EMORY UNIVERSITY

Introduction to Business Data Analytics

Homework #1

Carl Xi

(put your name above)

Note: This is an individual homework. Discussing this homework with your classmates is a violation of the Honor Code. If you borrow code from somewhere else, please add a comment in your code to make it clear what the source of the code is (e.g., a URL would sufficient). If you borrow code and you don't provide the source, it is a violation of the Honor Code.

Total grade: ________ out of ___150___ points

This homework has <u>seven</u> questions. Please answer them all and submit your assignment using Canvas. In particular, you need to submit:

a) ***This file with your answers. Please transform the doc file to a PDF file. Then, submit the PDF File in the Assignment labeled as "Homework 1"***
b) ***Your code and/or Rapidminer repositories for the hands-on exercise. Submit the (1) Python code in Jupyter format (2) the Rapidminer repositories and (3) the dataset(s) of the Assignment labeled as "Homework 1_Code & Repositories"***

1) (24 points) Choose the data technology (Q, H, U, or S) that is most appropriate for each of the following business questions/scenarios.

Q – SQL Querying
H – Statistical Hypothesis Testing
U – Unsupervised Data Mining/Pattern Finding
S – Supervised Data Mining

a) _H_ For my on-line advertising the decision tree model yields a response rate of 0.5% and the old, manual targeting model yields 0.3%. Is the decision tree model really better?

b) _Q_ I want to know which of my customers are the most profitable.

c) _Q_ I need to get data on all my on-line customers who were exposed to the special offer, including their registration data, their past purchases, and whether or not they purchased in the 15 days following the exposure.

d)_U_ I would like to segment my customers into groups based on their demographics and prior purchase activity. I am not focusing on improving a particular task, but would like to generate ideas.

e) _S_ I have a budget to target 10,000 existing customers with a special offer. I would like to identify those customers most likely to respond to the special offer.

f) _U_ I want to know what characteristics differentiate my most profitable customers.

2) (16 points) Label each case as describing either data mining (DM), or the use of the results of data mining (Use).

g) _Use_ Choose customers who are most likely to respond to an on-line ad.

h) _DM_ Discover rules that indicate when an account has been defrauded.

i) _DM_ Find patterns indicating what customer behavior is more likely to lead to response to an on-line ad.

j) _Use_ Estimate probability of default for a credit application.

3) (15 points) MTC (MegaTelCo) has decided to use supervised learning to address its problem of churn in its wireless phone business. As a consultant to MTC, you realize that a main task in the business understanding/data understanding phases of the data mining process is to define the target variable. In one or two sentences, please suggest a definition for the target variable. Be as precise as possible—someone else will be implementing your suggestion. *(Remember: it should make sense from a business point of view, and it should be reasonable that MTC would have data available to know the value of the target variable for historical customers.)*

The target variable is essentially the attribute within a dataset that you are trying to predict with the rest of the attributes in your available data. In this case, the target variable would be the churn variable of MTC's wireless phone business dataset. We will be trying to create a predictive model (a formula) that estimates whether a customer will churn within 3 months of contract expiration (the target variable) given a specific set of attributes. In this case, using class probability estimation methods will probably yield better results than binary probability estimation methods.

4) (20 points) A predictive model has been applied to a test dataset and has classified 87 records as fraudulent (31 correctly so) and 953 as non-fraudulent (919 correctly so).

- **Present the confusion matrix for this scenario.**

Predicted\Actual	**Positive (Fraudulent)**	**Negative (Non-Fraudulent)**	**Total**
Positive (Fraudulent)	31 (True Positives [TP])	56 (False Positives [FP])	87
Negative (Non-Fraudulent)	34 (False Negatives [FN])	919 (True Negatives [TN])	953

Total	65	975	1040

- **Calculate the error rate and accuracy rate.**

Accuracy = (TP+TN)/(TP+FP+FN+TN) = (31+919)/(31+56+34+919) = 0.9135 or 91.35%
Error rate = 1 – Accuracy = 1-0.9135 = 0.0865 or 8.65%

- **Calculate the precision, recall, and f-measure values for each of the two outcome classes (i.e., fraudulent and non-fraudulent records);**

Precision $_{\text{Positive}}$ = TP / (TP + FP)	**Precision** $_{\text{Negative}}$ = TN / (TN + FN)
Recall $_{\text{Positive}}$ = TP / (TP + FN)	**Recall** $_{\text{Negative}}$ = TN / (TN + FP)
F-Measure $_{\text{Positive}}$ $=2 * \frac{\textbf{Precision}[\text{Positive}] * \textbf{Recall}[\text{Positive}]}{\textbf{Precision}[\text{Positive}] * \textbf{Recall}[\text{Positive}]}$	**F-Measure** $_{\text{Negative}}$ $=2 * \frac{\textbf{Precision}[\text{Negative}] * \textbf{Recall}[\text{Negative}]}{\textbf{Precision}[\text{Negative}] * \textbf{Recall}[\text{Negative}]}$

Positive = Fradulent, Negative = Non-Fradulent

Precision $_{\text{Positive}}$ = 31/(31+56) = $\frac{31}{87} \approx 0.3563$	**Precision** $_{\text{Negative}}$ = 919/(919+34) = $\frac{919}{953} \approx 0.9643$
Recall $_{\text{Positive}}$ = 31/(31+34) = $\frac{31}{65} \approx 0.4769$	**Recall** $_{\text{Negative}}$ = 919/(919+56) = $\frac{919}{975} \approx 0.9426$
F-Measure $_{\text{Positive}}$ $=2 * \frac{0.3563 * 0.4769}{0.3563 + 0.4769} \approx 0.4079$	**F-Measure** $_{\text{Negative}}$ $=2 * \frac{0.9643 * 0.9426}{0.9643 + 0.9426} \approx 0.9533$

- **Also, calculate the accuracy rate that would be achieved by naïve (majority) rule on this data.**

The naïve rule basically states that all data records will be marked as 'non-fraudulent', or that they belong to the majority (most prevalent class).

Naïve accuracy rate = 56+919 = $\frac{(56+919)}{\text{total}} = \frac{(56+919)}{(31+34+56+919)}$ = 0.9375

5) (50 points) [Implement this exercise with both RapidMiner (20 points) and Python (30 points)] Use the decision tree classification technique on the *HW1* dataset. This dataset is provided on the course website and contains data about consumers and their decisions to terminate a contract (i.e., consumer churn problem).

Data description:

Col.	Var. Name	Var. Description
1	revenue	Mean monthly revenue in dollars
2	outcalls	Mean number of outbound voice calls
3	incalls	Mean number of inbound voice calls
4	months	Months in Service
5	eqpdays	Number of days the customer has had his/her current equipment
6	webcap	Handset is web capable
7	marryyes	Married (1=Yes; 0=No)
8	travel	Has traveled to non-US country (1=Yes; 0=No)
9	pcown	Owns a personal computer (1=Yes; 0=No)
10	creditcd	Possesses a credit card (1=Yes; 0=No)
11	retcalls	Number of calls previously made to retention team
12	churndep	Did the customer churn (1=Yes; 0=No)

Build a decision tree model that predicts whether a consumer will terminate his/her contract. In particular, I would like for you to create a decision tree using entropy with no max depth. Explore how well the decision trees perform for several different parameter values (e.g., for different splitting criteria). Interpret the model (decision tree) that provides the best predictive performance. Some possible issues / hints to think about: using training vs. test datasets. Present a brief overview of your predictive modeling process, explorations, and discuss your results. Make sure you present information about the model "goodness" (please report the confusion matrix, predictive accuracy, classification error, precision, recall, f-measure).

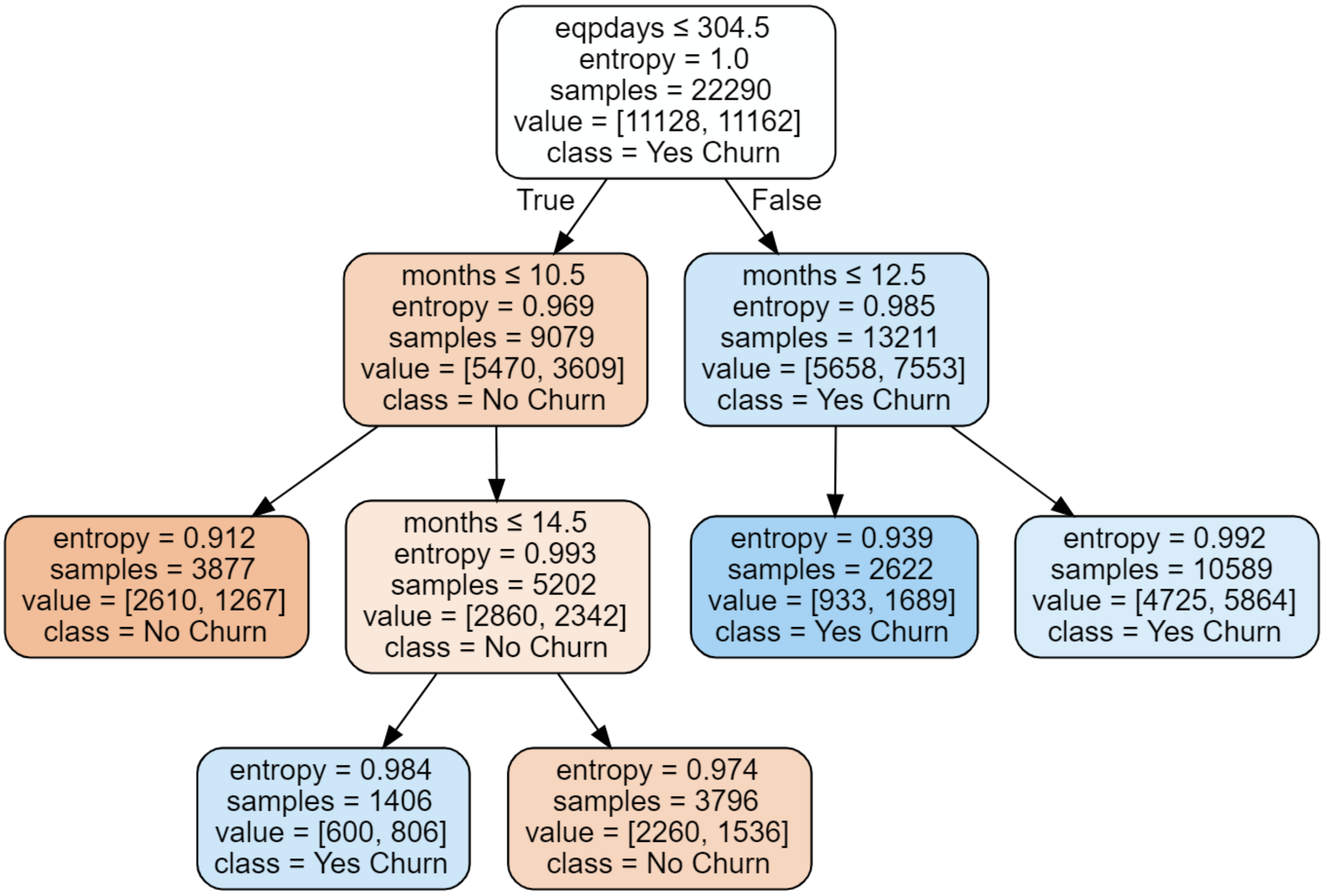
Present a brief overview of your predictive modeling process. That is, you need to lay out the steps you have taken in order to build and evaluate the decision tree model. For instance, how did you explore the data set before you built the model? Write this report in a way that the upper level management of the team would understand what you are doing. Why is the decision tree an appropriate model for this problem? How can we evaluate the predictive ability of the decision tree? If you build decision trees with different splitting criteria, which decision tree would you prefer to use in practice?

Business problem & data set overview:

Following our answer in question 3, we are essentially trying to build a model that predicts whether a customer will churn given a set of variables. We can immediately identify this as a supervised classification problem as we have a specific purpose and existing data on the target variable. Benefits of decision trees for this problem are discussed at the bottom of this problem.

In terms of our data, we can see that 4 of our variables are 0/1 or binary, while the rest are all floating numbers. Our target variable is also 0/1 or binary.

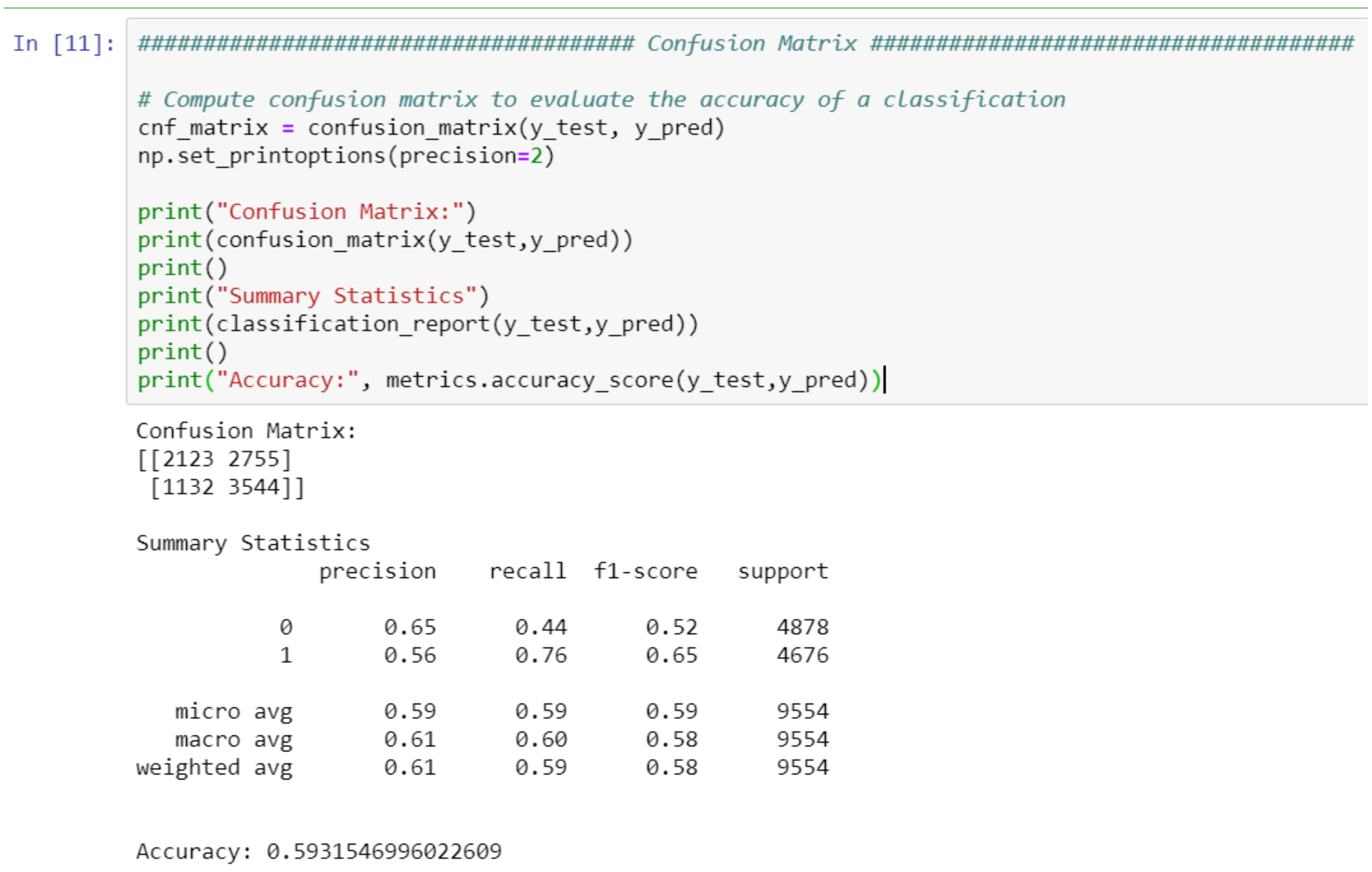
[Python Model]
First, here is my decision tree in python:



Here are the variables I used in my DecisionTreeClassifier call:
DecisionTreeClassifier(criterion ='entropy', random_state= 42, max_depth = None, min_samples_split = 0.081, min_impurity_decrease = 0.002)

- Note that we have set max_depth to 'None', but that other variables like min_samples_split and min_impurity_decrease will still affect the total number of branches.

Here are the results of my best decision tree:

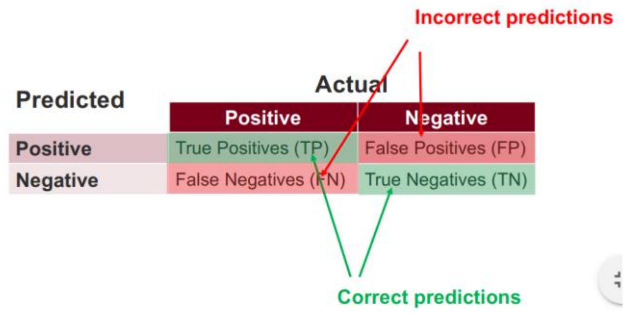


```
In [11]: ########################################## Confusion Matrix ########################################

         # Compute confusion matrix to evaluate the accuracy of a classification
         cnf_matrix = confusion_matrix(y_test, y_pred)
         np.set_printoptions(precision=2)

         print("Confusion Matrix:")
         print(confusion_matrix(y_test,y_pred))
         print()
         print("Summary Statistics")
         print(classification_report(y_test,y_pred))
         print()
         print("Accuracy:", metrics.accuracy_score(y_test,y_pred))
```

```
Confusion Matrix:
[[2123 2755]
 [1132 3544]]

Summary Statistics
              precision    recall  f1-score   support

           0       0.65      0.44      0.52      4878
           1       0.56      0.76      0.65      4676

   micro avg       0.59      0.59      0.59      9554
   macro avg       0.61      0.60      0.58      9554
weighted avg       0.61      0.59      0.58      9554


Accuracy: 0.5931546996022609
```

A quick review on the definitions of the statistics:

- Predictive accuracy: % of correct classifications out of total.
- Classification error: % of wrong classifications out of total.
- An accuracy of 0.5932 is very close to the accuracy I achieved with my RapidMiner model (which I will discuss below). The confusion matrix can be interpreted using the diagram below:

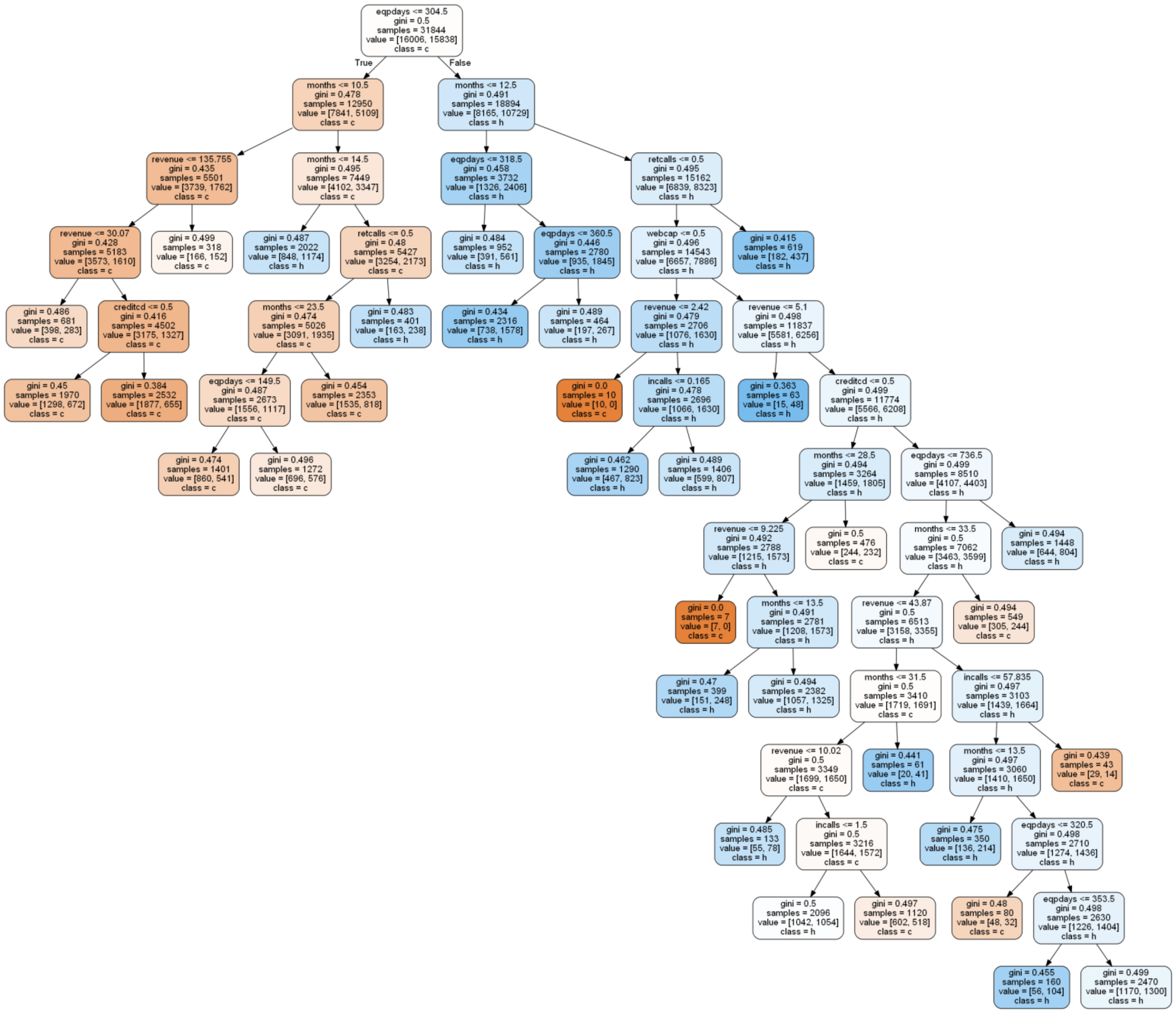


We can see that there are a sizable number of False Positives and False Negatives, but in-line with our 0.5932 accuracy. Our f-measure of 0.52 and 0.65 for no churn and churn respectively show us that our decision tree performs above average on a scale of 0 to 1.

Before I got my final decision tree, I played around with the different variables in the DecisionTreeClassifier call. I have produced about 10 different decision trees, but to save space I have only included 3 in my writeup. With my final tree excluded, the other two notable ones are below:

1. With nothing but random state, the call prints out a scary big and long tree that is effectively illegible.

2. My next significant improvement to my model is when I set the minimal sample split to 0.08. It produced a legible model that gets the job done, but I was still unhappy with how big it is.


eqpdays <= 304.5 gini = 0.5 samples = 31844 value = [16006, 15838] class = c
True
False
months <= 10.5 gini = 0.478 samples = 12950 value = [7841, 5109] class = c
months <= 12.5 gini = 0.491 samples = 18894 value = [8165, 10729] class = h
revenue <= 135.755 gini = 0.435 samples = 5501 value = [3739, 1762] class = c
months <= 14.5 gini = 0.495 samples = 7449 value = [4102, 3347] class = c
eqpdays <= 318.5 gini = 0.458 samples = 3732 value = [1326, 2406] class = h
retcalls <= 0.5 gini = 0.495 samples = 15162 value = [6839, 8323] class = h
revenue <= 30.07 gini = 0.428 samples = 5183 value = [3573, 1610] class = c
gini = 0.499 samples = 318 value = [166, 152] class = c
gini = 0.487 samples = 2022 value = [848, 1174] class = h
retcalls <= 0.5 gini = 0.48 samples = 5427 value = [3254, 2173] class = c
gini = 0.484 samples = 952 value = [391, 561] class = h
eqpdays <= 360.5 gini = 0.446 samples = 2780 value = [935, 1845] class = h
webcap <= 0.5 gini = 0.496 samples = 14543 value = [6657, 7886] class = h
gini = 0.415 samples = 619 value = [182, 437] class = h
gini = 0.486 samples = 681 value = [398, 283] class = c
creditcd <= 0.5 gini = 0.416 samples = 4502 value = [3175, 1327] class = c
months <= 23.5 gini = 0.474 samples = 5026 value = [3091, 1935] class = c
gini = 0.483 samples = 401 value = [163, 238] class = h
gini = 0.434 samples = 2316 value = [738, 1578] class = h
gini = 0.489 samples = 464 value = [197, 267] class = h
revenue <= 2.42 gini = 0.479 samples = 2706 value = [1076, 1630] class = h
revenue <= 5.1 gini = 0.498 samples = 11837 value = [5581, 6256] class = h
gini = 0.45 samples = 1970 value = [1298, 672] class = c
gini = 0.384 samples = 2532 value = [1877, 655] class = c
eqpdays <= 149.5 gini = 0.487 samples = 2673 value = [1556, 1117] class = c
gini = 0.454 samples = 2353 value = [1535, 818] class = c
gini = 0.0 samples = 10 value = [10, 0] class = c
incalls <= 0.165 gini = 0.478 samples = 2696 value = [1066, 1630] class = h
gini = 0.363 samples = 63 value = [15, 48] class = h
creditcd <= 0.5 gini = 0.499 samples = 11774 value = [5566, 6208] class = h
gini = 0.474 samples = 1401 value = [860, 541] class = c
gini = 0.496 samples = 1272 value = [696, 576] class = c
gini = 0.462 samples = 1290 value = [467, 823] class = h
gini = 0.489 samples = 1406 value = [599, 807] class = h
months <= 28.5 gini = 0.494 samples = 3264 value = [1459, 1805] class = h
eqpdays <= 736.5 gini = 0.499 samples = 8510 value = [4107, 4403] class = h
revenue <= 9.225 gini = 0.492 samples = 2788 value = [1215, 1573] class = h
gini = 0.5 samples = 476 value = [244, 232] class = c
months <= 33.5 gini = 0.5 samples = 7062 value = [3463, 3599] class = h
gini = 0.494 samples = 1448 value = [644, 804] class = h
gini = 0.0 samples = 7 value = [7, 0] class = c
months <= 13.5 gini = 0.491 samples = 2781 value = [1208, 1573] class = h
revenue <= 43.87 gini = 0.5 samples = 6513 value = [3158, 3355] class = h
gini = 0.494 samples = 549 value = [305, 244] class = c
gini = 0.47 samples = 399 value = [151, 248] class = h
gini = 0.494 samples = 2382 value = [1057, 1325] class = h
months <= 31.5 gini = 0.5 samples = 3410 value = [1719, 1691] class = c
incalls <= 57.835 gini = 0.497 samples = 3103 value = [1439, 1664] class = h
revenue <= 10.02 gini = 0.5 samples = 3349 value = [1699, 1650] class = c
gini = 0.441 samples = 61 value = [20, 41] class = h
months <= 13.5 gini = 0.497 samples = 3060 value = [1410, 1650] class = h
gini = 0.439 samples = 43 value = [29, 14] class = c
gini = 0.485 samples = 133 value = [55, 78] class = h
incalls <= 1.5 gini = 0.5 samples = 3216 value = [1644, 1572] class = c
gini = 0.475 samples = 350 value = [136, 214] class = h
eqpdays <= 320.5 gini = 0.498 samples = 2710 value = [1274, 1436] class = h
gini = 0.5 samples = 2096 value = [1042, 1054] class = h
gini = 0.497 samples = 1120 value = [602, 518] class = c
gini = 0.48 samples = 80 value = [48, 32] class = c
eqpdays <= 353.5 gini = 0.498 samples = 2630 value = [1226, 1404] class = h
gini = 0.455 samples = 160 value = [56, 104] class = h
gini = 0.499 samples = 2470 value = [1170, 1300] class = h

3. By further playing with the different variables available to the DeceitionTreeClassifier call (documented here), I found that min_impurity_decrease siginifically reduces the complexity of my tree while keeping the impact to my accuracy very small. The final tree is shown at the very beginning of this answers.

Across all my models, my accuracy mostly stayed around 0.59, with classification error, precision, recall and f-measure mostly stayed about the same. Generally, precision stayed above 0.55 for both churn and no-churn. Recall and f-score for churn stayed above 0.65 across all models, and classification error, being literally 1 – accuracy, stayed consistent along with accuracy across all my models.

We know that with the problem given (churn/no churn), we are looking for a supervised datamining technique. Of all supervised DM techniques, decision trees are easy to understand, implement, used and are computationally cheap. As well, most datamining packages natively include decision tree processes.

Most importantly, knowing that we are trying to communicate with non-DM-savvy stakeholders aka upper level management of the team, decision trees are great for their easy of understanding.

Even if I build decision trees with different splitting criteria, I think I would still stick with my final tree (the one at the very beginning of this answer) for its ability to deliver competitive results compared to the most complex one (minimal accuracy loss) while being still super easy to understand (only 3-4 levels) and visually pleasing.

[Python process]

First, I installed all the necessary packages:

```
In [1]: # To write a Python 2/3 compatible codebase, the first step is to add this line to the top of each module
        from __future__ import division, print_function, unicode_literals

        ########################################## Imports ##########################################
        # Numpy is the fundamental package for scientific computing with Python.
        # SciPy (pronounced "Sigh Pie") is an open source Python library used for scientific computing and technical computing.
        # Os module provides a portable way of using operating system dependent functionality.
        import numpy as np # np is an alias pointing to numpy
        import scipy as sp # sp is an alias pointing to scipy
        import pandas as pd # pd is an alias point to pandas

        from sklearn import metrics, tree
        from sklearn.metrics import confusion_matrix, classification_report
        from sklearn.model_selection import train_test_split
        from sklearn.tree import export_graphviz
        # If you don't have graphviz package, you need to install it https://anaconda.org/anaconda/graphviz
        # How to install Graphviz with Anaconda 1
        # conda install -c anaconda graphviz
        from IPython.display import Image
        import matplotlib.pyplot as plt
        !pip install graphviz
        import itertools
        import graphviz
        import os

        # Seed the generator to make this notebook's output stable across runs
        np.random.seed(42)

        # To plot pretty figures
        %matplotlib inline
        import matplotlib
        import matplotlib.pyplot as plt #pyplot is matplotlib's plotting framework https://matplotlib.org/users/pyplot_tutorial.html
```

I then imported the dataset in its raw state

```
############################################ Imports ############################################
hw1data = pd.read_csv("HW1_Data.csv")
```

I know from my work in RapidMiner that there are some outliers and bad data, so I wanted to examine the data and eliminate them:

```
In [2]: #Looking at how big our dataset is
        hw1data.shape

Out[2]: (31891, 12)
```

```
In [3]: #taking a peek at how the data looks like
        hw1data.head()
```

Out[3]:

	revenue	outcalls	incalls	months	eqpdays	webcap	marryyes	travel	pcown	creditcd	retcalls	churndep
0	83.53	20.00	1.0	31	745	1	0	0	0	0	4	1
1	29.99	0.00	0.0	52	1441	0	0	0	1	1	3	1
2	37.75	2.67	0.0	25	572	0	0	0	1	1	3	1
3	5.25	0.00	0.0	45	1354	0	0	0	0	0	2	1
4	42.71	8.67	0.0	27	224	1	0	0	0	0	3	1

```
In [4]: #Looking at any potential problems/outliers
        hw1data.describe()
```

Out[4]:

	revenue	outcalls	incalls	months	eqpdays	webcap	marryyes	travel	pcown	creditcd	re
count	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.0
mean	58.665179	24.951385	8.065277	18.761908	391.222633	0.894704	0.363175	0.057163	0.184817	0.676931	0.0
std	44.163859	34.790147	16.610589	9.548019	254.998478	0.306939	0.480922	0.232158	0.388155	0.467656	0.2
min	-5.860000	0.000000	0.000000	6.000000	-5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	33.450000	3.000000	0.000000	11.000000	212.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.0
50%	48.380000	13.330000	2.000000	17.000000	341.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.0
75%	71.040000	33.330000	9.000000	24.000000	530.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.0
max	861.110000	610.330000	404.000000	60.000000	1812.000000	1.000000	1.000000	1.000000	1.000000	1.000000	4.0

I noticed some problems, such as how the min for revenue and eqpdays were negative and removed these bad data points from my dataset.

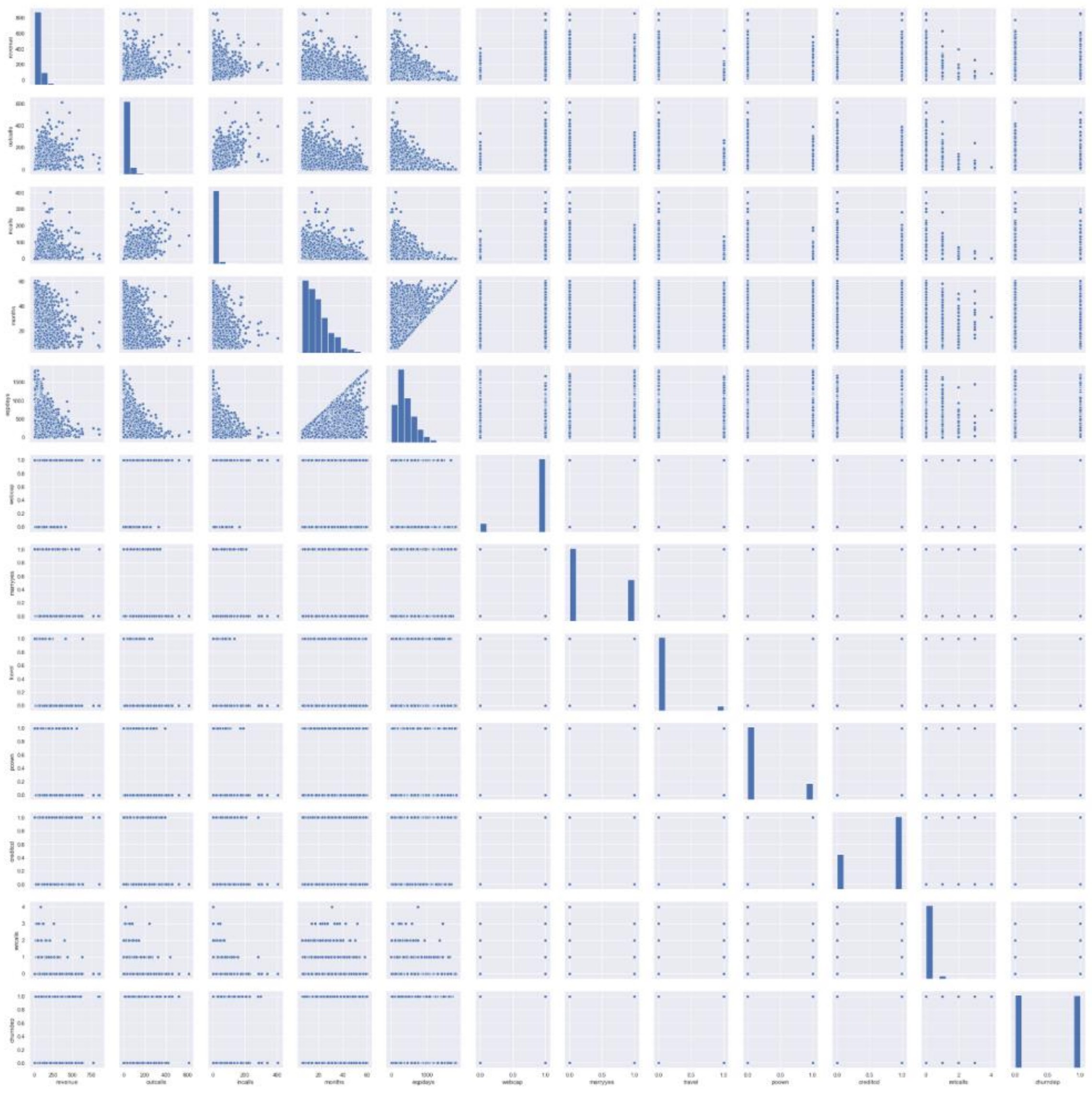
```
In [5]: #data cleaning
        baddata = hw1data[(hw1data['revenue']<0) | (hw1data['eqpdays']<0)].index

        hw1data.drop(baddata, inplace=True)
```

Then I imported the packages for decision tree making, and isolated my target variable from the dataset:

```
In [6]: from sklearn.tree import DecisionTreeClassifier # The sklearn.tree module includes decision tree-based models for
        # classification and regression
        # Documentation for decision Tree Classifier
        # http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

        # Seaborn is a Python data visualization library based on matplotlib.
        # Seaborn documentation can be found here https://seaborn.pydata.org/generated/seaborn.set.html
        import seaborn as sns # sns is an alias pointing to seaborn
        sns.set(color_codes=True) #Set aesthetic parameters in one step. Remaps the shorthand color codes (e.g. “b”, “g”, “r”, etc.) to
        from scipy import stats #Documentation stats package of scipy https://docs.scipy.org/doc/scipy/reference/stats.html#module-scipy

        ########################################## Isolating the Target Variable ##########################################
        # Retrieving Attributes
        X = hw1data.iloc[:,:-1]
        # Retriving Target Variable
        y = hw1data.iloc[:,-1]
```

With the data split and cleaned, I decided to take a final look at the trends in my data to see if there are patterns that might influence our model, as well as cross-variable relationships. Here is a pairplot matrix :

```
In [7]: sns.pairplot(hw1data)

Out[7]: <seaborn.axisgrid.PairGrid at 0x1addfb91668>
```



From the plot, I noticed that revenue, outcalls and incalls are heavily skewed towards to the left (smaller values). Additionally, most entries have web capable handsets, have traveled to non-US countries, owns a personal computer and previously did not call the retention team. Our target variable (churndep) does not have obvious correlations with any other variables in our dataset.

Explore how well the decision trees perform for several different parameter values (e.g., for different splitting criteria). Interpret the model (decision tree) that provides the best predictive performance.

Why is the decision tree an appropriate model for this problem? How can we evaluate the predictive ability of the decision tree?

I then split my data into 70% train and 30% test, and ran my model:

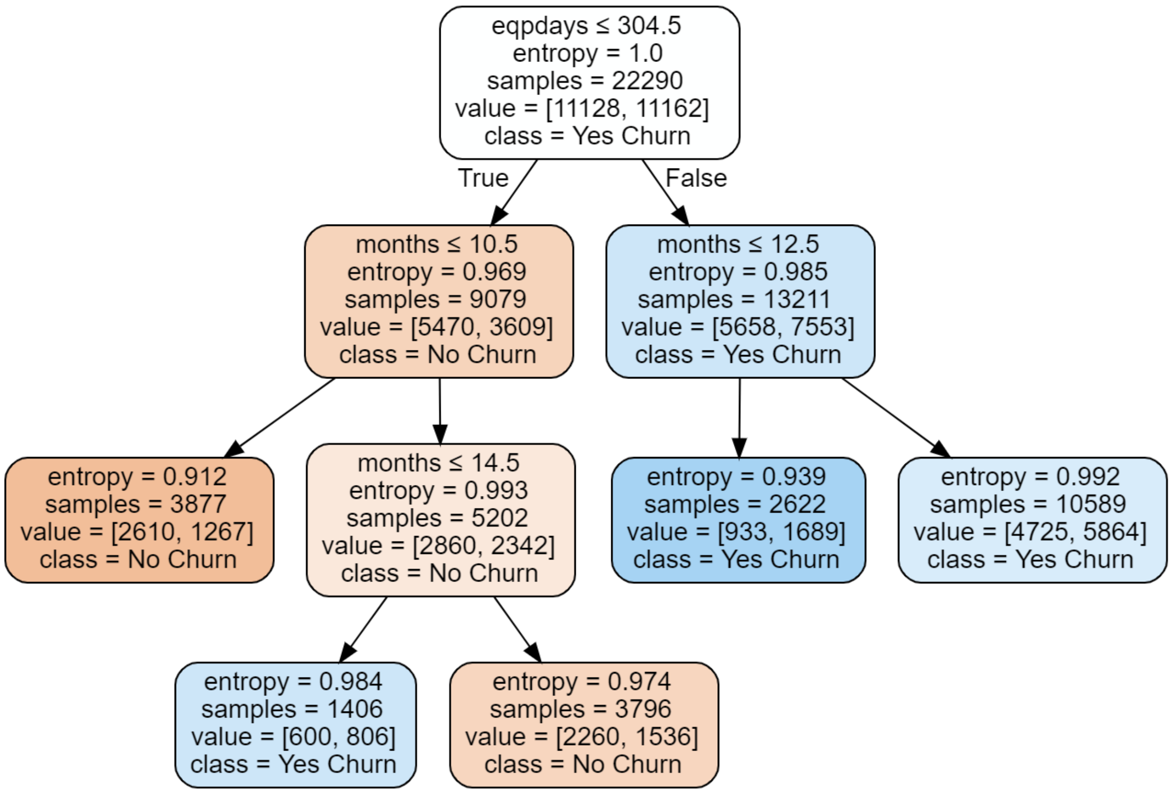
```
In [8]: ####################################### Split the Data #######################################

        # Split the data into a training set and a test set
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
        # Note: Always a good idea to shuffle the dataset before you split it into training and testing
        # train_test_split performs shuffling by default
```

```
In [9]: ########################################## Model Training ##########################################
        ############################# Build Model & Apply it to the Test Set #######################
        #Build the decision tree
        clf3 = DecisionTreeClassifier(criterion ='entropy', random_state= 42,max_depth = None, min_samples_split=0.081, min_impurity_deci
        # "clf3.fit(X_train, y_train)"" fits the model and then
        # ".predict(X_test)" makes predicitions based on the test set
        y_pred = clf3.fit(X_train, y_train).predict(X_test)
```

With my model successfully ran, I printed my tree to examine it visually:

```
In [10]: ################################# Visualization of Decision Tree ###########################
         banana = tree.export_graphviz( clf3,
                 out_file=None, #image_path("HW1.dot")
                 feature_names=list(X.columns),
                 class_names=["No Churn","Yes Churn"],
                 rounded=True,
                 filled=True,
                 special_characters=True)

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

Out[10]:



Lastly, I printed out all my evaluation metrics to see how well my model performs. This includes the confusion matrix, f-score, precision, and accuracy. I have discussed the meaning of these results in more detail at the beginning of this question.

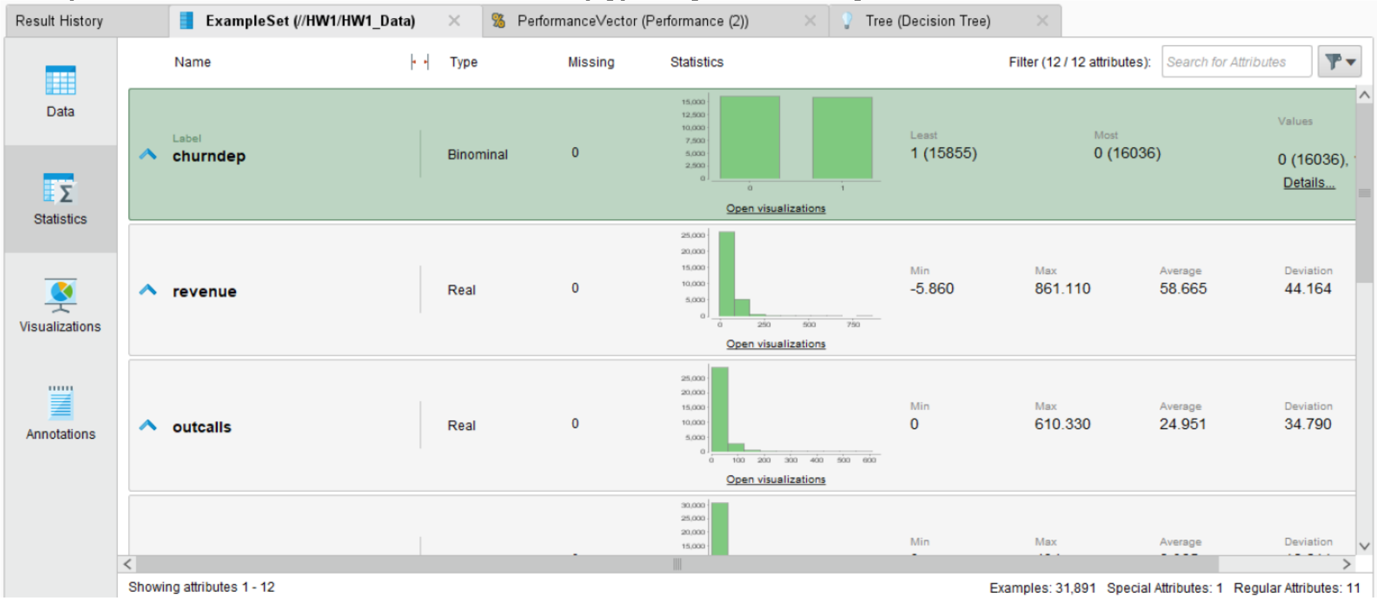
```
In [11]: ######################################## Confusion Matrix ########################################

         # Compute confusion matrix to evaluate the accuracy of a classification
         cnf_matrix = confusion_matrix(y_test, y_pred)
         np.set_printoptions(precision=2)

         print("Confusion Matrix:")
         print(confusion_matrix(y_test,y_pred))
         print()
         print("Summary Statistics")
         print(classification_report(y_test,y_pred))
         print()
         print("Accuracy:", metrics.accuracy_score(y_test,y_pred))
```

```
Confusion Matrix:
[[2123 2755]
 [1132 3544]]

Summary Statistics
              precision    recall  f1-score   support

           0       0.65      0.44      0.52      4878
           1       0.56      0.76      0.65      4676

   micro avg       0.59      0.59      0.59      9554
   macro avg       0.61      0.60      0.58      9554
weighted avg       0.61      0.59      0.58      9554


Accuracy: 0.5931546996022609
```

[RapidMiner Process]
I first created a new repository in RapidMiner and moved my data into it. I then saved a copy of the in class-exercise to build off of for my RapidMiner model:

images (carlj)
HW1_Data (carlj - v1, 9/10/19 9:39 PM - 1.2 MB)
HW1Prediction (carlj - v1, 9/13/19 12:32 AM - 6 kB

I then looked at my variables and tried to identify any concerning datapoints. For example, revenue and eqpdays have negative numbers for their mins. I kept this in mind as I took a look at the histograms of all my variables. I noticed that many of them are heavily skewed to the left, as I have mentioned in my python process writeup:

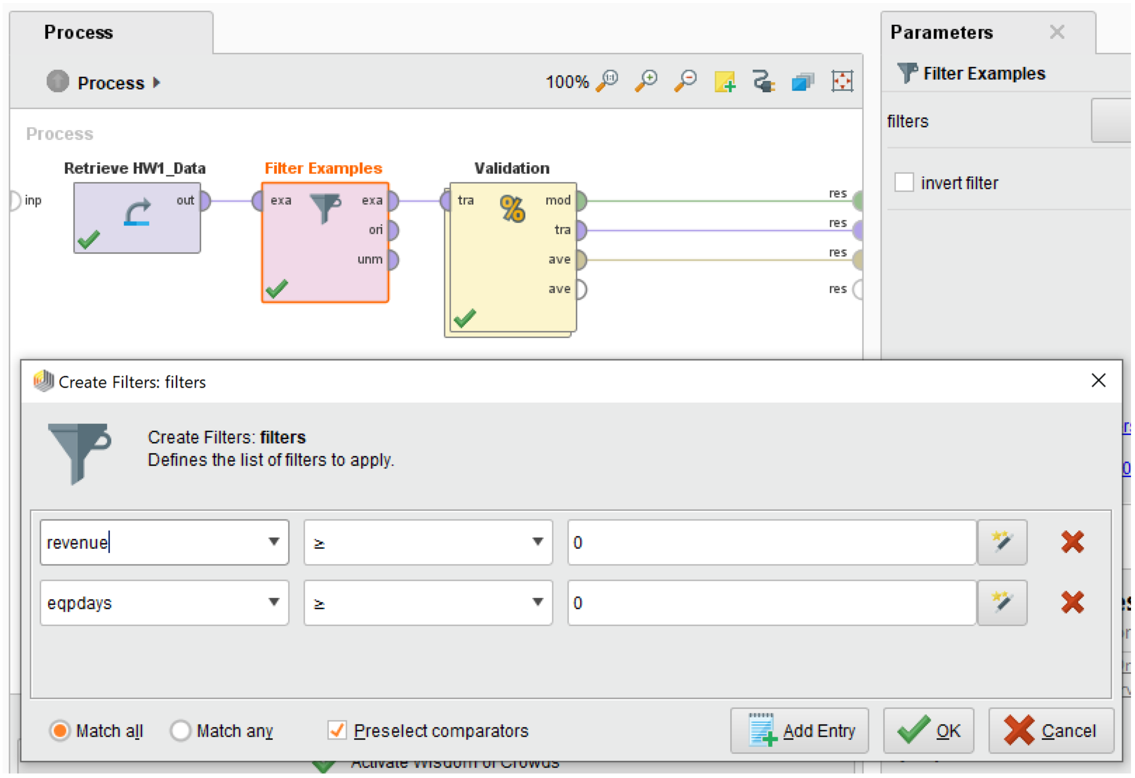


Filter (12 / 12 attributes):

Name	Type	Missing	Statistics				
Label churndep	Binominal	0	Least 1 (15855)	Most 0 (16036)	Values 0 (16036),	Details...	
revenue	Real	0	Min -5.860	Max 861.110	Average 58.665	Deviation 44.164	
outcalls	Real	0	Min 0	Max 610.330	Average 24.951	Deviation 34.790	
			Min	Max	Average	Deviation	

Showing attributes 1 - 12
Examples: 31,891 Special Attributes: 1 Regular Attributes: 11

Going back to my process map, I added a filter to remove all the bad data points mentioned in my previous step:



filters

☐ invert filter

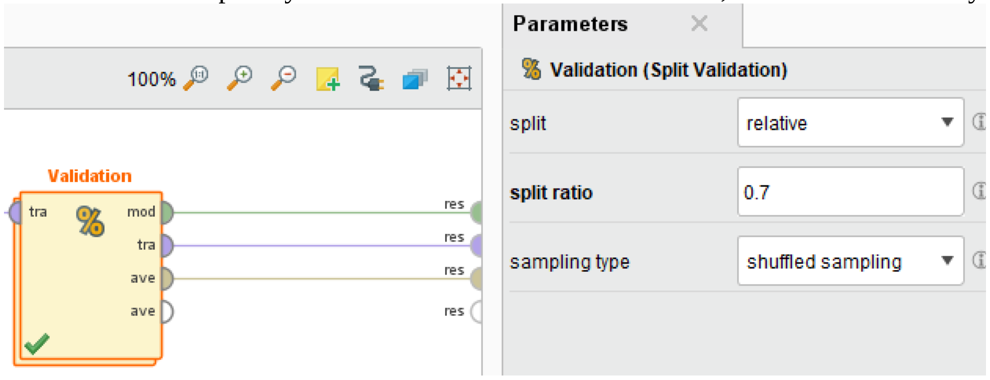
Create Filters: filters

Create Filters: **filters**
Defines the list of filters to apply.

revenue	≥	0
eqpdays	≥	0

[✓ Match all] Match any ☑ Preselect comparators

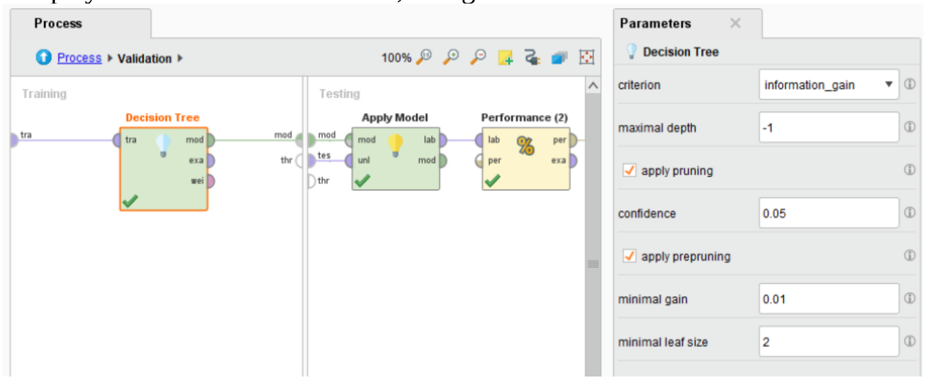
I made sure that I split my data between 70% train and 30% test, and that I shuffled my samples.



Validation (Split Validation)

split	relative
split ratio	0.7
sampling type	shuffled sampling

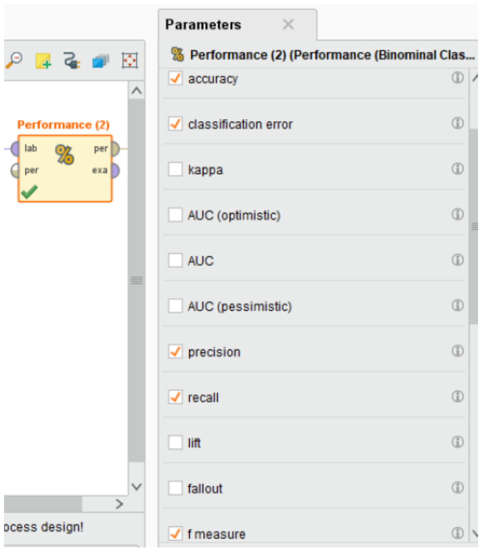
In the validation module, I made sure to change my criterion to information gain, max depth to -1 (this basically means no limit), and played around with confidence, min gain and min leaf size.



Decision Tree

criterion	information_gain
maximal depth	-1
☑ apply pruning	
confidence	0.05
☑ apply prepruning	
minimal gain	0.01
minimal leaf size	2

I then made sure that I got the output statistics on my model that I wanted by selecting them under the performance module:

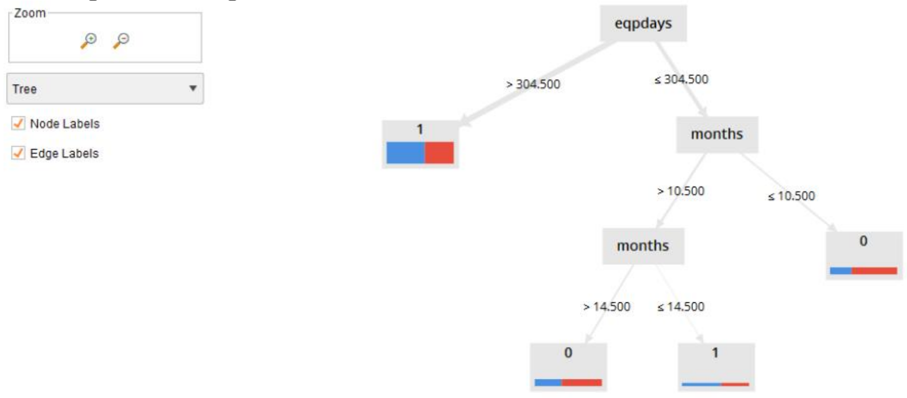


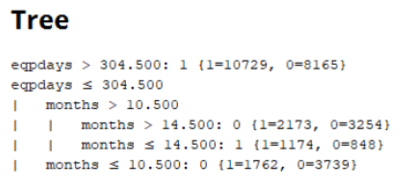
Parameters

Performance (2) (Performance (Binominal Clas...

- [✓] accuracy
- [✓] classification error
- [ ] kappa
- [ ] AUC (optimistic)
- [ ] AUC
- [ ] AUC (pessimistic)
- [✓] precision
- [✓] recall
- [ ] lift
- [ ] fallout
- [✓] f measure

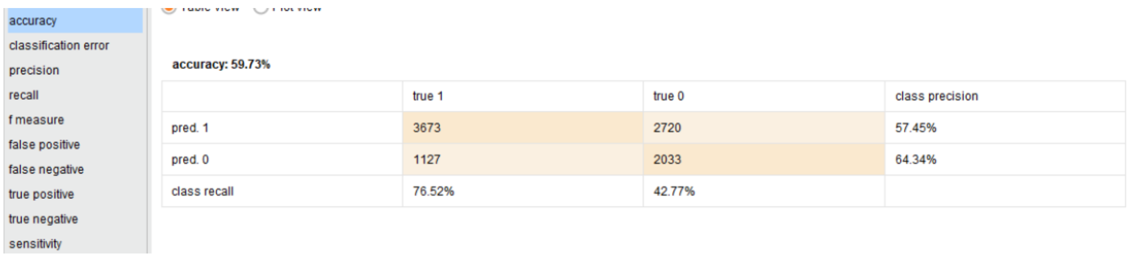
Running the model gave us our model decision tree, and its statistics below:





Tree

```
eqpdays > 304.500: 1 {1=10729, 0=8165}
eqpdays ≤ 304.500
|   months > 10.500
|   |   months > 14.500: 0 {1=2173, 0=3254}
|   |   months ≤ 14.500: 1 {1=1174, 0=848}
|   months ≤ 10.500: 0 {1=1762, 0=3739}
```



accuracy
classification error
precision
recall
f measure
false positive
false negative
true positive
true negative
sensitivity
Table View
Plot View

accuracy: 59.73%

	true 1	true 0	class precision
pred. 1	3673	2720	57.45%
pred. 0	1127	2033	64.34%
class recall	76.52%	42.77%	

All the individual statistics are in their own tab in the screenshot above. To save space, I have just typed them out below:
Classification error: 40.27%
Precision: 64.34% (positive)
Recall: 42.77% (positive)
f-measure: 51.38 (positive)
All True negative, True positive, false positive, sales negative values can be seen in the confusion matrix in the screenshot above.
Sensitivity: 42.77% (positive)

Similar to the summary statistics from my decision tree from the python model, our model is 'above' average given our f-measure, recall and precision values. The decision tree is an appropriate model for this problem because it helps us maximize information gain and is relatively efficient at classification problems. We can evaluate the predictive ability of the decision tree mostly by looking at the accuracy. Both my RapidMiner and Python models have an accuracy of around 59%, meaning that 59% of the time the model can correctly identify whether someone will churn or not. Given churn/not churn is 50/50, it's a small but still noticeable improvement of 9%. In practice, I would prefer making a decision tree with parameters that are the most effective at maximizing information gain and accuracy, while keeping the tree concise. When we did not have an additional parameter in the python model, the resulting decision tree looked like something straight out of a horror film. Also, upper management will want an answer/model that's the easiest to understand while being the most effective. An overly complex one will not satisfy the first requirement.

6) (20 points) Is a node's entropy generally lower or greater than its parent's? Is it ever possible for a node's entropy to be higher than its parent's entropy? Please justify your answer. Be precise but concise.

While a node's entropy is generally lower than the entropy of its parent, it is possible for a node's entropy to be higher than its parent's. This is dependent on how you split the parent node.

However, with the goal of increasing the purity of the data set in mind, typically a node's entropy should be lower than that of the parent's as smaller entropy leads to higher dataset purity.

Here's an example:

	Parent Node (10x, 5y)	
Child Node 1 (5x, 4y)		Child Node 2 (5x, 1y)

Parent Entropy = -(5/15)*log2(5/15)-(10/15)*log2(10/15)= 0.9183
Child 1 Entropy = -(5/9)*log2(5/9)-(4/9)*log2(4/9) = 0.9911
Child 2 Entropy = -(5/6)*log2(5/6)-(1/6)*log2(1/6) = 0.6500

Total information gain: 0.9183-(9/15)* 0.9911-(6/15)*0.6500 = 0.06364
Despite there being information gain, one of the two child nodes have a higher entropy than the parent node.

7) (5 points) What are the differences between supervised and unsupervised methods in machine learning? Is the decision tree algorithm a supervised or an unsupervised method? Be precise but concise.

Usually, supervised data mining results produces a model that predicts a quantifiable target variable of interest. Unsupervised methods are conducted with a specific target or purpose in mind while supervised methods do not. Supervised methods also require data on the target variable. Also, sometimes the results of unsupervised data mining methods are not useful or meaningful at all for any prediction purpose while supervised results are typically useful to a certain degree.

Decision tree algorithm falls under supervised methods because it is used for classification problems.