ML Article – E20027- Sanketh Udupi PGPDS Bangalore

Demystifying Decision Trees with an example problem on classification

A decision tree has many examples in real life. It has influenced many areas in Machine learning including Classification and Regression. A discussion on decision trees is best understood with an analogy of our daily lives. Think about how we're often put in situations where we make choices based on certain conditions, where one choice leads to a specific result or consequence. A Decision tree can visually represent decisions and decision making.

What Is a Decision Tree?

A decision tree is a map of the possible outcomes of a series of related choices. It allows an individual or organization to weigh possible actions against one another based on their costs, probabilities, and benefits.

As the name goes, it uses a tree-like model of decisions. They can be used either to drive informal discussion or to map out an algorithm that predicts the best choice mathematically.

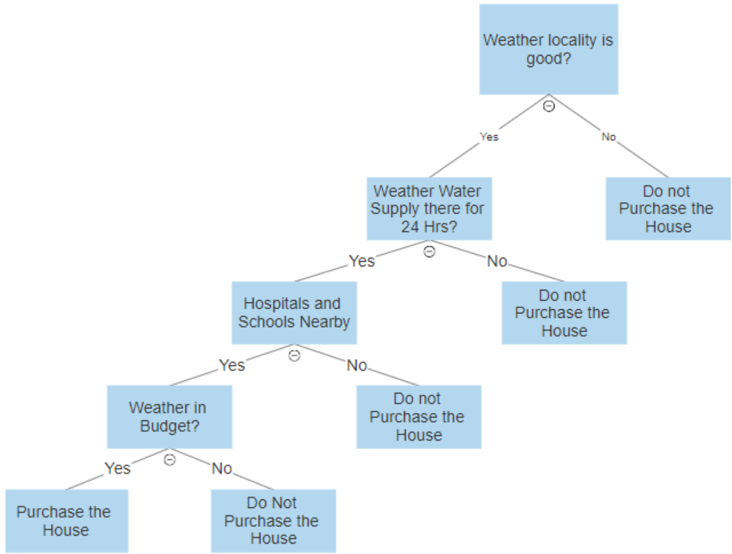
A decision tree typically starts with a single node, which branches into possible outcomes. Each of those outcomes leads to additional nodes, which branch off into other possibilities. This gives it a tree-like shape.

Creating a decision tree

Let us consider a scenario where a person wants to buy a new house. There are *n* number of deciding factors which need to be taken into consideration while purchasing the house.

These factors can be whether the locality is good or not. Weather 24 hrs supply water is there or not. Weather Hospital and schools are there around the locality or not. If it fits the budget or not. If the society is old, how old it is? if the construction is new weather, they have been registered in RERA or not etc.

Let us create a decision tree to find out whether the person should buy the house or not.



This is an Example of the Decision tree we created.

If we consider the above example, logical step-by-step approach is used to arrive at the final stage. What we necessarily do is apply a logical approach to break down a complicated situation or data set. This same approach of logical decision making is applied in decision trees.

Each leaf node is assigned a class variable. A class variable is the final output which leads to our decision.

Important Terminology related to Tree based Algorithms

Root Node: It represents entire population or sample and this further gets divided into two or more homogeneous sets.

Splitting: It is a process of dividing a node into two or more sub-nodes.

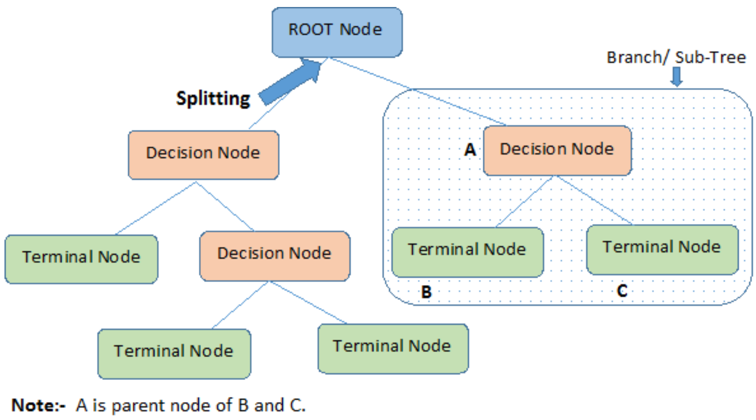
Decision Node: When a sub-node splits into further sub-nodes, then it is called decision node. It is also called as Internal Nodes.

Leaf/ Terminal Node: Nodes do not split is called Leaf or Terminal node.

Pruning: When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.

Branch / Sub-Tree: A sub section of entire tree is called branch or sub-tree.

Parent and Child Node: A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node.



Source- Analytics Vidhya

Measures for Selecting the Best Split

The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria is different for classification and regression trees.

Decision trees use multiple algorithms to decide to split a node in two or more sub-nodes. The creation of sub-nodes increases the purity with respect to the target variable.

Gini

Gini says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.

1. It works with categorical target variable "Success" or "Failure".
2. It performs only Binary splits
3. Higher the value of Gini higher the homogeneity.

Gini Impurity = 1-Gini

Chi-Square Test

It is an algorithm to find out the statistical significance between the differences between sub-nodes and parent node. We measure it by sum of squares of standardized differences between observed and expected frequencies of target variable.

1. It works with categorical target variable "Success" or "Failure".
2. It can perform two or more splits.
3. Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node.
4. Chi-Square of each node is calculated using formula,
5. Chi-square = ((Actual – Expected)^2 / Expected)^1/2

Information Gain:

If the sample is completely homogeneous, then the entropy is zero and if the sample is an equally divided (50% – 50%), it has entropy of one.

Entropy can be calculated using formula:

$$\text{Entropy} = -p\log_2 p - q\log_2 q$$

Entropy is also used with categorical target variable. It chooses the split which has lowest entropy compared to parent node and other splits. The lesser the entropy, the better it is.

We can derive **information gain** from entropy as **1- Entropy.**

Overfitting in Decision trees

Overfitting is one of the key challenges faced while using tree-based algorithms. If there is no limit set of a decision tree, it will give you 100% accuracy on training set because in the worst case it will end up making 1 leaf for each observation. Thus, preventing overfitting is pivotal while modelling a decision tree and it can be done in 2 ways:

1.Setting constraints on tree size

- **Minimum samples for a node split**
- **Minimum samples for a terminal node (leaf)**
- **Maximum depth of tree (vertical depth)**
- **Maximum number of terminal nodes**
- **Maximum features to consider for split**

This is greedy approach to the problem. we are specifying constraints so when it reaches the specific constraints it stops.

2.Pruning the Decision trees

Here we run the decision tree to maximum depth and then prune the tree which is not necessary.

We will be taking a dataset of loan data where we would be running Decision tree classifier and would be predicting the outcomes.

Decision Tree

```
1. Import Libraries
2. Load Data
3. Understand the Data
4. Data Preprocessing
5. Exploratory Data Analysis
6. Model Building
```

This would be our steps for model building. We would be using Scikit learn Library to do the analysis.

1. Import Libraries

```
In [1]:
import numpy as np #importing numpy library
import pandas as pd  #importing pandas library
pd.set_option('display.max_rows', 800)   #Limiting the max rows and columns
pd.set_option('display.max_columns', 500)

import seaborn as sns  #importing the seaborn library
import matplotlib.pyplot as plt
%matplotlib inline

# import all libraries and dependencies for machine learning
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
```

```
C:\Users\Sanketh\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19: FutureWarning: pandas.util.testing is deprecate
d. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Importing the required libraries and moving ahead to load the data.

2. Load Data

```
In [4]:
df = pd.read_csv('C:\\Users\\Sanketh\\Desktop\\Decision tree article and loan prediction\\loan_data.csv')
```

3. Understanding the data

```
In [5]:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   credit.policy      9578 non-null   int64
 1   purpose            9578 non-null   object
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   fico               9578 non-null   int64
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64
 11  delinq.2yrs        9578 non-null   int64
 12  pub.rec            9578 non-null   int64
 13  not.fully.paid     9578 non-null   int64
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

As we can see we have the data and we have loaded the data into the data frame.

In [6]:

```
df.describe()
```

Out[6]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yı
count	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9.578000e+03	9578.000000	9578.000000	9578.00000
mean	0.804970	0.122640	319.089413	10.932117	12.606679	710.846314	4560.767197	1.691396e+04	46.799236	1.577469	0.16370
std	0.396245	0.026847	207.071301	0.614813	6.883970	37.970537	2496.930377	3.375619e+04	29.014417	2.200245	0.54621
min	0.000000	0.060000	15.670000	7.547502	0.000000	612.000000	178.958333	0.000000e+00	0.000000	0.000000	0.00000
25%	1.000000	0.103900	163.770000	10.558414	7.212500	682.000000	2820.000000	3.187000e+03	22.600000	0.000000	0.00000
50%	1.000000	0.122100	268.950000	10.928884	12.665000	707.000000	4139.958333	8.596000e+03	46.300000	1.000000	0.00000
75%	1.000000	0.140700	432.762500	11.291293	17.950000	737.000000	5730.000000	1.824950e+04	70.900000	2.000000	0.00000
max	1.000000	0.216400	940.140000	14.528354	29.960000	827.000000	17639.958330	1.207359e+06	119.000000	33.000000	13.00000

In [7]:

```
df.head()
```

Out[7]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec	not
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0	0	0	
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0	0	0	
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1	0	0	
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1	0	0	
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0	1	0	

In [8]:

```
num_col = df.select_dtypes(include=np.number).columns
print("Numerical columns: \n",num_col)

cat_col = df.select_dtypes(exclude=np.number).columns
print("Categorical columns: \n",cat_col)
```

```
Numerical columns: 
 Index(['credit.policy', 'int.rate', 'installment', 'log.annual.inc', 'dti',
       'fico', 'days.with.cr.line', 'revol.bal', 'revol.util',
       'inq.last.6mths', 'delinq.2yrs', 'pub.rec', 'not.fully.paid'],
      dtype='object')
Categorical columns: 
 Index(['purpose'], dtype='object')
```

We categorise the features into numerical and categorical variable.

4. Data Pre-processing

In [9]:

```
# Let's do one hot encoding for the column `purpose` as model would be expecting numeric features

df = pd.get_dummies(prefix='purpose',data=df,columns=['purpose'])
```

In [10]:

```
num_col = df.select_dtypes(include=np.number).columns
print("Numerical columns: \n",num_col)

cat_col = df.select_dtypes(exclude=np.number).columns
print("Categorical columns: \n",cat_col)
```

```
Numerical columns: 
 Index(['credit.policy', 'int.rate', 'installment', 'log.annual.inc', 'dti',
       'fico', 'days.with.cr.line', 'revol.bal', 'revol.util',
       'inq.last.6mths', 'delinq.2yrs', 'pub.rec', 'not.fully.paid',
       'purpose_all_other', 'purpose_credit_card',
       'purpose_debt_consolidation', 'purpose_educational',
       'purpose_home_improvement', 'purpose_major_purchase',
       'purpose_small_business'],
      dtype='object')
Categorical columns: 
 Index([], dtype='object')
```

We do one hot encoding as model would be expecting numeric features.

In [11]:

```
print(df.isna().sum())
print(df.shape)
```

```
credit.policy                 0
int.rate                      0
installment                   0
log.annual.inc                0
dti                           0
fico                          0
days.with.cr.line             0
revol.bal                     0
revol.util                    0
inq.last.6mths                0
delinq.2yrs                   0
pub.rec                       0
not.fully.paid                0
purpose_all_other             0
purpose_credit_card           0
purpose_debt_consolidation    0
purpose_educational           0
purpose_home_improvement      0
purpose_major_purchase        0
purpose_small_business        0
dtype: int64
(9578, 20)
```

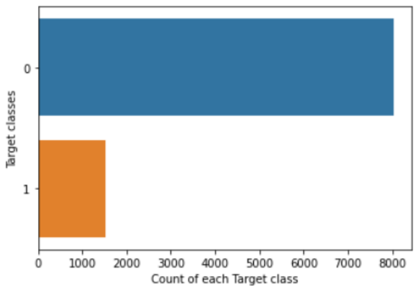
We see if there are any missing values so that we can impute it.

We see there is no missing values according to above, the data is clean so we move ahead.

5. Exploratory Data Analysis ¶

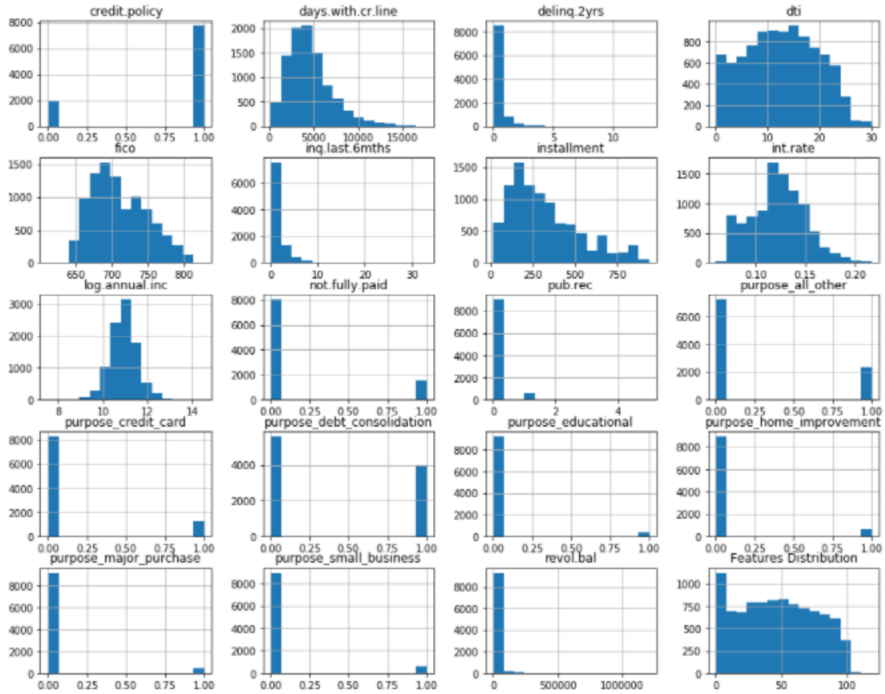
In [12]:

```
# Check the distribution of y variable to see if it's a case of unbalanced class

sns.countplot(y=df['not.fully.paid'] ,data=df)
plt.xlabel("Count of each Target class")
plt.ylabel("Target classes")
plt.show()
```



In [14]:

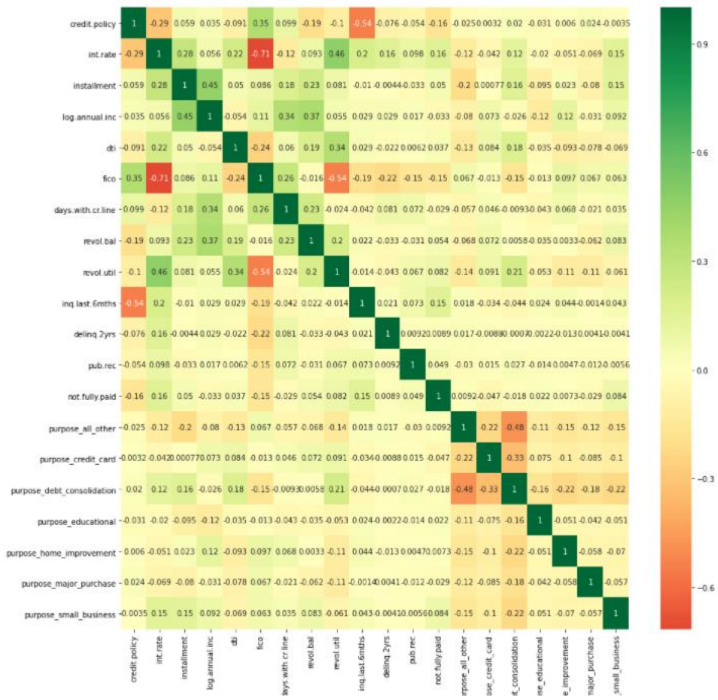
```
# Check the distribution of all the features

df.hist(figsize=(15,12),bins = 15)
plt.title("Features Distribution")
plt.show()
```



We perform exploratory data analysis on the dataset to find out the relation between the features.

In [13]:

```
# Let's check the multicollinearity of features by checking the correlation matric

plt.figure(figsize=(15,15))
p=sns.heatmap(df.corr(), annot=True,cmap='RdYlGn',center=0)
```



We perform Multicollinearity check through seaborn library using heatmap.

6. Model Building

In [14]:
```python
# Train test split
X = df.drop(['not.fully.paid'], axis = 1)
y = df['not.fully.paid']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=500)
```

Decision Tree with criterion = gini

In [15]:
```python
clf = DecisionTreeClassifier(criterion='gini',random_state=0)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Confusion Matrix : \n ",confusion_matrix(y_test, y_pred))
print("\n Accuracy Score : \n ",accuracy_score(y_test,y_pred))
print("\n Classification Report : \n",classification_report(y_test, y_pred))
```

```
Confusion Matrix :
  [[2018  374]
 [ 369  113]]

 Accuracy Score :
  0.7414752957550452

 Classification Report :
               precision    recall  f1-score   support

           0       0.85      0.84      0.84      2392
           1       0.23      0.23      0.23       482

    accuracy                           0.74      2874
   macro avg       0.54      0.54      0.54      2874
weighted avg       0.74      0.74      0.74      2874
```

Decision Tree with criterion = entropy

In [16]:
```python
clf = DecisionTreeClassifier(criterion='entropy',random_state=0)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Confusion Matrix : \n ",confusion_matrix(y_test, y_pred))
print("\n Accuracy Score : \n ",accuracy_score(y_test,y_pred))
print("\n Classification Report : \n",classification_report(y_test, y_pred))
```

```
Confusion Matrix :
  [[2034  358]
 [ 396   86]]

 Accuracy Score :
  0.7376478775226165

 Classification Report :
               precision    recall  f1-score   support

           0       0.84      0.85      0.84      2392
           1       0.19      0.18      0.19       482

    accuracy                           0.74      2874
   macro avg       0.52      0.51      0.51      2874
weighted avg       0.73      0.74      0.73      2874
```

After we do the EDA we fit the model and classify the dataset into decision tree with criterion gini and entropy.

We get the f1 score accuracy as 74%.

We can reduce the imbalance and run the decision tree classifier again and do some feature engineering and hyper parameter tuning to get more accuracy.

References for the Article

Analytics Vidhya Article- https://www.analyticsvidhya.com/blog/2016/04/tree-based-algorithms-complete-tutorial-scratch-in-python/

https://www.upgrad.com/blog/how-to-create-decision-tree-algorithm-examples/

https://www-users.cs.umn.edu/~kumar001/dmbook/ch4.pdf

http://ucanalytics.com/blogs/decision-tree-cart-retail-case-example-part-5/

https://dzone.com/articles/how-to-create-a-perfect-decision-tree

https://www.edureka.co/blog/decision-trees/