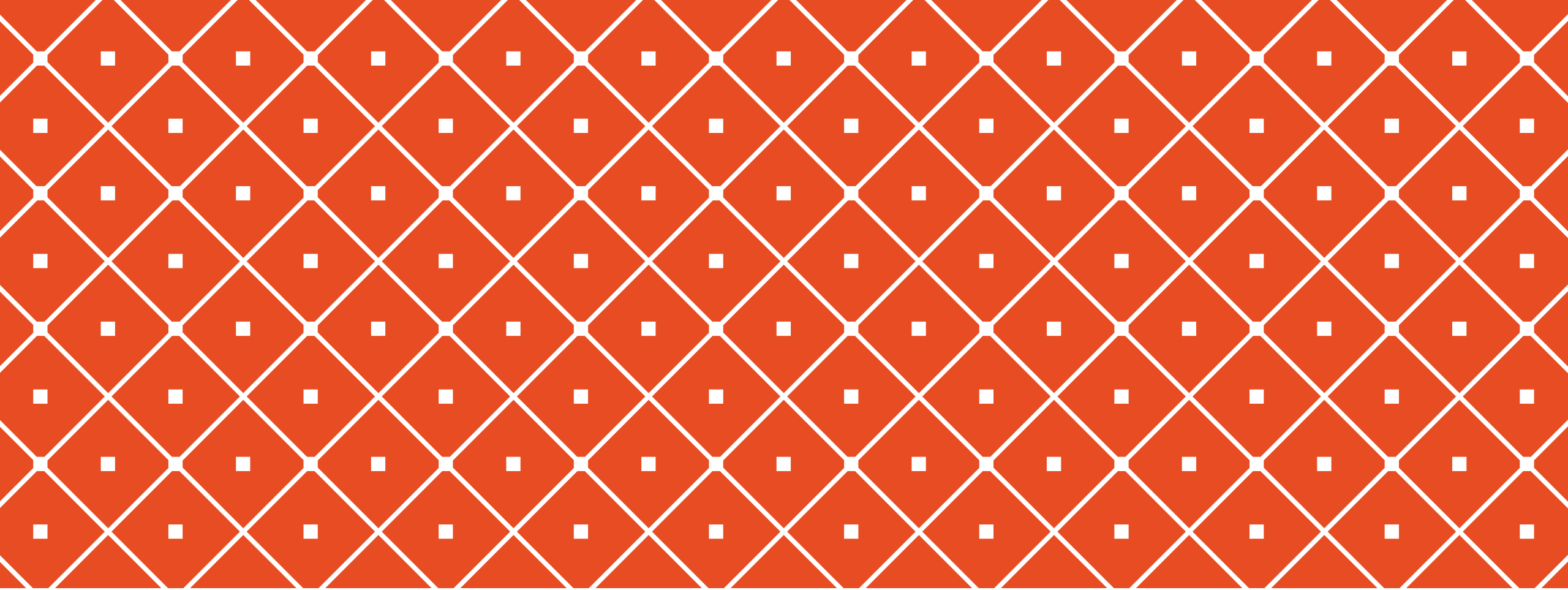
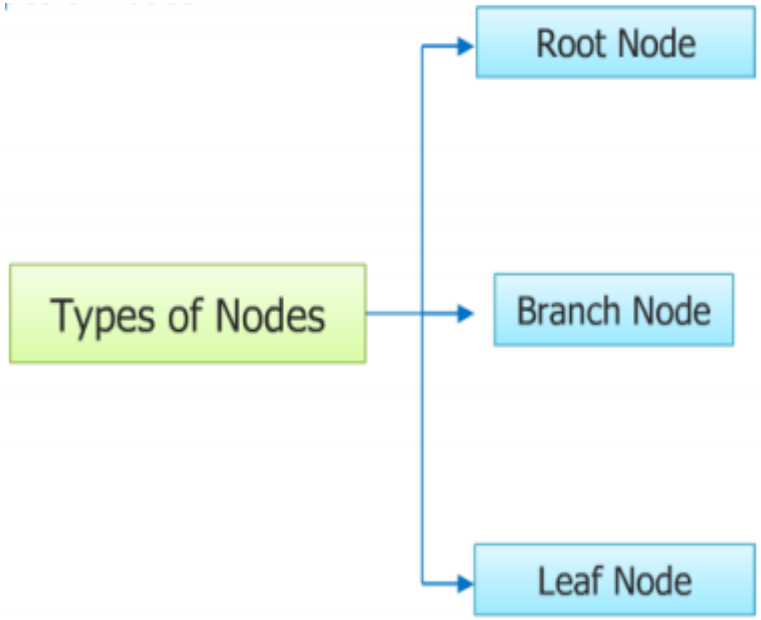


DECISION TREES

Anshu Pandey

Decision Trees

A decision tree is a tree like graphical representation of possible solutions to a decision based on certain conditions.



Decision Trees

A decision tree has three main components :
Root Node : The top most node is called Root Node. It implies the best predictor (independent variable).

Decision / Internal Node : The nodes in which predictors (independent variables) are tested and each branch represents an outcome of the test

Leaf / Terminal Node : It holds a class label (category) - Yes or No (Final Classification Outcome).

How Decision Tree works:

1. Pick the variable that gives the best split (based on lowest Gini Index)
2. Partition the data based on the value of this variable
3. Repeat step 1 and step 2. Splitting stops when CART detects no further gain can be made, or some pre-set stopping rules are met. (Alternatively, the data are split as much as possible and then the tree is later pruned.

Entropy / Information Gain

Entropy: Amount of Uncertainty in decision making

Smaller value of Entropy signifies a good classification.

Information Gain : = *Entropy(parent) - Weighted Sum of Entropy(Children)*

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

Steps to calculate entropy for a split:

1. Calculate entropy of parent node
2. Calculate entropy of each individual node of split and calculate weighted average of all sub-nodes available in split.

Gini Index

Gini Index measures impurity in node. It varies between 0 and (1-1/n) where n is the number of categories in a dependent variable.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

GINI INDEX

Important Points :

1. Zero.gini index implies perfect classification.
2. (1 - (1/ No. of classes) implies worst classification
3. We want a variable split having a low Gini Index.

Chi-Square

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.

Chi-square = ((Actual – Expected)^2 / Expected)^1/2

It generates tree called CHAID (Chi-square Automatic Interaction Detector)

Steps to Calculate Chi-square for a split:

1. Calculate Chi-square for individual node by calculating the deviation for Success and Failure both
2. Calculated Chi-square of Split using Sum of all Chi-square of success and Failure of each node of the split

Regression Tree

The impurity of a node is measured by the Least-Squared Deviation (LSD), which is simply the within variance for the node.

$$\text{Variance} = \frac{\Sigma(X - \overline{X})^2}{n}$$

Above X-bar is mean of the values, X is actual and n is number of values.

Steps to calculate Variance:

1. Calculate variance for each node.
2. Calculate variance for each split as weighted average of each node variance.

Which is better - Entropy or Gini

Both splitting criterias are approximately similar and produces similar result in 95% of the cases. Gini is comparatively faster than Entropy as it does not require calculation of log.

How Decision Tree works?

Weather information of last 14 days –

Whether match was played on those days or not.

Now using decision tree we have to predict that for given conditions –

Outlook – Rainy

Humidity – High

Wind – Weak

Match - ?

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

Decision Trees

Set a metric that evaluates imputicity of a split of data. then minimize the metric on each node.

Classification

Gini impurity（CART）	$\sum_{k=1}^{K} p_k(1-p_k)$
Entropy（C4.5）	$-\sum_{k=1}^{K} p_k \ln p_k$

p_k: probability of an item with label k

K : number of class

Regression

Variance	$\frac{\lvert S_L \rvert}{\lvert S \rvert} Var(S_L) + \frac{\lvert S_R \rvert}{\lvert S \rvert} Var(S_R)$

$SD(S)$: standard varience of set S

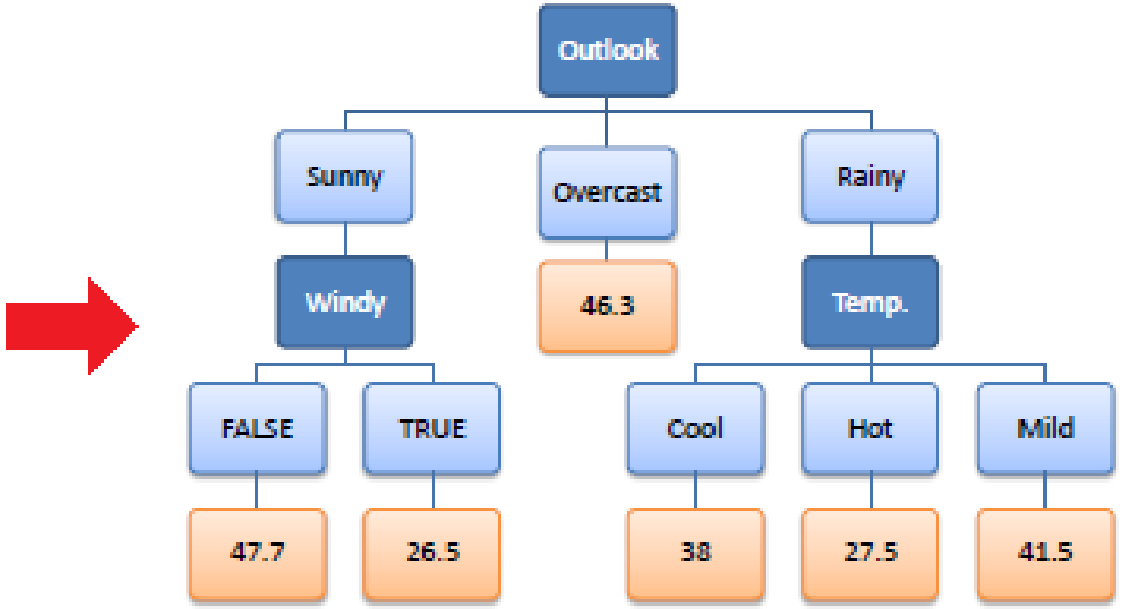
S_L, S_R : left and right split of a node

Decision Tree for regression

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.

Decision Tree for regression

Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	45
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30



Pruning : Correct Overfitting

It is a technique to correct overfitting problem. It reduces the size of decision trees by removing sections of the tree that provide little power to classify instances. It is used to remove anomalies in the training data due to noise or outliers.

Business Scenario and Advantage

→Among patients profile, who will respond better with such treatment

»So by putting rest of them into another kind of treatment

→Among customers, Find profile of those who will attrite vs. those will stay with the business

»So by targeting such customer you can reduce attrition?

→Among applicants, Find which are the applicants, who can be fraud (such as cases of account take over)

»So by working on few selected applications you can avoid lots of account take over fraud cases

Business Scenario and Advantage

→Among prospect of home loan pool, Find who are the prospects customer, who will switch over their home loan

»So by not working on few prospect, bank can quickly grow their portfolio by taking over existing home loans

→Find who among current base will move into delinquency

»So that their credit limit can be reduced to reduce exposure and losses

Stopping Criteria

- Maximum depth
- Minimum leaf nodes

Finding a good threshold for numerical data

- observed point of data
- the point that class labels are changed
- percentile of data

Advantages :

- Decision tree is easy to interpret.
- Decision Tree works even if there is nonlinear relationships between variables. It does not require linearity assumption.
- Decision Tree is not sensitive to outliers.

DISADVANTAGES :

Decision tree model generally overfits. It means it does not perform well on validation sample.

It assumes all independent variables interact each other, It is generally not the case every time.

Import libraries

```
import numpy
import pandas
```

Import data

```
data=pandas.read_csv(r"D:\AI\collabera\iris.data.txt",names=['s
epallength','sepalwidth','petallength','petalwidth','class'])
data.head()
xdata=data.drop(['class'],axis=1)
ydata=data['class']
```

Train test split

```
from sklearn.model_selection import train_test_split


xtr,xts,ytr,yts=train_test_split(xdata,ydata,test_size=0.1)
```

Decision Tree Algorithm

```
#Use decision tree algorithm and train it
from sklearn import tree
alg=tree.DecisionTreeClassifier()
alg.fit(xtr,ytr)
```

Confusion Matrix

```
from sklearn import metrics
yp=alg.predict(xts)
cm=metrics.confusion_matrix(yts,yp)
print(cm)
```

Evaluating Classification Report

```
from sklearn.metrics import classification_report


print(classification_report(yts, yp))
```

Decision Tree Visualization

```
from sklearn.externals.six import StringIO
import pydotplus
dot_data=StringIO()
fn=['sepallength','sepalwidth','petallength','petalwidth']
cn=['Iris-setosa','Iris-versicolor','Iris-virginica’]
from IPython.display import Image
```

Decision Tree Visualization

```
tree.export_graphviz(alg,
                     out_file=dot_data,
                     feature_names=fn,
                     class_names=cn,
                     filled=True,
                     rounded=True,
                     impurity=False)
graph=pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
graph.create_pdf('dtree.pdf')
```

Random Forest

Random forest (or **random forests**) is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.

Working of Random Forest

1. Assume number of cases in the training set is N. Then, sample of these N cases is taken at random but *with replacement*. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<M is specified such that at each node, m variables are selected at random out of the M. The best split on these m is used to split the node. The value of m is held constant while we grow the forest.
3. Each tree is grown to the largest extent possible and there is no pruning.
4. Predict new data by aggregating the predictions of the ntree trees (i.e., majority votes for classification, average for regression).

Features and Advantages of Random Forest

The advantages of random forest are:

1. It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
2. It runs efficiently on large databases.
3. It can handle thousands of input variables without variable deletion.
4. It gives estimates of what variables are important in the classification.
5. It generates an internal unbiased estimate of the generalization error as the forest building progresses.
6. It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

Import libraries

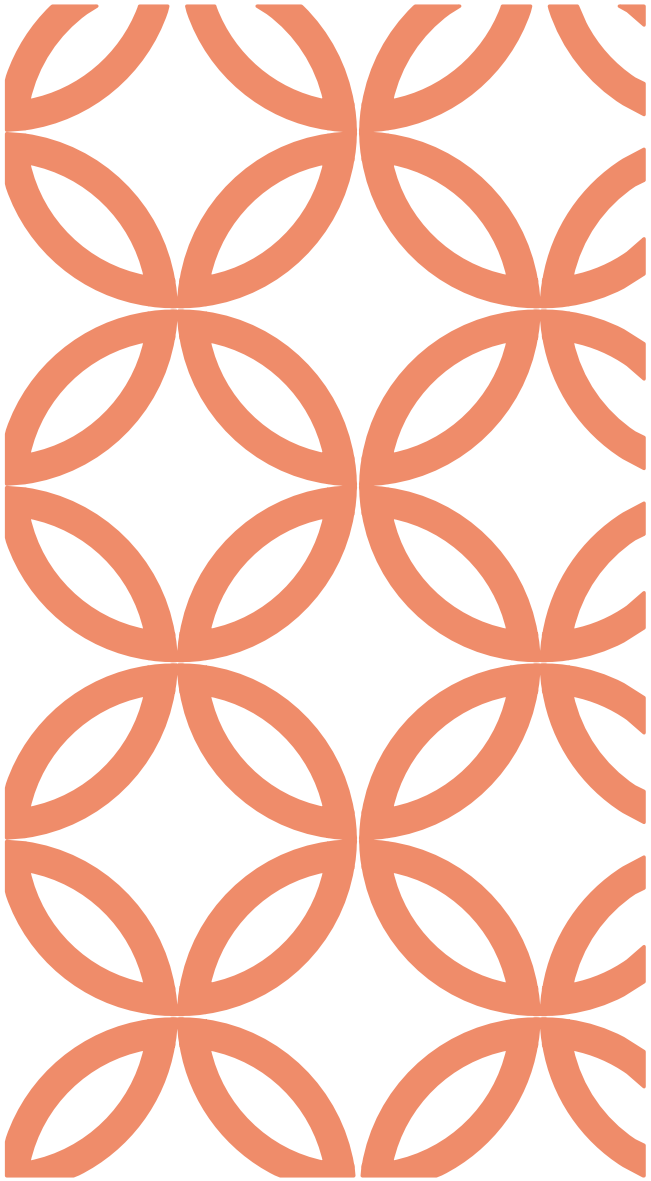
```
#Import Library
from sklearn.ensemble import RandomForestClassifier
#use RandomForestRegressor for regression problem
#Assumed you have, X (predictor) and Y (target) for training
data set and x_test(predictor) of test_dataset
```

Applying Random Forest

```
# Create Random Forest object
model= RandomForestClassifier(n_estimators=1000)
# Train the model using the training sets and check score
model.fit(X, y)
```

Predict output

```
#Predict Output
predicted= model.predict(x_test)
```



Stay tuned for programming on Decision Trees and Random Forest

THANK YOU