16- Tree Based Method - Decision Tree Learning

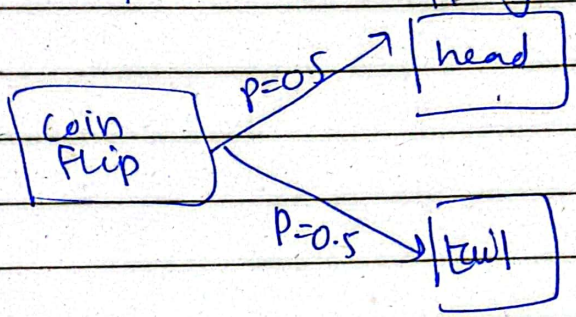
MTWTFSS DATE:

Three Main methods

- Decision Tree
- Random forest
- Boosted Trees

2- Decision Tree History

The general term "decision tree" can refer to a flowchart mapping out outcomes:



check the slides for more information.

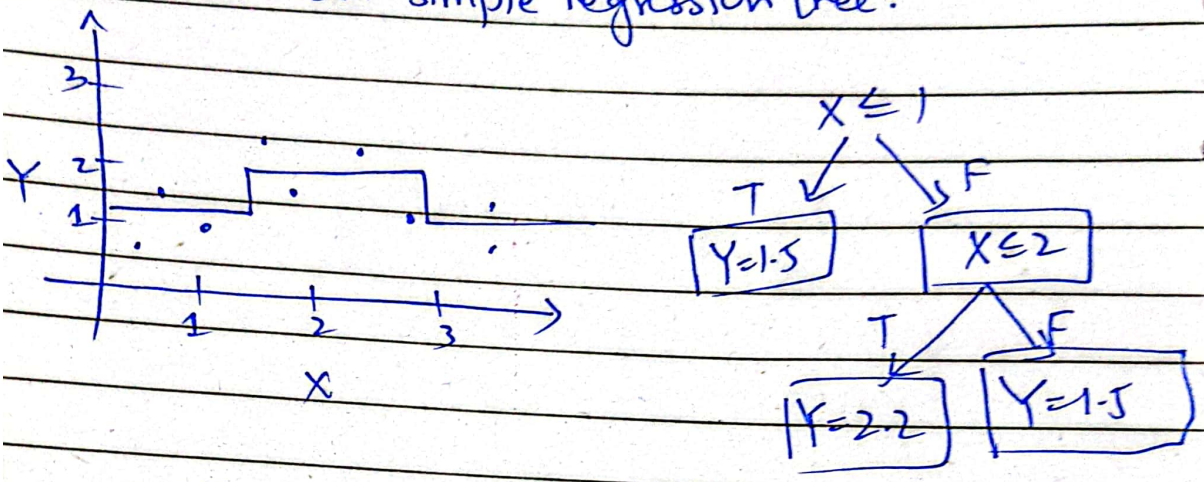
003 - Decision Tree - Terminology

Decision Tree Basics

M T W T F S DATE: _____

- To begin understanding a decision tree, we first need to review some terminology about the ~~dataset~~ decision tree components.

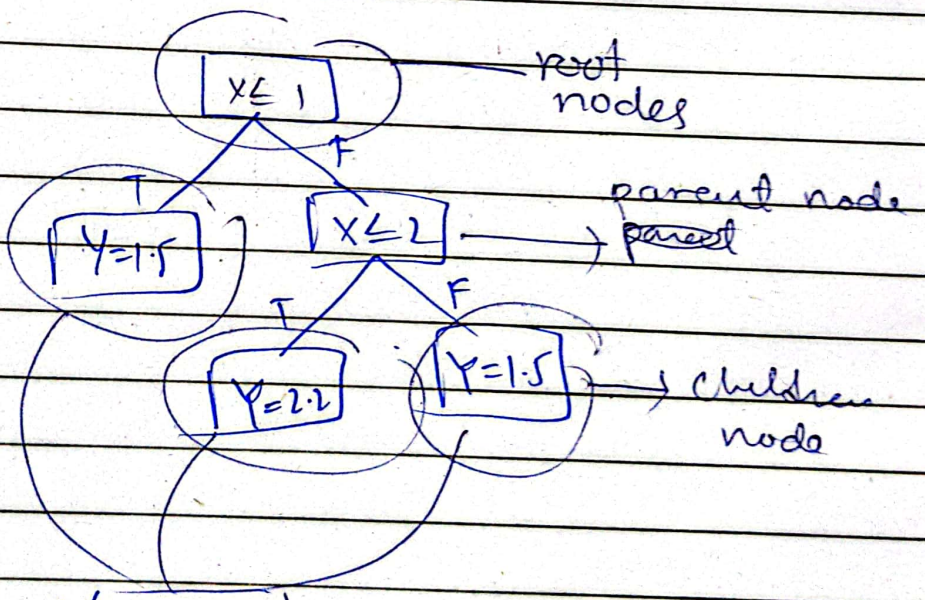
- Recall our simple regression tree:



- Splitting

↓

~~nodes~~ nodes



~~stops~~ splitting

pruning :- ~~stop~~ ~~stop~~ the node ~~data~~ from further splitting

4 - Decision Tree - Understanding Gini impurity

M T W T F S DATE:

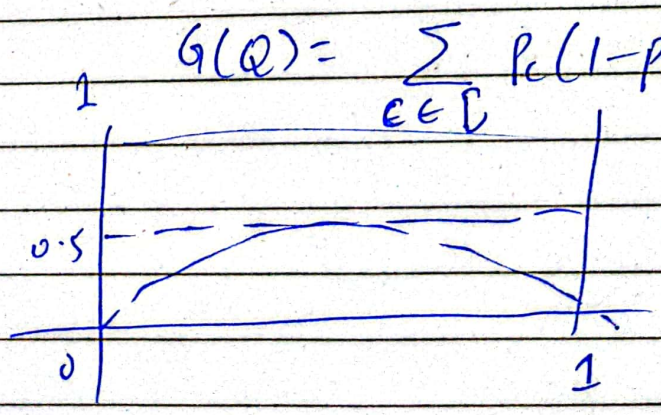
- Before we explore how splitting criterion is used in constructing decision trees, lets explore the common information measurement for decision for decision trees, gini impurity.

- Gini impurity is a mathematical measurement of how pure" the information in a data set is.
  - In regards to classification, we can think of this as a measurement of class uniformity.
  - Let see how this relates to the simplest case of two classes-

- Gini impurity for classification:
  - For a set of classes C for a given dataset Q, p_c is probability of class C.

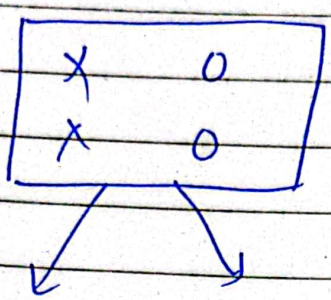
$$p_c = \frac{1}{N_Q} \sum_{x \in Q} 1(y_{class} = c)$$

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$



M T W T F S DATE:

- $G(Q) = \sum_{c \in C} P_c(1-P_c)$



class X
$(2/4)(1-2/4)$
$= 0.25$

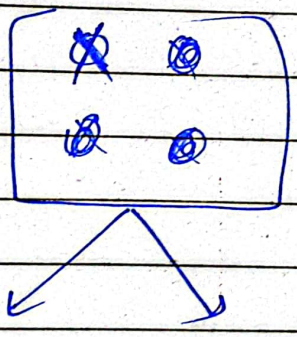
class O
$(2/4)(1-2/4)$
$= 0.25$

- "Maximum" Impurity possible

$G(Q) = \sum_{c \in C} P_c(1-P_c)$

gini impurity $= 0.25 + 0.25$
$= 0.5$

- Data is more "pure" (less impurity)



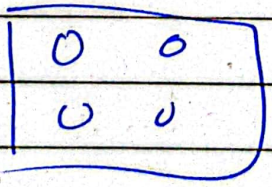
class X
$(1/4)(1-1/4)$
$= 0.1875$

$(3/4)(1-3/4) = 0.1875$

Gini impurity
$0.1875 + 0.1875$
$= 0.375$

Dataset is completly "pure" (no impurity)

$G(Q) = \sum_{c \in C} P_c(1-P_c)$



class X
$(0/4)(1-0) = 0$
+
class O
$(4/4)(1-4/4) = 0$
$0 + 0 = 0$

Gini impurity

M T W T F S DATE:

- If the goal of a decision tree is to seperate out classes, we can use gini impurity to decide on data split value-
- We want to minimize the gini impurity at leaf nodes-
- Mimimized impurity at leaf nodes mean we are seperating classes effectively!

5. Constructing Decision Tree with Gini Impurity

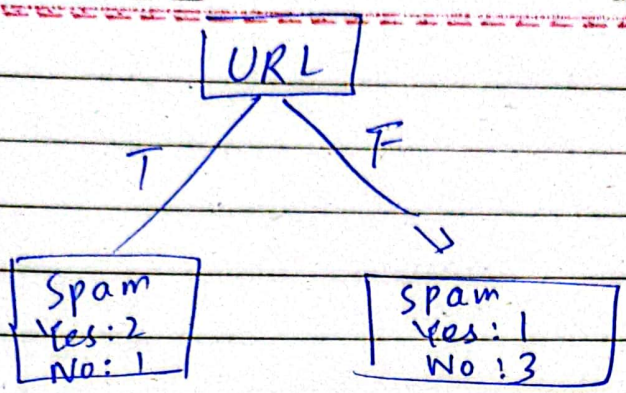
- When first constructing a tree, we need to decide what features will be used as the the root node-
- we can use gini impurity to compare the information contained within features for the training data-

$$p_c = \frac{1}{N_Q} \sum_{x \in Q} I(y_{class} = c) \qquad G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

- Let take a look at this data set:

X-URL link	Y-Spam
Yess	Yes
Yes	Yes
No	No
No	No
No	Yes
Yes	NO
	No

M T W T F S DATE:



Treat Yes Spam and No Spam as C classes

- Left leaf Node
- $(2/3)(1-2/3)+(1/3)(1-1/3)$
- Left Gini = 0.44

- Right leaf Node
- $(1/4)(1-1/4)+(3/4)(1-3/4)$

Right leaf node = 0.375

- Weight average of both:
  - left leaf Gini = 0.44
  - Right leaf Gini = 0.375

Total Email = (2+1) + (1+3) = 7

- left email = 3
- Right Email = 4
- $(3/7)*0.44+(4/7)*0.375 = 0.403$

M T W T F S DATE:

But if we have multiple features.

We still have issue to consider:

- Multiple features
- Continuous features
- Multi-class categorical features

we can incorporate the gini impurity to each of these to solve for best ~~possible~~ root nodes and best split parameters for leaves.

part II

- We explored how to ~~or~~ create gini impurity for a binary categorical feature (only cusisting of two category)

Now let us explore the following:

- continuous numeric features
- Multi-categorical features (N>2)
- choosing a root node feature

X- words in Email	Y Spam
~~15~~ 10	Yes
25 40	No
35 20	Yes
50	No
45 30	No
	N

M T W T F S DATE:

Words ≤ N

Words ≤ 15

Spam	spam
Yes = 1	Yes : 1
No = 0	No : 3

$G(Q) = (4/5)(0+0) + (4/5)((1/4)(1-1/4) + (3/4)(1-3/4))$

$= 0.3$

15	0.3	
25	0	Gini Impurity
35	0.26	
45	0.4	

Words ≤ 25

Spam	Spam
Yes = 2	Yes : 0
No 0	No : 3

$G(Q) = 0$

M I B T F S DATE:

- Lets explore gini impurity for a features that is multicategory.

X-sender	Y-spam
Abe	Yes
Bob	Yes
Clair	No
abe	No
bob	No

Senders=Abe

Spam
Yes 1
No 1

Spam
Yes 1
No: 2

Sender == Abe or bob

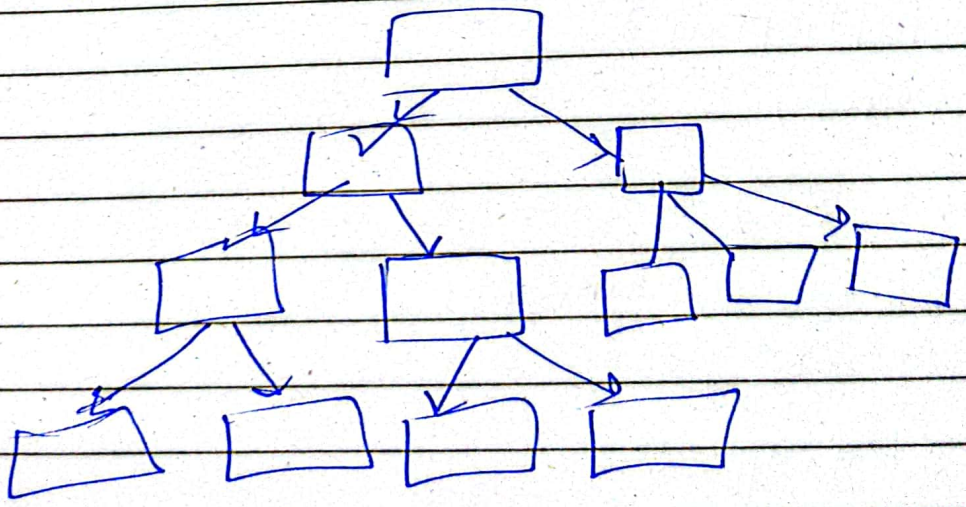
Sender == bob

Sender == Clair or bob

Sender == Clair

Sender == abe or cl

A Large overfitted tree:



M T W T F S DATE:

- Add minimum gini impurity decrease.
- We can also mandate a max depth!
  prune after after certain split

007 Coding Decision Tree — part One the data

```
df = pd.read_csv("penguinsize")
df.head()

df['species'].unique()

df.isnull().sum()
df.info()

df=df.dropna()
df.info()

df.head()
df["Island"].unique()
df["sex"].unique()
df[df["sex"]=='.'] describe()
df[df["species"]=='Gentoo'].groupby('sex')
```

M T W T F S

DATE:

```
df.at[336, 'sex'] = 'FEMALE'

sns.pairplot(df, hue = "sex")

sns.catplot(x='species', y='culmen_length',
            data = df, kind='box',
            col = 'sex')

X = pd.get_dummies(df.drop('species', axis=1),
                   drop_first=True)
y = df['species']

X_train, X_test, y_train, y_test = train_test_split
                                   (X, y, test_size=
```

part two

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
base_preds = model.fit(X_train, X_test)
base_pred = model.pred(X_test)
y. classification_report(y_test, ypreds)
confusion_matrix(y_test, ypred)
model.feature_importance
```

M T W T F S DATE:

X.columns

Index

pd.DataFrame(index = X.columns, data = model.f—
Columns: ['feature Imp']).sort_values
('feature
Imp')

from sklearn.tree import plot tree
plt.fisue(figsize =(12,8), dpi=200)
plot-tree (model);

feature_name = X.columns,
filled = True —

def report_model(model):
model_preds = model.predict(X_test)
print(classification_repn(y_test, mode
print('\n')
plt.figure(figsize=(12,8), dpi=200)
plot_tree(model, features=X.colum,
filled=True);

report_model(model)

pruned_tree = DecisionTreeClass(
max_depth = 2)
pruned_tre.fit(X_train, y_train)
Class report_model(model)

M U R T F S DATE:_____

max_leaf_tree = DecisionTreeClassifier(max_leaf_nodes = 3)
max_leaf_tree.fit( )

entropy_tree = DecisionTreeClassifier(criterion = 'entropy')