Introduction to Data mining: Exercise 3.11 - Page 186: Question #3

By Jimmy Nguyen - ADS-502 Assignment 2.1

Table 3.6. Data set for Exercise 3.

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

positives

negatives

3) Consider the training examples shown in Table 3.6 for a binary classification problem

a) What is the entropy of this collection of training examples with respect to the class attribute?

$$\text{Entropy} = -\sum_{i=0}^{c-1} p_i(t)\log_2 p_i(t)$$

- Relative frequency of positives "$p_i(t)$" $= 4/9$
- Relative frequency of negatives "$p_i(t)$" $= 5/9$

$$= -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$$

$$= 0.5199 + 0.4711$$

$$\boxed{= 0.9911}$$

Question #3 Cont.

b) What are the information gains of a_1 and a_2 relative to these training examples?

a_1 =

a_1	+	−
T	3	1
F	1	4

$$\Delta I = I(\text{parent}) - I(\text{children})$$

"Information gain"

Earlier $I(\text{parent}) = 0.9911$

Now we solve for $I(\text{children})$ which are a_1 and a_2

$$I(\text{children}) = \sum_{j=1}^{k} \frac{N(v_j)}{N} \underbrace{I(v_j)}_{\text{Entropy}}$$

$$= 4/9\left[-(3/4)\log_2(3/4) - (1/4)\log_2(1/4)\right]$$

$$+ 5/9\left[-(1/5)\log_2(1/5) - (4/5)\log_2(4/5)\right]$$

$$= I(a_1) = 0.7616$$

$$\Delta I_{a_1} = 0.9911 - 0.7616 = \boxed{0.2295}$$

a_2 =

a_2	+	−
T	2	3
F	2	2

$$= 4/9\left[-(2/4)\log_2(2/4) - (2/4)\log_2(2/4)\right]$$

$$+ 5/9\left[-(3/5)\log_2(3/5) - (2/5)\log_2(2/5)\right]$$

$$= 0.9839$$

$$\Delta I_{a_2} = 0.9911 - 0.9839 = \boxed{0.0072}$$

Chapter 3 Hw Cont.

c) For a_3, which is a continuous attribute, compute the information gain for every possible split.

$a_3 = [1.0, 6.0, 5.0, 4.0, 7.0, 3.0, 8.0, 7.0, 5.0]$

Table 3.6. Data set for Exercise 3.

Instance	a_1	a_2	a_3	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	−
4	F	F	4.0	+
5	F	T	7.0	−
6	F	T	3.0	−
7	F	F	8.0	−
8	T	F	7.0	+
9	F	T	5.0	−

Step 1: Sort the values

$a_3 = [1.0, 3.0, 4.0, 5.0, 5.0, 6.0, 7.0, 7.0, 8.0]$

class labels = +, −, +, −, −, +, +, −, −

Step 2: Discretization of the values

$a_3 = [1.0, 3.0, 4.0, 5.0, 5.0, 6.0, 7.0, 7.0, 8.0]$

split points = 2.0, 3.5, 4.5, 5.5, 6.5, 7.5

Step 3: Calculate the entropy for the split points

Entropy (Parent)

+ : 4/9

− : 5/9

$= -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$

$= 0.99107$

$I(\text{parent}) = 0.99107$

Entropy (child)

"≤ 2.0"

+ : 1

− : 0

$0\log_2 0 = 0$ in entropy

$= 1/8\left[-(1/1)\log_2(1/1) - (0/1)\log_2(0/1)\right]$

$= 1/8[0 - 0]$

$= 0$

Entropy (child)

"> 2.0"

+ : 3

− : 5

$= 8/9\left[-(3/8)\log_2(3/8) - (5/8)\log_2(5/8)\right]$

$= 8/9[0.98522]$

$= 0.84838$

$\Delta I = I(\text{parent}) - I(\text{children})$

$I(\text{children}) = 0 + 0.84838$

a_3 at split point 2.0 $\boxed{= 0.8484}$

$I(\text{parent}) = 0.9910$

$\boxed{\Delta I = 0.9910 - 0.8484 = 0.1426}$ → "Information gain"

↳ Repeat steps 3 for all split points

↳ final answer: Best split for a_3 is at split point = 2.0

d) What is the best split (among a_1, a_2, a_3) according to the information gain?

The best split occurs at a_1 with $\Delta I = 0.2295$

e) What is the best split (between a_1 and a_2) according to the misclassification rate?

Classification error $= 1 - \max_i [p_i(t)]$

Error $a_1 = 1 - \max\left[7/9, 2/9\right]$

$= 1 - 7/9$

$= 0.222$

node a_1	count
class = +	7
class = -	2

Error $a_2 = 1 - \max\left[4/9, 5/9\right]$

$= 1 - 5/9$

$= 0.444$

node a_2	count
class = +	4
class = -	5

Best split is at a_1
because of the lower classification error rate

f) What is the best split (between a_1 and a_2) according to the Gini index?

Gini index $= 1 - \sum_{i=0}^{c-1} p_i(t)^2$

Target class
+: 4
-: 5

a_1 =

a_1: True → +: 3, -: 1; False → +: 1, -: 4

$4/9\left[1-(3/4)^2+(1/4)^2\right] + 5/9\left[1-(1/5)^2+(4/5)^2\right]$

$= 0.3444$

Final answer
a_1 produces the best split b/c lower gini.

a_2 =

a_2: True → +: 2, -: 3; False → +: 2, -: 2

$5/9\left[1-(2/5)^2-(3/5)^2\right] + 4/9\left[1-(2/4)^2+(2/4)^2\right]$

$= 0.4889$

Module 2.1 Assignment - Jimmy Nguyen

March 15, 2021

1 Jimmy Nguyen - ADS 502 - Data Science Using Python and R: Chapter 4 Questions #21, 22, 23, 24, & 25 - EDA

[1]:
```
%load_ext rpy2.ipython
```

1.1 Packages in Python

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1.2 Packages in R

[3]:
```
%%R
library(readr)
library(ggplot2)
```

1.3 Dataset

Python code:

[4]:
```
bank_train = pd.read_csv("bank_marketing_training")
bank_train.head()
```

[4]:
```
   age          job  marital    education  default housing loan    contact  \
0   56    housemaid  married     basic.4y       no      no   no  telephone
1   57     services  married  high.school  unknown      no   no  telephone
2   41  blue-collar  married      unknown  unknown      no   no  telephone
3   25     services   single  high.school       no     yes   no  telephone
4   29  blue-collar   single  high.school       no      no  yes  telephone

  month day_of_week  …  campaign  days_since_previous  previous  \
0   may         mon  …         1                  999         0
1   may         mon  …         1                  999         0
2   may         mon  …         1                  999         0
3   may         mon  …         1                  999         0
```

```
4   may          mon   …          1                 999          0

   previous_outcome emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
0       nonexistent          1.1          93.994          -36.4      4.857
1       nonexistent          1.1          93.994          -36.4      4.857
2       nonexistent          1.1          93.994          -36.4      4.857
3       nonexistent          1.1          93.994          -36.4      4.857
4       nonexistent          1.1          93.994          -36.4      4.857

   nr.employed  response
0         5191        no
1         5191        no
2         5191        no
3         5191        no
4         5191        no

[5 rows x 21 columns]
```

R code:

[5]:
```
%%R
bank_train <- read_csv("bank_marketing_training")

head(bank_train)
```

```
R[write to console]: Parsed with column specification:
cols(
  .default = col_character(),
  age = col_double(),
  duration = col_double(),
  campaign = col_double(),
  days_since_previous = col_double(),
  previous = col_double(),
  emp.var.rate = col_double(),
  cons.price.idx = col_double(),
  cons.conf.idx = col_double(),
  euribor3m = col_double(),
  nr.employed = col_double()
)

R[write to console]: See spec(…) for full column specifications.


# A tibble: 6 x 21
    age job   marital education default housing loan  contact month day_of_week
  <dbl> <chr> <chr>
<chr>     <chr>   <chr>
<chr> <chr>   <chr>
```

```
<chr>
1     56 hous… married basic.4y  no      no      no    teleph… may
mon
2     57 serv… married high.sch… unknown no      no    teleph… may
mon
3     41 blue… married unknown   unknown no      no    teleph… may
mon
4     25 serv… single  high.sch… no      yes     no    teleph… may
mon
5     29 blue… single  high.sch… no      no      yes   teleph… may
mon
6     57 hous… divorc… basic.4y  no      yes     no    teleph… may
mon
# … with 11 more variables: duration <dbl>, campaign

<dbl>,

#   days_since_previous <dbl>, previous

<dbl>, previous_outcome <chr>,

#   emp.var.rate <dbl>, cons.price.idx

<dbl>, cons.conf.idx <dbl>,

#   euribor3m <dbl>, nr.employed <dbl>,

response <chr>
```

1.4 Question 21

Produce the following graphs. What is the strength of each graph? Weakness?

1.4.1 Question 21a.

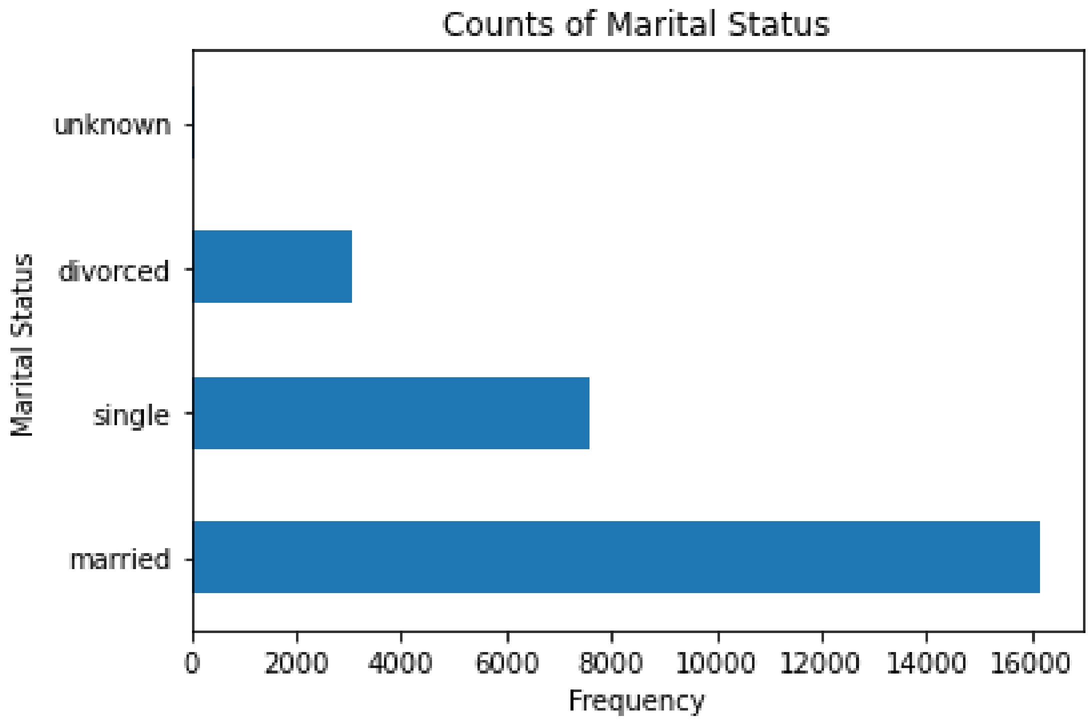
Bar graph of *marital*

Python Code:

[6]:
```
marital = bank_train['marital'].value_counts()
marital
```

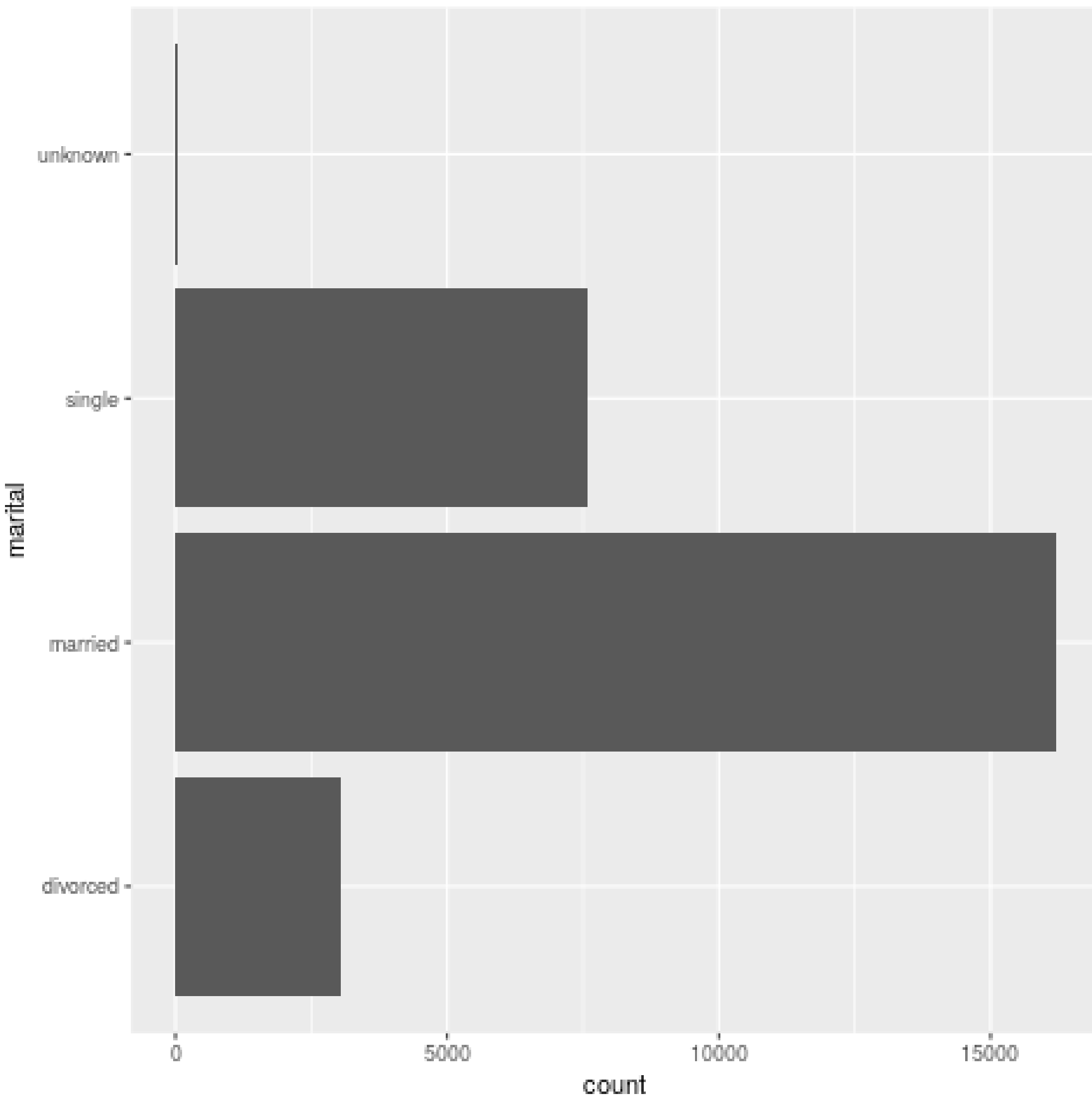
[6]:
```
married     16187
single       7575
divorced     3055
unknown        57
Name: marital, dtype: int64
```

[7]:
```
marital.plot(kind = 'barh')
plt.xlabel('Frequency'); plt.ylabel('Marital Status')
plt.title('Counts of Marital Status'); plt.show()
```



R Code:

[8]:
```
%%R
ggplot(bank_train, aes(marital)) +
geom_bar() + coord_flip()
```



Answer: For this graph, we can see the lengths of the values, but it does not tell us anything about the target variable.

Ex. **Married couples** are the **most frequent** while **divorced or unknown are the least**

1.4.2 Question 21b.

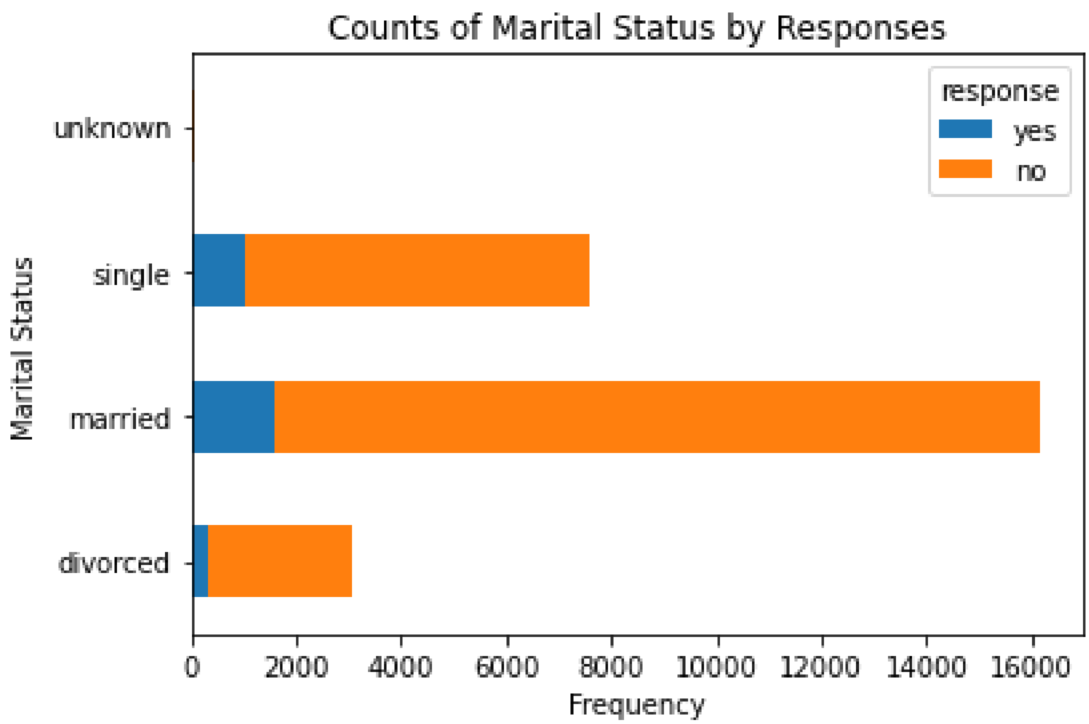
Bar graph of *marital*, with overlay of *response*

Python Code:

[9]:
```
crosstab_mar = pd.crosstab(bank_train['marital'], bank_train['response'])
response = ["yes", "no"]
crosstab_mar = crosstab_mar.reindex(response, axis="columns")
round(crosstab_mar * 100,1)
```

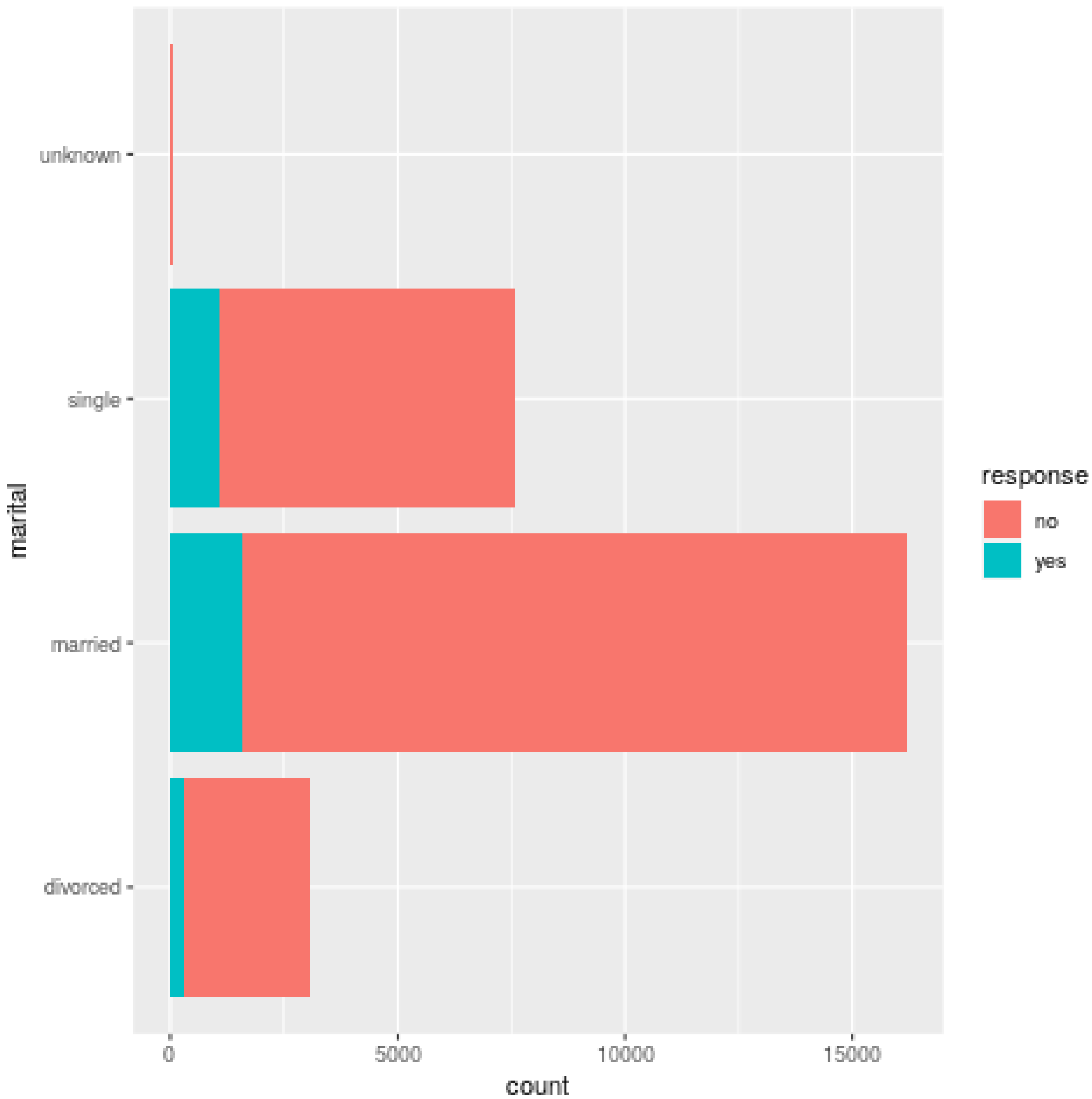
[9]:
```
response      yes       no
marital
divorced    31200   274300
married    160800  1457900
single     106100   651400
unknown       700     5000
```

[10]:
```
crosstab_mar.plot(kind = 'barh', stacked = True)
plt.xlabel('Frequency'); plt.ylabel('Marital Status')
plt.title('Counts of Marital Status by Responses'); plt.show()
```



R Code:

[11]:
```
%%R
ggplot(bank_train, aes(marital)) +
geom_bar(aes(fill=response)) + coord_flip()
```



Answer: This graph is useful for showing the distribution of the values of the categorical variable, however it not clear that we can see which category has the greater proportion.

1.4.3 Question 21c.

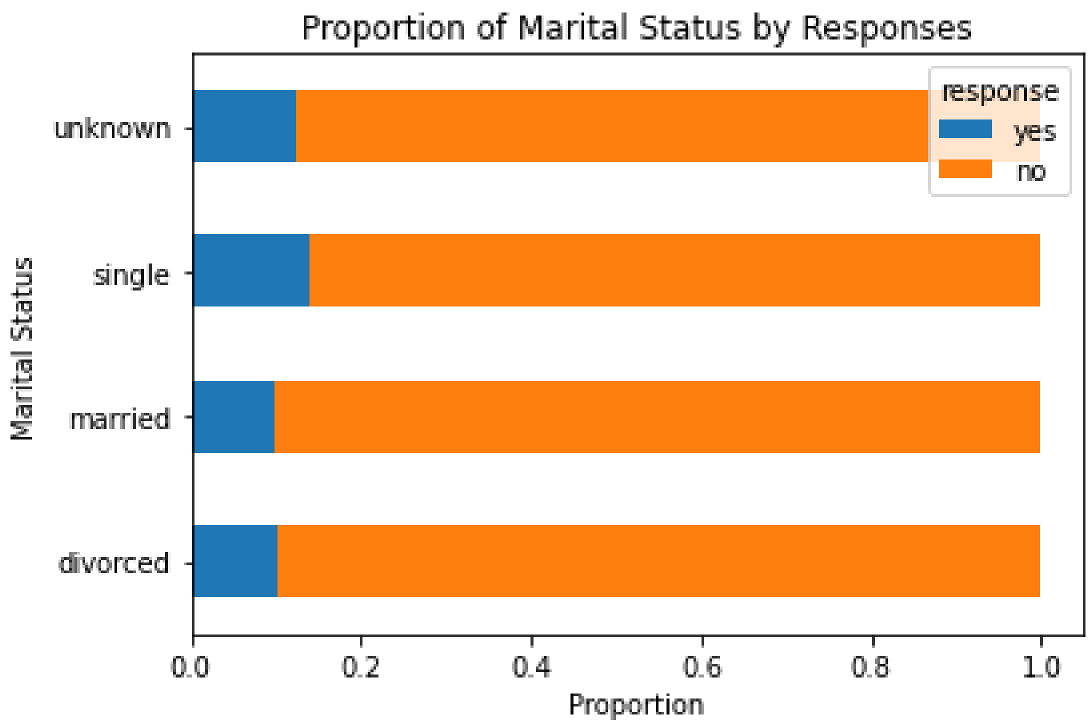
Normalized bar graph of *marital*, with overlay of *response*.

Python Code:

[12]:
```
mar_norm = crosstab_mar.div(crosstab_mar.sum(1), axis = 0)
response = ["yes", "no"]
mar_norm = mar_norm.reindex(response, axis="columns")
round(mar_norm * 100,1)
```

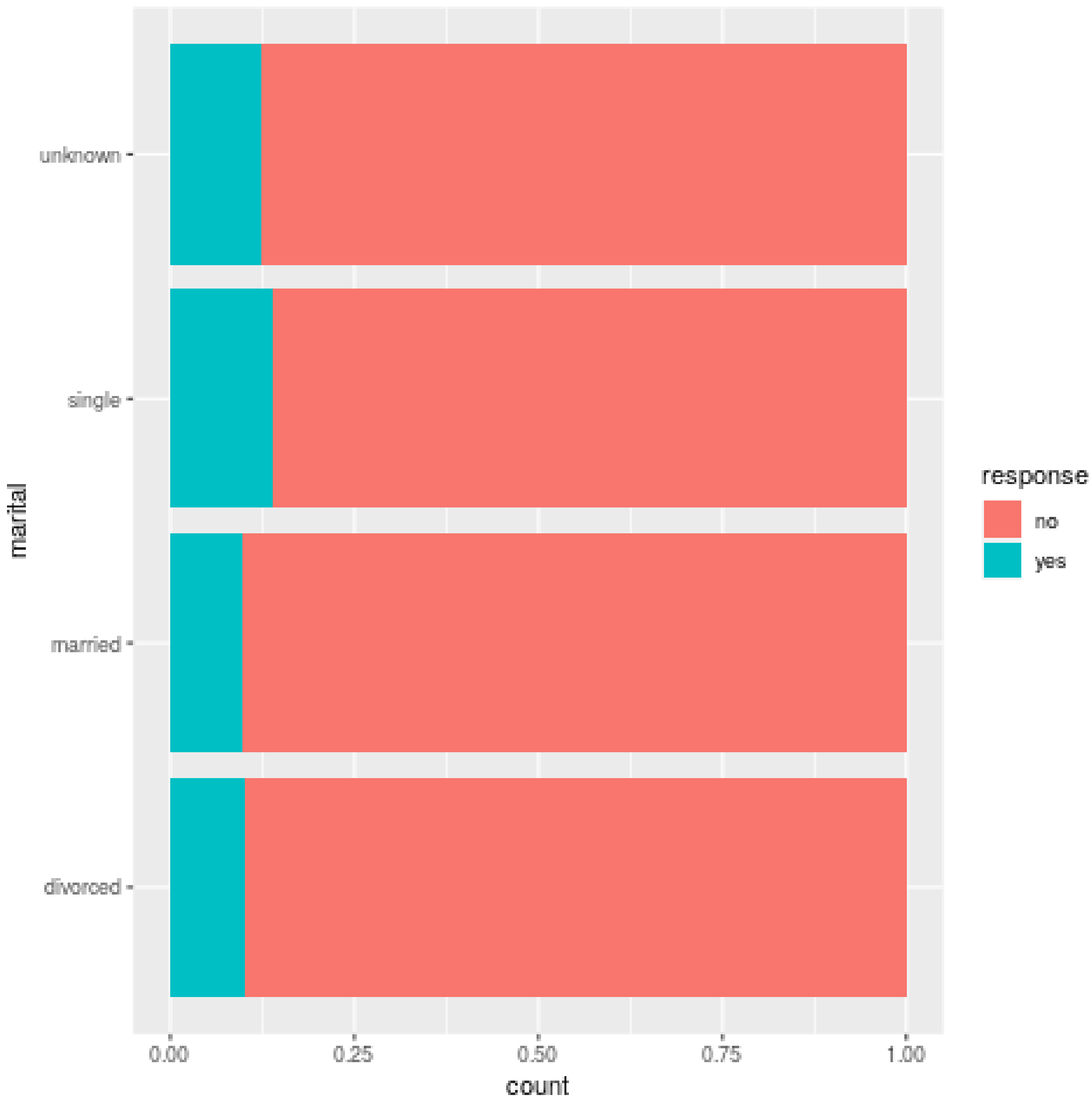
[12]:
```
response   yes    no
marital
divorced  10.2  89.8
married    9.9  90.1
single    14.0  86.0
unknown   12.3  87.7
```

[13]:
```
mar_norm.plot(kind = 'barh', stacked = True)
plt.xlabel('Proportion'); plt.ylabel('Marital Status')
plt.title('Proportion of Marital Status by Responses'); plt.show()
```



R Code:

[14]:
```
%%R

ggplot(bank_train, aes(marital)) +
geom_bar(aes(fill=response), position="fill") + coord_flip()
```



Answer:

This graph allows us the equalize the length of each bar, so that we may easily compare the response proportions.

Ex. **Single** status actually has the highest proportion of **yes** responses while **married** has the lowest proportion of **yes** responses.

1.5 Question 22

Using the graph from Exercise 21c, describe the relationship between marital and response.

Answer: Since the proportions of responses vary between the groups, we can conclude that the two variables are associated.

1.6 Question 23

Do the following with the variables *marital* and *response*

1.6.1 Question 23a

Build a contingency table, being careful to have the correct variables representing the rows and columns. Report the counts and the column percentages.

Python Code:

[15]:
```
real_crosstab = pd.crosstab(bank_train['response'], bank_train['marital']) #
↪This is what it should look like
real_crosstab
```

[15]:
```
marital   divorced  married  single  unknown
response
no            2743    14579    6514       50
yes            312     1608    1061        7
```

[16]:
```
col_crosstab_norm = round(real_crosstab.div(real_crosstab.sum(0), axis = 1) *
↪100,1)
col_crosstab_norm
```

[16]:
```
marital   divorced  married  single  unknown
response
no            89.8     90.1    86.0     87.7
yes           10.2      9.9    14.0     12.3
```

R Code:

[17]:
```
%%R
counts <- table(bank_train$response, bank_train$marital)
counts
```

```
     divorced married single unknown
 no      2743   14579   6514      50
 yes      312    1608   1061       7
```

[18]:
```
%%R
prop <- round(prop.table(counts, margin = 2) * 100,1)
prop
```

```
     divorced married single unknown
 no      89.8    90.1   86.0    87.7
 yes     10.2     9.9   14.0    12.3
```

1.6.2 Question 23b.

Describe what the contingency table is telling you.

Answer: - **Married couples** appears to have the response with **no** more frequently than **yes** responses, just like the other marital status with more **no** than **yes** responses.*

1.7 Question 24

Repeat the previous exercise, this time reporting the row percentages. Explain the difference between the interpretation of this table and the previous contingecy table.

Python Code:

```
[19]: row_crosstab_norm = round(real_crosstab.div(real_crosstab.sum(1), axis = 0) *␣
      ↪100,1)
      row_crosstab_norm
```

```
[19]: marital   divorced  married  single  unknown
      response
      no            11.5     61.0    27.3      0.2
      yes           10.4     53.8    35.5      0.2
```

R code

```
[20]: %%R
      prop_row <- round(prop.table(counts, margin = 1) * 100,1)
      prop_row
```

```
     divorced married single unknown
 no       11.5    61.0   27.3     0.2
 yes      10.4    53.8   35.5     0.2
```

Answer:

- While the normalization of the contingency table tells us that **married** has the highest proportion of **yes** responses, **unknown** has the lowest proportion of **yes** responses. The difference between this table and the previous is the we read the proportions. Simply, this table give us the proportions of marital status within each responses, thus we can see which marital status has the highest or lowest response for no or yes. While the previous table tells us the proportion of responses within each marital status. For example, we can see that married were actively responding **no** than **yes**.

1.8 Question 25

Produce the following graphs. What is the strength of each graph? Weakness?

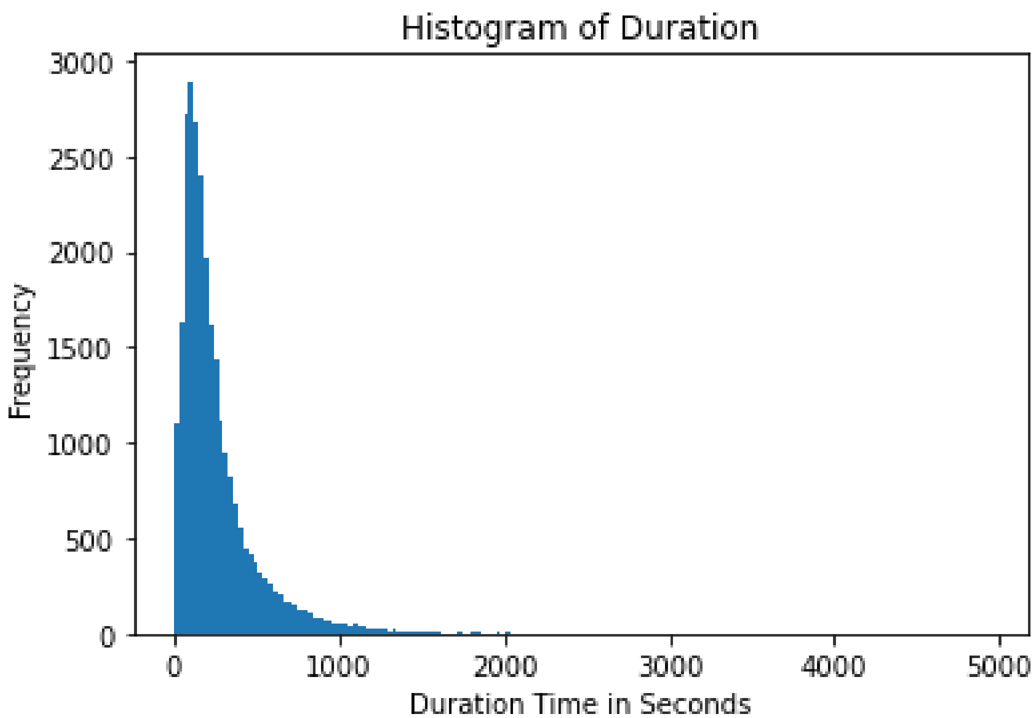
1.8.1 Question 25a.

Histogram of *duration*

Python Code:

[21]:
```
from math import *
n_bins = round(sqrt(len(bank_train['duration'])))
print(n_bins)

plt.hist(bank_train['duration'], bins = n_bins)
plt.xlabel('Duration Time in Seconds'); plt.ylabel('Frequency'); plt.
↪title('Histogram of Duration')
plt.show()
```

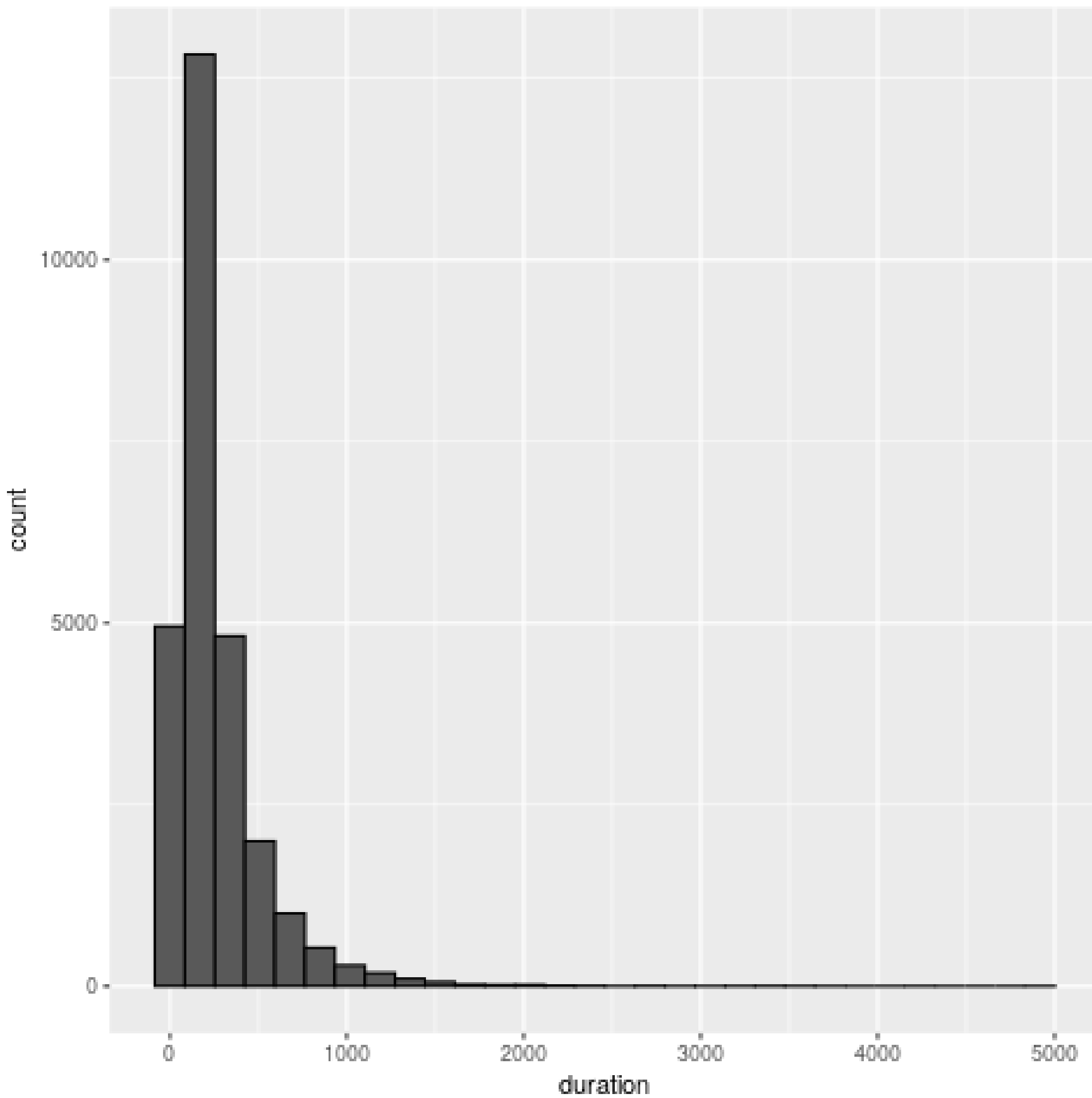
```
164
```



R code:

[22]:
```
%%R
ggplot(bank_train, aes(duration)) + geom_histogram(color = "black")
```

```
R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



Answer:

This histogram is useful for seeing the distribution of the values of duration, however it does not tell us anything about our target variable.

Ex. The shape of the data appears to be **right skewed** looks to be **unimodal**. Therefore, we can see that there are common phone calls are around 500 seconds or less from the last contact duration. That means phone calls with the customers tend to be shorter and under 500 seconds which is around 8 minutes.

1.8.2 Question 25b.

Histogram of *duration*, with overlay of *response*

Python Code:

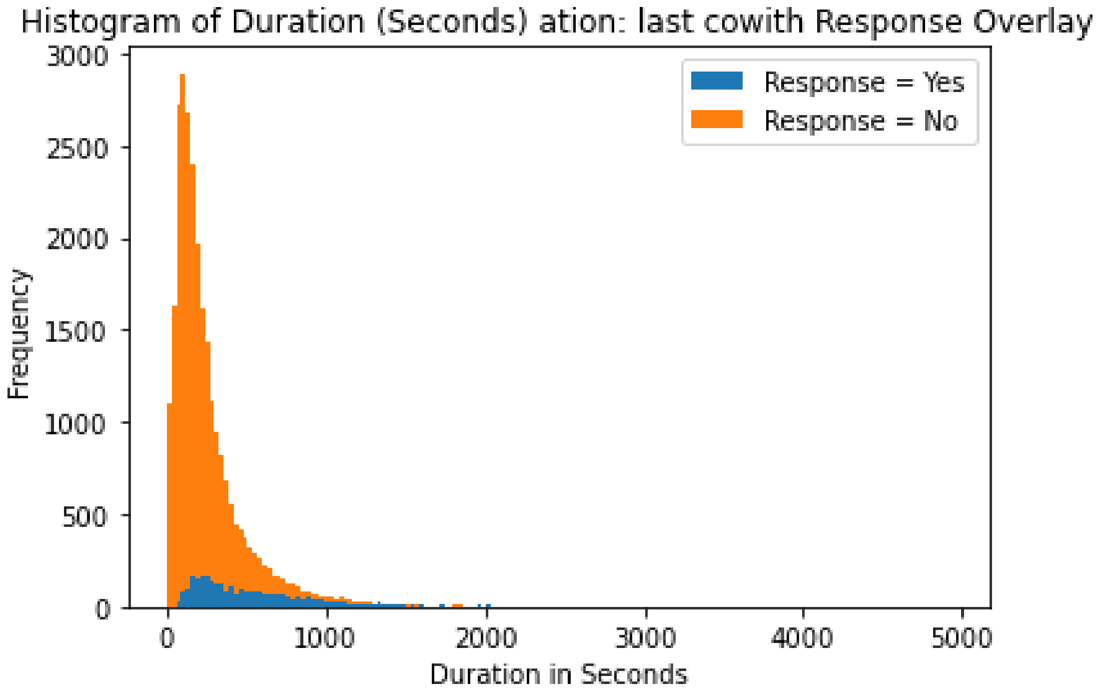
[23]:
```python
bt_dur_y = bank_train[bank_train.response == "yes"]['duration']
bt_dur_y
```

[23]:
```
42       1575
48       1042
51       1467
175       935
239      1030
         …
26865     112
26866     353
26867     329
26870     281
26872     334
Name: duration, Length: 2988, dtype: int64
```

[24]:
```python
bt_dur_n = bank_train[bank_train.response == "no"]['duration']
bt_dur_n
```

[24]:
```
0        261
1        149
2        217
3        222
4        137
        …
26864    293
26868    124
26869    254
26871    112
26873    383
Name: duration, Length: 23886, dtype: int64
```

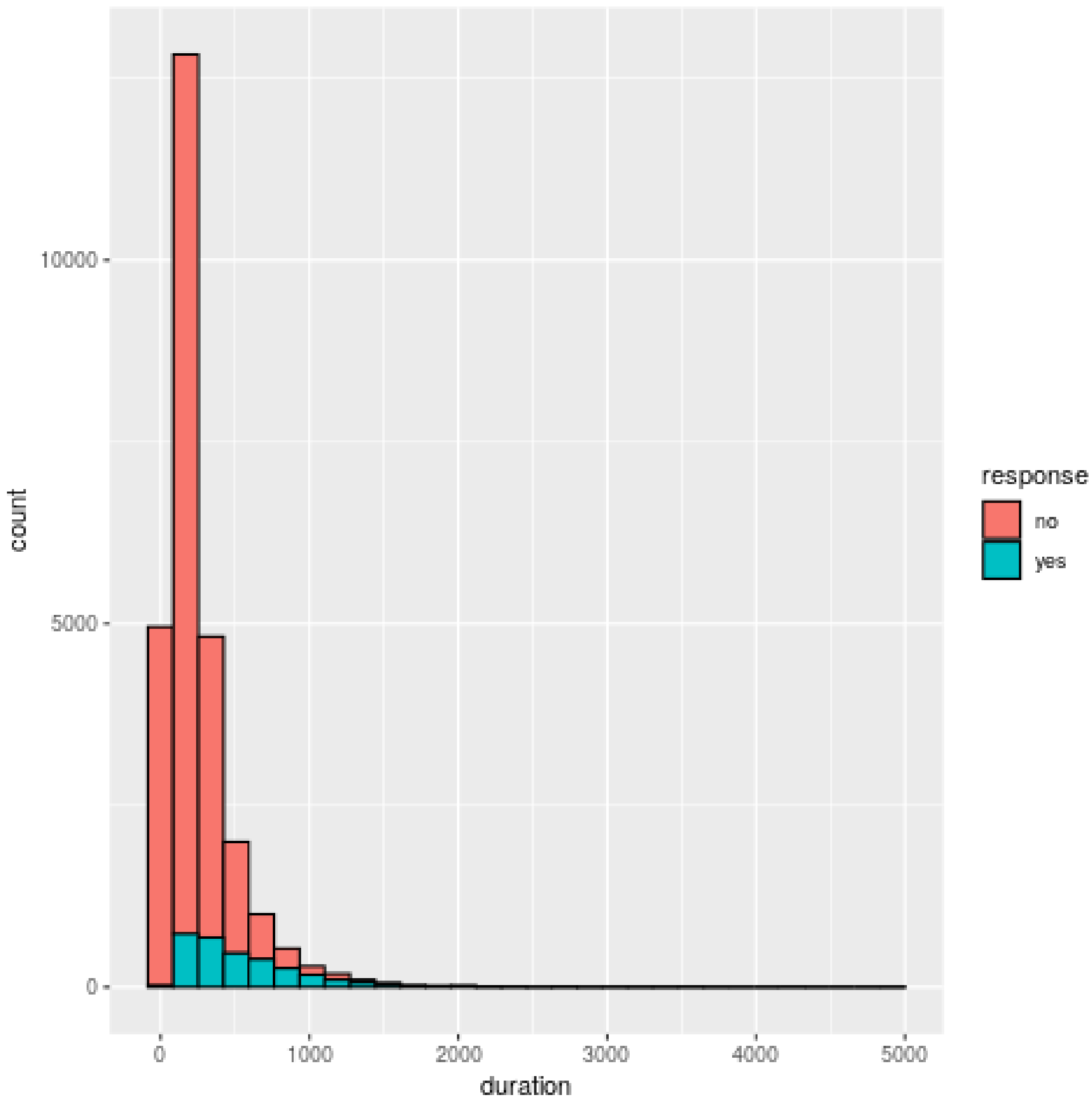
[25]:
```python
plt.hist([bt_dur_y,bt_dur_n], bins = n_bins, stacked = True)
plt.legend(['Response = Yes', 'Response = No'])
plt.title('Histogram of Duration (Seconds) ation: last cowith Response Overlay')
plt.xlabel('Duration in Seconds'); plt.ylabel('Frequency'); plt.show()
```



R code:

[26]:
```
%%R
ggplot(bank_train, aes(duration)) + geom_histogram(aes(fill = response), color␣
↪= "black")
```

```
R[write to console]: `stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



Answer:

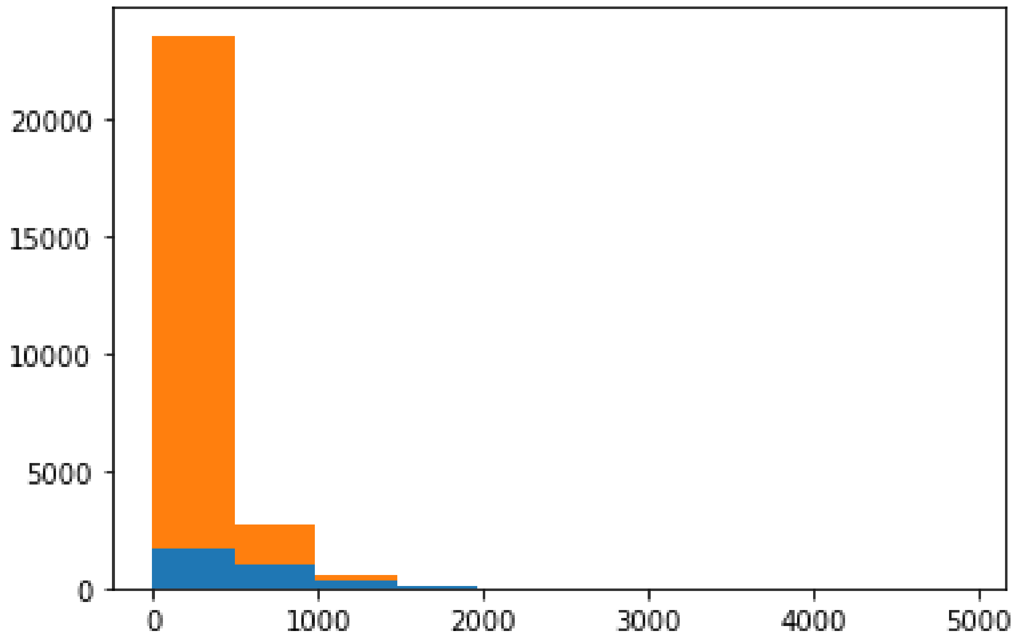
This histogram is useful for seeing the distribution of the responses, but it does not tell us anything about the patterns in the response proportions.

1.8.3 Question 25c.

Normalized histogram of *duration*, with overlay of *response*

Python code:

[27]:
```
(n, bins, patches) = plt.hist([bt_dur_y,bt_dur_n], bins = 10, stacked = True)
```

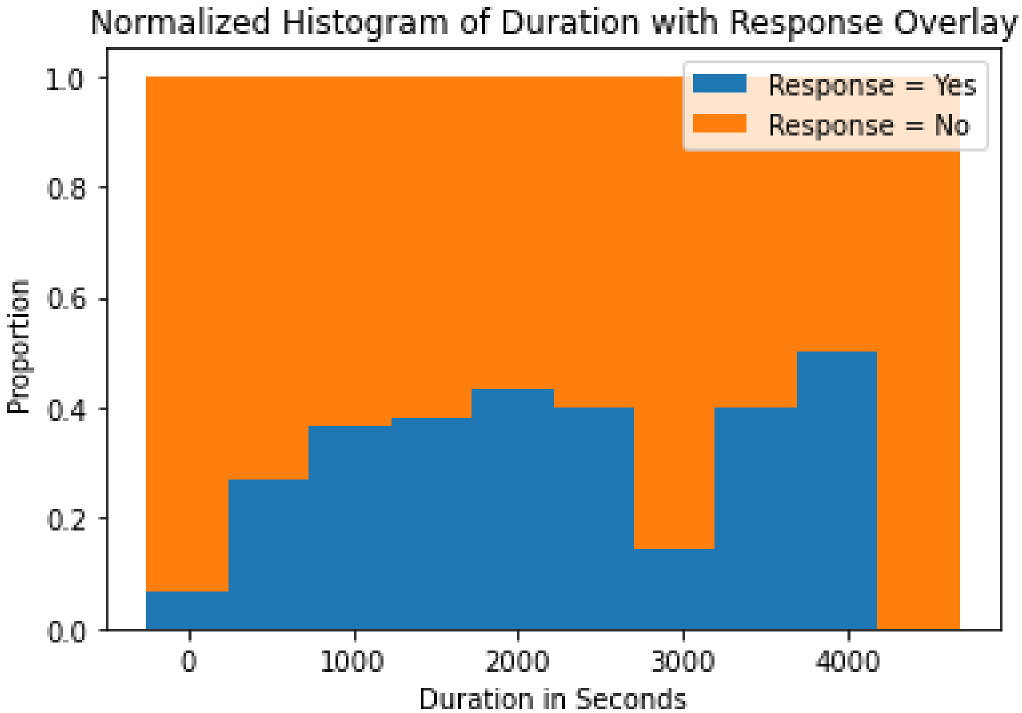


[28]:
```
n_table = np.column_stack((n[0], n[1]))
```

[29]:
```
n_norm =n_table / n_table.sum(axis = 1)[:,None]
```

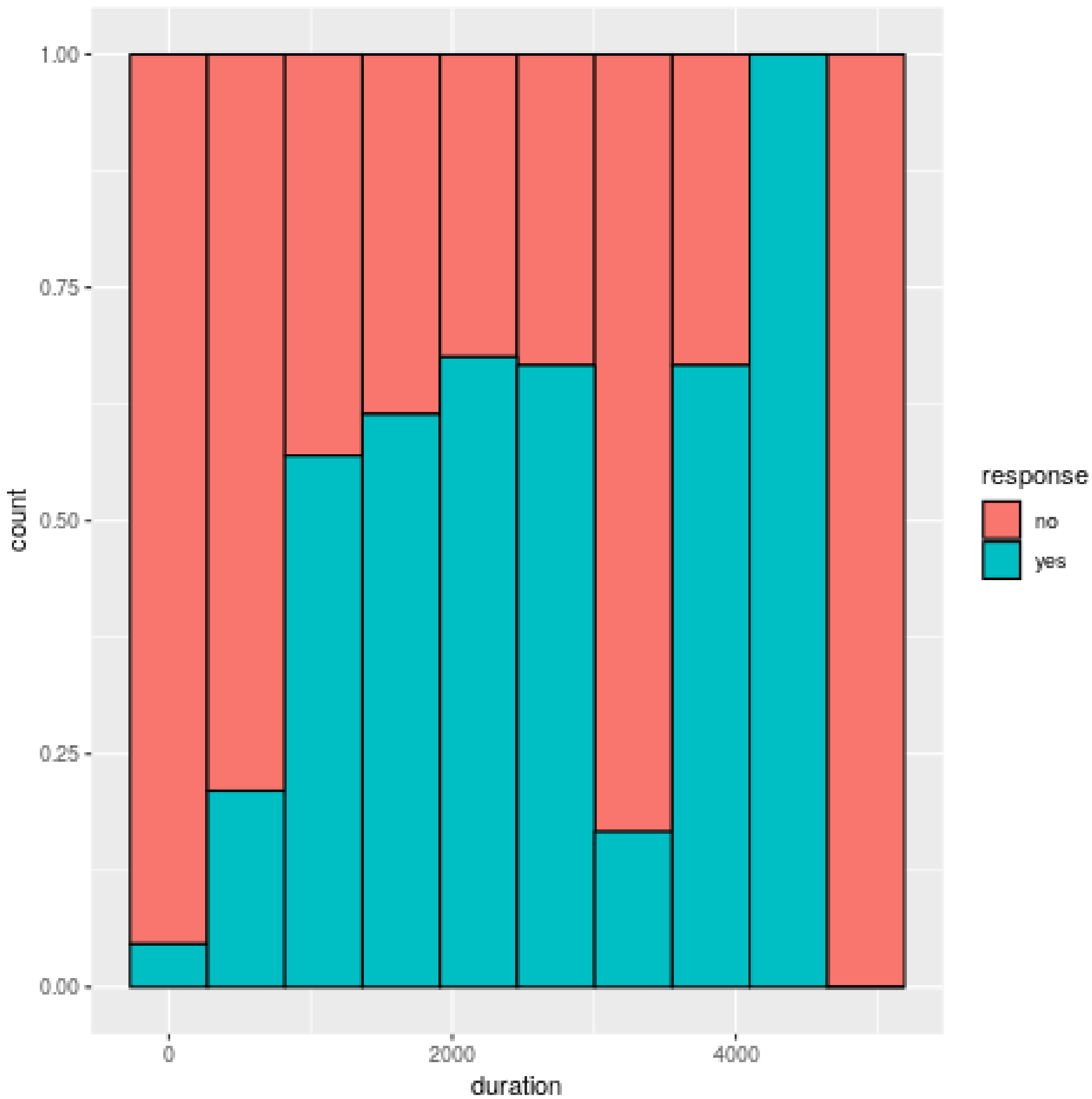
[30]:
```
ourbins =  np.column_stack((bins[0:10], bins[1:11]))
```

[31]:
```
p1 = plt.bar(x = ourbins[:,0], height = n_norm[:,0], width =ourbins[:,1] -␣
 ↪ourbins[:,0])
p2 = plt.bar(x = ourbins[:,0], height = n_norm[:,1], width =ourbins[:,1] -␣
 ↪ourbins[:,0], bottom = n_norm[:10,0])
plt.legend(['Response = Yes', 'Response = No'])
plt.title('Normalized Histogram of Duration with Response Overlay')
plt.xlabel('Duration in Seconds'); plt.ylabel('Proportion'); plt.show()
```



R code:

[32]:
```
%%R
ggplot(bank_train, aes(duration)) +
  geom_histogram(bins = 10, aes(fill = response),
                 color = "black", position = "fill")
```



Answer:

This normalized histogram help us better clarify these response proportions as we can see that customers with **yes** responses tend to spend more time on the phone with us. However, the weakness with this normalized histogram is that we can see its true distribution shape.

2 Data Science Using Python and R: Chapter 6 - Page 93: Questions #14, 15, 16, & 17 - Decision Trees

2.1 Packages in Python

[33]:
```
from sklearn import tree
import statsmodels.tools.tools as stattools
from sklearn.tree import DecisionTreeClassifier, export_graphviz
```

2.2 Packages in R

[34]:
```
%%R
library(rpart)
library(rpart.plot)
library(readr)
library(C50)
```

2.3 Dataset

Python code:

[35]:
```
adult_tr = pd.read_csv("adult_ch6_training")
adult_tr.head()
```

[35]:

	Marital status	Income	Cap_Gains_Losses
0	Never-married	<=50K	0.02174
1	Divorced	<=50K	0.00000
2	Married	<=50K	0.00000
3	Married	<=50K	0.00000
4	Married	<=50K	0.00000

[36]:
```
y = adult_tr[['Income']]
y.head()
```

[36]:

	Income
0	<=50K
1	<=50K
2	<=50K
3	<=50K
4	<=50K

R code:

[37]:
```
%%R
adult_tr <- read_csv("adult_ch6_training")

colnames(adult_tr)[1] <- "maritalStatus"
```

```
adult_tr$Income <- factor(adult_tr$Income)
adult_tr$maritalStatus <- factor(adult_tr$maritalStatus)

head(adult_tr)
```

```
R[write to console]: Parsed with column specification:
cols(
  `Marital status` = col_character(),
  Income = col_character(),
  Cap_Gains_Losses = col_double()
)


# A tibble: 6 x 3
  maritalStatus Income Cap_Gains_Losses
  <fct>         <fct>
<dbl>
1 Never-married <=50K           0.0217
2 Divorced      <=50K           0
3 Married       <=50K           0
4 Married       <=50K           0
5 Married       <=50K           0
6 Married       >50K            0
```

2.4 Question 14

Create a CART model using the training data set that predicts income using marital status and capital gains and losses. Visualize the decision tree (that is, provide the decision tree output). Describe the first few splits in the decision tree.

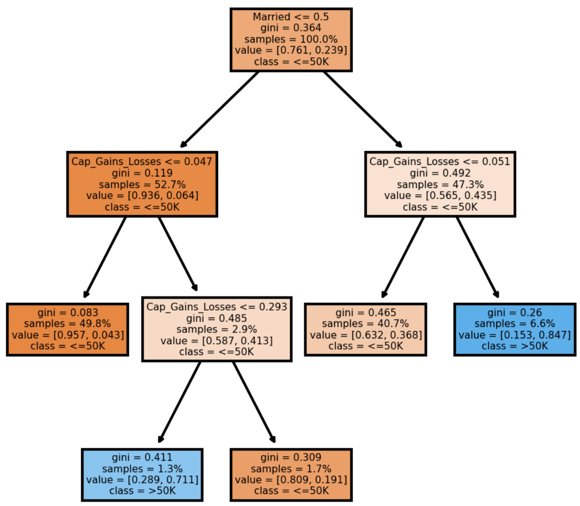
Python code:

[38]:
```
mar_np = np.array(adult_tr['Marital status'])
(mar_cat, mar_cat_dict) = stattools.categorical(mar_np, drop = True, dictnames␣
↪= True)
mar_cat_pd = pd.DataFrame(mar_cat)
X = pd.concat((adult_tr[['Cap_Gains_Losses']], mar_cat_pd), axis = 1)
X_names = [X.columns[0]] + list(mar_cat_dict.values())
y_names = list(y['Income'].unique())
X = X.rename(columns=mar_cat_dict)
X.head()
```

[38]:

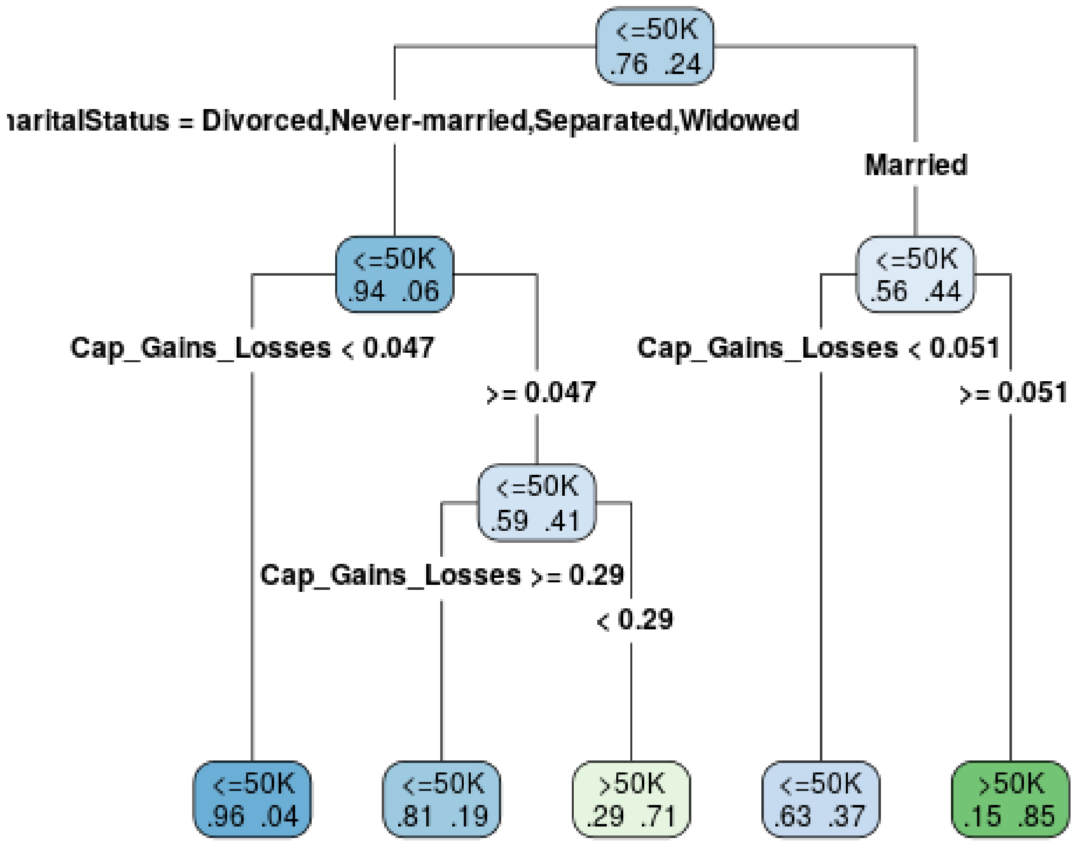
	Cap_Gains_Losses	Divorced	Married	Never-married	Separated	Widowed
0	0.02174	0.0	0.0	1.0	0.0	0.0
1	0.00000	1.0	0.0	0.0	0.0	0.0
2	0.00000	0.0	1.0	0.0	0.0	0.0
3	0.00000	0.0	1.0	0.0	0.0	0.0
4	0.00000	0.0	1.0	0.0	0.0	0.0

[39]:
```python
cart01 = DecisionTreeClassifier(criterion = "gini",
                                max_leaf_nodes = 5).fit(X,y)

fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)

tree.plot_tree(cart01,
               feature_names = X_names,
               class_names=y_names,
               filled = True, proportion = True)
fig.savefig('cart01_tree.png')
```



R code:

[40]:
```r
%%R
cart01 <- rpart(data = adult_tr,
                formula = Income ~ maritalStatus + Cap_Gains_Losses,
                method = "class")

rpart.plot(cart01, type =4, extra = 4 )
```



Answer: The root tells us that 76% of the records in the *adult_ch6_training* data set have low income (<50K). Thus each node tells us the proportion of low-income records in that node. At the root split, since we are using the CART algorithm, it will always find the best possible binary split as seperating them into two groups. Here we see in the root node split, we are dividing the dataset into marital status where adults who are not married (single, divorcedd, widowed) vs. adults who are married. On the left hand side, we have 94% of couples who are not married adults have low-income, while on the right side we have 56% of married couples with low income. Back to the left side, for unmarried couples, we see that if their capital gains and losses are below <0.047 then 96% of the records have low income. This is our interpretation of this dataset using the CART algorithm.

2.5 Question 15

Develope a CART model using the test data set that utilizes the same target and predictor variables. Visualize the decision tree. Compare the decision trees. Does the test data result match the training data result?

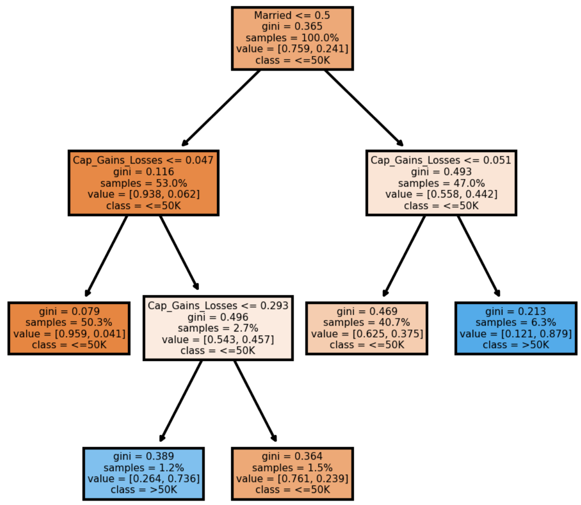
Python Code:

[41]:
```
adult_test = pd.read_csv("adult_ch6_test")
y = adult_test[['Income']]
mar_np = np.array(adult_test['Marital status'])
(mar_cat, mar_cat_dict) = stattools.categorical(mar_np, drop = True, dictnames␣
 ↪= True)
mar_cat_pd = pd.DataFrame(mar_cat)
X = pd.concat((adult_test[['Cap_Gains_Losses']], mar_cat_pd), axis = 1)
X_names = [X.columns[0]] + list(mar_cat_dict.values())
y_names = list(y['Income'].unique())
X = X.rename(columns=mar_cat_dict)
X.head()
```

[41]:

	Cap_Gains_Losses	Divorced	Married	Never-married	Separated	Widowed
0	0.000000	0.0	1.0	0.0	0.0	0.0
1	0.051781	0.0	1.0	0.0	0.0	0.0
2	0.000000	0.0	0.0	1.0	0.0	0.0
3	0.000000	1.0	0.0	0.0	0.0	0.0
4	0.000000	0.0	1.0	0.0	0.0	0.0

[42]:
```
cart02 = DecisionTreeClassifier(criterion = "gini",
                                max_leaf_nodes = 5).fit(X,y)
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)

tree.plot_tree(cart02,
               feature_names = X_names,
               class_names=y_names,
               filled = True,
              proportion = True)

fig.savefig('cart02_tree.png')
```



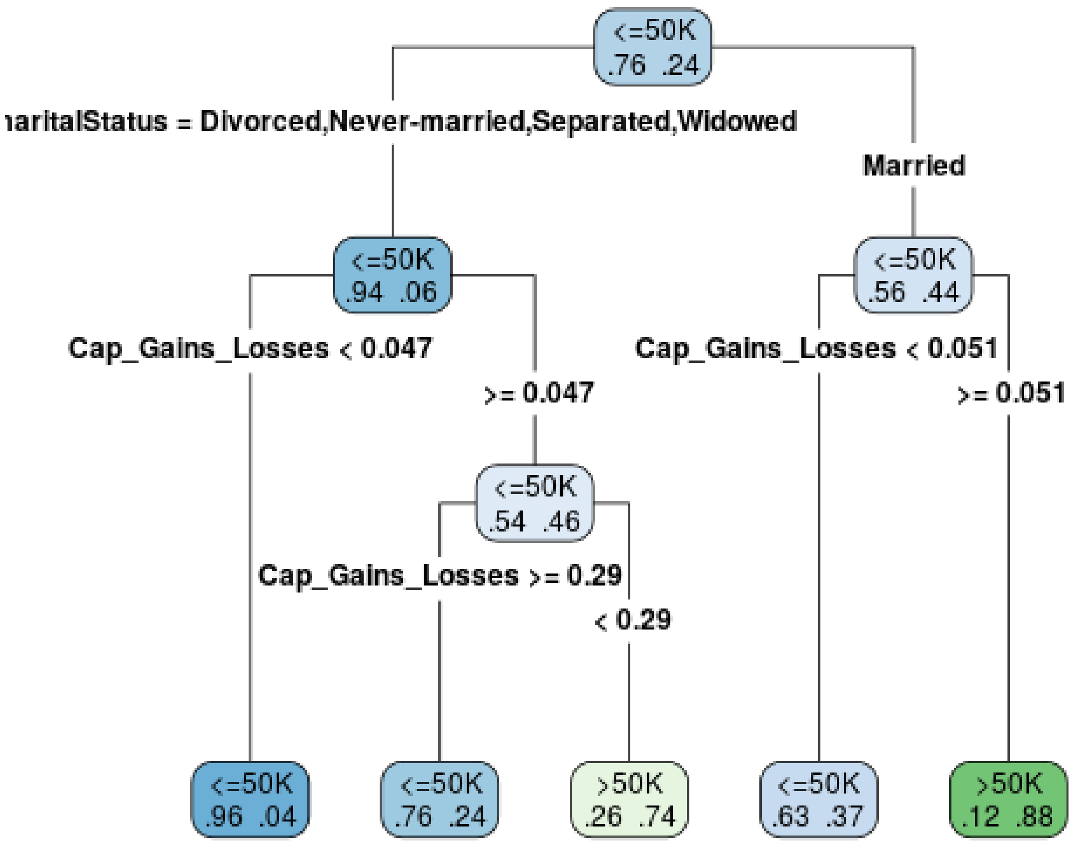
R Code:

[43]:
```
%%R
adult_test <- read_csv("adult_ch6_test")

colnames(adult_test)[1] <- "maritalStatus"

adult_test$Income <- factor(adult_test$Income)
adult_test$maritalStatus <- factor(adult_test$maritalStatus)

head(adult_test)
```

```
R[write to console]: Parsed with column specification:
cols(
  `Marital status` = col_character(),
  Income = col_character(),
  Cap_Gains_Losses = col_double()
)


# A tibble: 6 x 3
  maritalStatus Income Cap_Gains_Losses
  <fct>         <fct>
<dbl>
1 Married       <=50K            0
2 Married       >50K             0.0518
3 Never-married <=50K            0
4 Divorced      >50K             0
```

```
5 Married       >50K            0
6 Married       <=50K           0
```

[44]:
```
%%R
cart02 <- rpart(data = adult_test,
                formula = Income ~ maritalStatus + Cap_Gains_Losses,
                method = "class")

rpart.plot(cart02, type =4, extra = 4 )
```



Answer: The decision trees match each other exactly with the same splits and leaf nodes. The only difference between them is the amount of data avaliable to partition, but regardless both decision trees are identical in structure such root split at the beginning.

2.6 Question 16

Use the training data set to build a C5.0 model to predict income using marital status and capital gians and losses. Specify a minimum of 75 cases per terminal node. Visualize the decision tree. Describe the first few splits in the decision tree.

Python Code:

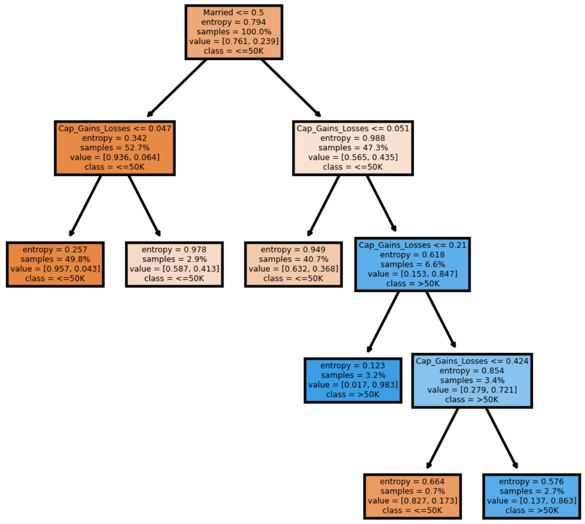
[45]:
```
adult_tr = pd.read_csv("adult_ch6_training")
y = adult_tr[['Income']]
mar_np = np.array(adult_tr['Marital status'])
(mar_cat, mar_cat_dict) = stattools.categorical(mar_np, drop = True, dictnames␣
↪= True)
mar_cat_pd = pd.DataFrame(mar_cat)
X = pd.concat((adult_tr[['Cap_Gains_Losses']], mar_cat_pd), axis = 1)
X_names = [X.columns[0]] + list(mar_cat_dict.values())
y_names = list(y['Income'].unique())
X = X.rename(columns=mar_cat_dict)
X.head()
```

[45]:

	Cap_Gains_Losses	Divorced	Married	Never-married	Separated	Widowed
0	0.02174	0.0	0.0	1.0	0.0	0.0
1	0.00000	1.0	0.0	0.0	0.0	0.0
2	0.00000	0.0	1.0	0.0	0.0	0.0
3	0.00000	0.0	1.0	0.0	0.0	0.0
4	0.00000	0.0	1.0	0.0	0.0	0.0

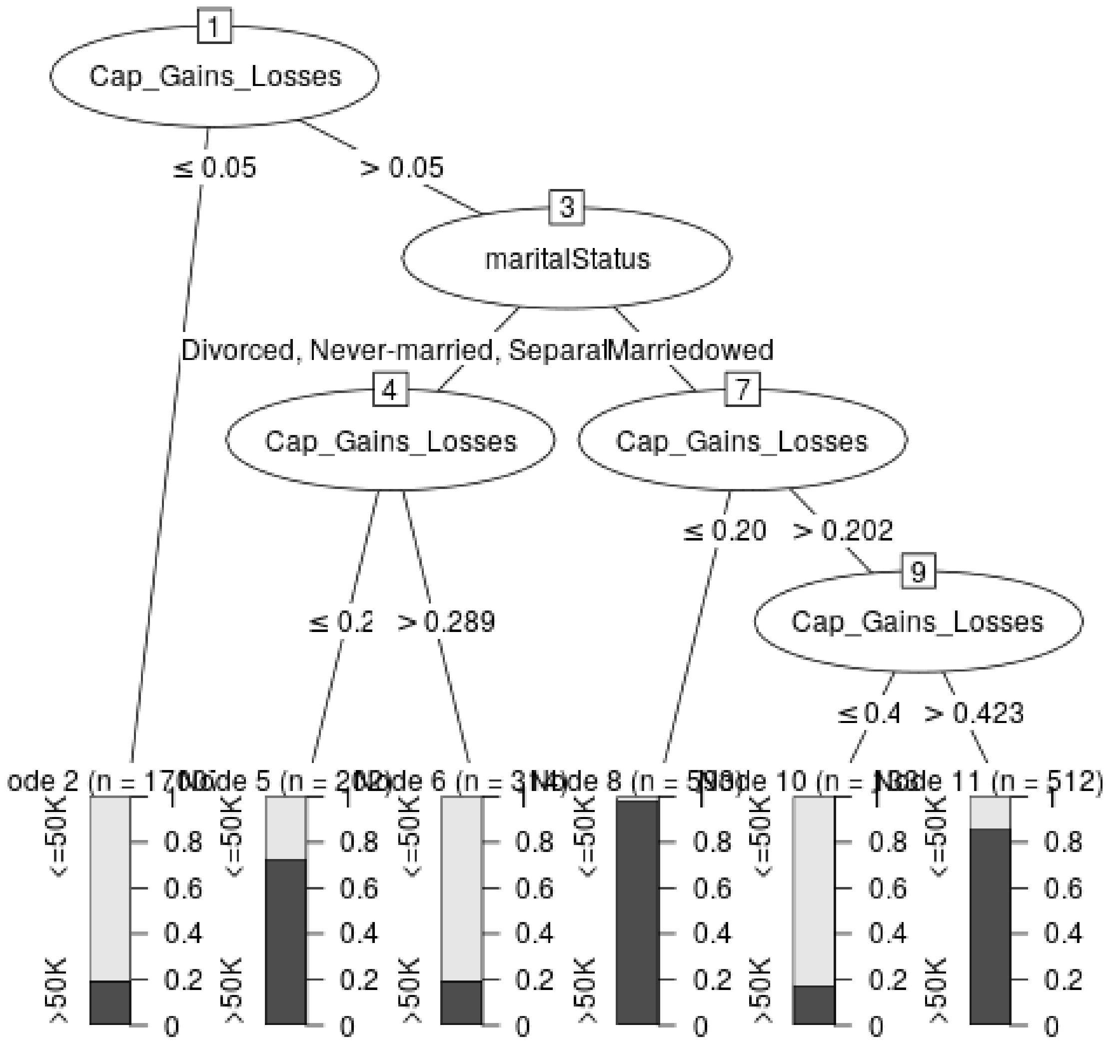
[46]:
```
c50_01 = DecisionTreeClassifier(criterion = "entropy",
                                min_samples_leaf = 75, max_leaf_nodes=6).
↪fit(X,y)
```

[47]:
```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)

tree.plot_tree(c50_01,
               feature_names = X_names,
               class_names=y_names,
               filled = True, proportion = True)

fig.savefig('c50_01_tree.png')
```



R code:

[48]:
```
%%R
C5 <- C5.0(formula = Income ~ maritalStatus + Cap_Gains_Losses, data = adult_tr,
           control = C5.0Control(minCases = 75))
plot(C5)
```



Answer:: At the root node split, we see that we are partitioning whether or not capital gains and losses exceeds 0.05 (>0.05). If it doesn't then we can see right away that adults who have capital gains and losses below 0.05 (<0.05) are low income. The proportion of low incomes are nearly 90% in node 2. Going back to node 1, if the node split condition for the capital gains and losses exceeds 0.05 (>0.05), then we will test the condition of their marital status, asking if they are married or not. As we can see if they are married and moves ahead with node 7, we test the condition if their captail gains and losses lower than or equal to 0.202. If so, we then land on node 8 with the proportion of high income is 90%.

2.7 Question 17

How does your C5.0 model compare to the CART model? Describe the similarities and differences.

Answer:: The first difference we noted was that the root nodes are different. For the first root split in the CART model, we test the condition of marital status to partition the data into binary groups of married vs. everything else. Due to the CART's models nature, it will always optimize binary splitting using the gini index. While the C5.0 model deals with information gain or entropy reduction to find the best optimal splits. The similarities we noted are the outcomes from the test conditions. There are very similar test conditions with the same outcomes such as splitting marital status up into married vs. everything else. Other similarities in test conditions are with capital gains and losses.

3 Data Science Using Python and R: Chapter 11 - Page 165: Questions #34,35,36,37,38,39,40, &41- Regression

3.1 Packages in Python

[49]:
```
from sklearn.metrics import mean_absolute_error
import statsmodels.api as sm
```

3.2 Packages in R

[50]:
```
%%R
library(readr)
library(MASS)
library(MLmetrics)
```

```
R[write to console]:
Attaching package: ‘MLmetrics’


R[write to console]: The following object is masked from ‘package:base’:

    Recall
```

3.3 Datasets

Python Code:

[51]:
```
bank_reg_training = pd.read_csv('bank_reg_training')
bank_reg_training.head()
```

[51]:

	Approval	Credit Score	Debt-to-Income Ratio	Interest	Request Amount
0	F	695.0	0.47	2700.0	6000.0
1	F	775.0	0.03	6300.0	14000.0
2	T	703.0	0.21	3600.0	8000.0
3	T	738.0	0.18	8100.0	18000.0
4	T	685.0	0.16	7650.0	17000.0

[52]:
```
bank_reg_test = pd.read_csv('bank_reg_test')
bank_reg_test.head()
```

[52]:
```
  Approval  Credit Score  Debt-to-Income Ratio  Interest  Request Amount
0        T         767.0                  0.05    2700.0          6000.0
1        F         707.0                  0.05   12150.0         27000.0
2        T         664.0                  0.08     900.0          2000.0
3        F         652.0                  0.05    9000.0         20000.0
4        T         664.0                  0.27    5850.0         13000.0
```

R Code:

[53]:
```
%%R
bank_reg_training <- read_csv("bank_reg_training")
names(bank_reg_training)[names(bank_reg_training) == "Credit Score"] <-␣
 ↪"CreditScore"
names(bank_reg_training)[names(bank_reg_training) == "Debt-to-Income Ratio"] <-␣
 ↪"Debt_to_Income_Ratio"
names(bank_reg_training)[names(bank_reg_training) == "Request Amount"] <-␣
 ↪"RequestAmount"
head(bank_reg_training)
```

```
R[write to console]: Parsed with column specification:
cols(
  Approval = col_logical(),
  `Credit Score` = col_double(),
  `Debt-to-Income Ratio` = col_double(),
  Interest = col_double(),
  `Request Amount` = col_double()
)


# A tibble: 6 x 5
  Approval CreditScore Debt_to_Income_Ratio Interest RequestAmount
  <lgl>          <dbl>
<dbl>    <dbl>
<dbl>
1 FALSE            695                 0.47     2700
6000
2 FALSE            775                 0.03     6300
14000
3 TRUE             703                 0.21     3600
8000
4 TRUE             738                 0.18     8100
18000
5 TRUE             685                 0.16     7650
17000
6 TRUE             725                 0.31     8550
```

```
19000
```

[54]:
```
%%R
bank_reg_test <- read_csv("bank_reg_test")
names(bank_reg_test)[names(bank_reg_test) == "Credit Score"] <- "CreditScore"
names(bank_reg_test)[names(bank_reg_test) == "Debt-to-Income Ratio"] <-
 ↪"Debt_to_Income_Ratio"
names(bank_reg_test)[names(bank_reg_test) == "Request Amount"] <-
 ↪"RequestAmount"
head(bank_reg_test)
```

```
R[write to console]: Parsed with column specification:
cols(
  Approval = col_logical(),
  `Credit Score` = col_double(),
  `Debt-to-Income Ratio` = col_double(),
  Interest = col_double(),
  `Request Amount` = col_double()
)


# A tibble: 6 x 5
  Approval CreditScore Debt_to_Income_Ratio Interest RequestAmount
  <lgl>          <dbl>
<dbl>    <dbl>
<dbl>
1 TRUE             767                 0.05     2700
6000
2 FALSE            707                 0.05    12150
27000
3 TRUE             664                 0.08      900
2000
4 FALSE            652                 0.05     9000
20000
5 TRUE             664                 0.27     5850
13000
6 TRUE             725                 0.23     8550
19000
```

3.4 Question 34

Use the training set to run a regression predicting *Credit Score*, based on *Debt-to-Income Ratio* and *Request Amount*. Obtain a summary of the model. Do both predictors belong in the model?

Python Code:

[55]:
```
X = bank_reg_training[['Debt-to-Income Ratio', 'Request Amount']]
y = bank_reg_training[['Credit Score']]
X = sm.add_constant(X)
```

[56]:
```
model01 = sm.OLS(y, X).fit()
model01.summary()
```

[56]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           Credit Score   R-squared:                       0.028
Model:                            OLS   Adj. R-squared:                  0.028
Method:                 Least Squares   F-statistic:                     156.2
Date:                Mon, 15 Mar 2021   Prob (F-statistic):           1.37e-67
Time:                        21:46:58   Log-Likelihood:                -59972.
No. Observations:               10693   AIC:                         1.199e+05
Df Residuals:                   10690   BIC:                         1.200e+05
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
const                  668.4562      1.336    500.275      0.000     665.837
671.075
Debt-to-Income Ratio   -48.1262      4.785    -10.058      0.000     -57.505
-38.747
Request Amount           0.0011   6.84e-05     15.727      0.000       0.001
0.001
==============================================================================
Omnibus:                     1658.575   Durbin-Watson:                   1.991
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2844.250
Skew:                          -1.021   Prob(JB):                         0.00
Kurtosis:                       4.487   Cond. No.                     1.24e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.24e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

R Code:

[57]:
```
%%R
model01 <- lm(formula = CreditScore ~ Debt_to_Income_Ratio + RequestAmount,
              data = bank_reg_training)
```

```
summary(model01)
```

```
Call:
lm(formula = CreditScore ~ Debt_to_Income_Ratio + RequestAmount,
    data = bank_reg_training)

Residuals:
    Min      1Q  Median      3Q     Max
-279.13  -25.11   10.87   39.93  175.32

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           6.685e+02  1.336e+00  500.27   <2e-16 ***
Debt_to_Income_Ratio -4.813e+01  4.785e+00  -10.06   <2e-16 ***
RequestAmount         1.075e-03  6.838e-05   15.73   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 66 on 10690 degrees of freedom
Multiple R-squared:  0.02839,   Adjusted R-squared:  0.02821
F-statistic: 156.2 on 2 and 10690 DF,  p-value: < 2.2e-16
```

Answer:

- Since both predictors does not have p-values lower than the cutoff (0.05), we are able to retain those variables in the training set without omitting any. (By the way, we are not make any statistical inference, but rather a different approach to cross-validate with the test set)

3.5 Question 35

Validate the model from the previous exercise.

Python Code:

[58]:
```
X_test = bank_reg_test[['Debt-to-Income Ratio', 'Request Amount']]
X_test = sm.add_constant(X_test)
```

[59]:
```
y_test = bank_reg_test[['Credit Score']]
```

[60]:
```
model01_test = sm.OLS(y_test, X_test).fit()
```

[61]:
```
model01_test.summary()
```

[61]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:            Credit Score   R-squared:                       0.038
Model:                             OLS   Adj. R-squared:                  0.038
Method:                  Least Squares   F-statistic:                     215.4
Date:                 Mon, 15 Mar 2021   Prob (F-statistic):           1.94e-92
Time:                         21:46:58   Log-Likelihood:                -60395.
No. Observations:                10775   AIC:                         1.208e+05
Df Residuals:                    10772   BIC:                         1.208e+05
Df Model:                            2
Covariance Type:             nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
const                  665.4987      1.328    501.265      0.000     662.896
668.101
Debt-to-Income Ratio   -52.1374      4.826    -10.803      0.000     -61.597
-42.677
Request Amount           0.0013   6.85e-05     19.013      0.000       0.001
0.001
==============================================================================
Omnibus:                     1792.693   Durbin-Watson:                   1.985
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3194.120
Skew:                          -1.067   Prob(JB):                         0.00
Kurtosis:                       4.600   Cond. No.                     1.25e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.25e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

R code:

[62]:
```
%%R
model01_test <- lm(formula = CreditScore ~ Debt_to_Income_Ratio + RequestAmount,
                   data = bank_reg_test)
summary(model01_test)
```

```
Call:
lm(formula = CreditScore ~ Debt_to_Income_Ratio + RequestAmount,
    data = bank_reg_test)

Residuals:
```

```
    Min      1Q  Median      3Q     Max 
-288.16  -24.49   11.08   39.47  199.84 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)           6.655e+02  1.328e+00  501.26   <2e-16 ***
Debt_to_Income_Ratio -5.214e+01  4.826e+00  -10.80   <2e-16 ***
RequestAmount         1.302e-03  6.849e-05   19.01   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 65.78 on 10772 degrees of freedom
Multiple R-squared:  0.03845,   Adjusted R-squared:  0.03827 
F-statistic: 215.4 on 2 and 10772 DF,  p-value: < 2.2e-16
```

Answer:

- Since the p-values in the test does not exceed the cutoff, we can say with certainty that these variables do work and should stay. (By the way, we are not make any statistical inference, but rather a different approach to cross-validate with the test set)

3.6 Question 36

Use the regression equation to complete this sentence: “The estimated Credit Score equals…”

Answer: *The estimated Credit Score equals* **(from training data summary)**

- *Credit Score = 668.4562 dollars -48.1262(Debt-to-Income Ratio) + 0.0011(Request Amount)*
- 668.46 dollars minus 48.13 dollars times the debt-to-income ratio plus 0.0011 times the amount they requested to borrow.

3.7 Question 37

Interpret the coefficient for *Debt_to_Income Ratio*.

Answer: - The estimated increase in Credit Score for a customer is due to a **decrease** of their Debt-to-Income Ratio by 48.13 dollars when the Request Amount is held constant.

3.8 Question 38

Interpret the coefficient for Request Amount

Answer: - For each increase of a requested amount in the average of requested amounts, the estimated increase in Credit Score is by 0.0013, when their Debt-To-Income ratio is held constant.

3.9 Question 39

Find and interpret the value of s.

Python Code:

[63]:
```
np.sqrt(model01.scale)
```

[63]: 66.00195259717187

R Code:

[64]:
```
%%R
summary(model01)
```

```
Call:
lm(formula = CreditScore ~ Debt_to_Income_Ratio + RequestAmount,
    data = bank_reg_training)

Residuals:
    Min      1Q  Median      3Q     Max
-279.13  -25.11   10.87   39.93  175.32

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           6.685e+02  1.336e+00  500.27   <2e-16 ***
Debt_to_Income_Ratio -4.813e+01  4.785e+00  -10.06   <2e-16 ***
RequestAmount         1.075e-03  6.838e-05   15.73   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 66 on 10690 degrees of freedom
Multiple R-squared:  0.02839,   Adjusted R-squared:  0.02821
F-statistic: 156.2 on 2 and 10690 DF,  p-value: < 2.2e-16
```

Answer: - Here s = 66.00, meaning the size of the model's typical prediction error is about 66.00 for credit score.

4 Question 40

Find and interpret R-squared adjusted. Comment

Python Code:

[65]:
```
model01_test.summary()
```

[65]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           Credit Score   R-squared:                       0.038
Model:                            OLS   Adj. R-squared:                  0.038
Method:                 Least Squares   F-statistic:                     215.4
```

```
Date:                Mon, 15 Mar 2021   Prob (F-statistic):          1.94e-92
Time:                        21:46:58   Log-Likelihood:               -60395.
No. Observations:               10775   AIC:                        1.208e+05
Df Residuals:                   10772   BIC:                        1.208e+05
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
========
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
const                665.4987      1.328    501.265      0.000     662.896
668.101
Debt-to-Income Ratio  -52.1374      4.826    -10.803      0.000     -61.597
-42.677
Request Amount         0.0013   6.85e-05     19.013      0.000       0.001
0.001
==============================================================================
Omnibus:                     1792.693   Durbin-Watson:                   1.985
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3194.120
Skew:                          -1.067   Prob(JB):                         0.00
Kurtosis:                       4.600   Cond. No.                     1.25e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.25e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[66]:
```
ytrue = bank_reg_test[['Credit Score']]
ypred = model01.predict(X_test)
print("R-squared adjusted:", round(0.038 * 100,1), "%")
```

```
R-squared adjusted: 3.8 %
```

R Code:

[67]:
```
%%R
summary(model01_test)
```

```
Call:
lm(formula = CreditScore ~ Debt_to_Income_Ratio + RequestAmount,
    data = bank_reg_test)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-288.16  -24.49   11.08   39.47  199.84

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           6.655e+02  1.328e+00  501.26   <2e-16 ***
Debt_to_Income_Ratio -5.214e+01  4.826e+00  -10.80   <2e-16 ***
RequestAmount         1.302e-03  6.849e-05   19.01   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 65.78 on 10772 degrees of freedom
Multiple R-squared:  0.03845,   Adjusted R-squared:  0.03827
F-statistic: 215.4 on 2 and 10772 DF,  p-value: < 2.2e-16
```

Answer: R squared adjusted is approximately equal to 3.8% as this interprets the proportion of the variability in the response by the variables used in this model: Debt-to-Income ratio and Request Amount. This means that the variables used in this model suggest that there are other factors involved affecting credit scores.

5 Question 41

Find the MAE baseline and MAE regression, and determine whether the regression model outperformed its baseline model.

Python Code:

[68]:
```python
ytrue = bank_reg_test[['Credit Score']]
ypred = model01.predict(X_test)

y_mean = list(bank_reg_test[['Credit Score']].mean()) *␣
↪len(bank_reg_test[['Credit Score']])

MAE_Baseline = mean_absolute_error(y_true = ytrue, y_pred = y_mean)
MAE_Baseline
```

[68]: 48.60024069637869

[69]:
```python
MAE_Regression = mean_absolute_error(y_true = ytrue, y_pred = ypred)
MAE_Regression
```

[69]: 47.79066993781929

[70]:
```python
MAE_Regression < MAE_Baseline
```

[70]: True

R code:

[71]:
```
%%R
y_mean <- rep(mean(bank_reg_test$CreditScore),
              times= length(bank_reg_test$CreditScore))
ytrue <- bank_reg_test$CreditScore

MAE_Baseline <- MAE(y_pred = y_mean, y_true = ytrue)
MAE_Baseline
```

```
[1] 48.60024
```

[72]:
```
%%R
X_test <-data.frame(Debt_to_Income_Ratio = bank_reg_test$Debt_to_Income_Ratio,
                    RequestAmount =bank_reg_test$RequestAmount)
ypred = predict(object = model01, newdata = X_test)

MAE_Regression <- MAE(y_pred = ypred, y_true = ytrue)
MAE_Regression
```

```
[1] 47.79067
```

[73]:
```
MAE_Regression < MAE_Baseline
```

[73]:
```
True
```

Answer: Since $47.79 =$ MAE for regression is less than $48.70 =$ MAE baseline, then our regression model did beat the baseline model.