Department of Industrial Engineering & Operations Research
Applied Data Science with Venture Applications
IEOR 135 / 290-002
Spring 2018

Homework 04

Entropy, Natural Language Processing & SQL

Table of Contents

Introduction

Due Date: 24rd April 2018
Student's Name: Diego Sapunar
Student's Cal ID: 013109070
Student's Data-X GitHub: Github

Part I

1. Identify the first feature

The dataset shown below represents bank customers with 3 features and the label corresponding to each customer identifies whether they've defaulted or not.

Description:

HasJob: Binary value, equal to 0 when a person has no job and 1 otherwise.
HasFamily: Binary value, equal to 0 when a person has no family and 1 otherwise.
IsAbove30years: Binary value, equal to 0 when a person's age is 30 or below and 1 otherwise.
Defaulter is also a binary valued label which is equal to 1 if a person is a defaulter and 0 otherwise.

Use this dataset to **identify the best feature to do the first split** in a binary decision tree, so as to maximize the information gain in the next split. Show your calculations.

1) Formulas - Reference: PUC Chile Artificial Intelligence Course (IIC2613):
   a. $Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$ (1)
   b. $Entropy(T,X) = \sum_{c \in X} P(c)E(c)$ (2)
   c. $Gain(T,X) = Entropy(T) - Entropy(T,X)$ (3)
   d. Convention:
      i. Defaulter = D
      ii. HasJob = J
      iii. HasFamily = F
      iv. IsAbove30years = A
      v. Entropy(S) = E(S)
      vi. Entropy(T,X) = E(T,X)

2) Calculate Entropy of target, Defaulter in this case. Using (1):

$$Entropy(Defaulter) = Entropy(4,4)$$
$$E(D) = E(0.5, 0.5)$$
$$E(D) = -(0.5 \log_2 0.5) - (0.5 \log_2 0.5)$$
$$E(D) = -\big(0.5 \cdot (-1)\big) - (0.5 \cdot (-1))$$
$$\boldsymbol{E(D) = 1}$$

3) Calculate the entropy for each attribute (branch). Using (2). And then the Gain with (3):

a. HasJob (J):

$$E(2,3) = E(0.4,0.6)$$
$$E(2,3) = -(0.4\log_2 0.4) - (0.6\log_2 0.6)$$
$$E(2,3) = -(-0.53) - (-0.44)$$
$$\boldsymbol{E(2,3) = 0.97}$$

$$E(2,1) = E(0.66,0.33)$$
$$E(2,1) = -(0.66\log_2 0.66) - (0.33\log_2 0.33)$$
$$E(2,1) = -(-0.4) - (-0.53)$$
$$\boldsymbol{E(2,1) = 0.93}$$

$$E(D,J) = P(1)\cdot E(2,3) + P(0)\cdot E(2,1)$$
$$E(D,J) = P\left(\frac{5}{8}\right)\cdot 0.97 + P\left(\frac{3}{8}\right)\cdot 0.93$$
$$E(D,J) = 0.60625 + 0.34875$$
$$\boldsymbol{E(D,J) = 0.955}$$

$$Gain(D,J) = 1 - 0.955$$
$$\boldsymbol{Gain(D,J) = 0.045}$$

b. HasFamily (F):

$$E(1,3) = E(0.25,0.75)$$
$$E(1,3) = -(0.25\log_2 0.25) - (0.75\log_2 0.75)$$
$$E(1,3) = -(-0.5) - (-0.31127775)$$
$$\boldsymbol{E(1,3) = 0.81127775}$$

$$E(3,1) = E(1,3)$$
$$\boldsymbol{E(3,1) = 0.81127775}$$

$$E(D,F) = P(1)\cdot E(1,3) + P(0)\cdot E(3,1)$$
$$E(D,F) = P\left(\frac{1}{2}\right)\cdot 0.81127775 + P\left(\frac{1}{2}\right)\cdot 0.81127775$$
$$E(D,F) = 0.405638875 + 0.405638875$$
$$\boldsymbol{E(D,F) = 0.81127775}$$

$$Gain(D,F) = 1 - 0.81127775$$
$$\boldsymbol{Gain(D,F) = 0.18872225}$$

c. IsAbove30years (A):

$$\boldsymbol{E(3,3) = 1} \qquad \textbf{(By definition)}$$

$$\boldsymbol{E(1,1) = 1} \qquad \textbf{(By definition)}$$

$$E(D,A) = P(1)\cdot E(3,3) + P(0)\cdot E(1,1)$$
$$E(D,A) = P\left(\frac{6}{8}\right)\cdot 1 + P\left(\frac{2}{8}\right)\cdot 1$$
$$\boldsymbol{E(D,A) = 1}$$

$$Gain(D,A) = 1 - 1$$
$$\boldsymbol{Gain(D,A) = 0}$$

4) Choose attribute with the largest information gain as the decision node
We see that the largest information gain is HasFamily (F) with $\mathbf{0.18872225}$, so is the best feature to do the first split.

2. Entropy of S

Given a signal of three symbols **S = (A, B, C)** and P(A)=0.7, P(B)=0.2, P(C)=0.1, What is the entropy of S? What does it mean according to the *Source coding Theorem*?

1) Formula

$$E(x) = \sum_x P(x)\log_2\left(\frac{1}{P(x)}\right)$$

2) Calculate:

$$E(A,B,C) = P(A)\cdot\log_2\frac{1}{P(A)} + P(B)\cdot\log_2\frac{1}{P(B)} + P(C)\cdot\log_2\frac{1}{P(C)}$$
$$E(A,B,C) = 0.7\cdot\log_2\frac{1}{0.7} + 0.2\cdot\log_2\frac{1}{0.2} + 0.1\cdot\log_2\frac{1}{0.1}$$
$$E(A,B,C) = 0.3602011 + 0.4643856 + 0.3321928$$
$$\boldsymbol{E(A,B,C) = 1.1567795}$$

3) This means that by theory, the minimum number of bits necessary to represent each symbol is 1.1567795

Part II

1. What is the difference between a Bag Of words Model in NLP and a Word2vec Model, discuss advantages of one over the other?

Bag-of-words, for a given document extract the unigram words to just create a random list of words. It doesn't take care of the positions, the syntax, semantics, anything, it just put them in a list or array.

In the other hand, according to Towards Data Science, we have Word to vector model, which is a more sophisticated way to aim NLP. Basically, given a bag of words that you got from the document, you create a real vector, as a feature vector, where each feature is a word and the feature's value is a term weight.

Also, using Bag of Words you assign word frequency to document-term matrix element and in Vector Space Model document-term matrix elements are quite general as long as operations (dot product) in vector space make sense, getting the word position.

2. What is a word vector? What is a word Embedding? On what factors does the word embedding of a word depend (explain it from a NLP perspective)?

A mathematical way of representing words is as vectors. Where these vectors are in a continuous space.

Word Embedding, is a method where all the words and phrases of your vocabulary are related to a vector of real numbers. So, by theory, it implies the space with one dimension by word to a continuous vectoral space with less dimensions.

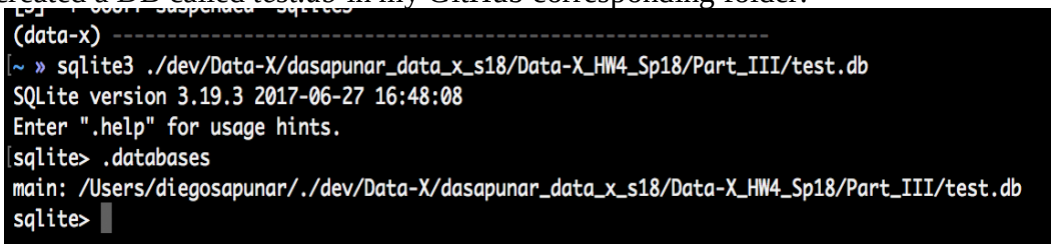
3. What is a corpus in NLP? How is the vocabulary of a model different from the corpus?

Corresponding to **StackOverFlow** forum, a *corpus*, in linguistics, is any coherent body of real-life text or speech being studied. So yes, a book is a corpus. The fact that it's in one string doesn't matter, as long as you don't randomly shuffle the characters. So, Corpus basically means a body, and in the context of Natural Language Processing (NLP), it means a body of text. On the other hand, a vocabulary are the specific words of your string, such as a long array.

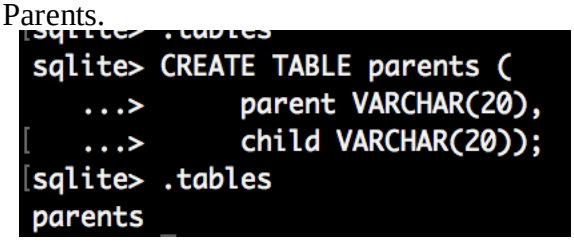
Part III

0. Creating from Scratch

1. I created a DB called *test.db* in my GitHub corresponding folder.



2. I created the table Parents.



```
sqlite> .schema
CREATE TABLE parents (
     parent VARCHAR(20),
     child VARCHAR(20));
```

3. I inserted data to the table Parents.

```
sqlite> INSERT INTO parents (parent, child)
   ...> VALUES ("abraham", "barack") UNION
   ...> VALUES ("abraham", "clinton") UNION
   ...> VALUES ("delano", "herbert") UNION
   ...> VALUES ("fillmore", "abraham") UNION
   ...> VALUES ("fillmore", "delano") UNION
   ...> VALUES ("fillmore", "grover") UNION
   ...> VALUES ("eisenhower", "fillmore");
```

1. Simple SELECTS (on the parents table)

1. SELECT all records in the table

```
sqlite> SELECT * from parents;
abraham|barack
abraham|clinton
delano|herbert
eisenhower|fillmore
fillmore|abraham
fillmore|delano
fillmore|grover
```

2. SELECT child and parent where Abraham is the parent

```
sqlite> SELECT * from parents WHERE parent="abraham";
abraham|barack
abraham|clinton
```

3. SELECT all children that have an ‘e’ in their name (hint: use LIKE ‘%e%’).

```
sqlite> SELECT child from parents WHERE child LIKE "%e%";
herbert
fillmore
delano
grover
```

4. SELECT all unique parents (use SELECT DISTINCT) and order them by name, descending order (i.e fillmore first).

```
sqlite> SELECT DISTINCT parent FROM parents ORDER BY parent DESC;
fillmore
eisenhower
delano
abraham
```

5. SELECT all dogs that are siblings (one-to-one relations). Only show a sibling pair once. To do this you need to select two times from the parents table.

```
sqlite> SELECT child FROM parents WHERE parent="fillmore";
abraham
delano
grover
```

```
sqlite> SELECT child FROM parents WHERE parent="abraham";
barack
clinton
```

OTHER METHOD WITH JOINS:

```
sqlite> SELECT DISTINCT p1.child, p2.child FROM parents AS p1
   ...> INNER JOIN parents AS p2 ON p1.parent = p2.parent
   ...> WHERE p1.child < p2.child;
barack|clinton
abraham|delano
abraham|grover
delano|grover
```

2. JOINS

0. Created the new table dogs.

```
sqlite> CREATE TABLE dogs AS
   ...> SELECT "abraham" AS name, "long" as fur Union
   ...> SELECT "barack", "short" UNION
   ...> SELECT "clinton", "long" UNION
   ...> SELECT "delano", "long" UNION
   ...> SELECT "eisenhower", "short" UNION
   ...> SELECT "fillmore", "curly" UNION
   ...> SELECT "grover", "short" UNION
   ...> SELECT "herbert", "curly";
```

1. COUNT the number of short haired dogs.

```
sqlite> SELECT COUNT(*) FROM dogs WHERE fur="short";
3
```

2. JOIN tables parents and dogs and SELECT the parents of curly dogs

```
sqlite> SELECT parent FROM parents INNER JOIN dogs ON parents.child = dogs.name WHERE dogs.fur="curly";
eisenhower
delano
```

3. JOIN tables parents and dogs and SELECT the parents and children that have the same fur type. Only show them once.

```
sqlite> SELECT p.parent, p.child FROM parents AS p
   ...> INNER JOIN dogs AS d1 ON d1.name = p.parent
   ...> INNER JOIN dogs AS d2 ON d2.name = p.child
   ...> WHERE d1.fur = d2.fur;
abraham|clinton
```

3. Aggregate functions, numerical logic and grouping

0. Created the new table animals.

```
sqlite> CREATE TABLE animals AS
   ...> SELECT "dog" AS kind, 4 AS legs, 20 AS weight UNION
   ...> SELECT "cat", 4, 10 UNION
   ...> SELECT "ferret", 4, 10 UNION
   ...> SELECT "parrot", 2, 6 UNION
   ...> SELECT "penguin", 2, 10 UNION
   ...> SELECT "t-rex", 2, 12000;
```

1. SELECT the animal with minimum weight. Display kind and min_weight.

```
sqlite> SELECT kind, MIN(weight) FROM animals;
parrot|6
```

2. Use the aggregate function AVG to display a table with the average number of legs and the average weight.

```
sqlite> SELECT AVG(legs), AVG(weight) FROM animals;
3.0|2009.33333333333
```

3. SELECT the animal kind(s) that have more than two legs, but weight less than 20. Display kind, weight and legs.

```
sqlite> SELECT kind, weight, legs FROM animals
   ...> WHERE legs>2 AND weight<20;
cat|10|4
ferret|10|4
```

4. SELECT the average weight for all the animals with 2 legs and the animals with 4 legs (by using GROUP BY).

```
sqlite> SELECT legs, AVG(weight) FROM animals GROUP BY legs;
2|4005.33333333333
4|13.3333333333333
```