Homework 1 - Machine Learning

Decision trees

1. Mathematics behind the decision tree with wild cards

Let a group of laws: $R = \{r_1, r_2, \dots, r_k\}$. Every law r_i has m_i^{ϕ} don't care bits (ϕ).

The number of times each law appears depends on the number of don't care bits it has: $\boldsymbol{N_{r_i} = 2^{m_i^{\phi}}}$

In total the number of laws (after duplicates) is: $\boldsymbol{N_{tot} = \sum_{i=1}^{k} N_{r_i}}$

Our goal is to compute the Information Gain (IG) - $\boldsymbol{IG(R, b_j) = H(R) - H(R \mid b_j)}$

$$\begin{aligned}
H(R) &= -\sum_{i=1}^{k} \frac{N_{r_i}}{N_{tot}} \cdot log_2\left(\frac{N_{r_i}}{N_{tot}}\right) \\
&= \frac{1}{N_{tot}}\left(log_2(N_{tot}) \cdot \sum_{i=1}^{k} N_{r_i} - \sum_{i=1}^{k} N_{r_i}\, log_2(N_{r_i})\right) \\
&= log_2(N_{tot}) - \frac{1}{N_{tot}} \sum_{i=1}^{k} N_{r_i}\, log_2(N_{r_i})
\end{aligned}$$

To compute $\boldsymbol{H(R \mid b_j)}$ we need to evaluate separately the cases for which $b_j = 0$ and $b_j = 1$.

- $b_j = 0$: Contains the laws for which $b_j = 0$ and the ones where $b_j = \phi$ -> In total N_0 laws

In[∘]:=

- Same for $b_j = 1$
- $N_{tot} = N_o + N_1$

$$H(R \mid b_j = 0) = -\sum\nolimits_{b_j=0} \frac{N_{r_i}}{N_0} \cdot log_2\left(\frac{N_{r_i}}{N_0}\right) - \sum\nolimits_{b_j=\phi} \frac{\frac{N_{r_i}}{2}}{N_0} \cdot log_2\left(\frac{\frac{N_{r_i}}{2}}{N_0}\right)$$

$$= \frac{1}{N_0}\left[N_0 \cdot log_2(N_0) - \sum\nolimits_{b_j=0} N_{r_i}\, log_2(N_{r_i}) - \frac{1}{2}\sum\nolimits_{b_j=\phi} N_{r_i}\left(log_2(N_{r_i}) - log_2(N_0) - 1\right)\right]$$

$$= log_2(N_0) - \frac{1}{N_0}\left[\sum\nolimits_{b_j=0} N_{r_i}\, log_2(N_{r_i}) + \sum\nolimits_{b_j=\phi} \frac{N_{r_i}}{2} log_2(N_{r_i}) - \sum\nolimits_{b_j=\phi} \frac{N_{r_i}}{2}\right]$$

$$H(R \mid b_j) = \frac{N_0}{N_{tot}} \cdot H(R \mid b_j = 0) + \frac{N_1}{N_{tot}} \cdot H(R \mid b_j = 1)$$

$$\begin{aligned}
= \frac{1}{N_{tot}}\Bigg[&N_0 \cdot log_2(N_0) - \sum\nolimits_{b_j=0} N_{r_i}\, log_2(N_{r_i}) - \sum\nolimits_{b_j=\phi} \frac{N_{r_i}}{2} log_2(N_{r_i}) + \sum\nolimits_{b_j=\phi} \frac{N_{r_i}}{2} + N_1 \\
&\cdot log_2(N_1) - \sum\nolimits_{b_j=1} N_{r_i}\, log_2(N_{r_i}) - \sum\nolimits_{b_j=\phi} \frac{N_{r_i}}{2} log_2(N_{r_i}) + \sum\nolimits_{b_j=\phi} \frac{N_{r_i}}{2}\Bigg]
\end{aligned}$$

$$\begin{aligned}
&= \frac{1}{N_{tot}}\Bigg[N_0 \cdot log_2(N_0) + N_1 \cdot log_2(N_1) + \sum\nolimits_{b_j=\phi} N_{r_i} - \sum\nolimits_{b_j=1} N_{r_i}\, log_2(N_{r_i}) \\
&\qquad - \sum\nolimits_{b_j=0} N_{r_i}\, log_2(N_{r_i}) - \sum\nolimits_{b_j=\phi} N_{r_i}\, log_2(N_{r_i})\Bigg] \\
&= \frac{1}{N_{tot}}\left[N_0 \cdot log_2(N_0) + N_1 \cdot log_2(N_1) + \sum\nolimits_{b_j=\phi} N_{r_i} - \sum_{i=1}^{k} N_{r_i}\, log_2(N_{r_i})\right]
\end{aligned}$$

Hence, we found:

$$IG(R, b_j) = H(R) - H(R \mid b_j) = \color{red}{log_2(N_{tot}) - \frac{1}{N_{tot}}\left[N_0 \cdot log_2(N_0) + N_1 \cdot log_2(N_1) + \sum\nolimits_{b_j=\phi} N_{r_i}\right]}$$

Where:

$$N_0 = \sum_{b_j=0} N_{r_i} + \frac{1}{2} \cdot \sum_{b_j=\phi} N_{r_i}, \qquad N_1 = \sum_{b_j=1} N_{r_i} + \frac{1}{2} \cdot \sum_{b_j=\phi} N_{r_i}$$

From the obtained result we can go a bit further by replacing the values of $N_{r_i} = 2^{m_i^\phi}$:

$$N_0 = \sum_{b_j=0} 2^{m_i^\phi} + \sum_{b_j=\phi} 2^{m_i^\phi - 1} = N_0^j + N_\phi^j$$

$$N_1 = \sum_{b_j=1} 2^{m_i^\phi} + \sum_{b_j=\phi} 2^{m_i^\phi - 1} = N_1^j + N_\phi^j$$

$$N_{tot} = N_0 + N_1 = N_0^j + N_1^j + 2N_\phi^j$$

$$IG(R, b_j) = log_2(N_{tot}) - \frac{1}{N_{tot}}\left[\left(N_0^j + N_\phi^j\right) log_2\left(N_0^j + N_\phi^j\right) + \left(N_1^j + N_\phi^j\right) log_2\left(N_1^j + N_\phi^j\right) + 2N_\phi^j\right]$$

Too see more, check out **Math.pdf**

Effect of the wild card ϕ

As we can see from the formula we found the more wild cards a column has the smaller the IG will be. Moreover, a wild cards prevents a good classification because when going down a level in the tree the rule will still appears on both sides.

2. Building the Decision Tree (best IG)

For this section, all the functions required to build the decision tree are in the **Functions.wl** module.

```
In[ ]:= << FileNameJoin[{NotebookDirectory[], "Functions.wl"}]
```

```
In[ ]:= path = FileNameJoin[{NotebookDirectory[], "Rule.tsv"}];
rules = LoadRules[path];
```

For section 2.1, we set the stopping point for groups of 64 rules **(threshold = 64).**
For section 2.2, we set the stopping point for an average length of 8 rules per group **(threshold = 8).**

2.1 Decision Tree with **best bit** on each branch

```
In[ ]:= DT1 = BuildTree[rules, threshold = 64, "Branch"]; // RepeatedTiming
Out[ ]= {77.4215, Null}
```

Decision Tree:

Out[ ]=
```
⟨|choice → b3, right → ⋯1⋯,
 left → ⟨|choice → b2, right → ⋯1⋯, left → ⟨|choice → b1, right → ⋯1⋯,
     left → ⟨|choice → b8, right → ⟨|choice → b7, ⋯1⋯, ⋯1⋯|⟩,
       left → ⟨|choice → b7, right → {⋯1⋯}, left → ⟨|choice → b6,
           right → ⟨|choice → b5, right → ⟨|choice → b4, right → ⟨|choice → b33,
                 right → ⋯1⋯, left → ⟨|choice → b34, right → ⟨|choice → b40,
                   ⋯1⋯, left → ⋯1⋯|⟩, left → {⋯1⋯}|⟩|⟩, left →
                  {⋯1⋯}|⟩, left → {⋯1⋯}|⟩, left → {⋯1⋯}|⟩|⟩|⟩|⟩|⟩|⟩
```
large output | show less | show more | show all | set size limit...

2.2 Decision Tree with **best IG** on the level

In[ ]:=
```
res2 = BuildTree[rules, threshold = 8, "Level"]; // RepeatedTiming
DT2 = res2["DT"]; choices2 = res2["choices"];
```

Out[ ]=
```
{40.259, Null}
```

Decision Tree:

Out[ ]=
```
⟨|{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} → {⋯1⋯},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1} → {⋯1⋯},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0} → {⋯1⋯},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1} → {⋯1⋯},
 {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0} → {⋯1⋯},
 {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1} → {⋯1⋯},
 {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0} → {⋯1⋯}, ⋯2034⋯,
 ⋯1⋯, {1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0} → {⋯1⋯},
 {1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1} → {⋯1⋯},
 {1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0} → {⋯1⋯},
 {1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1} → {⋯1⋯},
 {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0} → {⋯1⋯},
 {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1} → {⋯1⋯}|⟩
```
large output | show less | show more | show all | set size limit...

Choices for DT2 : {“b3”,”b2”,”b1”,”b8”,”b7”,”b5”,”b6”,”b4”,”b33”,”b34”,”b40”}

```
c = ToExpression[StringDrop[res["choices"], 1]];
DT2[Part[StripHeaders[rules][[206]], c]]; (*For predict*)
```

3. Building the Decision Tree with the highest conditional entropy criterion

As we have seen before, the formula of the conditional entropy is as followed:

$$H(R \mid b_j) = \frac{1}{N_{tot}}\left[(N_0^j + N_\phi^j)\cdot log_2\left(N_0^j + N_\phi^j\right) + (N_1^j + N_\phi^j)\cdot log_2\left(N_1^j + N_\phi^j\right) + 2N_\phi^j - \sum_{i=1}^{k} m_i^\phi \cdot 2^{m_i^\phi}\right]$$

3.1 Effect of the wild card ϕ

In this case the effect of the wild card is the complete opposite. Indeed the more ϕ the column has the highest the entropy will be. However in contradiction with the information gain, the higher the entropy is, the higher the uncertainty will be (bad). Hence building a tree using the highest conditional entropy criterion is **the less effective way** to build a decision tree.
However, if we choose to neglect the ϕ, some studies has shown surprising results for smaller batches of data:

$$H(R \mid b_j) = \frac{1}{N_0^j + N_1^j}\left[N_0^j \cdot log_2\left(N_0^j\right) + N_1^j \cdot log_2\left(N_1^j\right)\right]$$

Let’s build a tree using the highest entropy using this new definition of entropy:

3.2 Decision Tree with **best Entropy** on each branch

In[◦]:=
```
DT3 = BuildTree[rules, threshold = 64, "Entropy", "Branch"]; // RepeatedTiming
```
Out[◦]=
```
{48.3566, Null}
```

Decision Tree:

Out[◦]=
```
⟨|choice → b8, right → ⟨| ⋯ 1 ⋯ |⟩, left →
  ⟨|choice → b2, right → ⟨| ⋯ 1 ⋯ |⟩, left → ⟨|choice → b5, right → ⟨| ⋯ 1 ⋯ |⟩,
     left → ⟨|choice → b6, right → ⟨|choice → b1, right → ⟨|choice → b7,
            right → ⟨|choice → b4, right → ⋯ 1 ⋯ , left → ⟨| ⋯ 1 ⋯ |⟩ |⟩,
            left → ⟨|choice → b4, right → ⟨|choice → b3, right → ⟨|choice → b34,
                  right → ⟨|choice → b39, right → ⟨| ⋯ 1 ⋯ |⟩, left → { ⋯ 1 ⋯ } |⟩,
                  left → { ⋯ 1 ⋯ } |⟩, left → { ⋯ 1 ⋯ } |⟩, left → { ⋯ 1 ⋯ } |⟩ |⟩,
        left → { ⋯ 1 ⋯ } |⟩, left → { ⋯ 1 ⋯ } |⟩ |⟩ |⟩ |⟩
```

large output | show less | show more | show all | set size limit...

3.3 Decision Tree **best Entropy** on the level

```
In[ ]:= res4 = BuildTree[rules, threshold = 8, "Entropy", "Level"]; // AbsoluteTiming
DT4 = res4["DT"]; choices4 = res4["choices"];
Out[ ]= {19.7242, Null}
```

Decision Tree:

```
Out[ ]= <|{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} → {… 1 …},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1} → {… 1 …},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0} → {… 1 …},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1} → {… 1 …},
 {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0} → {… 1 …},
 {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1} → {… 1 …},
 {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0} → {… 1 …}, … 2034 …,
 … 1 …, {1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0} → {… 1 …},
 {1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1} → {… 1 …},
 {1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0} → {… 1 …},
 {1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1} → {… 1 …},
 {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0} → {… 1 …},
 {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1} → {… 1 …}|>

large output | show less | show more | show all | set size limit...
```

Choices for DT4 : {{"b8", "b2", "b5", "b7", "b1", "b6", "b4", "b3", "b34", "b38", "b39"}, 11}

In comparison for the same build with the best IG criterion, we got the following choices:

Choices for DT2 : {{"b3", " b2", "b1", "b8", "b7", "b5", "b6", "b4", "b33", "b34", "b40"}, 11}

As we can see both methods required 11 choices (to get a DT with an average of 8 rules per leaf) , and although the order differs, **9/11 choices** are the same:

Chosen by both methods (Intersection) : {{"b1", "b2", "b3", "b34", "b4", "b5", "b6", "b7", "b8"}, 9}

Free some memory for the Classify Function...

---

4. Classifying using the random forest's method

```
In[ ]:= file = FileNameJoin[{NotebookDirectory[], "Packets.txt"}];
fd = OpenRead[file];
lines = ToExpression[StringSplit[#, "\t"]] & /@ ReadList[fd, String, 300000];
data = DeleteDuplicates[lines];
Close[fd];
```

In[ ]:=
```
sourceIP = IntegerDigits[data[[All, 1]], 2, 32];
destinationIP = IntegerDigits[data[[All, 2]], 2, 32];
sourcePort = IntegerDigits[data[[All, 3]], 2, 16];
destinationPort = IntegerDigits[data[[All, 4]], 2, 16];
protocol = IntegerDigits[data[[All, 5]], 2, 8];
```

In[ ]:=
```
inputExamples = Flatten /@
   Transpose[{sourceIP, destinationIP, sourcePort, destinationPort, protocol}];
inputClasses = data[[All, 6]];
```

In[ ]:=
```
(* My computer couldn't process 1.6M training data so instead I used only 100K *)
trainingData = AssociationThread[
   Range[0, 3522],
   Table[Pick[inputExamples[[ ;; 100000]],
     Unitize@(inputClasses[[ ;; 100000]] - i), 0], {i, 0, 3522}]];
testData = Normal[AssociationThread[inputExamples[[100001 ;;]],
    inputClasses[[100001 ;;]]]];
```

In[ ]:=
```
Clear[file, fd, lines, data, sourceIP, sourcePort,
  destinationIP, destinationPort, protocol, inputExamples, inputClasses]
```

In[ ]:=
```
cf = Classify[trainingData, Method → "RandomForest", PerformanceGoal → "DirectTraining"]
```

Out[ ]=
```
ClassifierFunction[ Input type: BooleanVector (length: 104)
                    Number of classes: 3523
                    Method: RandomForest
                    Number of training examples: 100000 ]
```

In[ ]:=
```
cm = ClassifierMeasurements[cf, testData];
```

In[ ]:=
```
cm[{"Accuracy", "Error", "MeanCrossEntropy"}, ComputeUncertainty → True]
```

Out[ ]=
```
{0.8232 ± 0.0009, 0.1768 ± 0.0009, 0.881 ± 0.005}
```

As we can see, we got an accuracy of 82% which for the size of training data compared to the testing data (100000 : 200000) is in my opinion rather good. The classification could be improved with more training data. The result can also be explained by the fact that random forests (as decision trees) don’t manage well data they haven’t been trained with.