EMORY UNIVERSITY

Introduction to Business Data Analytics

Homework #1 (Part B)

Jane Doe

(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 be sufficient). If you borrow code and you don't provide the source, it is a violation of the Honor Code.

Total grade for Part 2:
__50___ out of ___50___ points

ATTENTION: HW1 has two parts. Please first complete the Quiz "HW1_Part1" on Canvas. Then, proceed with Part 2 in the following page. You will need to submit (a) a PDF file with your answers and screenshots of Python code snippets as well as Rapidminer repositories and (b) the Python code and Rapidminer repositories.

(50 points) [Implement this exercise with both Python (30 points) RapidMiner(20 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)

30 Points (Python):

a) (5 points) 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. Some possible issues / hints to think about: using training vs. test datasets.

b) (5 points) Explore how well the decision trees perform for several different parameter values (e.g., for different splitting criteria).

c) (5 points) Discuss the model (decision tree) that provides the best predictive performance from experimenting with different parameter values in question (b).

d) (15 points) Present a brief overview of your predictive modeling process, explorations, and discuss your results. 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? Make sure you present information about the model "goodness" (please report the confusion matrix, predictive accuracy, classification error, precision, recall, f-measure).

20 Points (Rapidminer):

As you discuss the results please make sure you provide screenshots of your corresponding Python code at the same time. At the end, also please provide the Rapidminer screenshots as well (Screenshot on how you split the data, how you built and evaluated the model, the Parameters panel for the Decision tree operator, all the corresponding performance metrics as well as the visualization of the decision tree).

Exploration:

In order to build a model to predict churn from the provided data set, we first need to examine the data. In order to do this we first print off basic summary statistics for the data set across all attributes. It is evident that there are no missing values from the counts, however, we can see that we have negative values for revenue and eqpdays. In this instance, we removed the observations with negative equipment days. However, we left the negative revenue values as they may have a more understandable business application, for instance they may represent reimbursements given back to the customer. We only filtered out 14 observations that happened to be negative for eqpdays.

```
hw1 = pd.read_csv('HW1_Data.csv')
print(hw1.head())
print(hw1.dtypes)
hw1.describe()
```

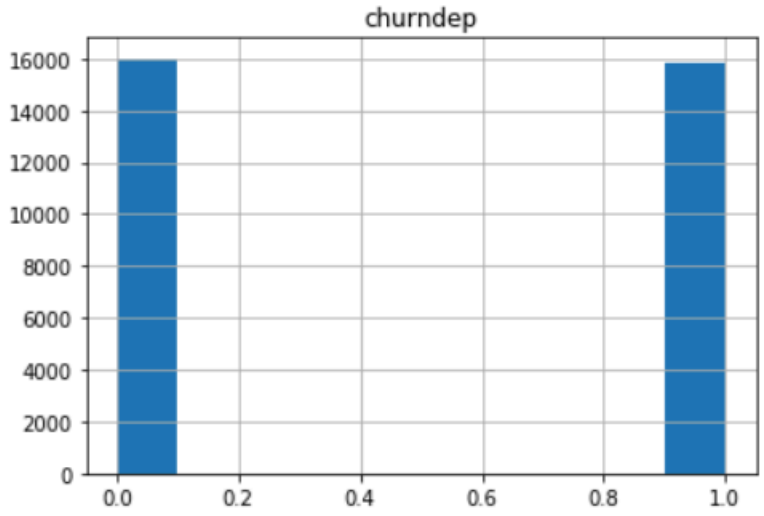
	revenue	outcalls	incalls	months	eqpdays	webcap	marryyes	travel	pcown	creditcd	
count	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891
mean	58.665179	24.951385	8.065277	18.761908	391.222633	0.894704	0.363175	0.057163	0.184817	0.676931	0
std	44.163859	34.790147	16.610589	9.548019	254.998478	0.306939	0.480922	0.232158	0.388155	0.467656	0
min	-5.860000	0.000000	0.000000	6.000000	-5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	33.450000	3.000000	0.000000	11.000000	212.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0
50%	48.380000	13.330000	2.000000	17.000000	341.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0
75%	71.040000	33.330000	9.000000	24.000000	530.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0
max	861.110000	610.330000	404.000000	60.000000	1812.000000	1.000000	1.000000	1.000000	1.000000	1.000000	4

```
hw1 = hw1[hw1.eqpdays >= 0]
```

A Histogram can help us to understand the distribution of our classes in the data. (i.e. churn or not churned). If the data is not evenly distributed, we may have to take additional steps to train and evaluate our classifiers.

Pandas has a simple, built-in histogram function that will allow us to quickly view the class distribution.

```
hw1.hist(column='churndep')
```

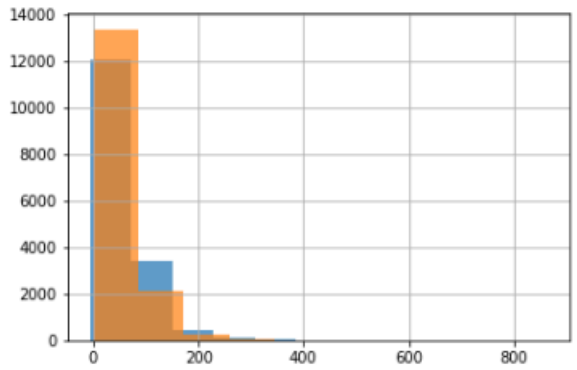


The distribution of the two classes is almost equal.

It can also be helpful to look at a histogram of each individual attribute/feature. This allows us to see the distribution of classes within each feature. This can help us determine if the individual feature itself would be useful in predicting churn (our target class). We can also check if a feature has a very skewed distribution of classes.

For example, to plot a histogram of revenue, we can do the following:

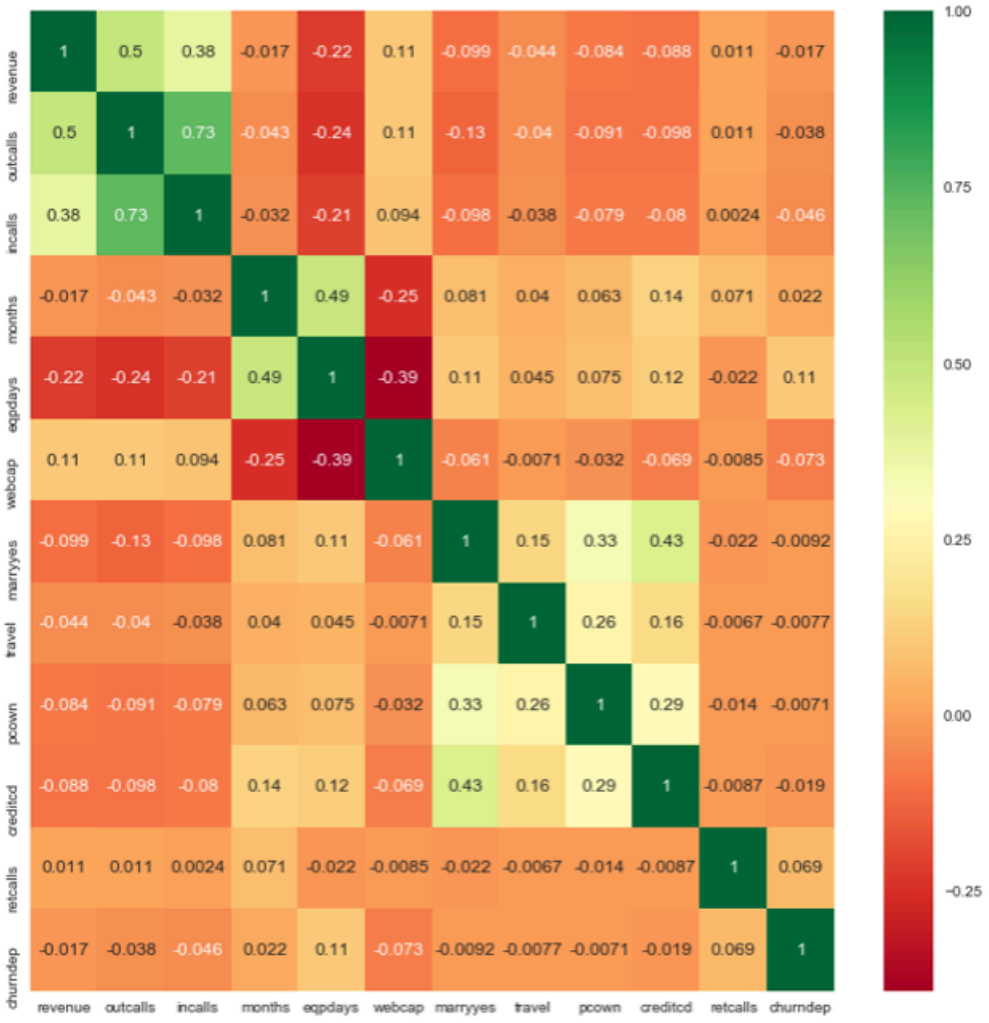
```
In [23]: hw1.groupby('churndep').revenue.hist(alpha=0.7)
Out[23]: churndep
         0    AxesSubplot(0.125,0.125;0.775x0.755)
         1    AxesSubplot(0.125,0.125;0.775x0.755)
         Name: revenue, dtype: object
```



To plot the other attributes, replace ‘revenue’ with the other attribute within our model.

For a deeper look at our data, we can also look at the correlation between features. This can help us identify which features are most related to our target class/variable. We are using the seaborn library to plot this correlation matrix.

```
import seaborn as sns
corrmatrix = hw1.corr()
top_correlated_features = corrmatrix.index
plt.figure(figsize=(12,12))
#plot heat map
plot=sns.heatmap(hw1[top_correlated_features].corr(),annot=True,cmap="RdYlGn")
```



Modeling:

In order to tackle this supervised learning problem, we plan to use Decision Trees. A tree model will provide us with a good prediction for our target variable of ChurnDep based on our other features.

However, before we start modeling we must split our data set in order to properly evaluate our predictions with out of sample testing. We split our data set into training and test data as shown below.

```
# Retrieving Attributes
X = df.iloc[:, :11]
# Retriving Target Variable
y = df.iloc[:, 11]

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, classification_report

# Split validation
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=1)
```

After properly splitting the data we can finally start modeling and testing out which parameters should be tuned to improve model performance. In order to compare different models we will look at Accuracy of the test set as a benchmark.

Max Leaf Nodes:

```
# try different max_leaf_nodes parameter values:
for i in range(2,50,1):
    clf1 = DecisionTreeClassifier(criterion='entropy', splitter='best', max_depth=None, max_leaf_nodes=i)
    clf1 = clf1.fit(X_train, y_train)
    y_pred1 = clf1.predict(X_test)
    rate = accuracy_score(y_test, y_pred1)
    print(str(i) +" " + str(rate))
```

```
2 0.5857023411371237
3 0.5857023411371237
4 0.5977215719063546
5 0.5977215719063546
6 0.6002299331103679
7 0.6002299331103679
8 0.6002299331103679
9 0.6002299331103679
```

Minimum Impurity Decrease:

```
# try different min_impurity_decrease values:
import numpy

for i in numpy.arange(0,0.011,0.001):
    clf2 = DecisionTreeClassifier(criterion='entropy', splitter='best', max_depth=None, min_impurity_decrease = i,
    clf2 = clf2.fit(X_train, y_train)
    y_pred2 = clf2.predict(X_test)
    rate = accuracy_score(y_test, y_pred2)
    print(str(i) + " " + str(rate))
```

```
0.0 0.6006479933110368
0.001 0.6002299331103679
0.002 0.5977215719063546
0.003 0.5977215719063546
0.004 0.5977215719063546
0.005 0.5857023411371237
0.006 0.5857023411371237
0.007 0.5857023411371237
0.008 0.5857023411371237
0.009000000000000001 0.5857023411371237
0.01 0.5857023411371237
```

By evaluating these models across different parameters in a For-loop we are able to identify where the maximum gain in accuracy occurs. Which happens to be at a max_leaf_nodes of 14,

and a min_impurity_decrease of .001. It is important to remember that our goal in calculating this measure is to ensure the purest partitions of our decision tree branches. This is so that ideally, all tuples within a partition would belong to the same class. This ensures that when new data is encountered, new data can be accurately compared to tree partitions, and tree partitions represent the characteristics of the data that determine Churn or Not Churn as closely as possible.

[**Note**: In practice nested hold-out validation or nested cross-validation would be the best way to choose the parameters of a data science method as we will learn later on during the course.]

Final Model

By stepping through these parameters and evaluating the tree model, we are able to be very confident in our predictions. It is important to note, that increasing the number of Max_leaf_nodes past 14 had diminishing returns. Essentially as the complexity of our model increased we received no benefit in performance. The final decision tree model is outlined below:

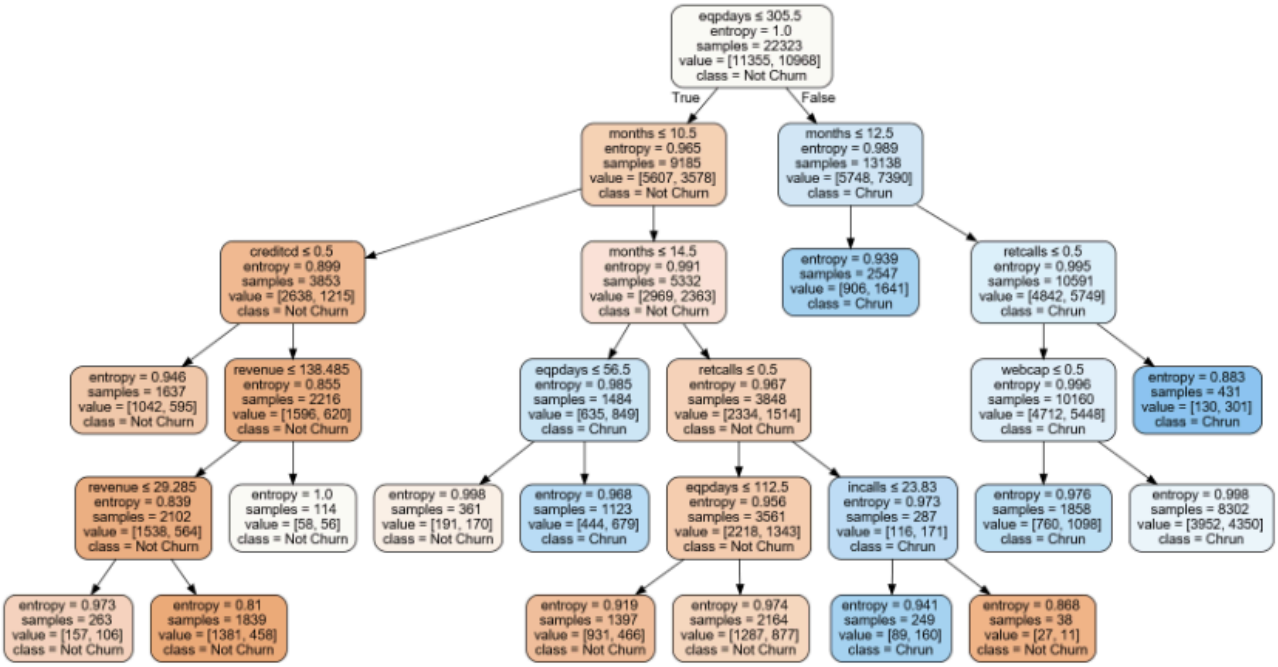


Figure 1

Evaluation and Conclusion:

In order to truly evaluate our model, we must look at all performance metrics. We have higher scores for Recall and F1, for our positive Churn class than we do for our negative "not churn". This may be beneficial as we are trying to predict whether customers will leave and therefore will likely be offering them a promotion of some sort to ensure their loyalty. In this instance, a False Negative (predicted not churn, true churn) will likely be the most expensive as we will be losing customers at that point. Looking at the confusion matrix, this class is minimized which is a plus for our predictor.

A decision tree is appropriate for this task because it can clearly represent how our model is making decisions. It will be easier to explain to others.

Moving forward, we should try different tactics to improve our predictions. This may be feature engineering, feature selection, or trying a whole new model altogether.

We could create a new feature that combined relevant attributes of our data. For example, it may be possible to combine ‘eqpdays’ and ‘months’ into one higher-level attribute. In terms of feature engineering, we may be able to apply a scaler to our data if we use models other than decision trees.

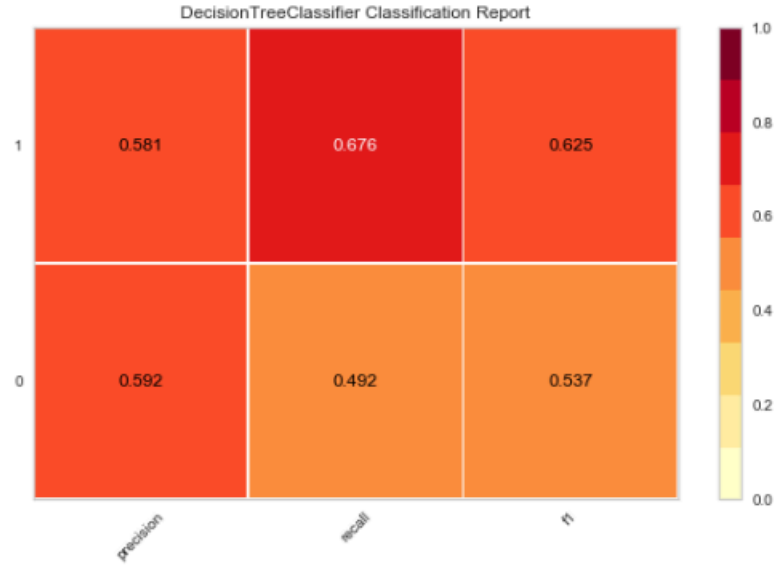
[Note: As we will learn later on in the class, we could also ‘prune’ attributes of the tree that are causing overfitting which in turn result in a drop in our model’s accuracy on new data. Pruning attributes could make our model more generalizable.]

```
# produce confusion matrix
from sklearn.metrics import confusion_matrix

pd.DataFrame(
    confusion_matrix(y_test, y_pred),
    columns=['Predicted Not Churn', 'Predicted Churn'],
    index=['True Not Churn', 'True Churn']
)
```

	Predicted Not Churn	Predicted Churn
True Not Churn	2097	2584
True Churn	1237	3650

We can also look at a heatmap. We are using the yellowbrick package to make the heatmap as shown below.

```
clf1 = DecisionTreeClassifier(criterion='entropy', splitter='best',max_depth=None, max_leaf_nodes=
clf1 = clf1.fit(X_train, y_train)
y_pred1 = clf1.predict(X_test)
#ate = accuracy_score(y_test, y_pred1)
visualizer = ClassificationReport(clf1)

visualizer.fit(X_train, y_train)  # Fit the visualizer and the model
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
visualizer.poof()
```




```
# precision, recall, f1-score
print (classification_report(y_test, y_pred, target_names=['Not Churn', 'Churn']))
```

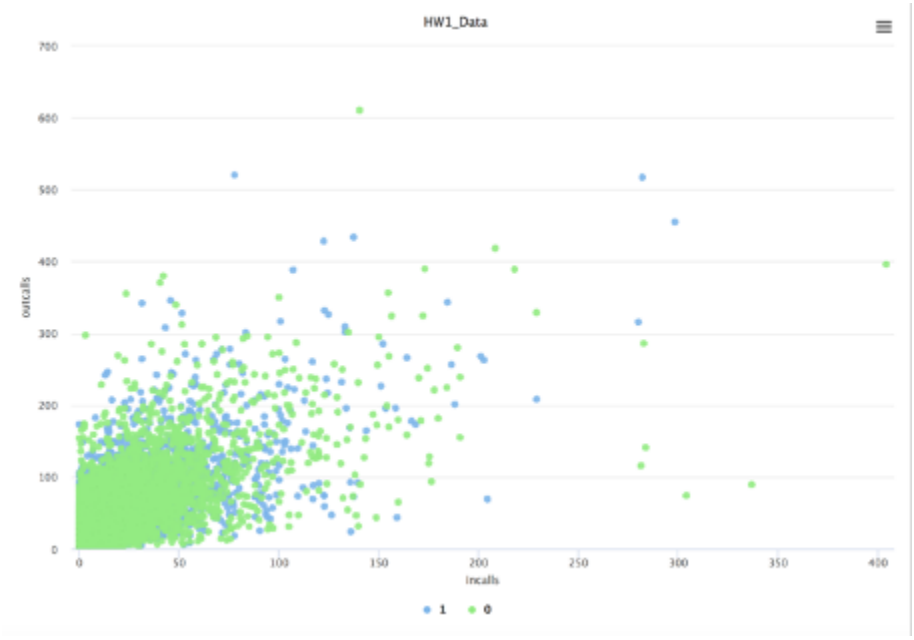
```
              precision    recall  f1-score   support

   Not Churn       0.63      0.45      0.52      4681
       Churn       0.59      0.75      0.66      4887

 avg / total       0.61      0.60      0.59      9568
```

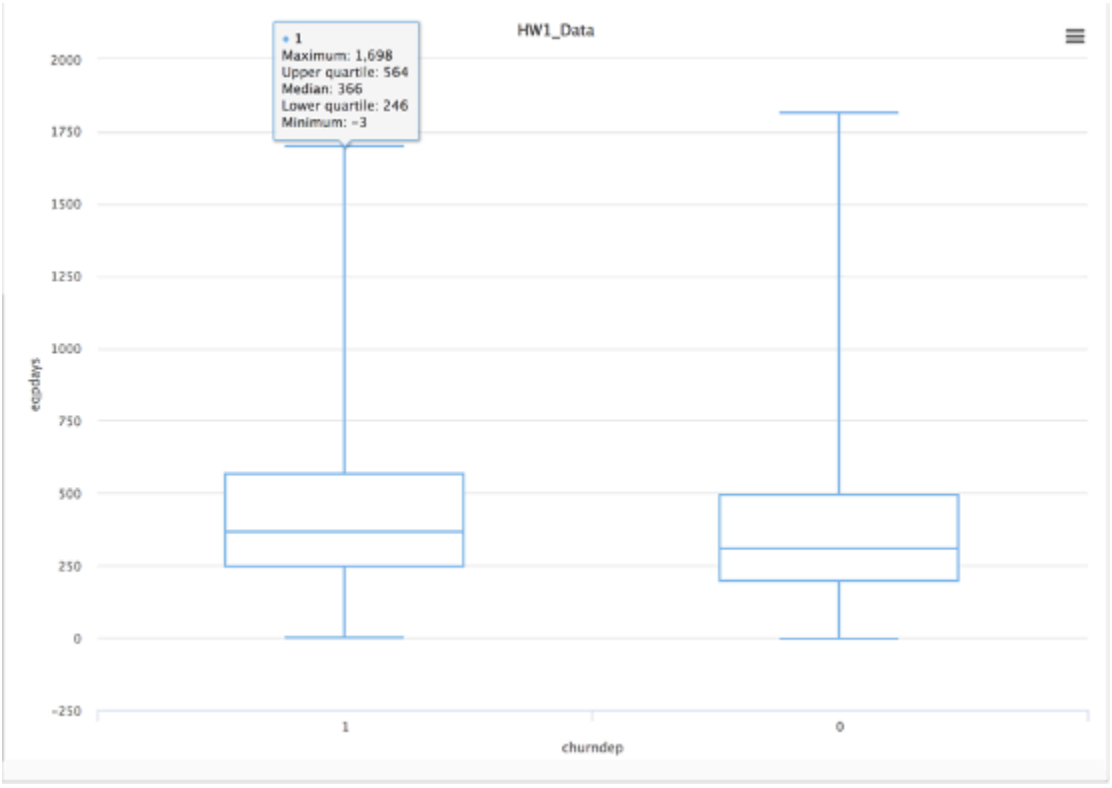
RapidMiner:

The process taken in RapidMiner mirrors that of Python. However, playing with the parameters proved to be more challenging as we had to update them by hand as opposed to being able to run it through a loop. Process image and evaluation metrics provided below.

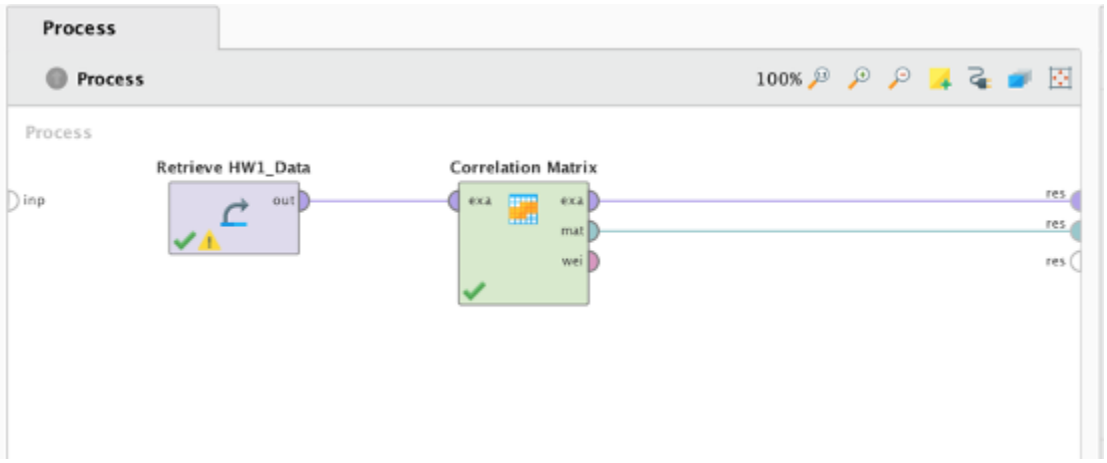
1. Please provide the following screenshots
Decision Tree Screenshots

Data Exploration Screenshots

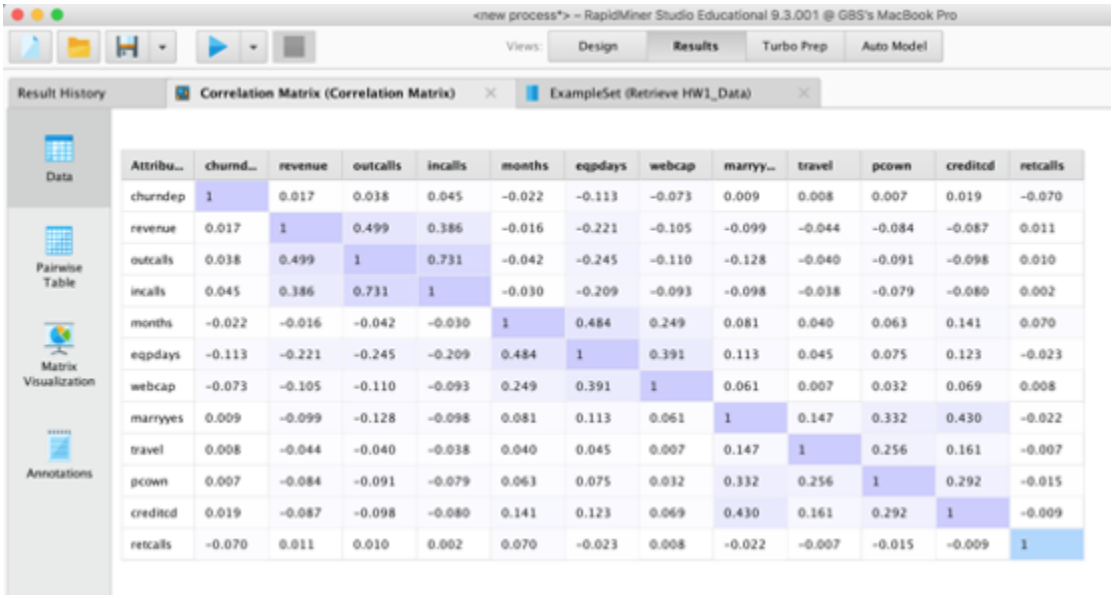




Similarly, I could explore the correlations in the data as follows:



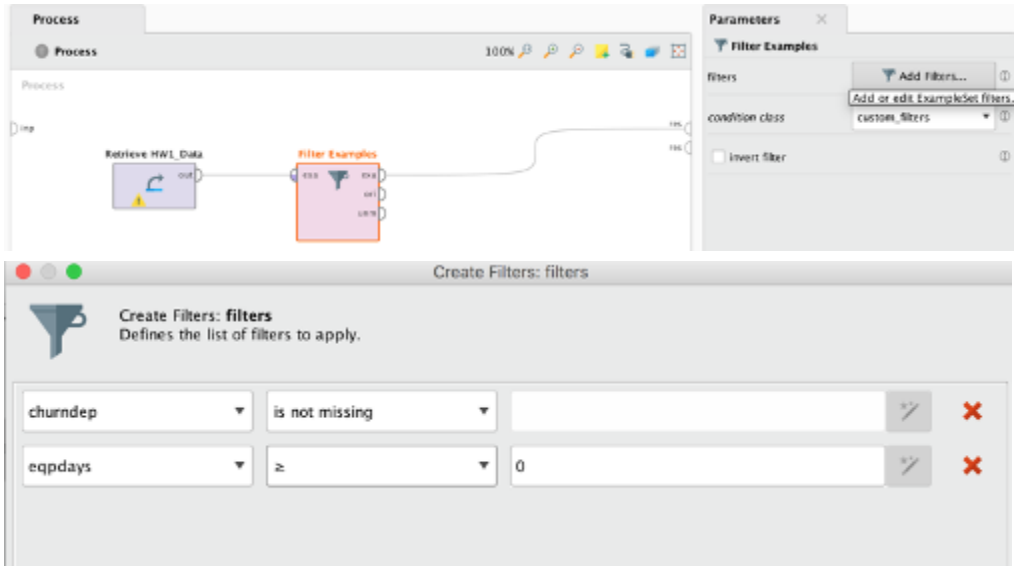
[Note: If you have declared your target variable (i.e., green column), Rapidminer will not include that column in the correlation matrix.]



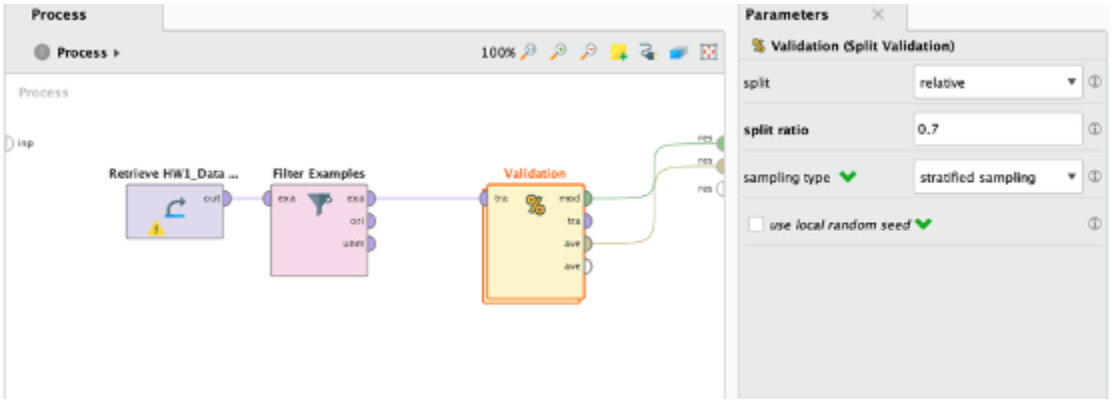
Attribu...	churnd...	revenue	outcalls	incalls	months	eqpdays	webcap	marryy...	travel	pcown	creditcd	retcalls
churndep	1	0.017	0.038	0.045	-0.022	-0.113	-0.073	0.009	0.008	0.007	0.019	-0.070
revenue	0.017	1	0.499	0.386	-0.016	-0.221	-0.105	-0.099	-0.044	-0.084	-0.087	0.011
outcalls	0.038	0.499	1	0.731	-0.042	-0.245	-0.110	-0.128	-0.040	-0.091	-0.098	0.010
incalls	0.045	0.386	0.731	1	-0.030	-0.209	-0.093	-0.098	-0.038	-0.079	-0.080	0.002
months	-0.022	-0.016	-0.042	-0.030	1	0.484	0.249	0.081	0.040	0.063	0.141	0.070
eqpdays	-0.113	-0.221	-0.245	-0.209	0.484	1	0.391	0.113	0.045	0.075	0.123	-0.023
webcap	-0.073	-0.105	-0.110	-0.093	0.249	0.391	1	0.061	0.007	0.032	0.069	0.008
marryyes	0.009	-0.099	-0.128	-0.098	0.081	0.113	0.061	1	0.147	0.332	0.430	-0.022
travel	0.008	-0.044	-0.040	-0.038	0.040	0.045	0.007	0.147	1	0.256	0.161	-0.007
pcown	0.007	-0.084	-0.091	-0.079	0.063	0.075	0.032	0.332	0.256	1	0.292	-0.015
creditcd	0.019	-0.087	-0.098	-0.080	0.141	0.123	0.069	0.430	0.161	0.292	1	-0.009
retcalls	-0.070	0.011	0.010	0.002	0.070	-0.023	0.008	-0.022	-0.007	-0.015	-0.009	1

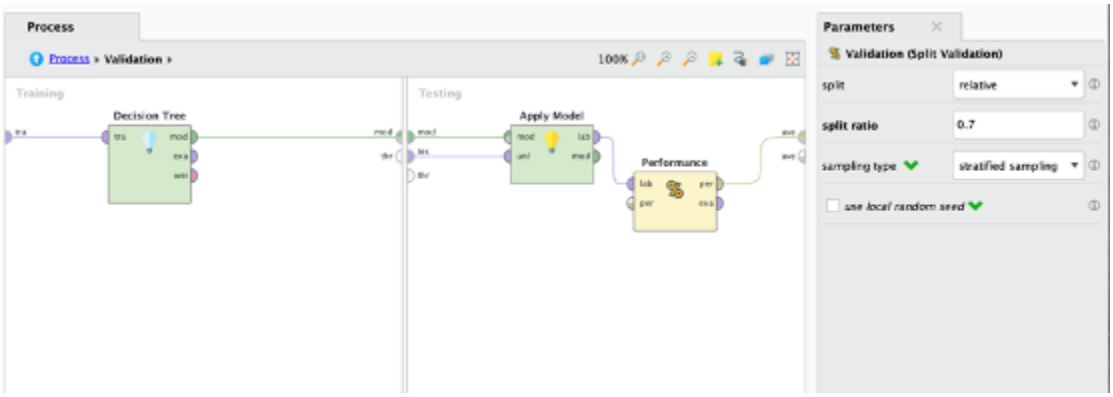
Irregular data points could be filtered with the “Filter Examples” operator as follows:

Once you click on “Filter Examples”, then click on Add Filter options in the Parameters section



i.Screenshots for Decision Tree Model Setup (All Relevant RapidMiner Processes)





ii.Screenshot for Decision Tree Performance Metrics

accuracy: 59.60%

	true 1	true 0	class precision
pred. 1	3580	2688	57.12%
pred. 0	1171	2114	64.35%
class recall	75.35%	44.02%	

classification_error: 40.40%

	true 1	true 0	class precision
pred. 1	3580	2688	57.12%
pred. 0	1171	2114	64.35%
class recall	75.35%	44.02%	

precision: 64.35% (positive class: 0)

	true 1	true 0	class precision
pred. 1	3580	2688	57.12%
pred. 0	1171	2114	64.35%
class recall	75.35%	44.02%	

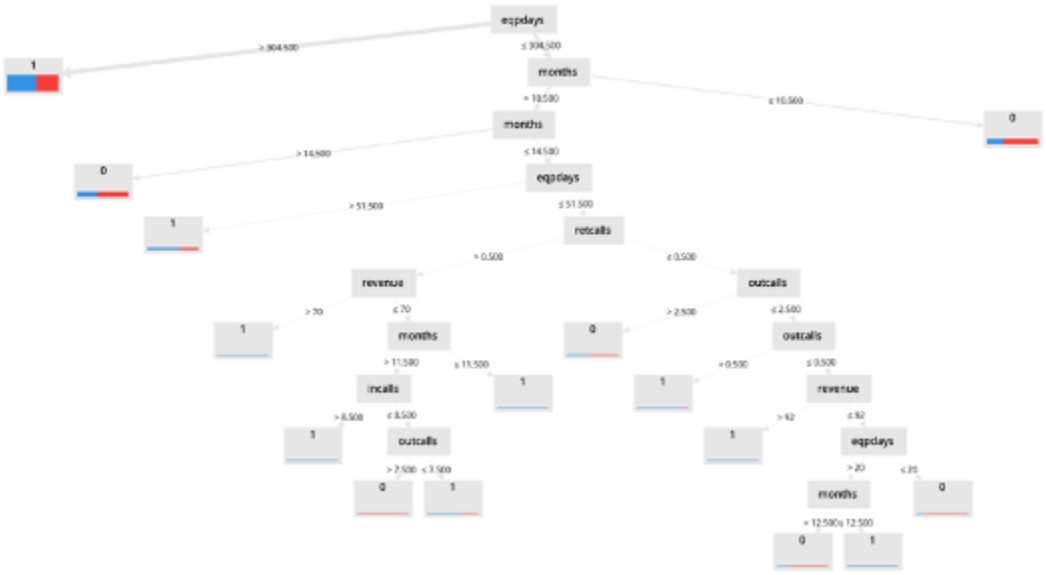
recall: 44.02% (positive class: 0)

	true 1	true 0	class precision
pred. 1	3580	2688	57.12%
pred. 0	1171	2114	64.35%
class recall	75.35%	44.02%	

f_measure: 52.28% (positive class: 0)

	true 1	true 0	class precision
pred. 1	3580	2688	57.12%
pred. 0	1171	2114	64.35%
class recall	75.35%	44.02%	

iii.Screenshot for Decision Tree Results (Visualization of Decision Tree)



iv.Screenshot for Decision Tree RapidMiner Operator Parameters (click on Decision Tree operator and then take a screenshot of the Parameters window on the right)