# CaRT Decision Trees

*Derek Andersen and Joanne Chau*

## `chefboost` Package

We will be utilizing the `chefboost` package to build our CaRT trees. `chefboost` is a simple to use Python package for building decision tree models that supports ID3, C4.5, CART, CHAID and regression tree algorithms.

In [ ]:

```
# Install chefboost package
!pip install chefboost
```

```
Collecting chefboost
  Downloading https://files.pythonhosted.org/packages/91/68/a
ab59c29bf619090ff4f6868070a19639e29b1bc882d35d104f7ee5d19f6/c
hefboost-0.0.6-py3-none-any.whl
Requirement already satisfied: numpy>=1.14.0 in /usr/local/li
b/python3.6/dist-packages (from chefboost) (1.18.5)
Requirement already satisfied: tqdm>=4.30.0 in /usr/local/li
b/python3.6/dist-packages (from chefboost) (4.41.1)
Requirement already satisfied: pandas>=0.22.0 in /usr/local/l
ib/python3.6/dist-packages (from chefboost) (1.1.4)
Requirement already satisfied: pytz>=2017.2 in /usr/local/li
b/python3.6/dist-packages (from pandas>=0.22.0->chefboost) (2
018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /us
r/local/lib/python3.6/dist-packages (from pandas>=0.22.0->che
fboost) (2.8.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/pyt
hon3.6/dist-packages (from python-dateutil>=2.7.3->pandas>=0.
22.0->chefboost) (1.15.0)
Installing collected packages: chefboost
Successfully installed chefboost-0.0.6
```

We will be using `chefboost` and `pandas` since CaRT models focus on implementation of data sets.

# Classification Tree

In [ ]:

```
from chefboost import Chefboost as chef
import pandas as pd
```

Load and view the dataset.

In [ ]:

```
# Load the dataset
classification = pd.read_csv("drive/My Drive/chefboost-master/tests/dataset/golf.txt")

# View the dataset
classification
```

Out[ ]:

| | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| **0** | Sunny | Hot | High | Weak | No |
| **1** | Sunny | Hot | High | Strong | No |
| **2** | Overcast | Hot | High | Weak | Yes |
| **3** | Rain | Mild | High | Weak | Yes |
| **4** | Rain | Cool | Normal | Weak | Yes |
| **5** | Rain | Cool | Normal | Strong | No |
| **6** | Overcast | Cool | Normal | Strong | Yes |
| **7** | Sunny | Mild | High | Weak | No |
| **8** | Sunny | Cool | Normal | Weak | Yes |
| **9** | Rain | Mild | Normal | Weak | Yes |
| **10** | Sunny | Mild | Normal | Strong | Yes |
| **11** | Overcast | Mild | High | Strong | Yes |
| **12** | Overcast | Hot | Normal | Weak | Yes |
| **13** | Rain | Mild | High | Strong | No |

## Training the model

The syntax for data configuration is:

`{"algorithm" : "[Type of Algorithm to Apply]"}`

There are more than just `CART` and `Regression` algorithms allowed for modeling training with the Chefboost package. I encourage you to explore all possible algorithms in their GitHub repository.

In [ ]:

```
config = {'algorithm': 'CART'} # Using the CART algorithm
class_model = chef.fit(classification.copy(), config)
```

```
CART  tree is going to be built...
Accuracy:  100.0 % on  14  instances
finished in  0.08057379722595215  seconds
```

## Testing the model

Now we can create some dummy data to evaluate with our trained model.

In [ ]:

```
# Instances are of the form [Outlook, Temp, Humidity, Wind]
test_set = [
            ['Sunny', 'Cool', 'High', 'Strong'],
            ['Sunny', 'Cool', 'High', 'Weak'],
            ['Overcast', 'Cool', 'High', 'Weak'],
            ['Overcast', 'Mild', 'Normal', 'Weak'],
            ['Rain', 'Hot', 'Normal', 'Strong'],
            ['Rain', 'Hot', 'High', 'Weak']
]

# Evaluate
print("Predictions:")
for instance in test_set:
  print(instance, "decision:", chef.predict(class_model, instance))
```

```
Predictions:
['Sunny', 'Cool', 'High', 'Strong'] decision: No
['Sunny', 'Cool', 'High', 'Weak'] decision: No
['Overcast', 'Cool', 'High', 'Weak'] decision: Yes
['Overcast', 'Mild', 'Normal', 'Weak'] decision: Yes
['Rain', 'Hot', 'Normal', 'Strong'] decision: No
['Rain', 'Hot', 'High', 'Weak'] decision: Yes
```

# Regression Tree

In [ ]:

```
# Load and preview the dataset we are working with to understand features
 being used

regression = pd.read_csv('drive/My Drive/chefboost-master/tests/dataset/go
lf4.txt')
regression
```

Out[ ]:

| | Outlook | Temp. | Humidity | Wind | Decision |
|---|---|---|---|---|---|
| **0** | Sunny | 85 | 85 | Weak | 25 |
| **1** | Sunny | 80 | 90 | Strong | 30 |
| **2** | Overcast | 83 | 78 | Weak | 46 |
| **3** | Rain | 70 | 96 | Weak | 45 |
| **4** | Rain | 68 | 80 | Weak | 52 |
| **5** | Rain | 65 | 70 | Strong | 23 |
| **6** | Overcast | 64 | 65 | Strong | 43 |
| **7** | Sunny | 72 | 95 | Weak | 35 |
| **8** | Sunny | 69 | 70 | Weak | 38 |
| **9** | Rain | 75 | 80 | Weak | 46 |
| **10** | Sunny | 75 | 70 | Strong | 48 |
| **11** | Overcast | 72 | 90 | Strong | 52 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| **11** | Overcast | 72 | 90 | Strong | 52 |
| **12** | Overcast | 81 | 75 | Weak | 44 |
| **13** | Rain | 71 | 80 | Strong | 30 |

## Training the model

This is where Chefboost can be confusing. Though it is simple to configure the data to understand what is needed, we demonstrated earlier that Classification trees use the `CART` algorithm. When using `chefboost`, remember that `CART` is for Classification trees and `Regression` is for Regression trees.

In [ ]:

```
# Train and configure the model using the Regression algorithm

config = {"algorithm" : "Regression"}
reg_model = chef.fit(regression.copy(), config)
```

```
Regression  tree is going to be built...
MAE:  3.4404761904761902
RMSE:  4.429339411136566
Mean:  39.785714285714285
MAE / Mean:  8.647516457211252 %
RMSE / Mean:  11.132989543251693 %
finished in  0.08156037330627441  seconds
```

Unlike the Classification Tree seen earlier, the Regression Tree shows how accurate they were and the margin of error between the instances. With this small training set, there is about a 8% error.

In [ ]:

```
# Compare our training to predictions
# For a better understanding of the error
# Utilizing line 4 from the dataset

test_instance = [ "Rain", 70, 96, "Weak"]
prediction = round(chef.predict(reg_model, test_instance), 1)
actual = regression.iloc[4]["Decision"]
print("Prediction: ", prediction, "| Actual : ", actual, "| Error: ", roun
d((actual - prediction), 1))
```

```
Prediction:  47.7 | Actual :  52 | Error:  4.3
```

In [ ]:

```
# Compare all predictions to accurate numbers
# Take the absolute value for error

for index, instance in regression.iterrows():
  prediction = round(chef.predict(reg_model, instance), 1)
  actual = instance["Decision"]

  print("Prediction: ", prediction, "| Actual : ", actual, "| Error: ", ro
und((abs(actual - prediction)),1))
```

```
Prediction:  25 | Actual :  25 | Error:  0
Prediction:  37.8 | Actual :  30 | Error:  7.8
```

```
Prediction:  37.8 | Actual :  30 | Error:  7.8
Prediction:  46.2 | Actual :  46 | Error:  0.2
Prediction:  47.7 | Actual :  45 | Error:  2.7
Prediction:  47.7 | Actual :  52 | Error:  4.3
Prediction:  26.5 | Actual :  23 | Error:  3.5
Prediction:  46.2 | Actual :  43 | Error:  3.2
Prediction:  37.8 | Actual :  35 | Error:  2.8
Prediction:  37.8 | Actual :  38 | Error:  0.2
Prediction:  47.7 | Actual :  46 | Error:  1.7
Prediction:  37.8 | Actual :  48 | Error:  10.2
Prediction:  46.2 | Actual :  52 | Error:  5.8


Prediction:  46.2 | Actual :  44 | Error:  2.2
Prediction:  26.5 | Actual :  30 | Error:  3.5
```

## Testing the model

Now we can create some dummy test set and feed it to our model for predictions. These test set instances are the similar to the ones we used for the Classification Tree example earlier.

In [ ]:

```
# Instances are of the form [Outlook, Temp, Humidity, Wind]
test_set = [
            ['Sunny', 50, 75, 'Strong'],
            ['Sunny', 54, 85, 'Weak'],
            ['Overcast', 43, 87, 'Weak'],
            ['Overcast', 65, 60, 'Weak'],
            ['Rain', 90, 55, 'Strong'],
            ['Rain', 98, 90, 'Weak']
]

# Evaluate
for instance in test_set:
  print(instance , "Prediction: ", round(chef.predict(reg_model, instance
)))
```

```
['Sunny', 50, 75, 'Strong'] Prediction:  38
['Sunny', 54, 85, 'Weak'] Prediction:  38
['Overcast', 43, 87, 'Weak'] Prediction:  46
['Overcast', 65, 60, 'Weak'] Prediction:  46
['Rain', 90, 55, 'Strong'] Prediction:  26
['Rain', 98, 90, 'Weak'] Prediction:  48
```