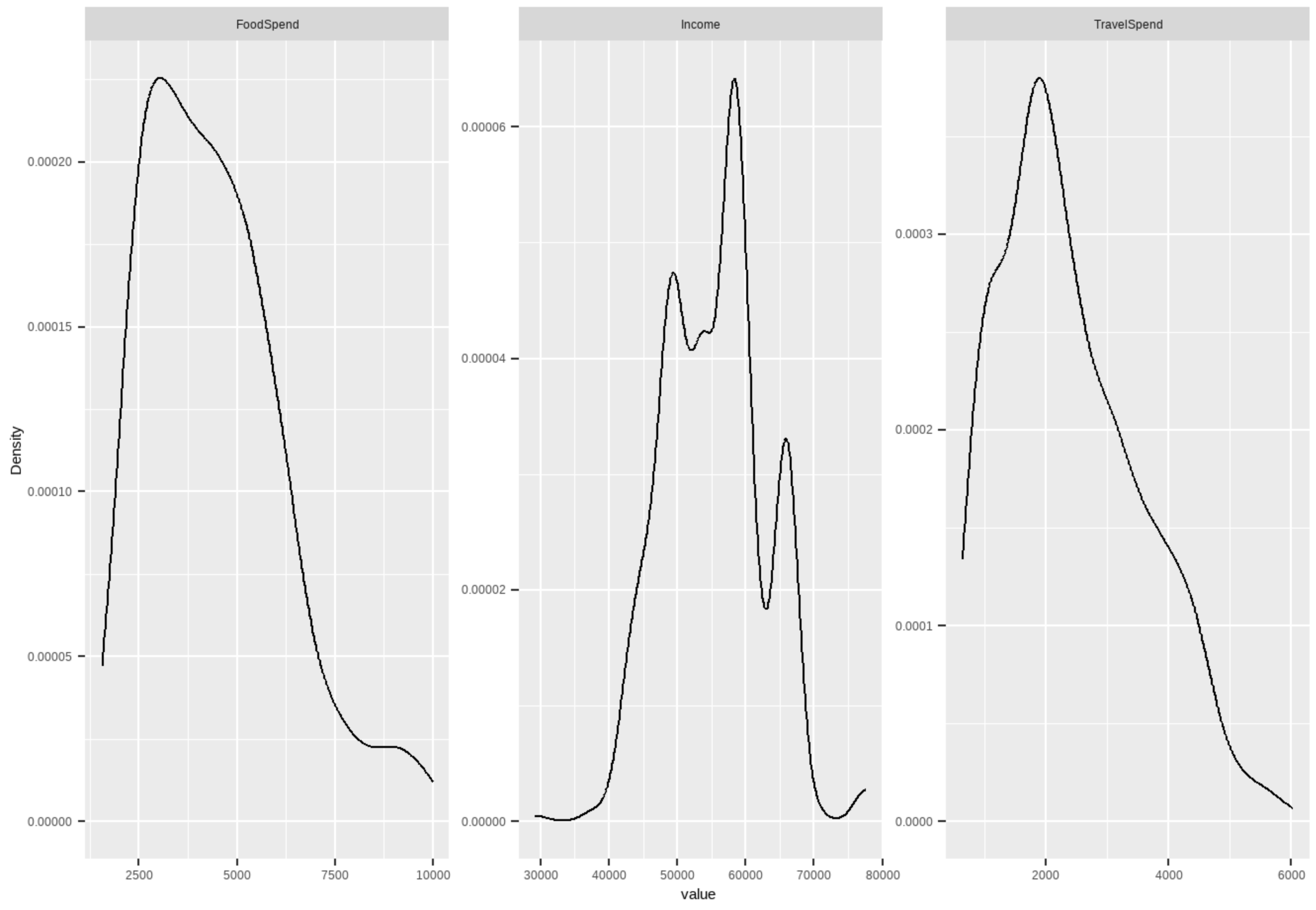
```r
prop.table(table(travel_data$College))
```

```
   No   Yes 
0.736 0.264 
```

```r
prop.table(table(travel_data$CreditCard)) ## Categorical Variables are disproportioned, dominant
         level is the "no" value for both variables.
```

```
   No   Yes 
0.686 0.314 
```

```r
travel_data %>% plot_density()
```



```r
skew_table <- data.frame(
  Variable = c("Food Spend", "Income", "Travel Spend"),
  Skewness = c(
    skewness(travel_data$FoodSpend),
    skewness(travel_data$Income),
    skewness(travel_data$TravelSpend)
  )
)

print(skew_table)
```

```
      Variable  Skewness
1   Food Spend 0.8343613
2       Income 0.1607550
3 Travel Spend 0.6157839
```

Comments on Variable EDA:
Both Categorical Variables are Binary and have observations in both levels. However, the levels are disproportionate with the dominant level being “no” for both College and Credit Card. The density plots for the numeric variables show that none of the variables are normally distributed. Food and Travel Spend show mild right skewness where most families spend lower on money on food and travel with a few families that spend higher amounts, pulling the distribution to the right. The Income variable shows some multiodality, which suggests possible subgroups within the data. The skewness values for these numeric variables are

positive and less than one, supporting right skewness but suggesting that these variables can be considered approximately normal.

Outliers

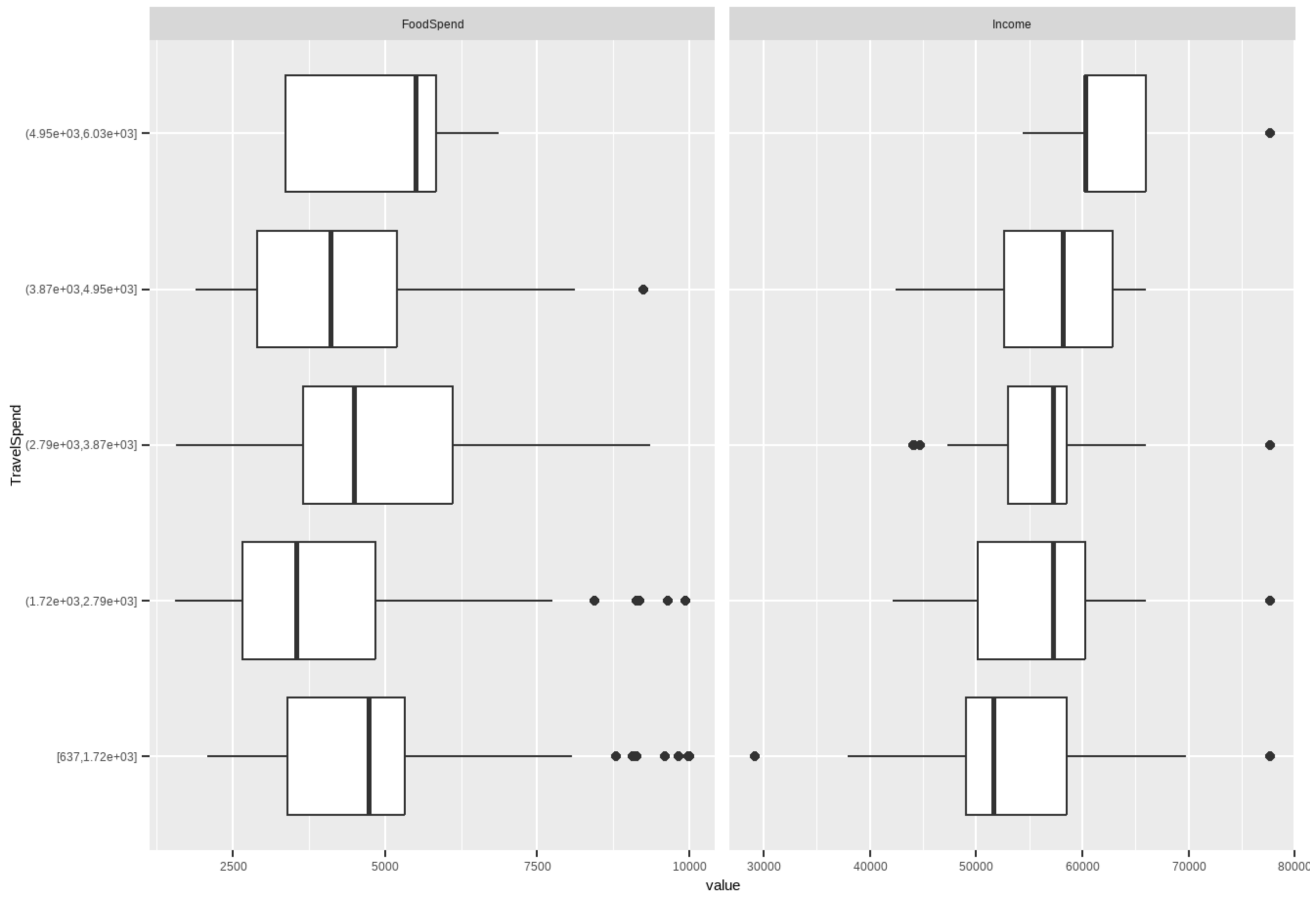
```
diagnose_outlier(travel_data)
```

```
# A tibble: 3 × 6
  variables   outliers_cnt outliers_ratio outliers_mean with_mean without_mean
  <chr>              <int>          <dbl>         <dbl>     <dbl>        <dbl>
1 FoodSpend             12            2.4         9516.     4420.        4295.
2 Income                 7            1.4        70708     55416.       55199.
3 TravelSpend            5            1           5679.     2402.        2369.
```

```
diagnose_outlier(travel_score)
```

```
# A tibble: 2 × 6
  variables outliers_cnt outliers_ratio outliers_mean with_mean without_mean
  <chr>            <int>          <dbl>         <dbl>     <dbl>        <dbl>
1 FoodSpend            0              0           NaN     4455.        4455.
2 Income               0              0           NaN    59984        59984
```

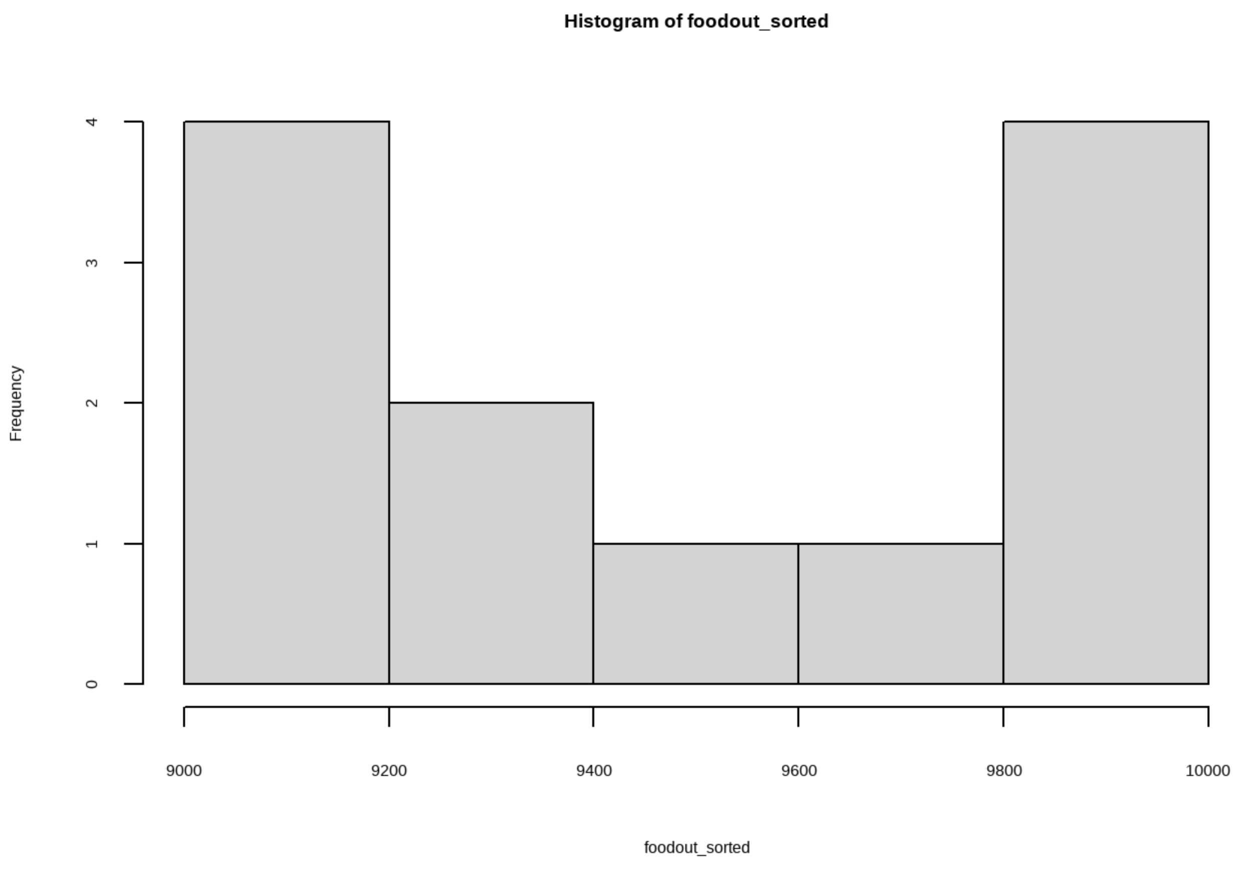
```
travel_data %>% plot_boxplot(by="TravelSpend")
```



```
foodout <- boxplot.stats(travel_data$FoodSpend)$out
foodout_sorted <- sort(foodout, decreasing = TRUE)
foodout_sorted
```

```
[1] 10000.00  9978.49  9935.07  9819.56  9649.01  9592.99  9350.13  9249.13
[9]  9185.28  9174.83  9130.73  9129.66
```

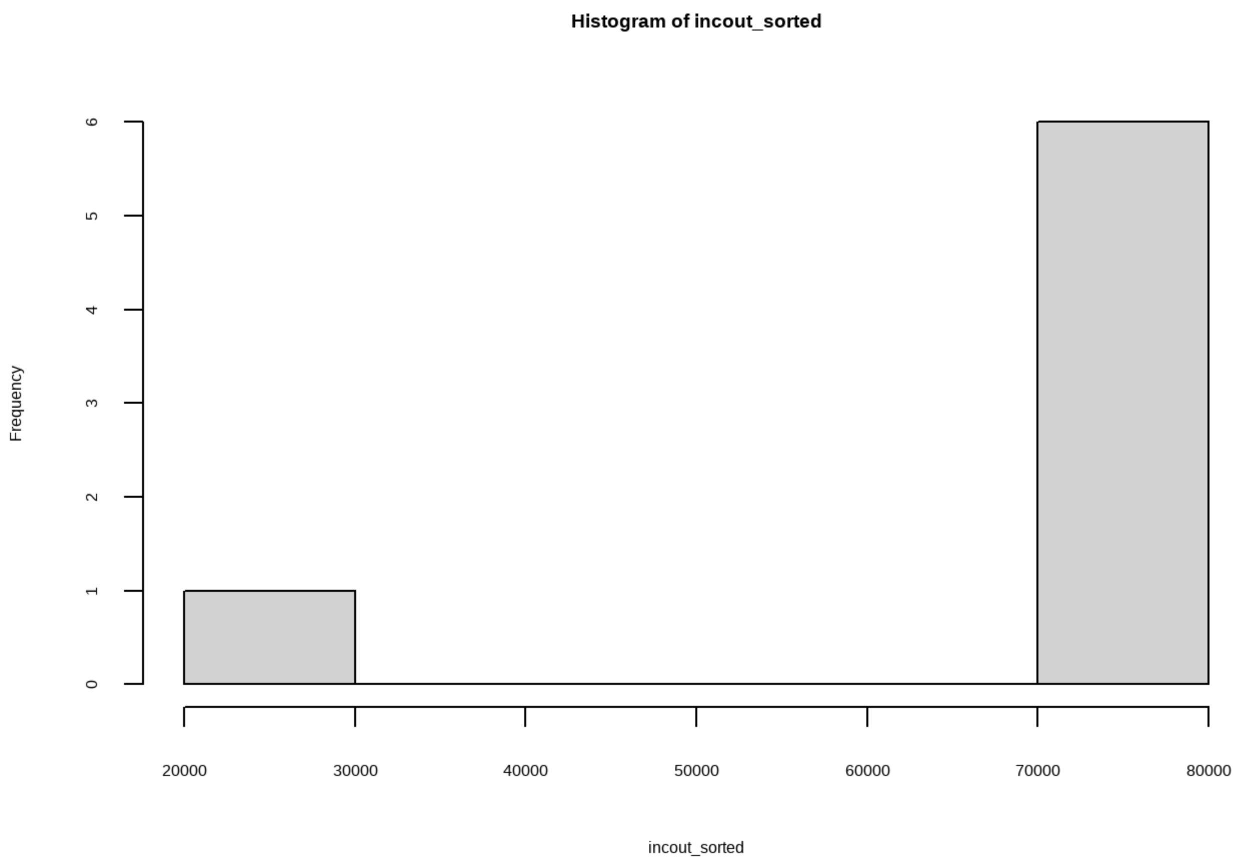
```
hist(foodout_sorted)
```



```r
incout <- boxplot.stats(travel_data$Income)$out
incout_sorted <- sort(incout, decreasing = TRUE)
incout_sorted
```

```
[1] 77626 77626 77626 77626 77626 77626 29200
```

```r
hist(incout_sorted)
```



Comments on Outliers:
There are no outliers in the score dataset. There are outliers in the modeling dataset but are not extreme enough to delete.

Mean of Dependent Variable

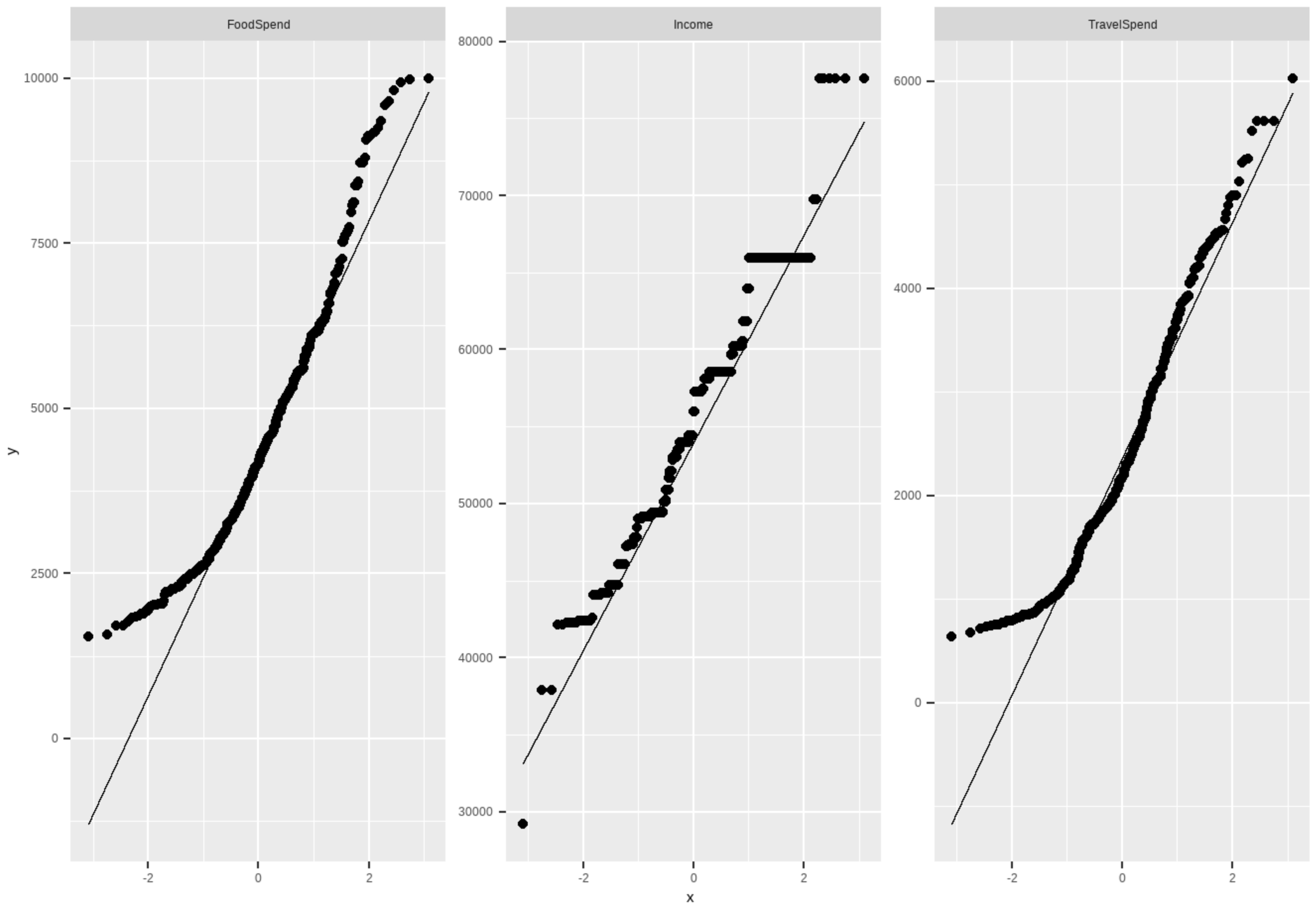
```
mean(travel_data$TravelSpend)
```

```
[1] 2402.287
```

The Average value of spending on travel products is 2402, this value will be used in comparison against the train and test sets mean value of the dependent variable.

Q-Q plot

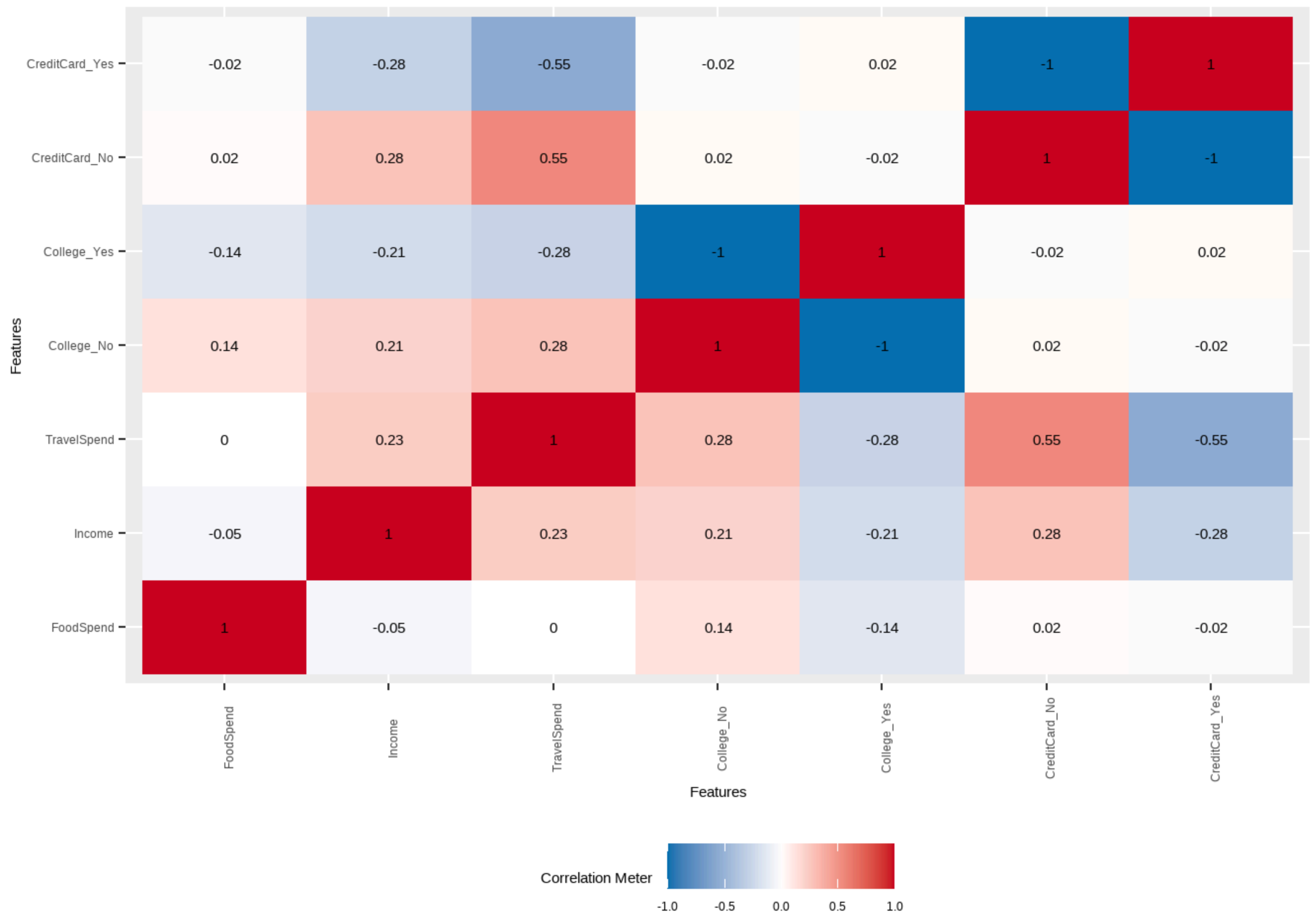
```
travel_data %>% plot_qq()
```



Comments on Q-Q plots:
The Q-Q plots of the numeric variables show that points follow closely along the diagonal reference line, with some deviations at the tail. This further supports that the variables are approximately normally distributed with mild right skewness.

Correlation Matrix

```
travel_data %>% plot_correlation()
```



	FoodSpend	Income	TravelSpend	College_No	College_Yes	CreditCard_No	CreditCard_Yes
CreditCard_Yes	-0.02	-0.28	-0.55	-0.02	0.02	-1	1
CreditCard_No	0.02	0.28	0.55	0.02	-0.02	1	-1
College_Yes	-0.14	-0.21	-0.28	-1	1	-0.02	0.02
College_No	0.14	0.21	0.28	1	-1	0.02	-0.02
TravelSpend	0	0.23	1	0.28	-0.28	0.55	-0.55
Income	-0.05	1	0.23	0.21	-0.21	0.28	-0.28
FoodSpend	1	-0.05	0	0.14	-0.14	0.02	-0.02

Comments on Correlation Matrix:
This correlation matrix shows some interesting relationships among the variables. It is interesting that not having a credit card would lead to higher travel spend and income then not having a credit card, this relationships seems counterintuitive similarly, although not necessarily counterintuitive but interesting to note that not going to college also shows higher spending and income then people that did go to college. Travel Spend, has weak positive correlation with travel spend, people that did not attend college, and people that do not have a credit card.

Data Preperation

Variable Transformation

```r
## Turn character variables into factors
travel_data$College <- as.factor(travel_data$College)
travel_data$CreditCard <- as.factor(travel_data$CreditCard)
travel_data %>% str()
```

```
tibble [500 × 5] (S3: tbl_df/tbl/data.frame)
 $ College    : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 1 1 1 1 ...
 $ CreditCard : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 1 2 1 1 ...
 $ FoodSpend  : num [1:500] 5472 9131 4451 5021 3408 ...
 $ Income     : num [1:500] 49150 47806 46050 42600 52092 ...
 $ TravelSpend: num [1:500] 827 864 1180 756 851 ...
```

Partition

```
set.seed(1)
myindex <- createDataPartition(travel_data$TravelSpend, p = 0.7, list = FALSE)
trainset <- travel_data[myindex,]
testset <- travel_data[-myindex,]

cat("Mean of the depvar Trainset")
```

```
Mean of the depvar Trainset
```

```
mean(trainset$TravelSpend)
```

```
[1] 2395.475
```

```
cat("Mean of the depvar Testset")
```

```
Mean of the depvar Testset
```

```
mean(testset$TravelSpend)
```

```
[1] 2418.489
```

Comments on Data Partition:
The modeling dataset was partitioned into a 70/30 split, 70% of the data will be used to train the model and 30% of the dataset will be used to test the model. The means of travel spend for the train and test dataset were both close which means both have similar distributions of the dependent variable.

Modeling (Regression Trees)

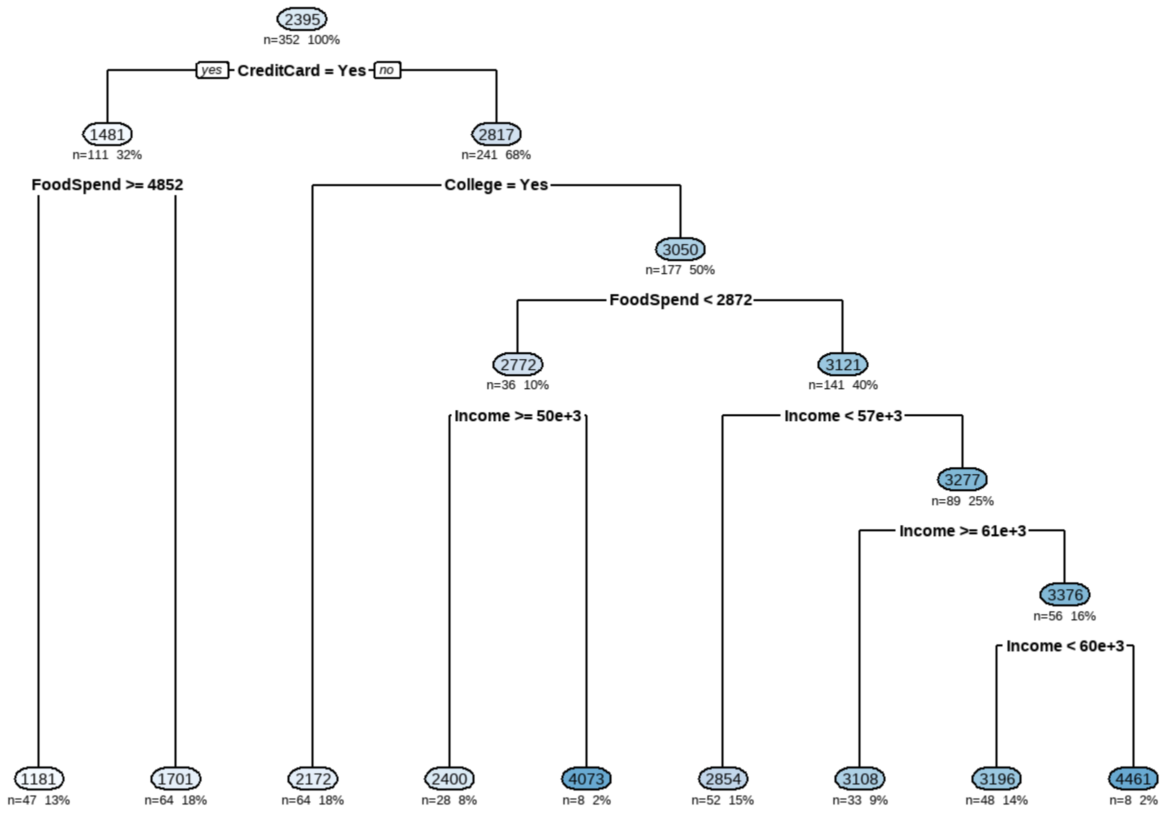
Cross Validation

```
myctrl <- trainControl(method = "cv", number = 10)
```

Default Tree (rpart)

```
## Default tree using cp = 0.01
set.seed(1)
```

```
tree_default <- rpart(TravelSpend~., data = trainset, method="anova")
rpart.plot(tree_default, type = 2, extra = 101, under = TRUE, fallen.leaves = TRUE)
```



```
## Find number of leaf nodes in the default tree
cat("Number of Leaf Nodes in the Default Tree: ")
```

```
Number of Leaf Nodes in the Default Tree:
```

```
sum(tree_default$frame$var == "<leaf>")
```

```
[1] 9
```

Answer to Question 37 part A:
A) How many leaf nodes are in the default tree? What are the predictor variable and split value for the first split of the default regression tree?

- How many Leaf Nodes are in the default tree: There are 9 leaf nodes in the default tree.
- The predictor variable for the first split of the default regression tree is credit card. The split values are when credit card = yes the predicted mean Travel Spend is 1481, and if credit card = no then the predicted mean income is 2817.

Full Grown Tree (rpart)

```
set.seed(1)
full_tree <- rpart(TravelSpend ~., data = trainset, control = rpart.control(cp = 0, minsplit =
         2, minbucket = 1, maxdepth = 30, xval = 10))

#xval = 10 is cross fold validation of 10, this is how to do it using rpart, used rpart to get
         correct values in the cp table

head(full_tree$cptable)
```

```
          CP nsplit rel error    xerror       xstd
1 0.30174408      0 1.0000000 1.0117564 0.06854470
2 0.08049551      1 0.6982559 0.7058791 0.05010272
3 0.01631113      2 0.6177604 0.6236386 0.04619739
4 0.01438172      3 0.6014493 0.6287099 0.04812099
5 0.01437510      8 0.5295407 0.6231581 0.04815957
6 0.01241362     10 0.5007905 0.6349426 0.05110058
```

```
## Where to Find Answers to Part B

min_error <- full_tree$cptable[which.min(full_tree$cptable[,"xerror"]),]
min_error
```

```
        CP     nsplit  rel error     xerror       xstd 
0.01437510 8.00000000 0.52954066 0.62315809 0.04815957 
```

Answer to Question 37 Part B:
B) Build a full-grown tree. Which cp value is associated with the lowest cross-validation error? How many splits are in the minimum-error tree? What is the minimum cross-validation error?

- CP value associated with the lowest cross-validation error: CP = 0.01437510.
- Number of splits that are in the minimum-error tree: 8.
- Minimum cross validation error: xerror = 0.62315809.

Best Pruned Tree (rpart)

```
## To find the best pruned tree, you take the minimum error found in the minimum error tree and
         add it to the minimum errors associated xstd value. These two values added together
         makes the best pruned trees threshold. Now we look for the lowest number of splits
         within the xerror threshold.

cpt <- full_tree$cptable

imin      <- which.min(cpt[,"xerror"]) ## Grabs the row that the minimum error is on and defines
         it with a variable
xerr_min  <- cpt[imin,"xerror"] ## Gets the xerror from row 5
xstd_min  <- cpt[imin,"xstd"] ## gets the xstd from row 5
```

```
threshold <- xerr_min + xstd_min ## Creates the xerror threshold for best pruned tree.
threshold ## lowest amount of splits while staying below an xerror value of 0.6713177
```

```
[1] 0.6713177
```

```
within_threshold <- which(cpt[,"xerror"] <= threshold)

best_pruned_index <- within_threshold[which.min(cpt[within_threshold, "nsplit"])]

best_pruned_cp <- cpt[best_pruned_index, "CP"]
best_pruned_xerror <- cpt[best_pruned_index, "xerror"]
best_pruned_nsplits <- cpt[best_pruned_index, "nsplit"]

cat("Best-Pruned Tree (1-SE Rule)\n")
```

```
Best-Pruned Tree (1-SE Rule)
```

```
cat("CP value:", best_pruned_cp, "\n")
```

```
CP value: 0.01631113
```

```
cat("Cross-validation error:", best_pruned_xerror, "\n")
```

```
Cross-validation error: 0.6236386
```

```
cat("Number of splits:", best_pruned_nsplits, "\n")
```

```
Number of splits: 2
```

Answer to Question 37 Part C:
c. Is there a simpler tree with a cross-validation error that is within one standard error of the minimum error? If there is, then which cp value is associated with the best-pruned tree?

Yes, there is a simpler tree with a cross-validation error within one standard error of the minimum error. The cp value associated with the best-pruned tree is 0.01631113.

Best tree (rpart)

Make Model

```
best_pruned_tree <- prune(full_tree, cp=0.01631113)
```

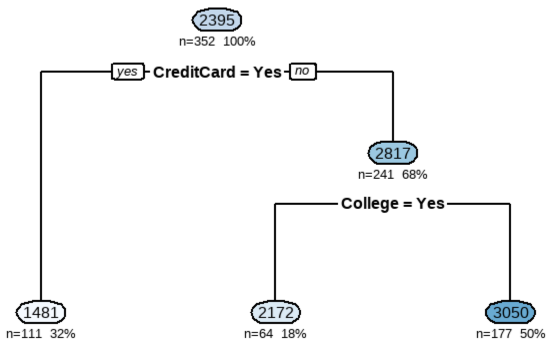
Variable Importance

```
caret::varImp(best_pruned_tree)
```

```
              Overall
College    0.20846187
CreditCard 0.30174408
FoodSpend  0.03065334
Income     0.06040484
```

Comments on Variable Importance:
Credit Card and College are the most important to the model. In the best pruned tree it can be expected that these variables will be predictor variables for splits.

```
rpart.plot(best_pruned_tree, type = 2, extra = 101, under = TRUE, fallen.leaves = TRUE)
```



```
cat("Number of Leaf Nodes in the Pruned Tree: ")
```

```
Number of Leaf Nodes in the Pruned Tree:
```

```
sum(best_pruned_tree$frame$var == "<leaf>")
```

```
[1] 3
```

Answer to Question 37 Part D:
D. Prune the full tree to the best-pruned tree or the minimum error tree if the answer to part c is “No.”

Display the tree. How many leaf nodes are in the pruned tree?

There are three leaf nodoes in the pruned tree.

Comments on Best Pruned Tree:
The Best pruned tree has Two Splits, The first split is whether or not the individual has a credit card, if the individual has a credit card then his annual travel spending is 1481. If the individual does not have a credit card then the annual travel spending is 2817. The second split is over whether an individual went to college or not, this split happens only if the person does not have a credit card. If the person did go to college the annual travel spending is 2172 whereas if the person did not go to college the annual travel spending is 3050.

Evaluation (rpart)

Evaluation Metrics

```r
predict_bpt <- predict(best_pruned_tree, testset)
bpt_metrics <- round(forecast::accuracy(predict_bpt, testset$TravelSpend),2)

## For inline code
ME_value   <- bpt_metrics["Test set", "ME"]
RMSE_value <- bpt_metrics["Test set", "RMSE"]
MAE_value  <- bpt_metrics["Test set", "MAE"]
MPE_value  <- bpt_metrics["Test set", "MPE"]
MAPE_value <- bpt_metrics["Test set", "MAPE"]

rownames(bpt_metrics) <- "Rpart Regression Tree"
print(bpt_metrics)
```

```
                         ME   RMSE    MAE    MPE  MAPE
Rpart Regression Tree -1.35 894.68 713.39 -14.75 34.74
```

Answer to Question 37 Part E:
E) What are the ME, RMSE, MAE, MPE, and MAPE of the pruned tree on the validation data?

- **ME Value:** ‘r ME_value’.
- **RMSE Value:** ‘r RMSE_value’.
- **MAE Value:** ‘r MAE_value’.
- **MPE Value:** ‘r MPE_value’%.
- **MAPE Value:** ‘r MAPE_value’%.

Measures of Accuracy:

- **RMSE (Root Mean Square Error = 894.68)** and **MAE (Mean Absolute Error = 713.39)** show that the average prediction error of the pruned tree is roughly between **$700–$900** units from the actual value.

- **MAPE (Mean Absolute Percentage Error = 34.74%)** indicates that on average, predictions are off by about **35%** from the true travel spend values.
- Overall, the model demonstrates **moderate predictive accuracy** — it captures general trends but still has some variance in prediction magnitudes.

Measures of Bias:

- **ME (Mean Error = -1.35)** is close to zero, suggesting that the model is **unbiased** — it does not consistently over- or under-predict.
- **MPE (Mean Percentage Error = -14.75%)** is slightly negative, meaning the model **tends to slightly under-predict** actual values on average.

Score (rpart)

```
score_rpart <- predict(best_pruned_tree, travel_score)
rpart_pip <- cbind(travel_score, score_rpart)
rpart_pip
```

```
  College CreditCard FoodSpend Income score_rpart
1      No        Yes   2892.90  65982    1480.546
2     Yes         No   6017.66  53986    2172.309
```

Best Tree (caret)

```
set.seed(1)
best_tree <- train(TravelSpend ~.,
                   data = trainset,
                   method = "rpart",
                   trControl = myctrl,
                   tuneLength = 25,
                   metric = "RMSE",
                  control = rpart::rpart.control(minsplit = 2, minbucket = 1, cp = 0))

best_tree$resample %>%
  arrange(RMSE)
```

```
       RMSE  Rsquared      MAE Resample
1  675.5641 0.6209010 513.9024   Fold08
2  783.7385 0.4410429 616.2088   Fold05
3  797.4464 0.5025473 562.8668   Fold10
4  864.7713 0.3207340 652.3549   Fold04
5  869.5082 0.4124079 691.6714   Fold02
6  908.2189 0.3394698 703.6228   Fold06
7  918.3013 0.4746556 688.2106   Fold09
8  973.0953 0.2522752 790.9866   Fold01
```

```
9  1015.4360 0.3027039 797.4441   Fold07
10 1025.5792 0.3539480 830.3901   Fold03
```

```
best_tree$finalModel$cptable
```

```
          CP nsplit rel error
1 0.30174408      0 1.0000000
2 0.08049551      1 0.6982559
3 0.01631113      2 0.6177604
4 0.01438172      3 0.6014493
5 0.01437510      8 0.5295407
6 0.01257267     10 0.5007905
```

CP Values

```
best_tree$results
```

```
           cp      RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
1  0.00000000  979.8144 0.3940203 609.5450 143.29946 0.12610666 109.18476
2  0.01257267  883.1659 0.4020686 684.7659 109.83998 0.11057390 102.95901
3  0.02514534  890.0210 0.3830431 717.3401 119.41866 0.13487652 105.91385
4  0.03771801  884.5279 0.3908443 715.2211 121.28057 0.13777580 105.54171
5  0.05029068  884.5279 0.3908443 715.2211 121.28057 0.13777580 105.54171
6  0.06286335  884.5279 0.3908443 715.2211 121.28057 0.13777580 105.54171
7  0.07543602  924.0716 0.3313033 761.2885  98.93832 0.12190218  89.51029
8  0.08800869  943.3660 0.3066676 778.3473 114.72822 0.11965967 102.89155
9  0.10058136  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
10 0.11315403  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
11 0.12572670  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
12 0.13829937  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
13 0.15087204  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
14 0.16344471  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
15 0.17601738  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
16 0.18859005  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
17 0.20116272  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
18 0.21373539  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
19 0.22630806  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
20 0.23888073  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
21 0.25145340  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
22 0.26402607  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
23 0.27659874  939.4593 0.3102737 783.7894 116.88836 0.11715062 100.17559
24 0.28917141  995.3909 0.2756474 830.0653 121.22026 0.10387564 102.28537
25 0.30174408 1042.4628 0.2371829 870.9928  89.14339 0.08918501  79.12694
```

```
cat("\nBest Tuned cp Value: ", best_tree$bestTune$cp)
```

```
Best Tuned cp Value:  0.01257267
```

Variable Importance

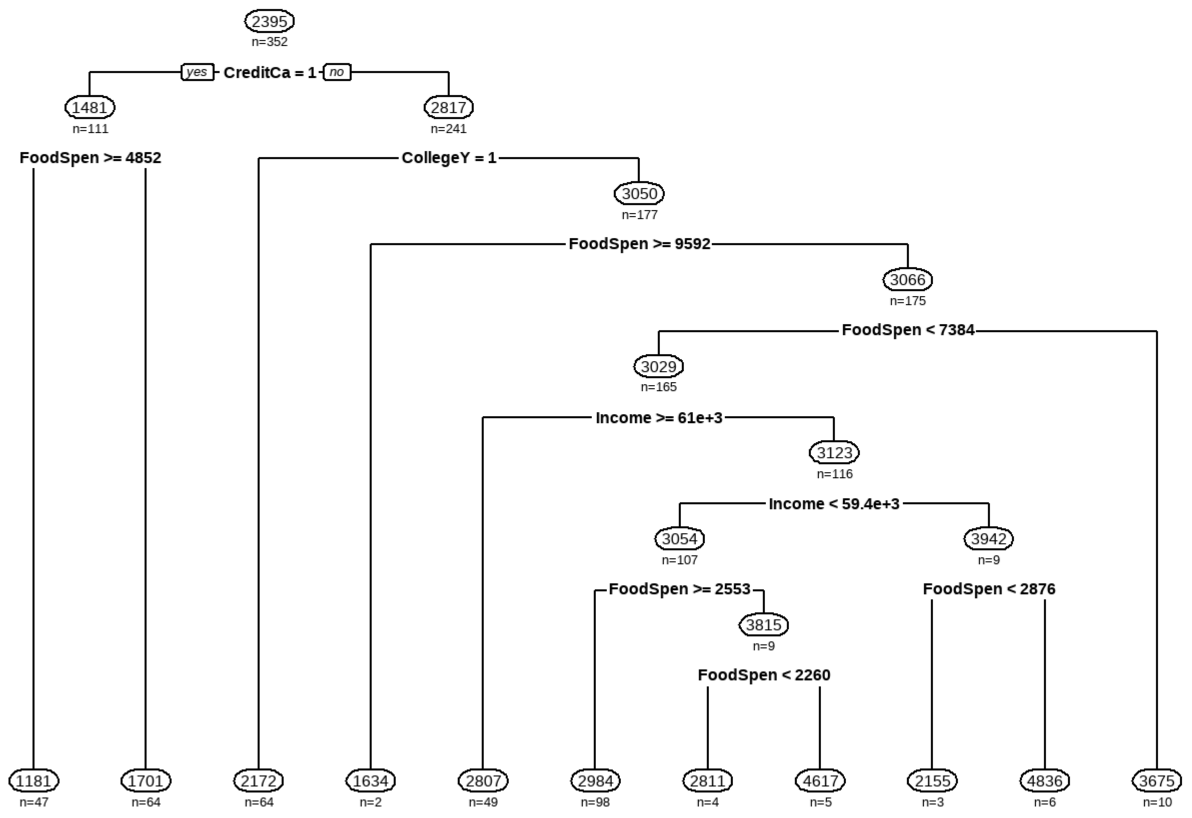
```
print(caret::varImp(best_tree))
```

```
rpart variable importance

              Overall
FoodSpend     100.000
Income         32.062
CreditCardYes   5.226
CollegeYes      0.000
```

Show Best Pruned Tree

```
prp(best_tree$finalModel, type = 2, extra = 1, under = TRUE, digits = 3, fallen.leaves = TRUE)
```



Number of Leaves in Best Tree (carert)

```
cat("Number of Leaf Nodes in the Best Tree: ")
```

```
Number of Leaf Nodes in the Best Tree:
```

```
sum(best_tree$finalModel$frame$var == "<leaf>")
```

```
[1] 11
```

Evaluation (Caret Regression Tree)

```
predict_bp <- predict(best_tree, testset)
validation_metrics_tree <- round(forecast::accuracy(predict_bp, testset$TravelSpend),2)
rownames(validation_metrics_tree) <- "Regression Tree"
print(validation_metrics_tree)
```

```
                    ME   RMSE    MAE    MPE  MAPE
Regression Tree -23.33 941.16 700.45 -14.27 32.47
```

Scoring (Caret Regrerssion Tree)

```
score_predict <- predict(best_tree, travel_score)
pip <- cbind(travel_score, score_predict)
knitr::kable(pip[order(pip$score_predict) , ])
```

College	CreditCard	FoodSpend	Income	score_predict
No	Yes	2892.90	65982	1700.855
Yes	No	6017.66	53986	2172.309

Data Partition (Random Forest)

```
p <- ncol(trainset)-1

mtry_center <- max(1, round(p/3))
mtry_grid <- data.frame(mtry = sort(unique(pmax(1, c(mtry_center-1, mtry_center, mtry_center+1,
        2, p)))))
```

Modeling (Random Forest)

```
set.seed(1)
rf_model <- train(TravelSpend ~ .,
                                                  data = trainset,
                                                  method = "rf",
                                                  trControl = myctrl,
                                                tuneGrid = mtry_grid,
                                                ntree = 1000,
                                                importance = TRUE)
```

Resample

```
rf_model
```

```
Random Forest

352 samples
  4 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 318, 317, 316, 316, 316, 318, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
  1     873.3978  0.4449263  715.9041
  2     820.1100  0.4795871  644.1123
  4     826.4942  0.4770504  607.5179

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 2.
```

Variable Importance

```
caret::varImp(rf_model)
```

```
rf variable importance

              Overall
CreditCardYes 100.000
CollegeYes     33.112
FoodSpend       3.641
Income          0.000
```

Partial Effects of Predictors

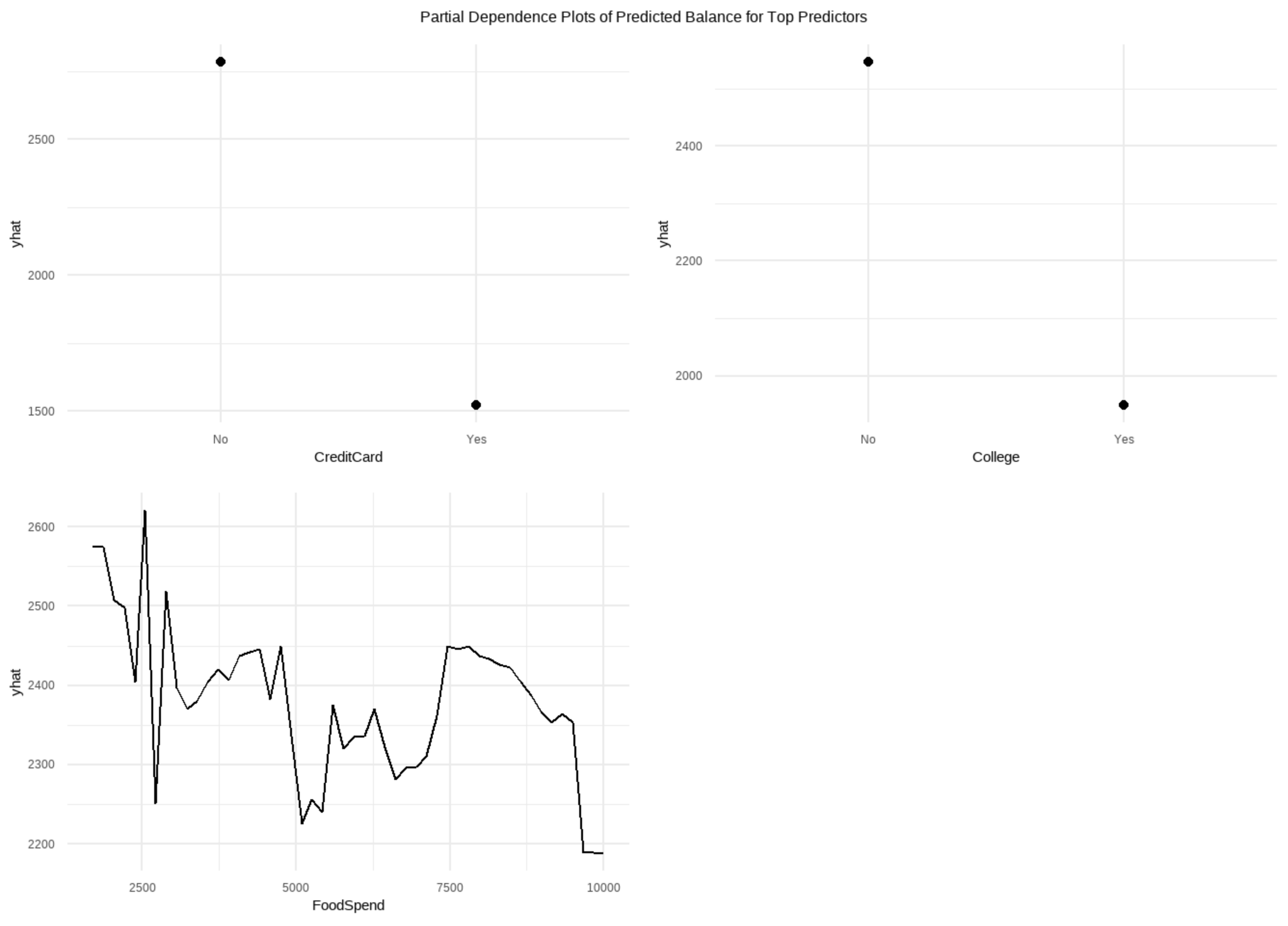
```
## Predictors College, CreditCard, FoodSpend, Income

train_x <- subset(trainset, select = -TravelSpend)

pd_FoodSpend <- partial(object = rf_model,
                       pred.var = "FoodSpend",
                              train = train_x,
                              grid.resolution = 50)
pd_ccd <- partial(object = rf_model,
                       pred.var = "CreditCard",
                       train = train_x,
                       grid.resolution = 50)
pd_college <- partial(object = rf_model,
                           pred.var = "College",
                           train = train_x,
                           grid.resolution = 50)

pd_fs <- autoplot(pd_FoodSpend) + theme_minimal()
pd_ccd <- autoplot(pd_ccd) + theme_minimal()
pd_col <- autoplot(pd_college) + theme_minimal()

grid.arrange(pd_ccd, pd_col, pd_fs, nrow = 2, ncol = 2, top = "Partial Dependence Plots of
         Predicted Balance for Top Predictors")
```



Evaluation (Random Forest)

```
predicted_rf <- predict(rf_model, testset)
test_rf <- round(forecast::accuracy(predicted_rf, testset$TravelSpend),2)
```

```
rownames(test_rf) <- "Random Forest"
test_rf
```

```
                  ME   RMSE    MAE    MPE  MAPE
Random Forest -18.57 829.31 629.59 -12.79 29.32
```

Compare Random Forest With Regression Tree

```
rbind(validation_metrics_tree, test_rf, bpt_metrics)
```

```
                          ME   RMSE    MAE    MPE  MAPE
Regression Tree       -23.33 941.16 700.45 -14.27 32.47
Random Forest         -18.57 829.31 629.59 -12.79 29.32
Rpart Regression Tree  -1.35 894.68 713.39 -14.75 34.74
```

Comments Random Forest vs Regression Tree:
The Random Forest performs better than the Regression tree. The Random forest is more accurate than the regression tree as its RMSE, MAE, and MAPE are lower than the regression trees. The Random Forest also has minimal bias compared to the regression tree as the Random Forests ME and MPE are lower than the Regression trees ME and MPE.

Score (Random Forest)

```
score_rf <- predict(rf_model, travel_score)
rf_pip <- cbind(travel_score, score_rf, score_predict, score_rpart)
rf_pip
```

```
  College CreditCard FoodSpend Income score_rf score_predict score_rpart
1      No        Yes   2892.90  65982 1816.319      1700.855    1480.546
2     Yes         No   6017.66  53986 1835.589      2172.309    2172.309
```

Comments on Comparing Random Forests to Regression Tree:
The Random Forest model's superior accuracy and low bias found in the key metrics suggest its travel spend predictions are more reliable than those of the regression tree. It is important to note that these predictions are not perfect and due to the Random Forests ME and MPE tend to slightly underpredict the true value.