Homework #1

Mayur Kumar

(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.

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?

1)BUSINESSS UNDERSTANDING:

This is very imperative step, because as we keep asking the questions, this would help us understand the problem statement. As we gain proper understanding of the problem, we would eventually and intuitively be able to select right variables as well.

MTC is facing an issue of churning here. We defined our target variable as whether a customer churns or not and encoded in binary form as 0 or 1.

Some questions that we can ask

- We need to understand why customers are churning
  Is it because of Poor service, Lack of usage, Better price available in the market?bad customer service? Quality? Quantity?

- which customers are churning
  particular region people? Demographics? Income? Etc

- when they are churning
  Immediately? After a long time?

- what are the plausible reasons that are causing to customers leave the service.

With this fundamental knowledge, we are better equipped now to look into dataset and solve the problem.

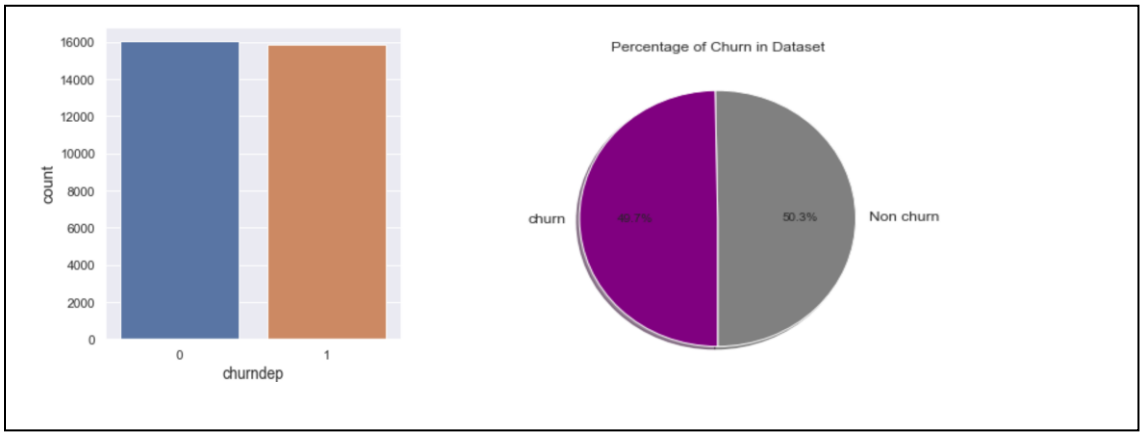
2)DATA UNDERSTANDING:

i) Structure of the complete dataset:

Number of Observations	Explanatory variables	Response variables
31891	11	1

Size of the dataset is fairly big enough to model and talk about the results confidently.

ii) Distribution of Dependent variable and it's statistics



Non churn : class 0	Churn : class 1
16006	15838

Our Target variable in our dataset is well balanced for each class, This helps us to think which sampling strategy we need to use while splitting the data. we can split the data into training and test dataset by using **shuffled or stratifying**. It wouldn't make much difference since the dataset is well balanced.

iii) Distribution of Independent Variables and it's statistics:

	revenue	outcalls	incalls	months	eqpdays	webcap	marryyes	travel	pcown	creditcd	retcalls
count	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000	31891.000000
mean	58.665179	24.951385	8.065277	18.761908	391.222633	0.894704	0.363175	0.057163	0.184817	0.676931	0.044088
std	44.163859	34.790147	16.610589	9.548019	254.998478	0.306939	0.480922	0.232158	0.388155	0.467656	0.224552
min	-5.860000	0.000000	0.000000	6.000000	-5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	33.450000	3.000000	0.000000	11.000000	212.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	48.380000	13.330000	2.000000	17.000000	341.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000
75%	71.040000	33.330000	9.000000	24.000000	530.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
max	861.110000	610.330000	404.000000	60.000000	1812.000000	1.000000	1.000000	1.000000	1.000000	1.000000	4.000000

- Looking at the statistics of each variable, we see that revenue and eqpdays are taking negative values, which is ideally not possible**.** We need to filter the dataset by removing these values so that we can train the model on a quality dataset
- Mean value of 0.89 for webcap says that most of the customers have web capable phones
- Mean value of 0.05 for travel says that a very few customers have travelled to non US countries

3)DATA CLEANING:

```
churn2= churn[(churn.revenue>=0) & (churn.eqpdays>=0)]
churn2.info()
```

Our new dataset now has better quality with 31844 number of observations. Now we can do data exploration and visualization on this filtered dataset to gain more insights

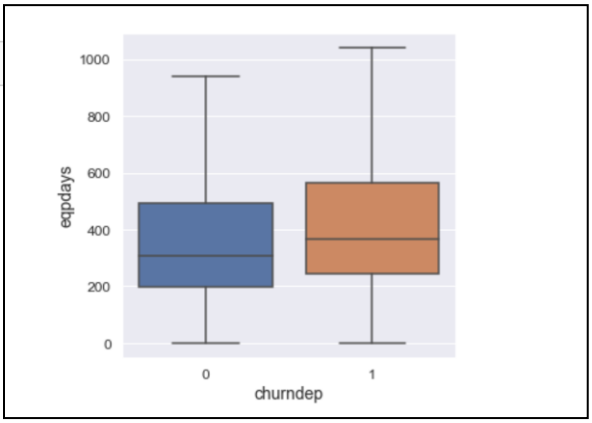
4)DATA EXPLORATION AND VISUALIZATION:

We plot graphs for each independent variable vs churndep. Most of the graphs didn't give much of the insights apart from the insights that we got already but two of the following graphs give some insights.

Graph 1)eqpdays vs churndep

```
print(churn2.groupby("churndep").mean())
```

churndep	revenue	outcalls	incalls	months	eqpdays	webcap	marryyes	travel	pcown	creditcd	retcalls
0	59.383509	26.232304	8.806221	18.544608	363.289704	0.917094	0.367550	0.059040	0.187680	0.685868	0.028614
1	57.852762	23.607592	7.289834	18.971840	420.591047	0.872080	0.358757	0.055436	0.182157	0.668266	0.059477



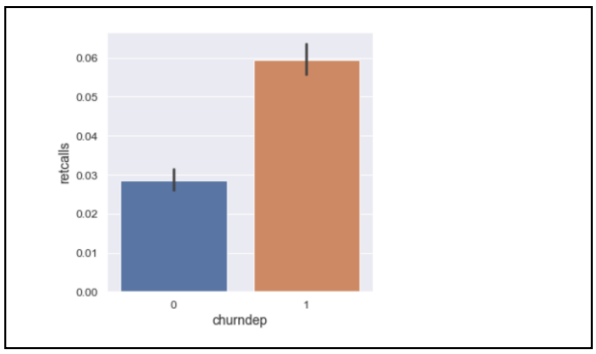
Insights:

- On an average churners are using the equipment more than non churner and on an aerage, after 420days of equipment usage, churning is happening. This would probably help us to think that we need to be mindful as we approach 420days so that we can send some vouchers before churning.

- we could check what exactly happening around 420 days

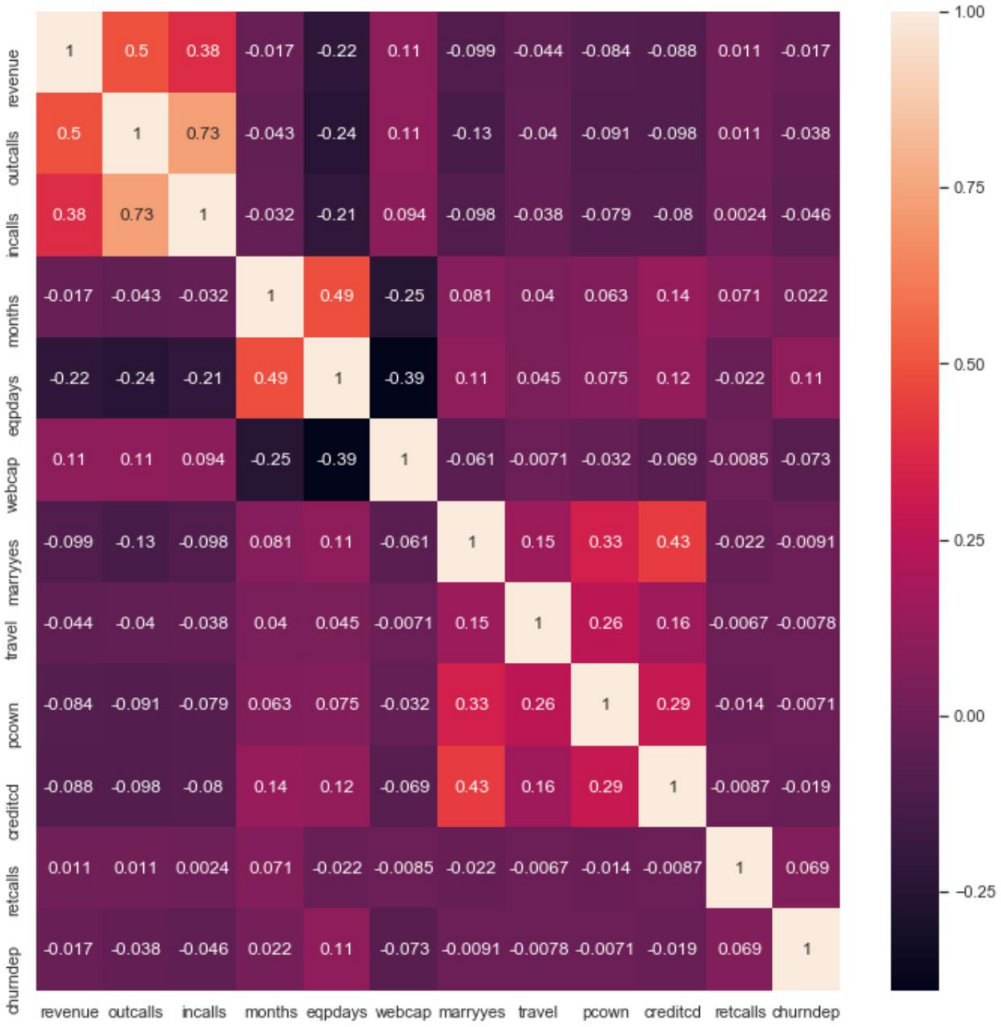
Graph 2) retcalls vs churndep

On an average, churners are making more calls to retention team than non churners. One of the plausible reasons that we could think of is that they might be not happy with the product or other sorts of complaints. But this plot definitely helps us getting closet to the problem and have gain more understanding.



Further we could find correlation between each variable and get more understanding of how each variable relates to other variables.

Correlation between each variable:



Insights:

None of the variables are strongly correlated to dependent variable. Two of the independent variables(outcalls and incalls) are highly correlated , saying that customers with inbound calls are the one with outbound calls. But this is not really helpful.

5)MODEL BUILDING

We are using decision tree to build the model.

- **Why Decision tree for this business problem?**

1)Our business context demands an understanding of which attributes are leading a customer to churn
2)Easy to interpret. For a given attributes of a particular customer, we can trace down the tree to see if he churns or not. It is not a black box

i)SPLIT THE DATASET

Now we are all set to model building. Our first step is to split the data into two sets.
1)Train dataset
2)Test dataset

We could split in

- 70% train data set and 30% test dataset or
- 80% train dataset and 20% test dataset

SPLIT THE DATA INTO TRAINING AND TEST DATA

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=0)
```

Our Train dataset size = 80% of the dataset
Test dataset size = 20 % of the datset

- We are splitting the data set into train and test datset by specifying the samplying type as **shuffled**. We could do stratifying as well since the dataset is well balanced.
- We now have X_train, y_train for train dataset , and X_test, y_test for test dataset

ii)TRAIN THE MODEL

Let's make two models and compare the performance.

Model 1 : Train with ***gini impurity*** criteria.
Model 2 : Train with **information gain** criteria.

a)TRAINING THE MODEL 1 WTH GINI IMPURITY.

We train the model 1 with train data set. Let's name the model as gini_clf
Code for training the model with gini impurity and getting the deicion tree graph

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
import graphviz

# Training the model with Gini criteria
gini_clf = tree.DecisionTreeClassifier()
gini_clf = gini_clf.fit(X_train, y_train)


#get the decision tree graph

feature_names = list(X.columns)   # we need column names in the list form
gini_tree = tree.export_graphviz(gini_clf, out_file=None,
                                 feature_names=feature_names,
                                 class_names= '01',
                                 filled=True, rounded=True,
                                 special_characters=True)

gini_graph = graphviz.Source(gini_tree)

gini_graph
```

Here we are not specifying the max_depth value, so the decision tree would keep splitting as much as it can, and prone to **over fitting** as it is trying to fit all the points in train dataset**.** Since we didn't specify the max_depth value, the decision tree is very big and it's hard to paste the graph here.

MODEL 1 EVALUATION:

Now that the model is trained with train dataset,
let's run this model on test data set and get the predicted values.

```
y_pred_gini = gini_clf.fit(X_train, y_train).predict(X_test)     #prediction for test data set
```

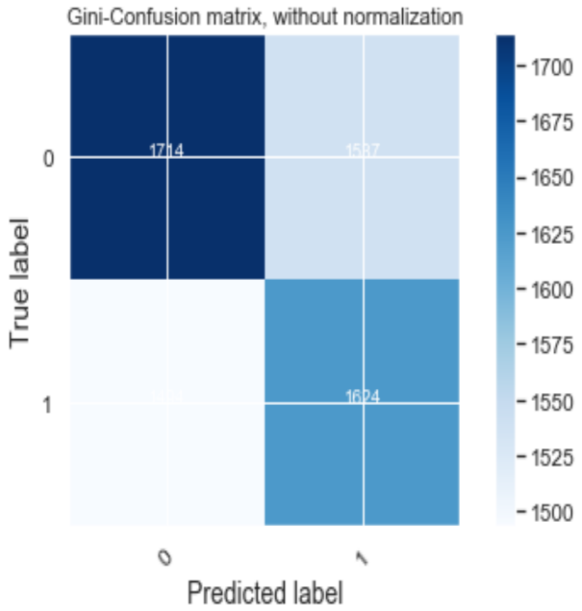
Then using confusion matrix, we will compare the predicted values for test data set with actual values for the test dataset to see how well our model is performing.

```
#generate confusion matrix
gini_matrix = confusion_matrix(y_test, y_pred_gini)
np.set_printoptions(precision=2)
```

Let's plot the normal confusion matrix and normalized confusion matrix using the following code.

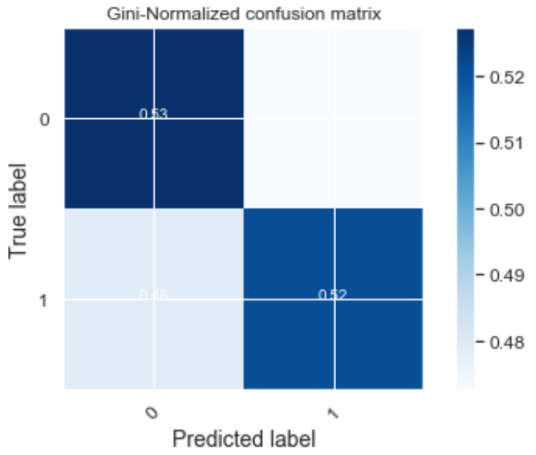
```
#plot confusion matix
plt.figure()
plot_confusion_matrix(gini_matrix, classes='01',
                      title='Gini-Confusion matrix, without normalization')

plt.figure()
plot_confusion_matrix(gini_matrix, classes='01', normalize=True,
                      title='Gini-Normalized confusion matrix')

plt.show()
```

Following is the confusion matrix without normalization that clearly shows true positive, true negative, false positive, false negative values



TP= 1624
TN=1714
FP=1587
FN= 1404

We can normalize this table to better understand the proportion of the above values.



Let's calculate the metrics such as ACCURACY, PRECISION, RECALL SCORE, F1 SCORE to evaluate this model.

```
gini_Accuracy   = accuracy_score(y_test,y_pred_gini)
gini_Precision = precision_score(y_test,y_pred_gini)
gini_Recall = recall_score(y_test,y_pred_gini)
gini_F1measure = f1_score(y_test,y_pred_gini)
```

ACCURACY METRIC = (TP+TN)/(TP+FP+FN+TN)

gini_Accuracy = 52%

- Accuracy is an appropriate measure for our business problem because we have symmetric datasets (falsenegatives & false positives counts are close), also**,** false negatives & false positives have similar costs**.**We have false positive =1587 and false negative =1404, so accuracy is a good metric to evaluate our business model. In that case, 52% Accuracy may say that this model isn't a great one to solve this business problem.

PRECISION METRIC =TP/(TP+FP)

gini_Precision = 51%

- Generally we choose precision as a metric to evaluate if we want to be more confident of our true positives i..e who are the exact customers that are churning**.** It's like I would rather have non churners predicted as churner than incorrectly predicting the churners as non churners. Hence, Given our business problem, this is definitely a good metric but doesn't encompass the entire importance.

RECALL METRIC = TP/(TP+FN)

gini_Recall = **52%**

- Generally we choose this metric**,** If the idea of false positives is far better than false negatives, in other words**,** if the occurrence of false negatives is unaccepted/intolerable**,** that We'd rather get some extra false positives(**Non churners)** over saving some false negatives.
  You'd rather get some churners labeled as non churners over leaving a churner labeled Non churner. Given our business problem, this is definitely a good metric but doesn't encompass the entire importance

F1 MEASURE = 2*(Recall * Precision) / (Recall + Precision)

gini_F1measure = **51%**

- This definitely a great measure if we have asymmetric class distribution which is not the case here.

SUMMARY

Inorder to evaluate this particular model Accuracy is the metric we choose in the context of our business problem , and 52% is not a great accuracy for this business problem.

b)TRAINING THE MODEL 2 WTH INFORMATION GAIN:

Following the same steps as we did for Model 1. This time our decision tree model is created with splitting criteria "entropy" indicating that it uses information gain to measure quality of split.

Train the model with the training dataset and keeping Information gain as the criteria. In this case as well, since we are not specifying the max_depth , the model is prone to overfit as it is trying to fit all the data points in train data set,

```
# Training the model with Information gain
clf_infoG = tree.DecisionTreeClassifier(criterion="entropy")
clf_infoG = clf_infoG.fit(X_train, y_train)
```

MODEL 2 EVALUAITON

Now that the model is trained with training data set, we want to see how it performs on test data set.
Let's apply this model on test data set , and get the predicted values.

```
y_pred_infoG = clf_infoG.fit(X_train, y_train).predict(X_test)
```

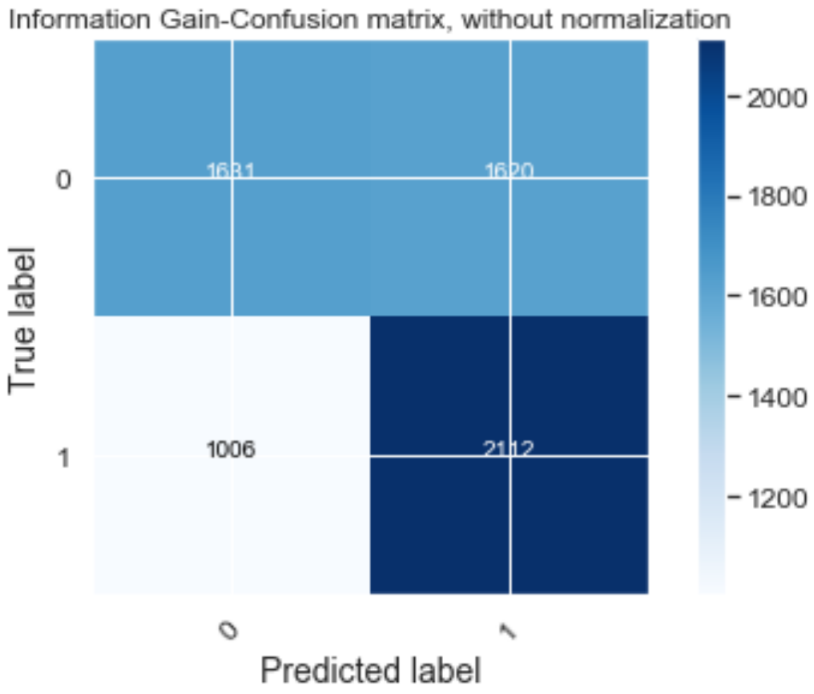
We want to evaluate the model by comparing the predicted values for test data set and the actual values in that test data set. we would do this with the help of confusion matrix.

```
infoG_matrix = confusion_matrix(y_test, y_pred_infoG)
np.set_printoptions(precision=2)
```

Let's plot the confusion matrix for this model with the following code:

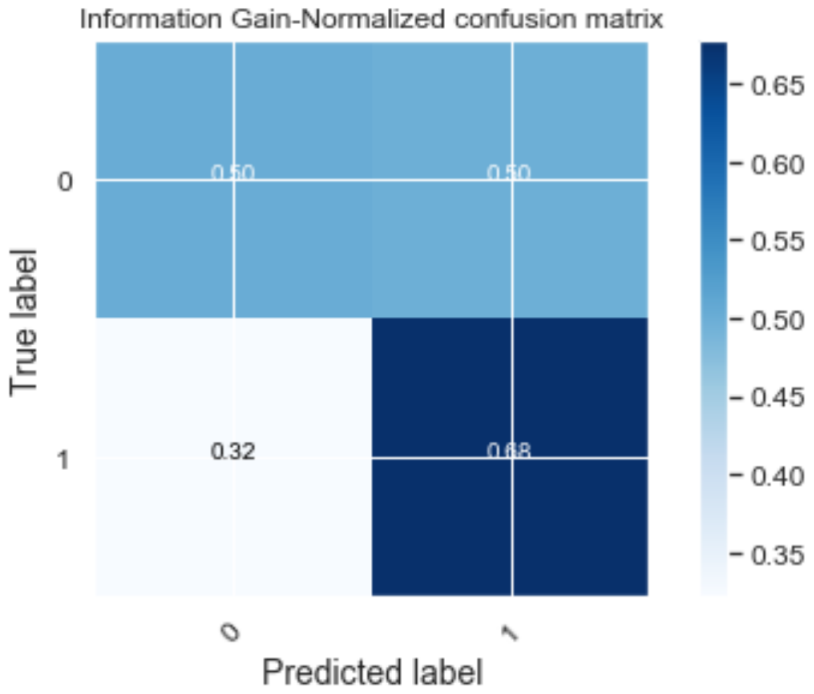
```
plt.figure()
plot_confusion_matrix(infoG_matrix, classes='01',
                      title='Information Gain-Confusion matrix, without normalization')

plt.figure()
plot_confusion_matrix(infoG_matrix, classes='01', normalize=True,
                      title='Information Gain-Normalized confusion matrix')

plt.show()
```

CONFUSION MATRIC WIHTOUT NORMALIZATION



TP=2112
TN=1681
FP = 1620
FN = 1006

CONFUSION MATRIX WITH NORMALIZATION:



From this confusion matrix, we can get the metrics which are needed to evaluate the model. The metrics are as follows.

```
infoG_A = accuracy_score(y_test,y_pred_infoG)
infoG_P = precision_score(y_test,y_pred_infoG)
infoG_R = recall_score(y_test,y_pred_infoG)
infoG_F = f1_score(y_test,y_pred_infoG)
```

ACCURACY = 53%
PRECISION = 52%
RECALL = 51%
F1MEASURE = 51.8%

As discussed earlier, if we were to evaluate this model, then f1 is a great metric as false positive and false negatives are unevenly distributed

COMPARING TWO MODELS:

Both of the models have fairly similar values for each metric. I would prefer model 2 with information gain criteria because it has slightly better f1 value and accuracy as well.

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.

Generally Entropy of Parent's node is greater than child's node. But other way is also possible.

Hence,

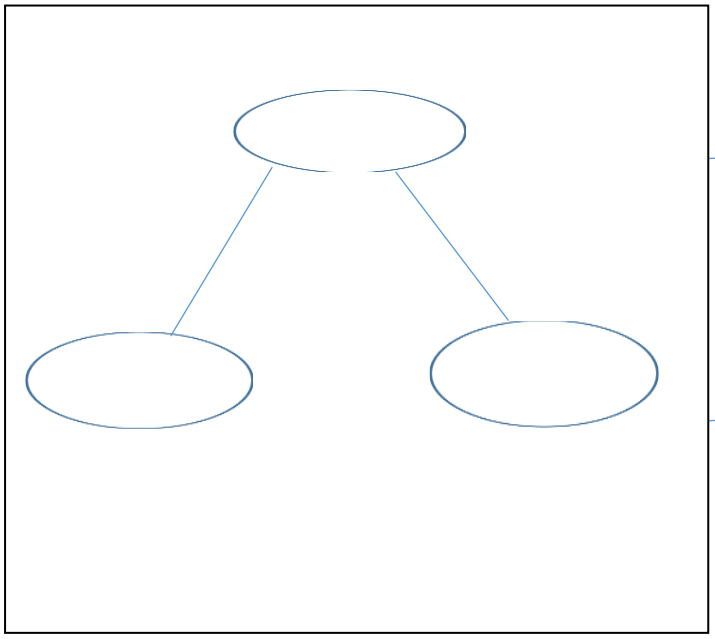
Nodes's entropy can be greater than parent's entropy

Nodes's entropy can also be lower than parent's entropy.

I would like to explain this with two possible scenarios.

CASE 1:

When entropy of child is lower than parent's entropy.



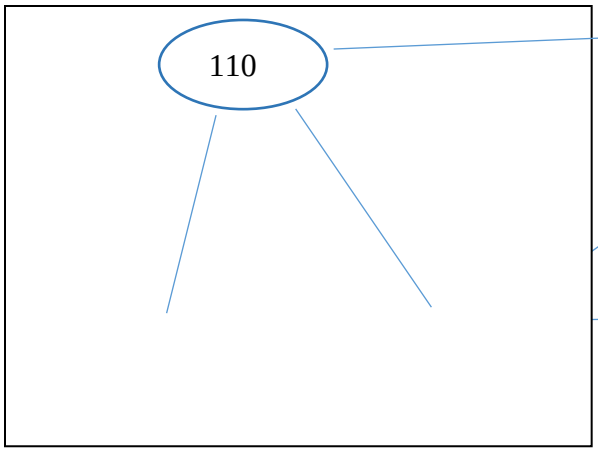
This split is actually possible, because information gain is >0.

Entropy of paren't s node =1

Entropy of both right and left child nodes =0

Information gain = 1. So this split is possible, and we proved **that Entropy of parent's node can be greater than child's node**

CASE2:



Entropy of the parent = -1/3 * log(1/3)- 2/3* log(2/3)

= -(0.33*1.59) –(0.66*0.59)

= 0.91 (impure)

Entropy of right child= 0

Entropy of Left child = 1 (very impure)

Total entropy of the weighted child nodes = 2/3*1 + 1/3*0 = 0.66

Information gain = 0.91 – 0.66 >0

Since information gain is positive, this scenario is feasible as well.

Here**, Entropy of left child is 1 which is greater than the parent's 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.

SUPERVISED LEARNING	UNSUPERVISED LEARNING
• Response Variable is defined and is available in the training dataset • Do Prediction, Inference	• Training data set just contains data with no response variable • Find hidden structure in data

Decision tree algorithm is a supervised method since we can predict a target variable with the help of the rules from the decision tree. The outcome of a decision tree is either classification or regression, hence it is supervised.