# Data Warehousing And Data Mining

## Major Project - Phase 2

INSTRUCTOR: Dr. Kamalakar Karlapalem
- Gunjan Karamchandani (20161086), Neal Karpe (20161159)

## Dataset Description

We have chosen the zoo dataset available on UCI archive.

### Items in the dataset

There are a total of 15 items which are all characteristics of different species of animals. The list of items present are:

1. Hair
2. Aquatic
3. Venomous
4. Feathers
5. Predator
6. Fins
7. Eggs
8. Toothed
9. Tails
10. Milk
11. Backbone
12. Domestic
13. Airborne
14. Breathes
15. Cat-size

### Description of transactions

Each transaction can be mapped to a particular animal species. The itemset in each transaction represents the characteristics of that animal.

The data set contains all the above listed items as boolean attributes. The itemset for a particular transaction is the set of attributes with value 1 in that row.  For example: A row with value of Feathers = 1, Eggs = 1 and rest attributes = 0, would indicate this itemset: *{Feathers, Eggs}*.

### Characteristics of dataset

Number of Transactions: 101

Number of Items: 15

## Implementation

### Algorithm/tool used

We used Apriori algorithm, given the small number of transactions and items in the dataset. The algorithm is initiated by considering all 1-item itemsets as the **candidate set**. The itemsets are validated and the infrequent ones are eliminated. Now, we combine all k-item frequent itemsets to form a **candidate set** for (k+1)-item itemsets. This is based on the fact that a superset of an infrequent itemset can never be frequent.

We implemented the algorithm without using any external library, in Python3. We included optimizations in the implementation (like string representation of itemsets, hashing, etc.) to improve the performance, so that our code provides results even for min_sup=1 within reasonable amount of time.

## Code

Available here for testing. Can adjust min support value to obtain different frequent itemsets.

```
def rowContainsItemSet(row, itemset):
    """(Bool) Check if a row contains a particular itemset"""
    for item in itemset:
         if row[item] != '1':
                 return False
    return True

def frequency(str_item_set,data):
    """(Int) Returns the number of rows the itemset is present in"""
    items = list(map(int, str_item_set.split("#")))
    count = 0
    for row in data:
         if rowContainsItemSet(row, items):   count += 1
    return count

def join(string1, string2):
    """Input: two itemsets (in string format) of same length
        Output: returns union itemset if it is length+1, else returns false"""
    itemset1 = set(list(map(int, string1.split("#"))))
    len1 = len(itemset1)
    itemset2 = set(list(map(int, string2.split("#"))))
    union = itemset1.union(itemset2)
    if len(union) != len1 + 1:   return False
    num_list = list(union)
    num_list.sort()
    str_list = [str(item) for item in num_list]
    return "#".join(str_list)

def frequentItemSets(items,data,min_sup):
    """return value is a set of itemsets (in string format)"""
    result = {}
    k = 1
    candidateItemSets = set([str(item) for item in items])
    while len(candidateItemSets) != 0 and k<=len(items):
         verifiedItemSets = set()
         for str_item_set in candidateItemSets:
                 item_set_frequency = frequency(str_item_set,data)
                 if item_set_frequency >= min_sup:
                         verifiedItemSets.add(str_item_set)
                         result[str_item_set] = item_set_frequency
         candidateItemSets = set()
         verifiedSetList = list(verifiedItemSets)
         n = len(verifiedItemSets)
         if n == 0:   break
         print("count of "+str(k)+"-item frequent itemsets:", n)
         for i in range(n-1):
                 for j in range(i+1,n):
                         joined_str_item_set = join(verifiedSetList[i], verifiedSetList[j])
                         if joined_str_item_set:   candidateItemSets.add(joined_str_item_set)
         k += 1
    return result

def parseResult(result, attributes_row):
    list_of_strings = list(result)
    list_of_lists = list(map(lambda x: list(map(int,x.split("#"))), list_of_strings))
```

```
    list_of_lists.sort()
    list_of_lists.sort(key=len)
    print("---------------Frequent itemsets along with their support count---------------")
    for numericItemList in list_of_lists:
         strItemList = "#".join(list(map(str,numericItemList)))
         support_count = result[strItemList]
         namedItemList = list(map(lambda x: attributes_row[x], numericItemList))
         print(", ".join(namedItemList), "- " + str(support_count))

items = [1,2,3,4,5,6,7,8,9,10,11,12,14,15,16]
data = []
with open('zoo.csv', 'r') as f:
    reader = list(csv.reader(f))
    attributes_row = list(reader[0])
    for i in range(1,len(reader)):
         data.append(list(reader[i]))

min_sup = 50 # change min support count value here
result = frequentItemSets(items, data, min_sup) # set of all itemsets (in string format)
parseResult(result, attributes_row)
```

# Results

## Frequent Itemsets

<table>
<tr><th>For min_sup = 40</th><th>For min_sup = 50</th></tr>
<tr><td>count of 1-item frequent itemsets: 9<br>count of 2-item frequent itemsets: 14<br>count of 3-item frequent itemsets: 7<br>count of 4-item frequent itemsets: 1</td><td>count of 1-item frequent itemsets: 6<br>count of 2-item frequent itemsets: 5<br>count of 3-item frequent itemsets: 2</td></tr>
<tr><td>hair - 43<br>eggs - 59<br>milk - 41<br>predator - 56<br>toothed - 61<br>backbone - 83<br>breathes - 80<br>tail - 75<br>catsize - 44<br>hair, breathes - 43<br>milk, toothed - 40<br>eggs, backbone - 42<br>milk, backbone - 41<br>milk, breathes - 41<br>predator, backbone - 47<br>predator, tail - 42<br>toothed, backbone - 61<br>toothed, breathes - 47<br>toothed, tail - 52<br>backbone, breathes - 69<br>backbone, tail - 74<br>backbone, catsize - 43<br>breathes, tail - 61<br>milk, toothed, backbone - 40<br>milk, toothed, breathes - 40<br>milk, backbone, breathes - 41<br>predator, backbone, tail - 41<br>toothed, backbone, breathes - 47<br>toothed, backbone, tail - 52<br>backbone, breathes, tail - 60<br>milk, toothed, backbone, breathes - 40</td><td>eggs - 59<br>predator - 56<br>toothed - 61<br>backbone - 83<br>breathes - 80<br>tail - 75<br>toothed, backbone - 61<br>toothed, tail - 52<br>backbone, breathes - 69<br>backbone, tail - 74<br>breathes, tail - 61<br>toothed, backbone, tail - 52<br>backbone, breathes, tail - 60</td></tr>
</table>

## Selection of min support and confidence

We chose min support as **0.5**. Our rationale behind this was that there were too many itemsets had support count in the range 40-50. Hence, we won't be able to derive much "knowledge" if too many itemsets are frequent. Thus, we decided to call an itemset 'frequent' if it is found in more than half of the transactions.

We chose confidence as **0.85**. It was interesting to find that even when min support was as high as 0.5, we found 4 association rules with almost 100% confidence. These rules are "always followed". We were also able to derive some interesting knowledge when we considered the other rules that had 85%-90% confidence. Hence, we set the min confidence as 0.85.

## Interesting association rules

1. **toothed -> backbone:** Confidence = 61/61 = 100%
    a. Interpretation: All toothed animals have a backbone.
    b. Explanation: Teeth for any animal consist of a mix of calcium, mineral salts and phosphorus. These materials are also present in the backbone, so any animal having teeth is also highly likely to have a backbone.
2. **toothed, tail -> backbone:** Confidence = 52/52 = 100%
    a. Interpretation: All animals with teeth and tail have a backbone.
    b. Explanation: Invertebrates usually do not have teeth or a tail. Hence, it is highly likely that an animal with teeth & tail is a vertebrate (i.e. has backbone).
3. **tail -> backbone:** Confidence = 74/75 = 98.66%
    a. Interpretation: Almost all animals with tail have a backbone.
    b. Explanation: In general, having a tail implies that the animal is a vertebrate (has backbone). However, one animal (Scorpion) has a tail & is not a vertebrate.
4. **breathes, tail -> backbone:** Confidence = 60/61 = 98.33%
    a. Interpretation: Animals that breathe through lungs and have a tail, almost always have a backbone.
    b. Explanation: We saw that all animals with tail except for Scorpion have a backbone. Some of these animals don't breathe with lungs (e.g. fish). Remove these cases, we still get a very high confidence.
5. **backbone -> tail:** Confidence = 74/83 = 89.15%
    a. Interpretation: Most vertebrates have a tail.
    b. Explanation: Vertebrates evolved from fish and all fish had tails. Hence, most vertebrates got tails. However, for some (e.g. homo sapiens), the tail wasn't that useful, and was eventually lost. Hence, this rule has around 90% confidence.
6. **backbone, breathes -> tail:** Confidence = 60/69 = 86.95%
    a. Interpretation: High portion of vertebrates that breathe with lungs have a tail.
    b. Explanation: Vertebrates that have lungs are mostly found on land (e.g. homo sapiens). Hence, they are less like fish. Since tails of vertebrates came from fish, land vertebrates have less probability of keeping a tail.
7. **backbone -> breathes:** Confidence = 69/80 = 86.25%
    a. Interpretation: Most vertebrates breathe with lungs.
    b. Explanation: Majority of the vertebrates are land animals (roughly 15% are aquatic). Hence, a lot of them breathe through lungs, and some portion of vertebrates don't.
8. **toothed -> tail:** Confidence = 52/61 = 85.24%
    a. Interpretation: Many animals having teeth have tail.
    b. Explanation: toothed->backbone & backbone->tail
9. **toothed -> tail, backbone:** Confidence = 52/61 = 85.24%
10. **toothed, backbone -> tail:** Confidence = 52/61 = 85.24%