Gabi Rivera

05Nov2022

ADS502-01

Exercise Questions:

Introduction to Data Mining – Exercise 3.11

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

Table 3.6. Data set for Exercise 4.

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	−

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

Answer: The entropy is 0.991.

Entropy = -(4/9)log2(4/9) – (5/9)log2(5/9)

b. What are the information gains of a1 and a2 relative to these training examples?

Answer: The info gain of a1 is 0.229 and a2 is 0.007.

A1 = 0.9911 – [(4/9)[-(3/4)log2(3/4)-(1/4)log3(1/4)] + (5/9)[-(1/5)log2(1/5)-(4/5)log2(4/5)]]

A2 = 0.9911 – [(5/9)[-(2/5)log2(2/5)-(3/5)log3(3/5)] + (4/9)[-(2/4)log2(2/4)-(2/4)log2(2/4)]]

c. For a3, which is a continuous attribute, compute the information gain for every possible split.

Answer:

a3	Possible Split	Entropy	Info Gain
1	(1+3)/2 = 2	0.848	0.143
3	(3+4)/2 = 3.5	0.989	0.003
4	(4+5)/2 = 4.5	0.918	0.073
5	(5+6)/2 = 5.5	0.984	0.007
5	(5+6)/2 = 5.5	0.984	0.007
6	(6+7)/2 = 6.5	0.973	0.018
7	(7+8)/2 = 7.5	0.889	0.102
7	(7+8)/2 = 7.5	0.889	0.102

d. What is the best split (among a1, a2, and a3) according to the information gain?

Answer: a1 has the best split at 0.229.

a1 = 0.229, a2 = 0.007, and a3 = 0.143 (highest)

e. What is the best split (between a1 and a2) according to the misclassification error rate?

Answer: The best split by misclassification error rate is a1 at 0.222.

		Positive	Negative
a1	True	3	1
a1	False	1	4
a2	True	2	3
a2	False	2	2

a1 Error rate = wrong predictions/total predictions = 2/9

a2 Error rate = 4/9

f. What is the best split (between a1 and a2) according to the Gini index?

Answer: a1 at 0.344 has the best split.

$A1 = 4/9[1 – (3/4)^2 – (1/4)^2] + 5/9[1 – (1/5)^2 – (4/5)^2] = 0.344$

$A2 = 5/9[1 – (2/5)^2 – (3/5)^2] + 4/9[1 – (2/4)^2 – (2/4)^2] = 0.489$

Module-2 Assignment

Gabi Rivera

2022-11-07

Data Science Using Python and R: Chapter 4 hands-on analysis

Load Libraries:

```
library(tidyverse)
library(skimr)
```

Import Bank Marketing Training dataset:

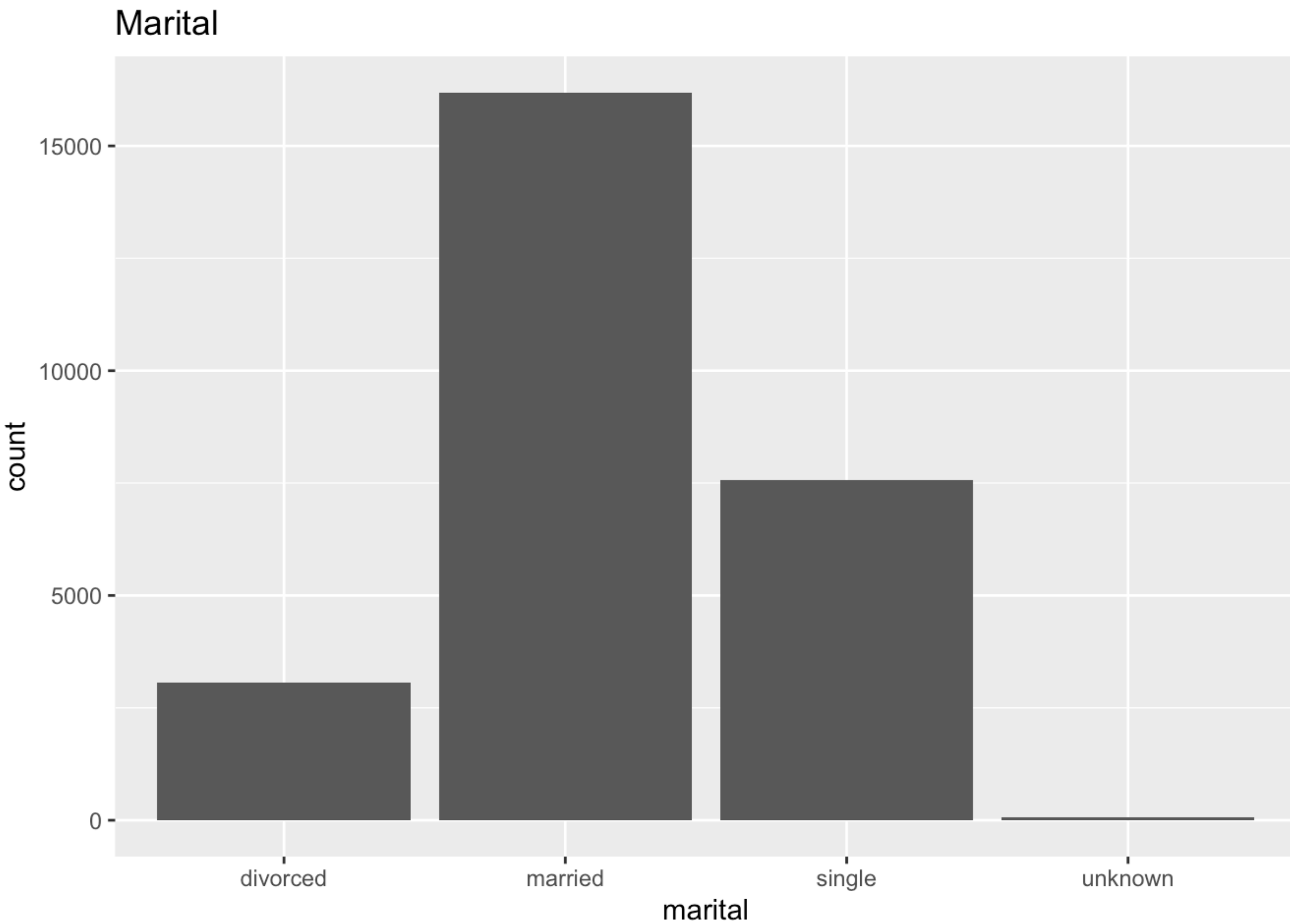
```
 bm_t = read.csv("bank_marketing_training", header = TRUE, sep = ",")
```

Question 21. What is the strength of each graph? Weakness?

Each graph is showing different perspective. The marital bar graph shows the overall view without the response attribute. The marital with overlayed response provides us with the original distribution and the proportion of responses across the marital statuses. We can clearly see which marital status responded with higher frequency of no and yes compared to each other. The normalized stacked view provides us with the proportion of responses in each marital status. So, we can tell that the proportion of no response across each marital status is relatively the same.

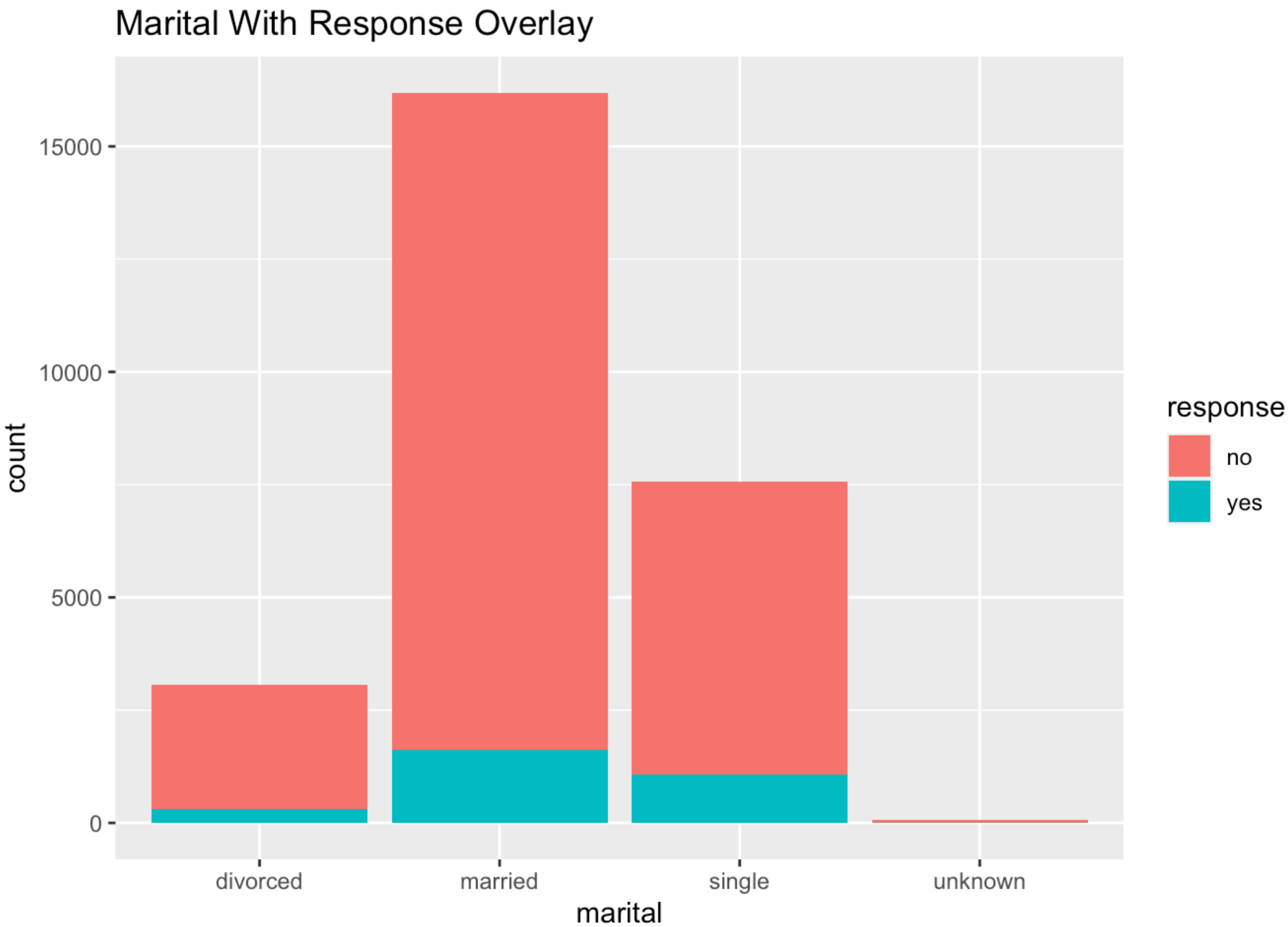
a. Bar graph of marital.

```
ggplot(bm_t, aes(marital)) + geom_bar() + ggtitle("Marital")
```



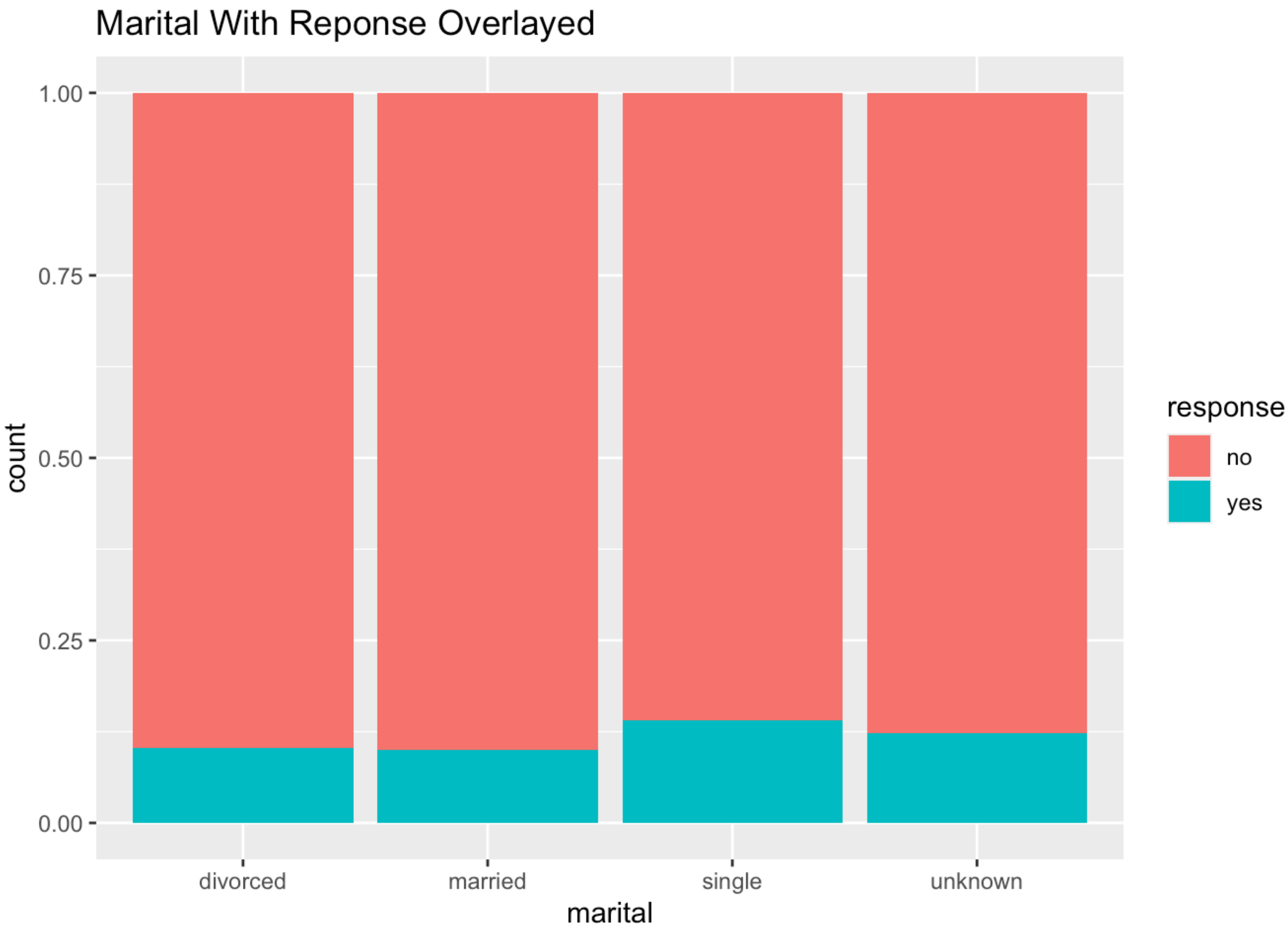
b. Bar graph of marital, with overlay of response.

```
ggplot(bm_t, aes(marital)) + geom_bar(aes(fill = response)) +
  ggtitle("Marital With Response Overlay")
```



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

```
ggplot(bm_t, aes(marital)) +
  geom_bar(aes(fill = response), position = 'fill') +
  ggtitle("Marital With Reponse Overlayed")
```



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

Across each marital statuses, no is the overwhelming response with relatively similar frequency. Only single marital statuses have a slight gain on yes response. But overall, yes and no response are relatively similar across marital status.

23. Do the following with the variables marital and response.

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

Marital and Response contingency table:

```
mare = table(bm_t$response, bm_t$marital)
head(mare)
```

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

Count Report:

```
mare_2 = addmargins(A = mare, FUN = list(total = sum), quiet = TRUE)
head(mare_2)
```

```
##
##         divorced married single unknown total
##   no        2743   14579   6514      50 23886
##   yes        312    1608   1061       7  2988
##   total     3055   16187   7575      57 26874
```

Column percentage added:

```
round(prop.table(mare, margin = 2)*100, 2)
```

```
##
##       divorced married single unknown
##   no     89.79   90.07  85.99   87.72
##   yes    10.21    9.93  14.01   12.28
```

```
head(mare_2)
```

```
##
##         divorced married single unknown total
##   no        2743   14579   6514      50 23886
##   yes        312    1608   1061       7  2988
##   total     3055   16187   7575      57 26874
```

b. Describe what the contingency table is telling you.

This contingency table is showing that the highest frequency for divorced status is the no response at 89.79% against the yes response at 10.21%. This is the same for married status which has 90.07% no response against 9.93% yes response. Overall, each marital status has no response as the highest proportion of responses.

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

For this one, the highest proportion of no response is from the married status at 61.04% and the lowest from unknown at 0.21%. The highest proportion of yes response is from the married status again at 53.82% while the lowest is at 0.23$ unknown status. The previous contingency table calculates the percentage by column of each marital statuses while this contingency table calculates the percentage by row of each responses.

Row percentage:

```
round(prop.table(mare, margin = 1)*100, 2)
```

```
##
##       divorced married single unknown
##   no     11.48   61.04  27.27    0.21
##   yes    10.44   53.82  35.51    0.23
```

```
head(mare_2)
```

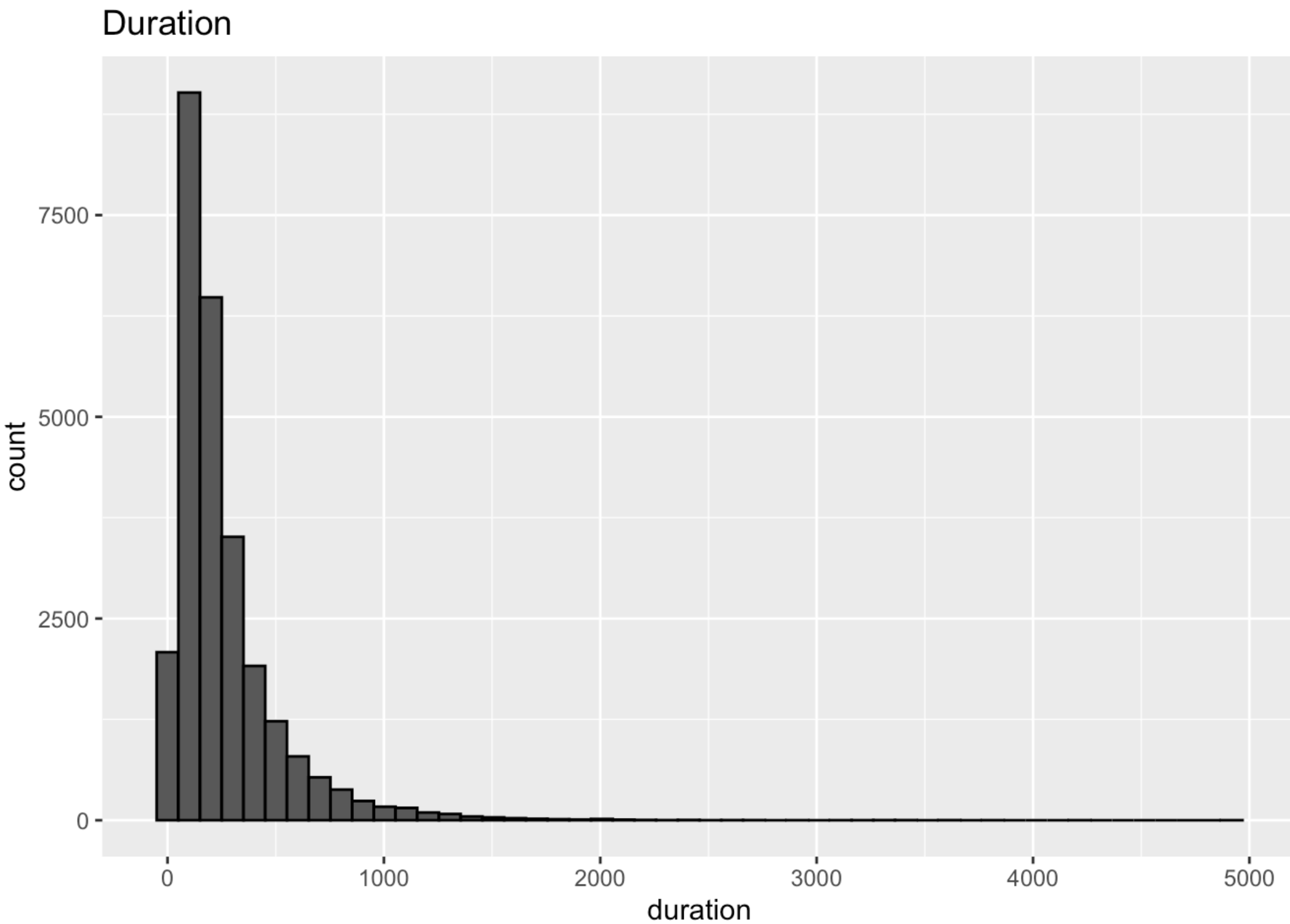
```
##
##         divorced married single unknown total
##   no        2743   14579   6514      50 23886
##   yes        312    1608   1061       7  2988
##   total     3055   16187   7575      57 26874
```

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

Duration histogram provides us with the count distribution of duration. Duration with overlayed response provides us with the original distribution and added proportion of yes and no response across duration. We can tell that no responses are more frequent at the beginning duration as it peaks and the trend goes down. The normalized graph provides the proportion of response at each duration instances. We can more finely say that no responses happen at the beginning duration then yes response seems to gain in proportion as the duration increases.

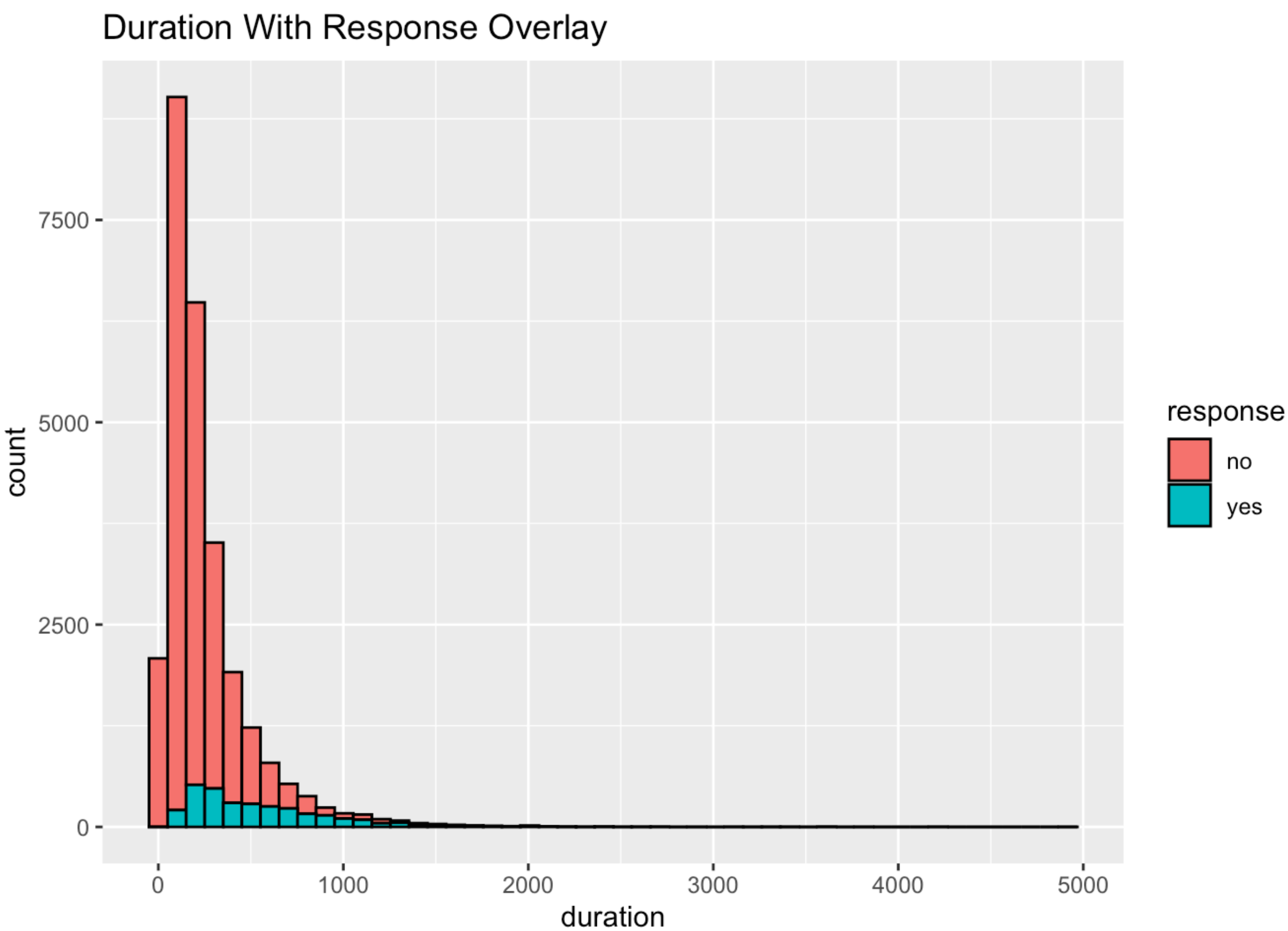
a. Histogram of duration.

```
ggplot(bm_t, aes(duration)) + geom_histogram(color="black", bins = 50) +
ggtitle("Duration")
```



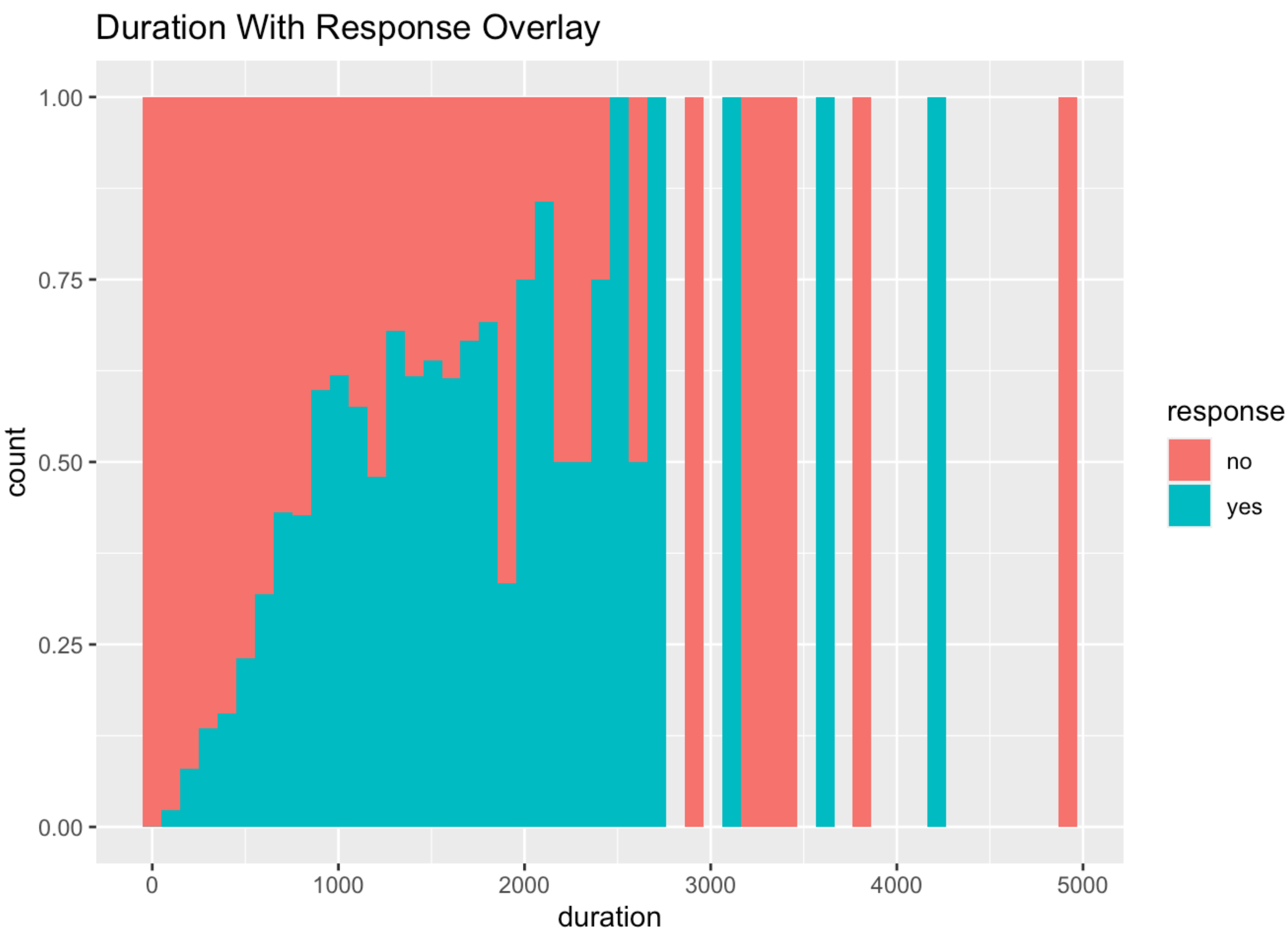
b. Histogram of duration, with overlay of response.

```
ggplot(bm_t, aes(duration)) + geom_histogram(aes(fill = response),color="black", bins
= 50) +
ggtitle("Duration With Response Overlay")
```



c. Normalized histogram of duration, with overlay of response.

```
ggplot(bm_t, aes(duration)) + geom_histogram(aes(fill = response),
    position = "fill", bins = 50) + ggtitle("Duration With Response Overlay")
```



Data Science Using Python and R: Chapter 6 hands-on analysis

Libraries:

```
library(rpart)
library(rpart.plot)
```

Import adult_ch6 Training dataset:

```
 ad_t = read.csv("adult_ch6_training", header = TRUE, sep = ",")

colnames(ad_t)[1] = "maritalStatus"
 ad_t$Income = factor(ad_t$Income)
 ad_t$maritalStatus = factor(ad_t$maritalStatus)
```

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.

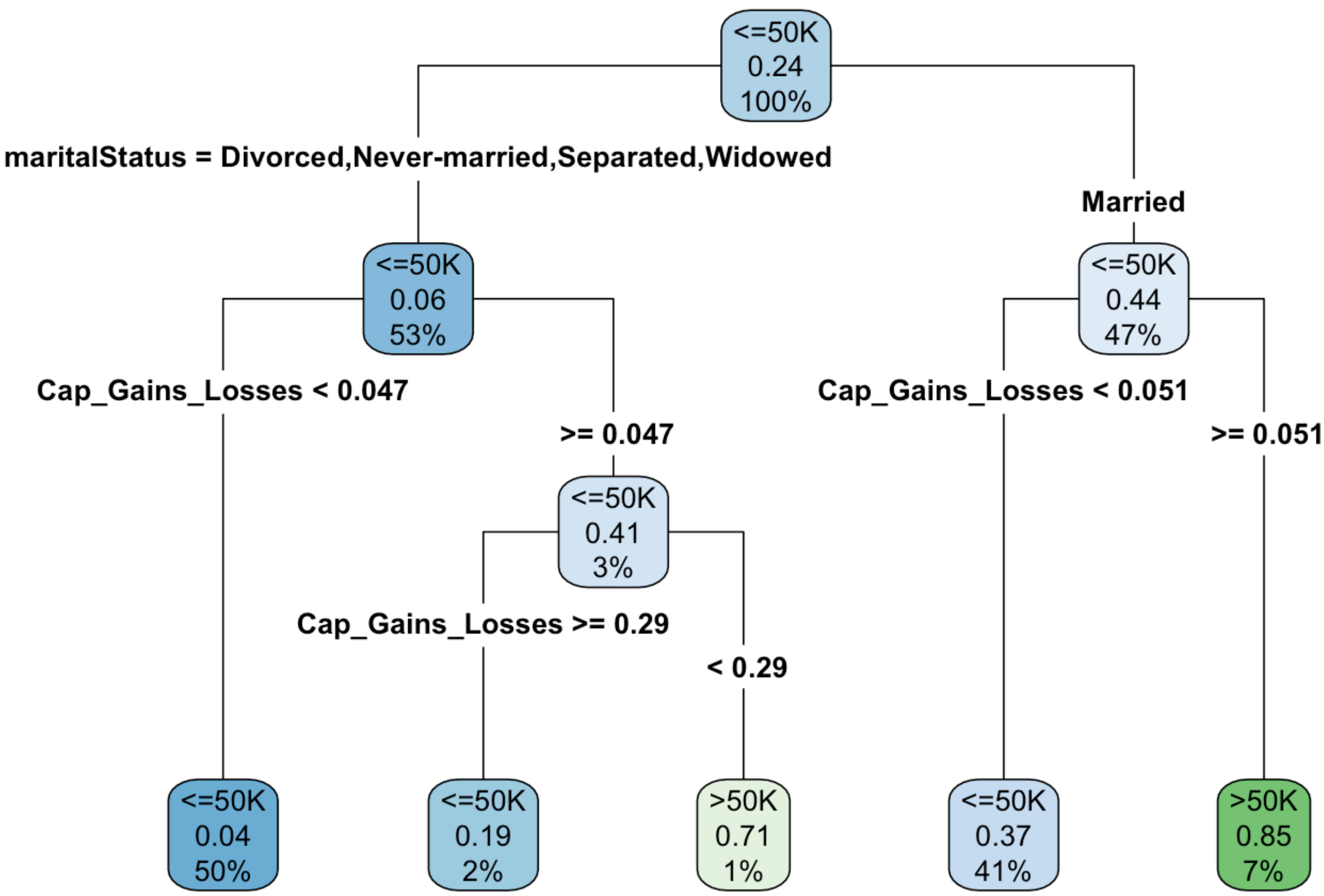
The top 100% is the root node where all of the data records are subjected against down to each of the internal nodes starting from the first split with 47% married status and 53% to others. It also tells us that 24% of the root node or the dataset population earns less than 50K. If we follow down the married status, the next split is determined by capital gain and losses where 41% earns less than 50K in income and 7% earns greater than 50K.

Cart Model: Training dataset

```
cart01 = rpart(formula = Income ~ maritalStatus + Cap_Gains_Losses,
               data = ad_t, method = "class")
```

Training Plot: Predicting income using marital and capital gains and losses

```
rpart.plot(cart01, type = 4, extra = 106)
```



15. Develop 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? Yes, the test dataset and the training dataset are almost identical.

Test data:

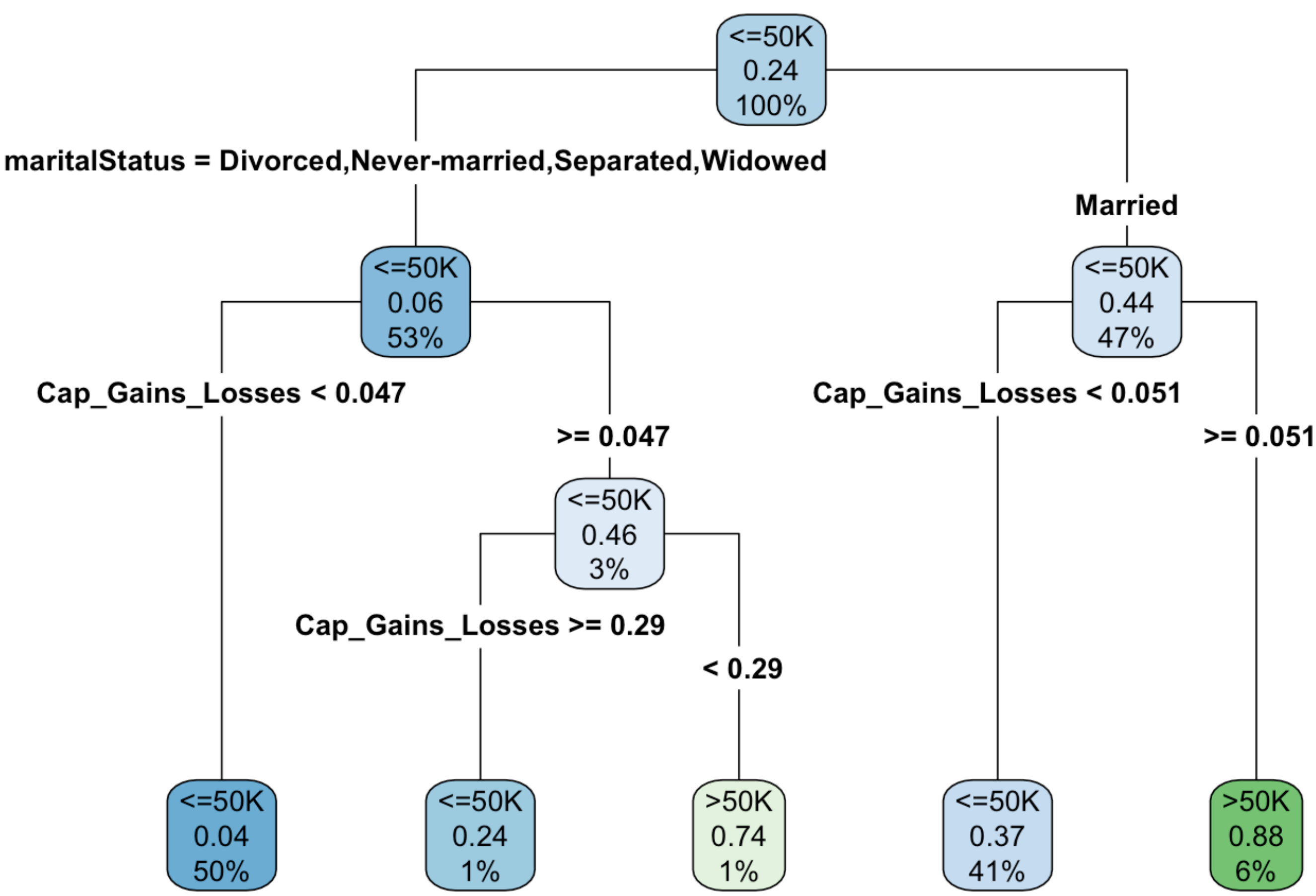
```
ad_test = read.csv("adult_ch6_test", header = TRUE, sep = ",")
colnames(ad_test)[1] = "maritalStatus"
ad_test$Income = factor(ad_test$Income)
ad_test$maritalStatus = factor(ad_test$maritalStatus)
```

CART Model: Test dataset

```
cart02 = rpart(formula = Income ~ maritalStatus + Cap_Gains_Losses,
               data = ad_test, method = "class")
```

Test Plot: Predicting income using marital and capital gains and losses

```
rpart.plot(cart02, type = 4, extra = 106)
```



16. Use the training data set to build a C5.0 model to predict income using marital status and capital gains and losses. Specify a minimum of 75 cases per terminal

node. Visualize the decision tree. Describe the first few splits in the decision tree.

The root nodes starts with capital gain and lose with split at less than and greater than 0.05. Less than 0.05 gain/losses immediately terminates at leaf node 2 where majority of the population are low income earners. Greater than 0.05 gains/losses leads to node 3 which splits the dataset based on marital status. If married, then the population are split again based on capital gains/losses at node 7.

Library:

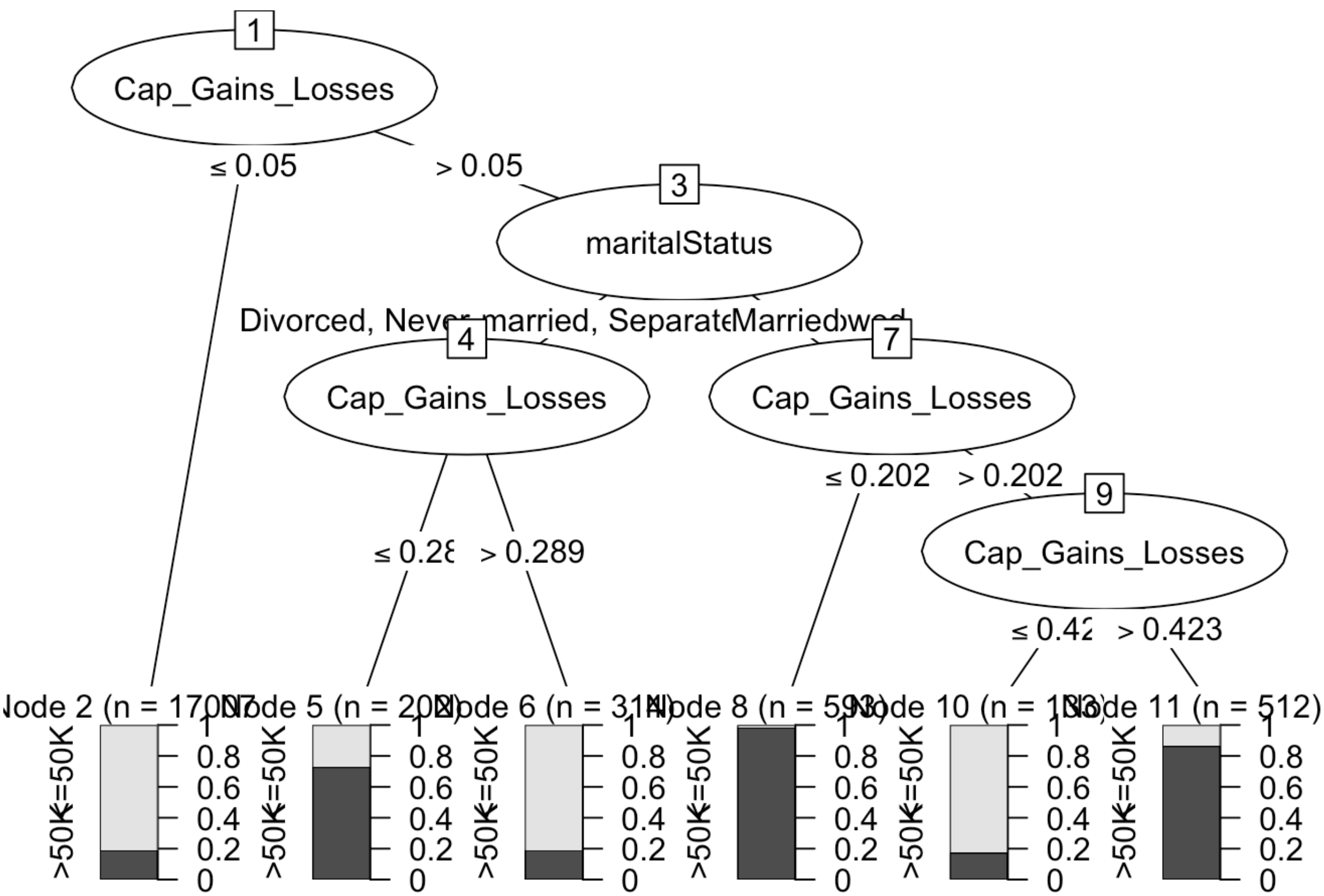
```
library(C50)
```

C5.0 model: Training dataset

```
C5 = C5.0(formula = Income ~ maritalStatus + Cap_Gains_Losses,
          data = ad_t, control = C5.0Control(minCases=75))
```

Training Plot: Predicting income using marital and capital gains and losses

```
plot(C5)
```



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

CART model decision tree splits the nodes in two compared to C5.0 which determines an optimal split value instead. So the CART plot looks more balanced compared to the C5.0 plot because it can accommodate multi-branches. In addition to having split nodes, C5.0 model have leaf nodes that shows bars of income attribute with n count compared to a simple summary stat shown in CART leaf nodes.

Data Science Using Python and R: Chapter 11 hands-on analysis

Import bank datasets:

```
bank_train = read.csv("bank_reg_training", header = TRUE, sep = ",")
bank_test = read.csv("bank_reg_test", header = TRUE, sep = ",")
```

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?

Yes, both predictors belong in the model. The Std Error and the t-value of the estimates are all relatively small. Both p-values of the parameters are less than 0.05 alpha which means that the values are statistically significant.

Train regression: Credit score based on Debit-to-income ratio and request amount

```
model01 = lm(formula = Credit.Score ~ Debt.to.Income.Ratio +
               Request.Amount, data = bank_train)
```

Summary: Training dataset

```
summary(model01)
```

```
##
## Call:
## lm(formula = Credit.Score ~ Debt.to.Income.Ratio + Request.Amount,
##     data = bank_train)
##
## 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 ***
## Request.Amount        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
```

35. Validate the model from the previous exercise.

Validate regression: Credit score based on Debit-to-income ratio and request amount

```
model01_test = lm(formula = Credit.Score ~ Debt.to.Income.Ratio +
               Request.Amount, data = bank_test)
```

Summary: Test dataset

```
summary(model01_test)
```

```
##
## Call:
## lm(formula = Credit.Score ~ Debt.to.Income.Ratio + Request.Amount,
##     data = bank_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 ***
## Request.Amount        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
```

36. Use the regression equation to complete this sentence: "The estimated Credit Score equals…."

Test bank dataset: The estimated Credit score equals 665.5 - 52.1(Debt-to-Income ratio) plus 0.001(Request Amount).

Training bank dataset: The estimated Credit score equals 668.5 - 48.1(Debt-to-Income ratio) plus 0.001(Request Amount).

37. Interpret the coefficient for Debt-to-Income Ratio.

The coefficient for debt-to-income ratio of the training is -48.1 which is relatively close to -52.1 coefficient of the test dataset. Both p-values of the parameters are less than 0.05 alpha which means that the values are statistically significant.

38. Interpret the coefficient for Request Amount.

The coefficient for request amount of the training and the test dataset are relatively the same at 0.001. Both p-values of the parameters are less than 0.05 alpha which means that the values are statistically significant.

39. Find and interpret the value of s.

The residual standard error for the training dataset is 66 while the test dataset has 65.78. Both are relatively the same and they are essentially very small in value meaning that the residual have small variance.

40. Find and interpret R^2 adj. Comment.

The adjusted r^2 for the training dataset is 0.02821 while the test dataset has 0.03827. Both are slightly lower than their multiple r^2 counterpart. But even so the results is that the parameters can only explain ~3-4% of the natural variance in the y variable. This is a very low or weak score.

41. Find MAE(Baseline) and MAE(Regression), and determine whether the regression model outperformed its baseline model.

MAE of baseline is 48.44 while MAE of regression is 47.79. Since the regression model is lower compared to the baseline, it outperforms the baseline.

Package:

```
library(MLmetrics)
```

Actual values:

```
X_test = data.frame(Debt.to.Income.Ratio = bank_test$Debt.to.Income.Ratio,
              Request.Amount= bank_test$Request.Amount)
```

Predicted values:

```
ypred = predict(object = model01, newdata = X_test)
ytrue = bank_test$Credit.Score
```

Mean absolute error: Regression

```
MAE(y_pred = ypred, y_true = ytrue)
```

```
## [1] 47.79067
```

MAE: baseline

```
ymean = mean(bank_train$Credit.Score)
MAE(y_pred= ymean, y_true = ytrue)
```

```
## [1] 48.44154
```

Module 2 Assignment

Gabi Rivera || 13Oct2022 || ADS501-01

In [83]:
```
import os
os.getcwd()
```

Out[83]:
```
'/Users/gabirivera/Desktop/MSADS2/ADS502-01/Module2/Assignment'
```

In [84]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Data Science Using Python and R: Chapter 4 hands-on analysis

Question 21. What is the strength of each graph? Weakness?

a. Bar graph of marital.

In [2]:
```
bm_train= pd.read_csv('bank_marketing_training', sep = ',')
bm_train.head(5)
```

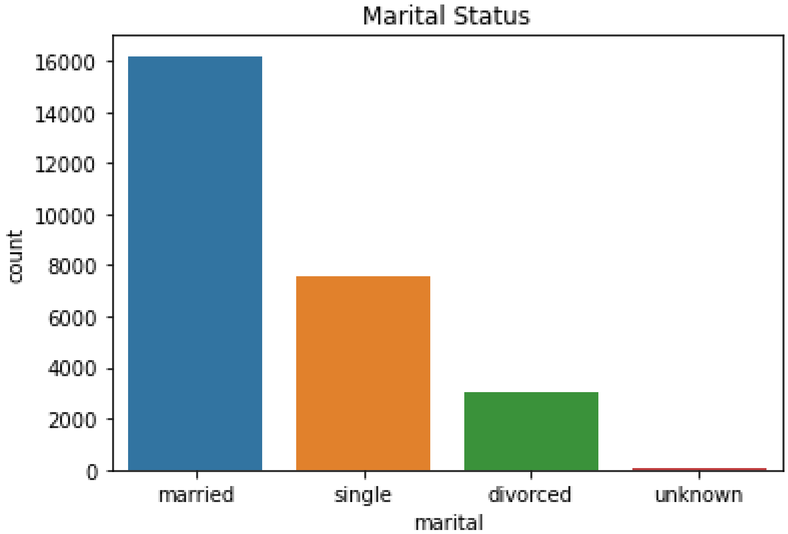
Out[2]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	cam
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	
2	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	
3	25	services	single	high.school	no	yes	no	telephone	may	mon	...	
4	29	blue-collar	single	high.school	no	no	yes	telephone	may	mon	...	

5 rows × 21 columns

In [3]:
```
sns.countplot(x= bm_train["marital"]).set(title='Marital Status')
```

Out[3]:
```
[Text(0.5, 1.0, 'Marital Status')]
```

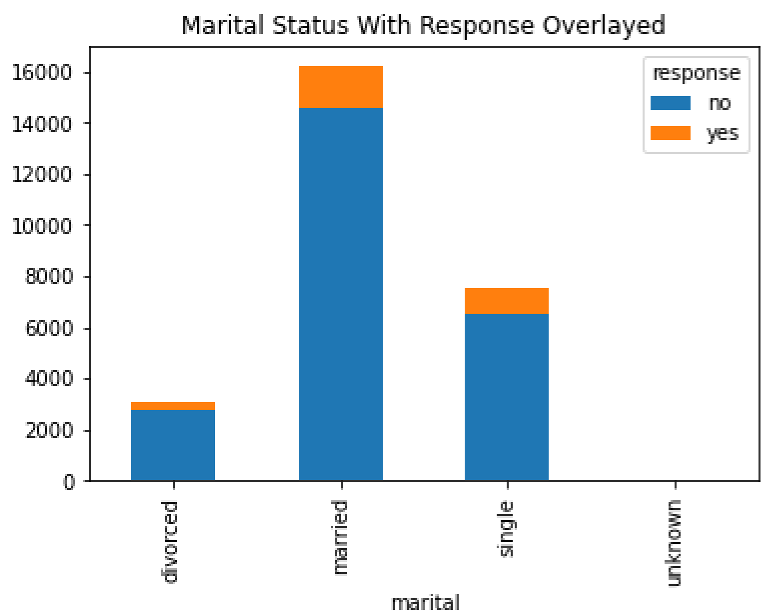


b. Bar graph of marital, with overlay of response.

In [4]:
```
mares = pd.crosstab(bm_train['marital'], bm_train['response'])
```

In [5]:
```
mares.plot(kind = 'bar', stacked = True)
plt.title('Marital Status With Response Overlayed')
```

Out[5]:
```
Text(0.5, 1.0, 'Marital Status With Response Overlayed')
```

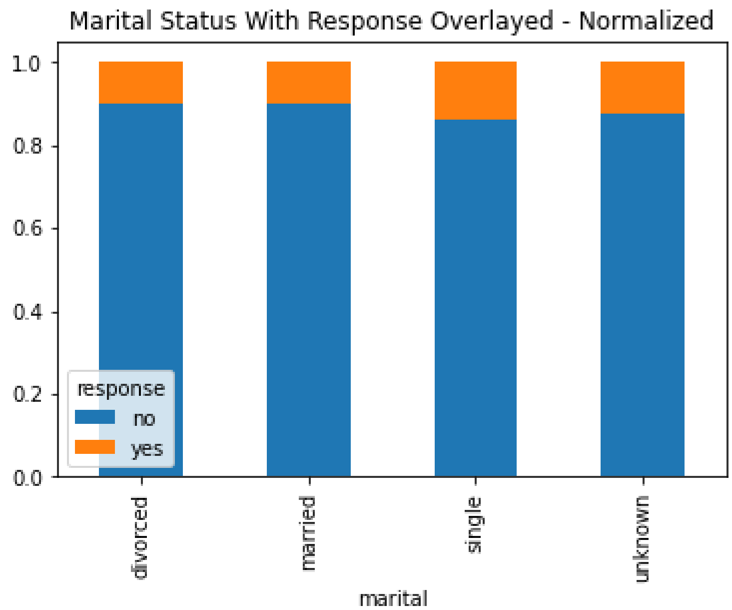


c. Normalized bar graph of marital, with overlay of response.

In [6]:
```
mares_norm = mares.div(mares.sum(1), axis = 0)

mares_norm.plot(kind = 'bar', stacked = True)
plt.title('Marital Status With Response Overlayed - Normalized')
```

Out[6]:
```
Text(0.5, 1.0, 'Marital Status With Response Overlayed - Normalized')
```



23. Do the following with the variables marital and response.

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

In [7]:
```
mares2 = pd.crosstab(bm_train['response'], bm_train['marital'])

round(mares2.div(mares2.sum(0), axis = 1)*100, 2)
```

Out[7]:

marital	divorced	married	single	unknown
response				
no	89.79	90.07	85.99	87.72
yes	10.21	9.93	14.01	12.28

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

In [8]:
```
mares2 = pd.crosstab(bm_train['marital'], bm_train['response'])

round(mares2.div(mares2.sum(0), axis = 1)*100, 1)
```

Out[8]:

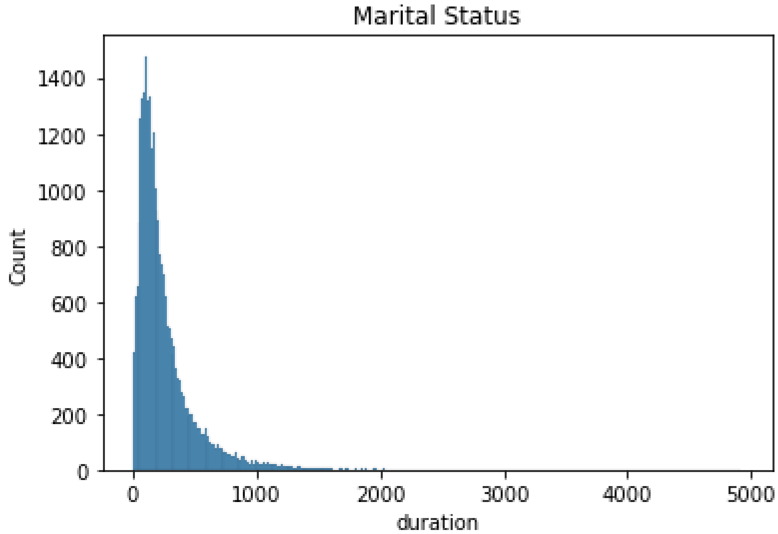
response	no	yes
marital		
divorced	11.5	10.4
married	61.0	53.8
single	27.3	35.5
unknown	0.2	0.2

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

a. Histogram of duration.

In [9]:
```
sns.histplot(x= bm_train["duration"]).set(title='Marital Status')
```

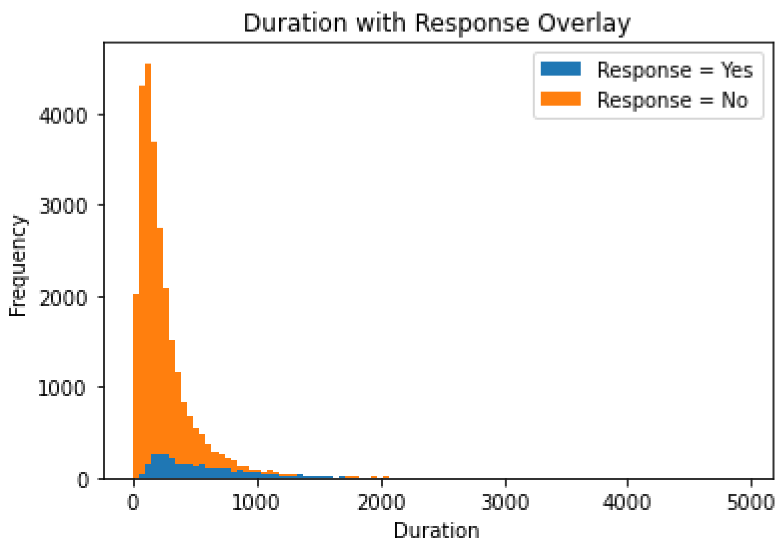
```
Out[9]: [Text(0.5, 1.0, 'Marital Status')]
```



b. Histogram of duration, with overlay of response.

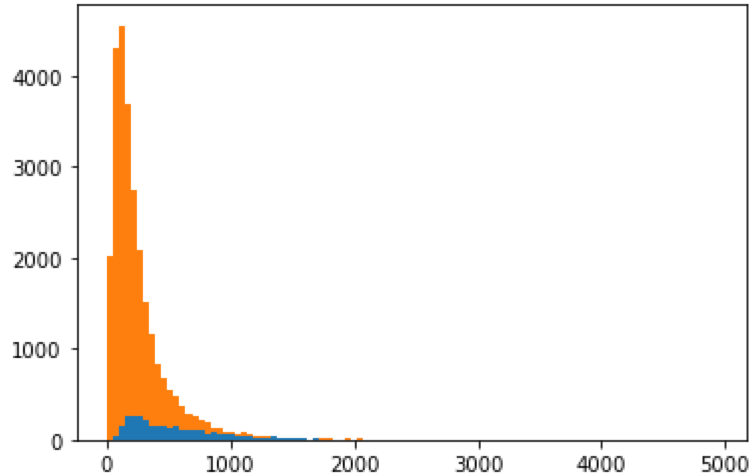
```
In [10]: bm_t_y = bm_train[bm_train.response == "yes"]['duration']
         bm_t_n = bm_train[bm_train.response == "no"]['duration']

         plt.hist([bm_t_y, bm_t_n], bins = 100, stacked = True)
         plt.legend(['Response = Yes', 'Response = No'])
         plt.title('Duration with Response Overlay')
         plt.xlabel('Duration')
         plt.ylabel('Frequency')
         plt.show()
```



c. Normalized histogram of duration, with overlay of response.

```
In [85]: (n, bins, patches) = plt.hist([bm_t_y, bm_t_n], bins =100, stacked = True)
```



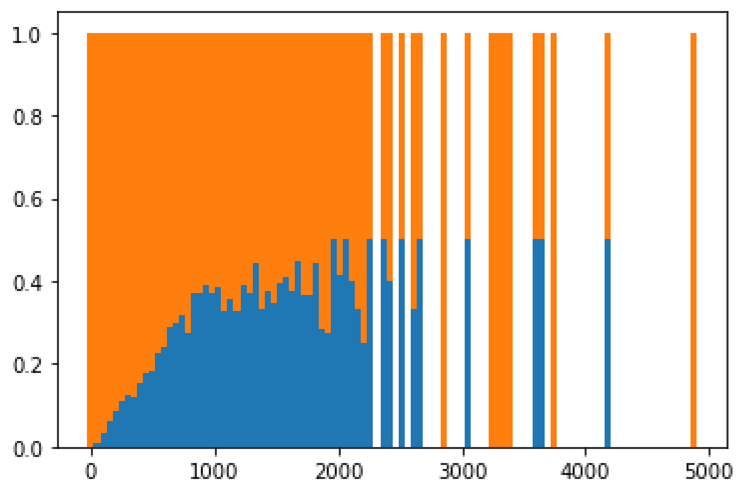

```
In [12]: n_table = np.column_stack((n[0], n[1]))
```

```
In [87]: n_norm = n_table / n_table.sum(axis=1)[:,None]
```

```
In [96]: ourbins = np.column_stack((bins[0:100], bins[1:101]))
```

```
In [98]: 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[:,0])
```



Data Science Using Python and R: Chapter 6 hands-on analysis

1. Develop 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?

```
In [17]: adult_tr= pd.read_csv('adult_ch6_training', sep = ',')
         adult_tr.head(5)
```

Out[17]:

	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

In [18]:
```
import statsmodels.tools.tools as stattools
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import tree
```

In [19]:
```
y = adult_tr[['Income']]
```

In [20]:
```
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)

mar_cat_dict


X_names = ["Cap_Gains_Losses", "Divorced", "Married", "Never-married",
           "Separated", "Widowed"]

y_names = ["<=50K", ">50K"]
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tools/tools.py:1
52: FutureWarning: categorical is deprecated. Use pandas Categorical to represent catego
rical data and can get_dummies to construct dummy arrays. It will be removed after relea
se 0.13.
  warnings.warn(
```

In [21]:
```
clf = tree.DecisionTreeClassifier(criterion = "gini", max_leaf_nodes=5)
clf = clf.fit(X, y)
```

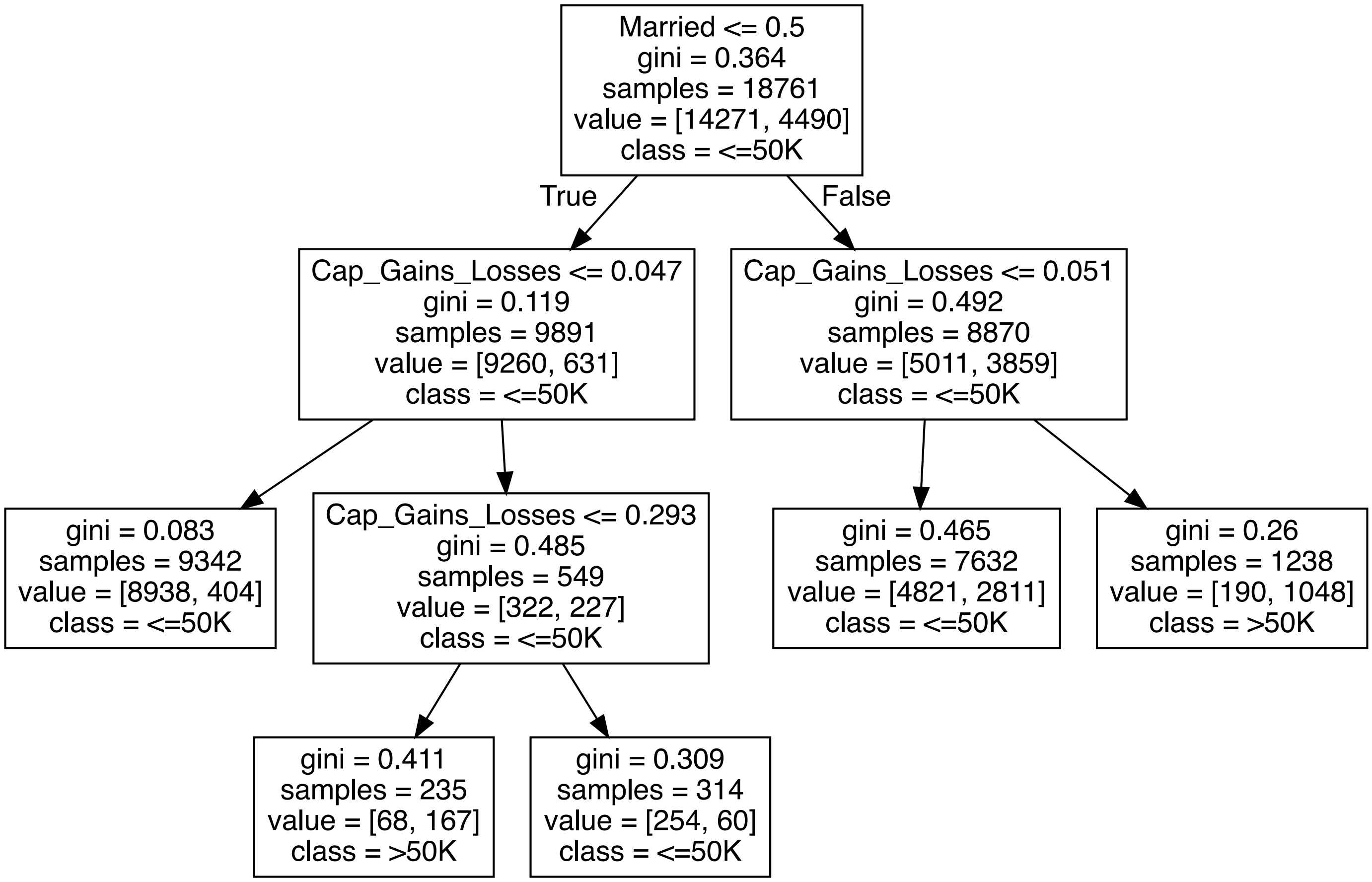
```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:
1858: FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [22]:
```
import graphviz


dot_data = tree.export_graphviz(clf, out_file=None, feature_names= X_names,
                                class_names= y_names)

graph = graphviz.Source(dot_data)
graph
```

Out[22]:



15. Develop 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?

In [23]:
```
adult_ts= pd.read_csv('adult_ch6_test', sep = ',')
adult_ts.head(5)
```

Out[23]:

	Marital status	Income	Cap_Gains_Losses
0	Married	<=50K	0.000000
1	Married	>50K	0.051781
2	Never-married	<=50K	0.000000
3	Divorced	>50K	0.000000
4	Married	>50K	0.000000

In [24]:
```
y1 = adult_ts[['Income']]
```

In [25]:
```
mar_np1 = np.array(adult_ts['Marital status'])

(mar_cat1, mar_cat_dict1) = stattools.categorical(mar_np1, drop=True, dictnames = True)

mar_cat_pd1 = pd.DataFrame(mar_cat1)

X1 = pd.concat((adult_ts[['Cap_Gains_Losses']], mar_cat_pd1), axis = 1)

mar_cat_dict1
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tools/tools.py:1
52: FutureWarning: categorical is deprecated. Use pandas Categorical to represent catego
rical data and can get_dummies to construct dummy arrays. It will be removed after relea
se 0.13.
  warnings.warn(
```

Out[25]:
```
{0: 'Divorced', 1: 'Married', 2: 'Never-married', 3: 'Separated', 4: 'Widowed'}
```

In [26]:
```
X1_names = ["Cap_Gains_Losses", "Divorced", "Married", "Never-married",
            "Separated", "Widowed"]

y1_names = ["<=50K", ">50K"]
```

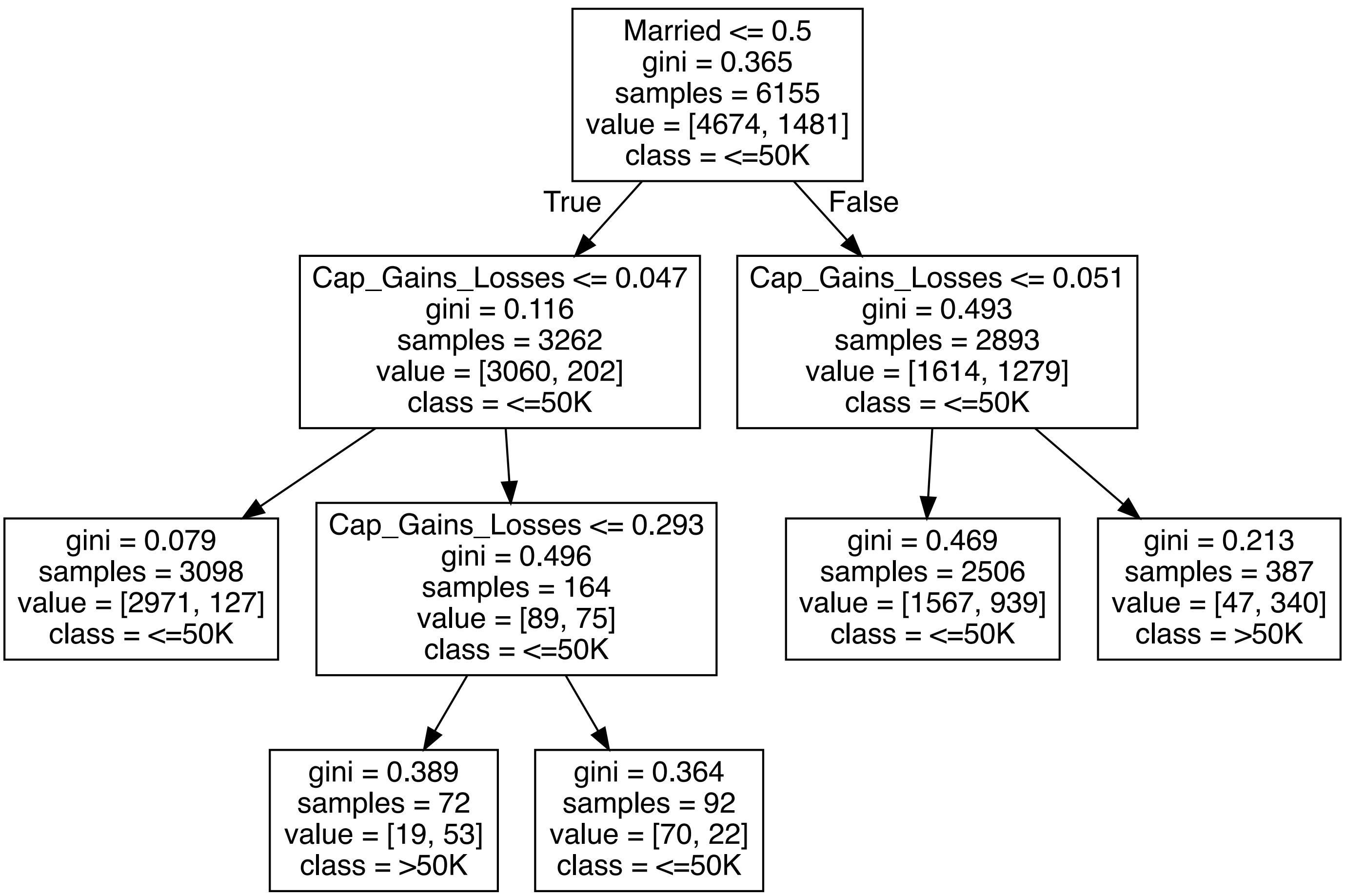
In [27]:
```
clf1 = tree.DecisionTreeClassifier(criterion = "gini", max_leaf_nodes=5)
clf1 = clf1.fit(X1, y1)
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:
1858: FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [28]:
```
dot_data = tree.export_graphviz(clf1, out_file=None, feature_names= X1_names,
                                class_names= y1_names)

graph = graphviz.Source(dot_data)
graph
```

Out[28]:



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

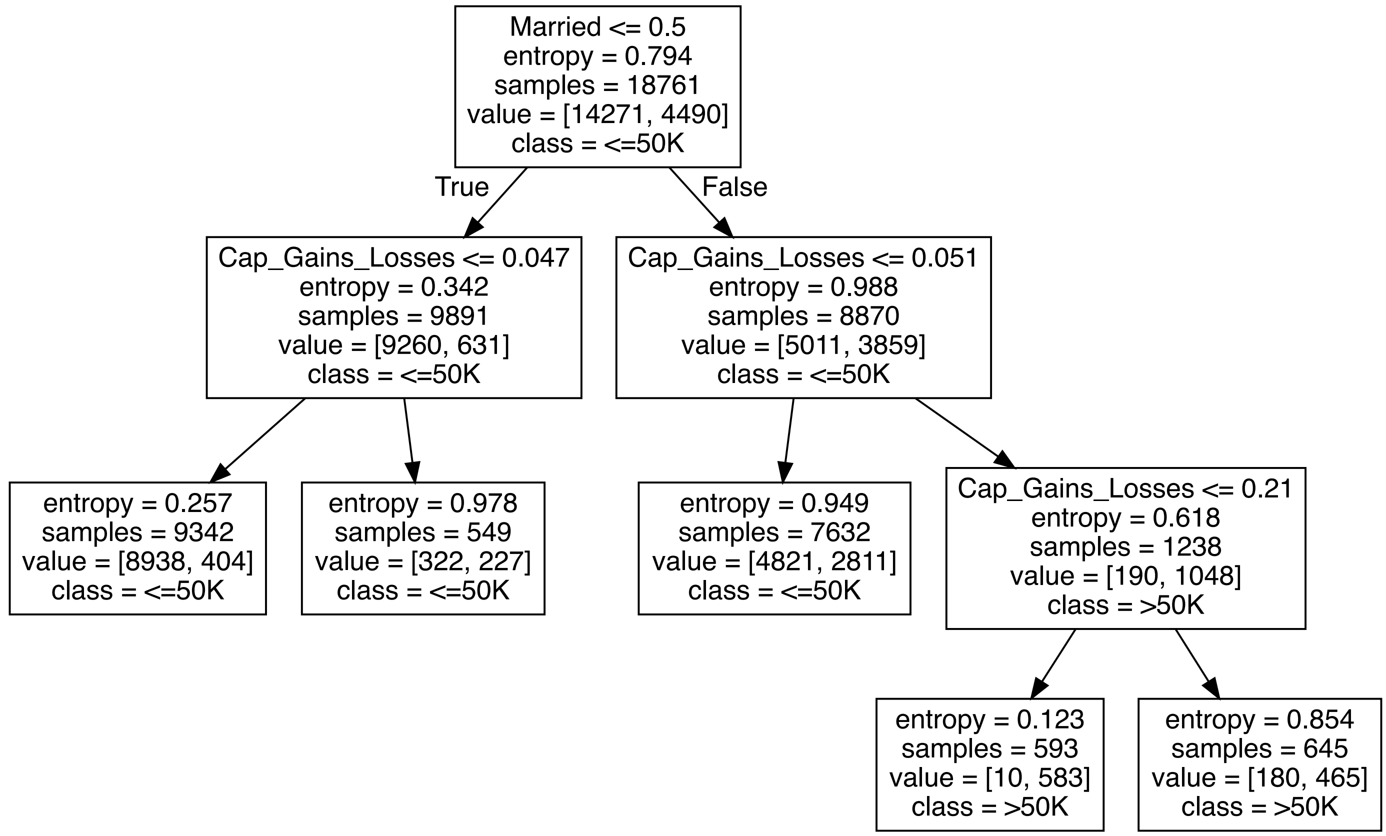
In [29]:
```
c50_01 = DecisionTreeClassifier(criterion="entropy", max_leaf_nodes=5).fit(X,y)
```

```
/Users/gabirivera/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py:
1858: FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

In [30]:
```
dot_data = tree.export_graphviz(c50_01, out_file=None, feature_names= X_names,
                                class_names= y_names)
```

```
graph = graphviz.Source(dot_data)
graph
```

Out[30]:



Data Science Using Python and R: Chapter 11 hands-on analysis

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?

In [31]:
```
bank_train= pd.read_csv('bank_reg_training', sep = ',')

bank_train.head(5)
```

Out[31]:

	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

In [32]:
```
import statsmodels.api as sm
```

In [33]:
```
a = pd.DataFrame(bank_train[['Credit Score']])
b = pd.DataFrame(bank_train[['Debt-to-Income Ratio', 'Request Amount']])
```

In [34]:
```
b = sm.add_constant(b)
```

In [35]:
```
model01 = sm.OLS(a, b).fit()
```

In [36]:
```
model01.summary()
```

Out[36]:

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, 07 Nov 2022	**Prob (F-statistic):**	1.37e-67
Time:	18:20:54	**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

Notes:
[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.

35. Validate the model from the previous exercise.

In [37]:
```
bank_test= pd.read_csv('bank_reg_test', sep = ',')
bank_test.head(5)
```

Out[37]:

	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

In [38]:
```
a1 = pd.DataFrame(bank_test[['Credit Score']])
b1 = pd.DataFrame(bank_test[['Debt-to-Income Ratio', 'Request Amount']])
```

In [39]:
```
b1 = sm.add_constant(b1)
```

In [40]:
```
model_01 = sm.OLS(a1, b1).fit()
```

In [41]:
```
model_01.summary()
```

Out[41]:

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, 07 Nov 2022	**Prob (F-statistic):**	1.94e-92
Time:	18:20:57	**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

Notes:
[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.

39. Find and interpret the value of s.

Bank test dataset s value:

In [42]:
```
np.sqrt(model_01.scale)
```

Out[42]:
```
65.77845809176674
```

Bank training dataset s value:

In [43]:
```
np.sqrt(model01.scale)
```

Out[43]:
```
66.00195259717187
```

41. Find MAE(Baseline) and MAE(Regression), and determine whether the regression model outperformed its baseline model.

MAE: Regression

In [44]:
```
ytrue = bank_test[['Credit Score']]
```

In [45]:
```
ypred = model01.predict(b1)
```

In [46]:
```
from sklearn.metrics import mean_absolute_error

mean_absolute_error(y_true = ytrue, y_pred = ypred)
```

Out[46]:
```
47.79066993785576
```