CITS3401 Data Warehousing
Project Report

Bruce How (22242664) & Haolin Wu (21706137)

Semester 1 2019

Contents

1 Design and Implementation

1.1 Concept Hierarchies & StarNet Model

In order to create our StarNet model, we had to first build a concept hierarchy for each dimension. Each concept hierarchy is of total order with the value "ALL" in the root node as shown in Figure 1.

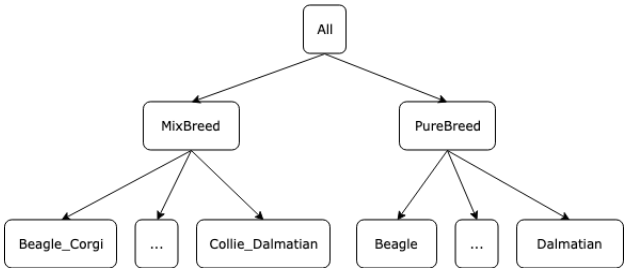


Figure 1: Concept Hierarchy for Breed

Each level on the concept hierarchy is then categorized and mapped to the corresponding dimension on the StarNet model (see Figure 14). For example, the levels in Figure 1 would be categorized to "All", "Breed Purity", and "Specific Breed" and each category would be appropriately mapped to the *Breed* dimension on the StarNet model.

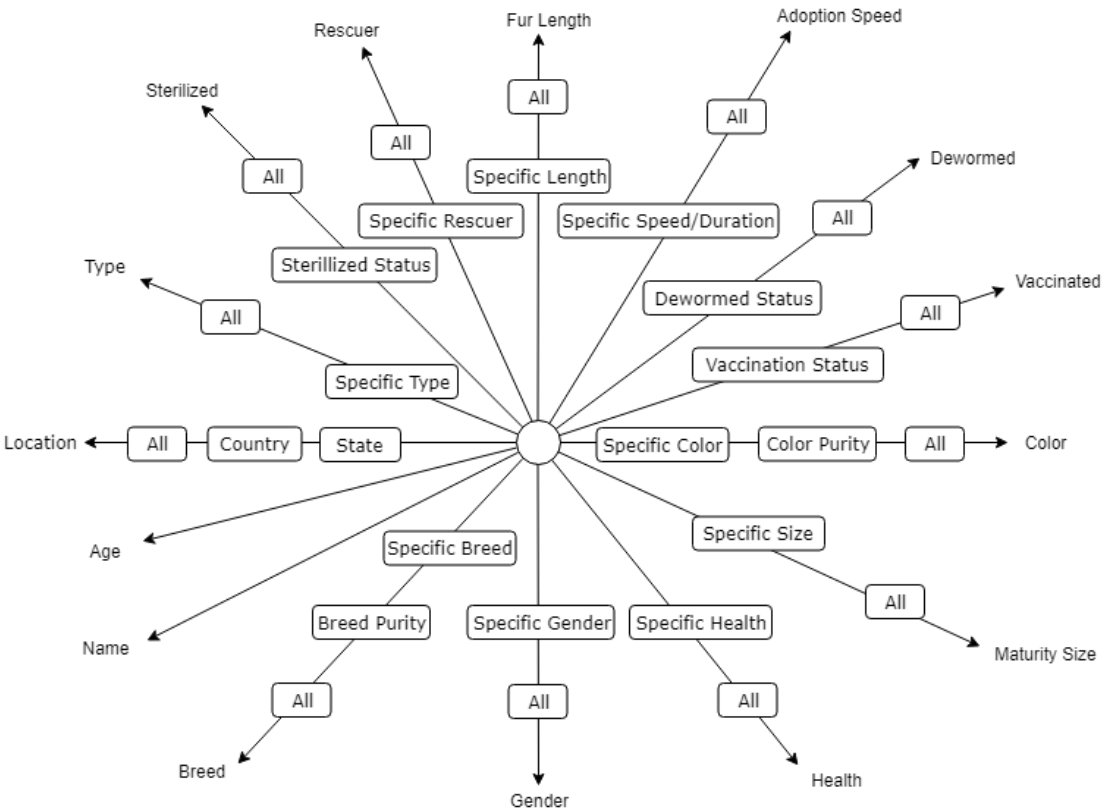


Figure 2: StarNet model

1.2 Business Queries

When designing our business queries, we wanted to ensure that they could be answered using our StarNet model regardless of how specific or broad they may be.

Our 5 business queries are as follows:

1. How many Pure and Mixed Coloured pets are there?
2. How many vaccinated pets are also sterilized?
3. How many female cats with long hair were adopted instantly?
4. How many pets from each state were adopted?
5. How many healthy Pure and Mixed Breed pets have been adopted slowly?

1.2.1 StarNet Footprints

As our business queries were based on both low and high level information, we are able to test the capabilities of our StarNet model. Our StarNet model had to be refined several times to be able to illustrate all our business query footprints. Figure 3 below illustrates the StarNet footprint for the fifth business query.

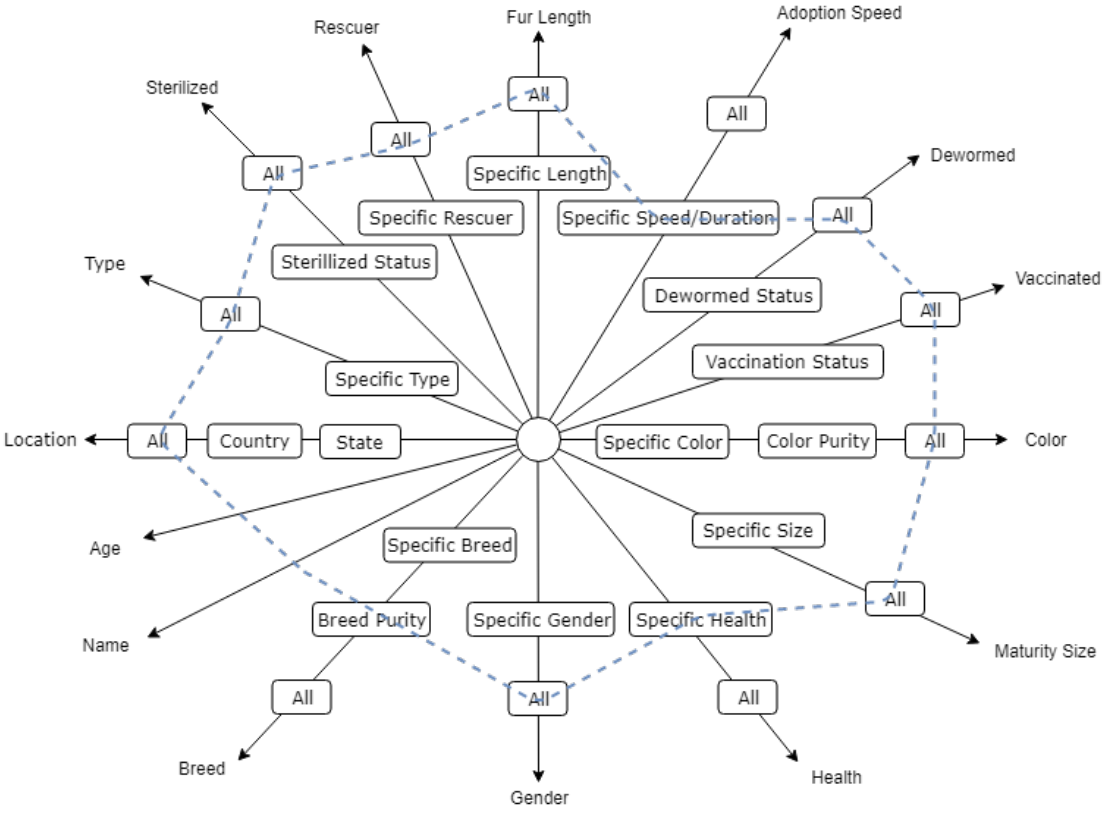


Figure 3: StarNet footprint for Business Query 5

1.3 Database Schema

Our database follows the structure of a Snowflake Schema (Figure 4). Rather than representing the adoption speed as a measure, we decided to use a dimension, *DimAdoptionSpeed*, instead. This allowed us to add additional columns to represent the speed of adoption such as *ListingDuration*, which represented the actual measure value, and *AdoptionSpeed*, which is an ENUM string used to describe the adoption speed (e.g Instant, Slow, etc.).

The idea behind this was so that adoption speeds could be grouped and easily visualized. This has also proven to be useful when generating the data cube. Additional columns in *DimBreed* and *DimColor* have also been added for the same reason.

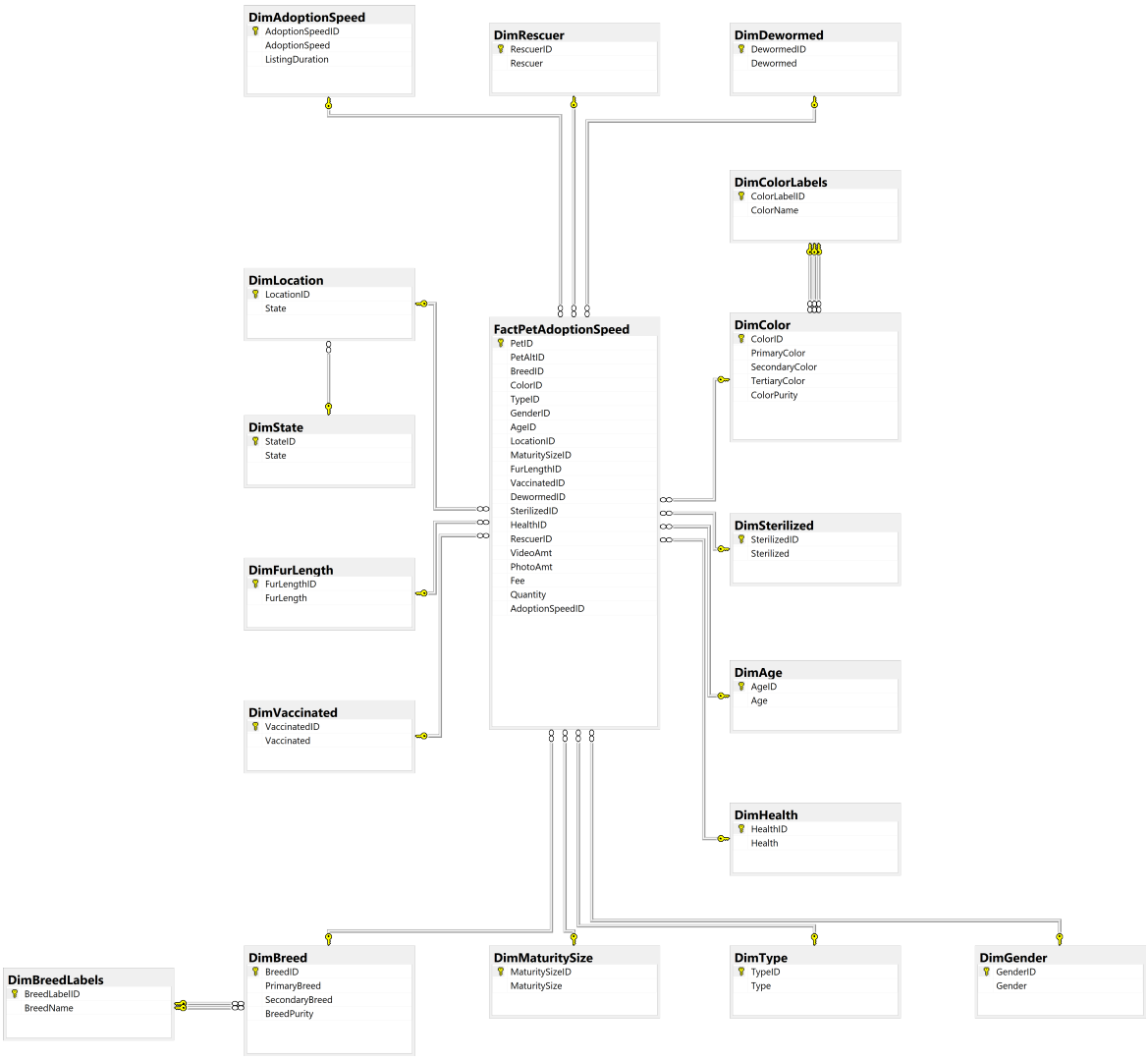


Figure 4: StarNet footprint for Business Query 5

1.4 ETL

The ETL process was done using a python script, *etl.py*, which utilises the csv module. Each row in the *train.csv* file is passed to several extraction functions where the appropriate data, *row[x]* (breed1 and breed2 in the example below), is extracted and written to it's respective *csv* file.

```
seen_breed = []

def breed_data(file, row):
    breed1, breed2 = row[3], row[4]

    # Check to see if we have already seen the breed
    if [breed1, breed2] in seen_breed:
        return # Already exists, don't add duplicates

    seen_breed.append([breed1, breed2])

    breed_purity = "Mixed"
    if breed2 == "0":
        breed_purity = "Pure"
        breed2 = "" # NULL color

    data = ["", breed1, breed2, breed_purity]
    file.writerow(data)
```

Each row of data is passed through a set of conditions to ensure that duplicate rows are not re-inserted into the processed *csv* file; and that columns with the value of "0" (not specified) are replaced with *NULL*. Global-scoped arrays are used to store data that are referenced by other dimensions to ensure that all foreign keys will reference the correct data. The extraction function also generates values for additional columns to conform with our database design. Some of these include *color_purity*, *listing_duration*, and *breed_purity* (shown in the example above).

Header rows are appended to each processed *csv* file to ensure that they are easily readable. These rows are ignored when bulk-inserted into the database.

```
write_breed.writerow(["BreedID", "PrimaryBreed", "SecondaryBreed"])
write_color.writerow(["ColorID", "PrimaryColor", "SecondaryColor",
    "TertiaryColor"])
```

The processed *csv* files are generate to the **csv** folder in the same directory as *etl.py*.

1.5 Implementation

1.5.1 Building the Database

The PetAdoption database can be created using the provided *build_database.sql* script. The script is responsible for creating the appropriate dimension and fact tables, setting up constraints, and referencing all primary and foreign keys accordingly. Prior to this, any existing PetAdoption databases are automatically dropped and re-created (see code below).

```
USE master;
GO

IF EXISTS (SELECT [name] FROM [master].[sys].[databases] WHERE [name] =
   N'PetAdoption')
ALTER DATABASE PetAdoption SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO
DROP DATABASE PetAdoption;
PRINT CHAR(13) + CHAR(10) + 'DROPPED␣EXISTING␣DATABASE';
```

Enumerated columns are defined in specific tables to ensure that data across all rows are consistent. The segment of code below shows the declaration of *DimColor*. In this example, the *ColorPurity* column is declared as an enum where it's value can only either be "Pure" or "Mixed".

```
CREATE TABLE DimColor (
  ColorID INT PRIMARY KEY IDENTITY,
  PrimaryColor INT NOT NULL,
  SecondaryColor INT,
  TertiaryColor INT,
  ColorPurity VARCHAR(5) NOT NULL CHECK (ColorPurity IN('Pure', 'Mixed'))
);
```

Foreign keys constraints are declared accordingly to ensure that data inserted into the tables are valid. As our database follows a Snowflake schema structure, foreign key constraints must also be added to the appropriate dimensions. These dimensions include *DimBreed*, *DimColor* and *DimLocation* which references the *DimBreedLabels*, *DimColorLabels* and *DimState* dimensions respectively (shown in the code below).

```
ALTER TABLE DimBreed ADD
CONSTRAINT FK_PrimaryBreed FOREIGN KEY (PrimaryBreed) REFERENCES
   DimBreedLabels(BreedLabelID),
CONSTRAINT FK_SecondaryBreed FOREIGN KEY (SecondaryBreed) REFERENCES
   DimBreedLabels(BreedLabelID);

ALTER TABLE DimColor ADD
CONSTRAINT FK_PrimaryColor FOREIGN KEY (PrimaryColor) REFERENCES
   DimColorLabels(ColorLabelID),
```

```
CONSTRAINT FK_SecondaryColor FOREIGN KEY (SecondaryColor) REFERENCES
   DimColorLabels(ColorLabelID),
CONSTRAINT FK_TertiaryColor FOREIGN KEY (TertiaryColor) REFERENCES
   DimColorLabels(ColorLabelID);

ALTER TABLE DimLocation ADD
CONSTRAINT FK_State FOREIGN KEY (State) REFERENCES DimState(StateID);
```

1.5.2 Populating the Database

The PetAdoption database can be populated with the appropriate data using the *populate_database.sql* script. This script utilizes BULK_INSERT to insert data from the processed *.csv* files to the respective tables. Because of the use of **:setvar SQLSourceDataPath**, SQLCMD mode must be enabled when used in SSMS. This variable is responsible for setting the directory containing the processed csv files and may require changing based on the testing environment.

```
BULK INSERT DimDewormed FROM '$(SQLSourceDataPath)DimDewormed.csv'
WITH (
   FIRSTROW=2,
   CHECK_CONSTRAINTS,
   DATAFILETYPE='char',
   FIELDTERMINATOR=',',
   ROWTERMINATOR='\n',
   TABLOCK
);
```

The BULK_INSERT segment of code above is based on the example provided in lab 3. The arguments used in BULK_INSERT include "FIRSTROW=2", to exclude the header row in the csv files, FILEDTERMINATOR=',' and ROWTERMINATOR='\n', to process the csv files appropriately, and CHECK_CONSTRAINTS to ensure that the data inserted is valid. This constraint has been omitted for tables using foreign keys to avoid errors (as all data are bulk inserted together).

1.6 Visualisation

Visualising PetAdoption as a data cube allows for Roll Up and Drill Down operations, making our business queries a lot more versatile, and explores information that normally would not be found without a data cube.

Using Visual Studio (SSDT), we are able to create the data cube for better visualization in Power BI (Figure 5).

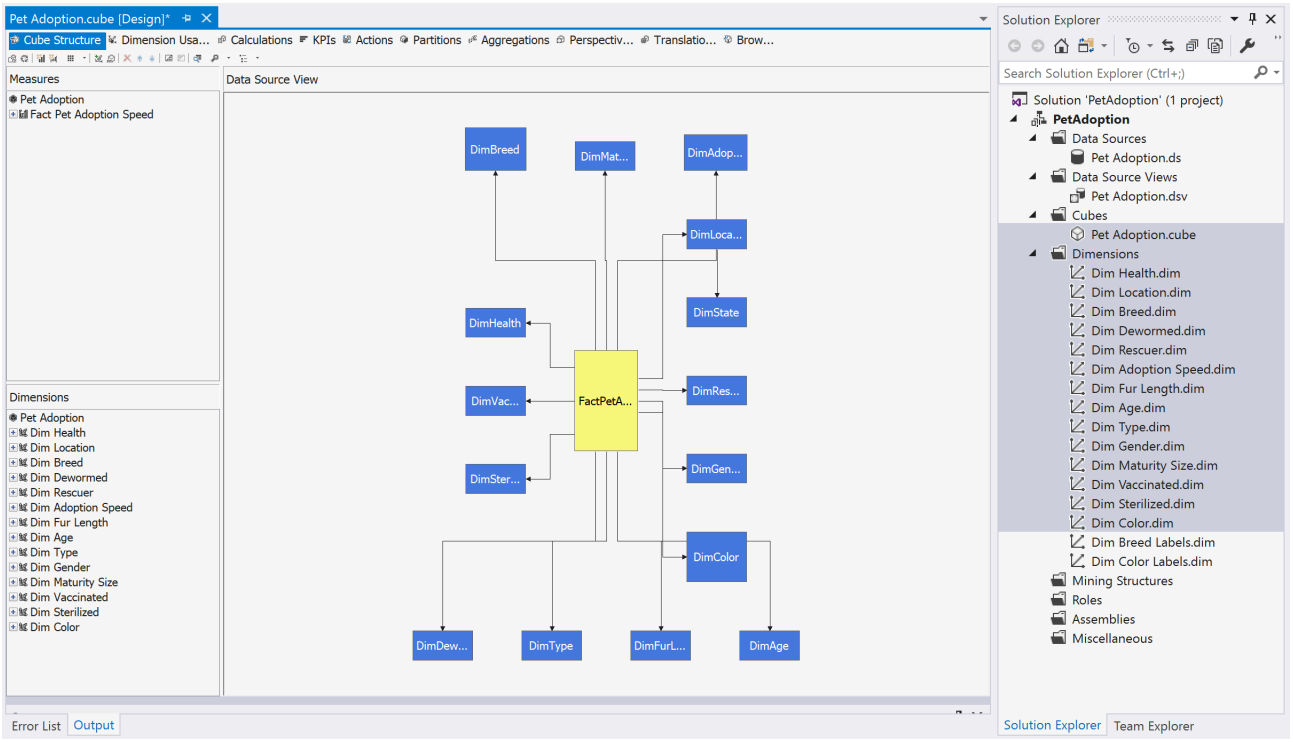


Figure 5: Data Cube for PetAdoption

Furthermore, creating hierarchies allows up to drill down further into the data, finding information that can prove useful in a business scenario. Color is an attribute that is made up of Primary, Secondary and Tertiary colors. By allowing a drill down on primary color, we can see the effect of not only purity, but primary color.

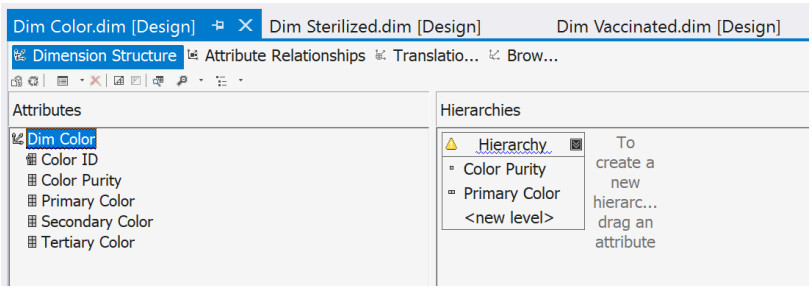


Figure 6: Hierarchy for PetAdoption

Unfortunately, *Color Labels* and *Breed Labels* were not link-able to the hierarchy. This means that our queries in PowerBI required a legend, to show what color each ID corresponded to.

Additionally, the cube needs to have the data loaded into the server using SSMS, before it can be deployed for use by PowerBI.

1.6.1 Business Query 1

Our first business query demonstrates the use of the Drill Down and Roll Up operations on our PowerBI visualisations.

Figure 7 illustrates a Donut graph containing the number of Pure and Mixed coloured pets. This graph has the ability to Drill Down to a lower level as shown in Figure 8, which illustrates the number of pets for each Primary Colour.

The below query was executed in SSDT to validate the visualisation results.

```
-- Business Query 1 (High Level)
SELECT ColorPurity, SUM(Quantity) Total
FROM FactPetAdoptionSpeed f, DimColor c
WHERE f.ColorID = c.ColorID
GROUP BY c.ColorPurity
ORDER BY Total DESC;

-- Drilled Down Query (Low Level)
SELECT ColorName, SUM(Quantity) Total
FROM FactPetAdoptionSpeed f, DimColor c, DimColorLabels l
WHERE f.ColorID = c.ColorID
AND c.PrimaryColor = l.ColorLabelID
GROUP BY ColorName
ORDER BY Total DESC;
```

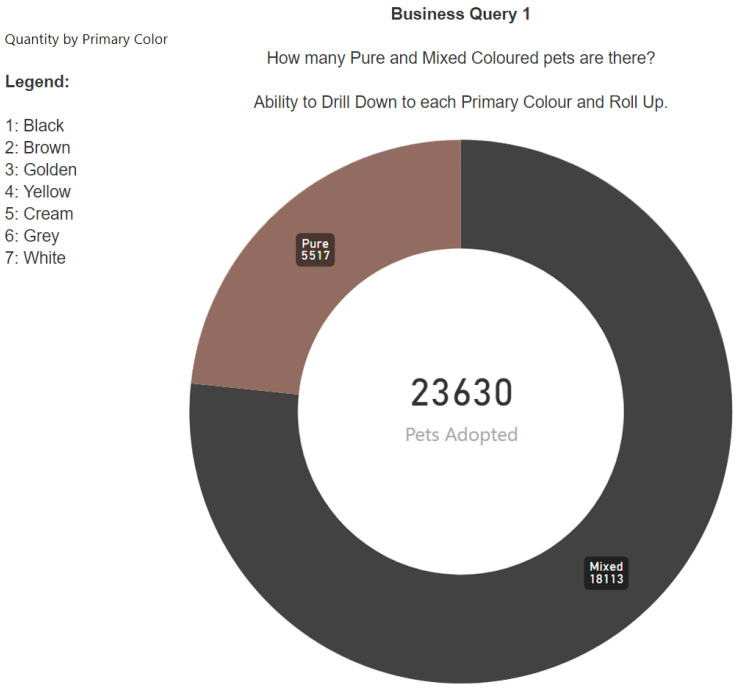


Figure 7: Business Query 1 Visualisation

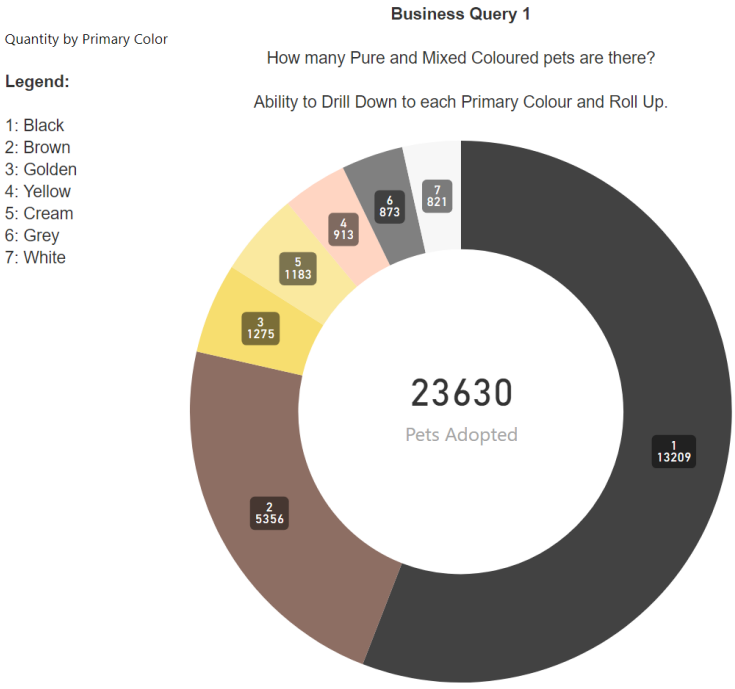


Figure 8: Business Query 1 Drilled Down Visualisation

1.6.2 Business Query 2

Figure 9 is an example of a high level query that can be visualised with our data. It is illustrated on a clustered bar chart for simplicity.

This query can be visualised by filtering the *Sterilized* and *Vaccinated* attributes for the value "Yes", and grouping by *Type*.

The below query was executed in SSDT to validate the visualisation results.

```
-- Business Query 2
SELECT Type, SUM(Quantity) Total
FROM FactPetAdoptionSpeed f, DimVaccinated v, DimSterilized s, DimType t
WHERE f.VaccinatedID = v.VaccinatedID
AND f.TypeID = t.TypeID
AND f.SterilizedID = s.SterilizedID
AND s.Sterilized = 'Yes'
AND v.Vaccinated = 'Yes'
GROUP BY Type;
```

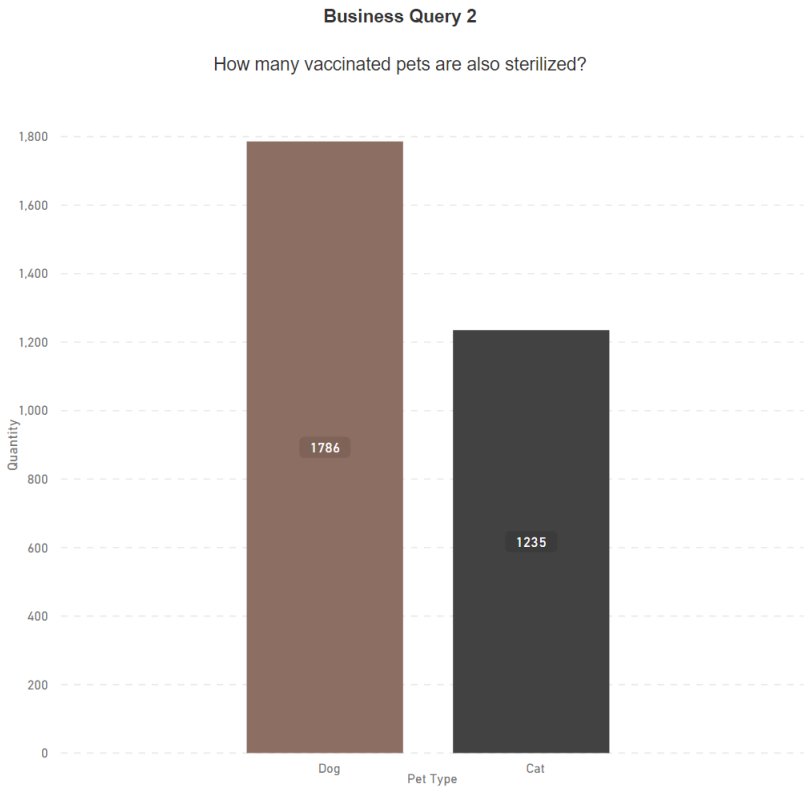


Figure 9: Business Query 2 Visualisation

1.6.3 Business Query 3

Figure 10 is an example of a low level query that can be visualised with our data. It is illustrated on a clustered bar chart for simplicity.

The below query was executed in SSDT to validate the visualisation results.

```
-- Business Query 3
SELECT SUM(Quantity) Total
FROM FactPetAdoptionSpeed f, DimType t, DimGender g, DimAdoptionSpeed a,
   DimFurLength l
WHERE f.TypeID = t.TypeID
AND f.GenderID = g.GenderID
AND f.AdoptionSpeedID = a.AdoptionSpeedID
AND f.FurLengthID = l.FurLengthID
AND l.FurLength = 'Long'
AND g.Gender = 'Female'
AND a.AdoptionSpeed = 'Instant'
AND t.Type = 'Cat';
```

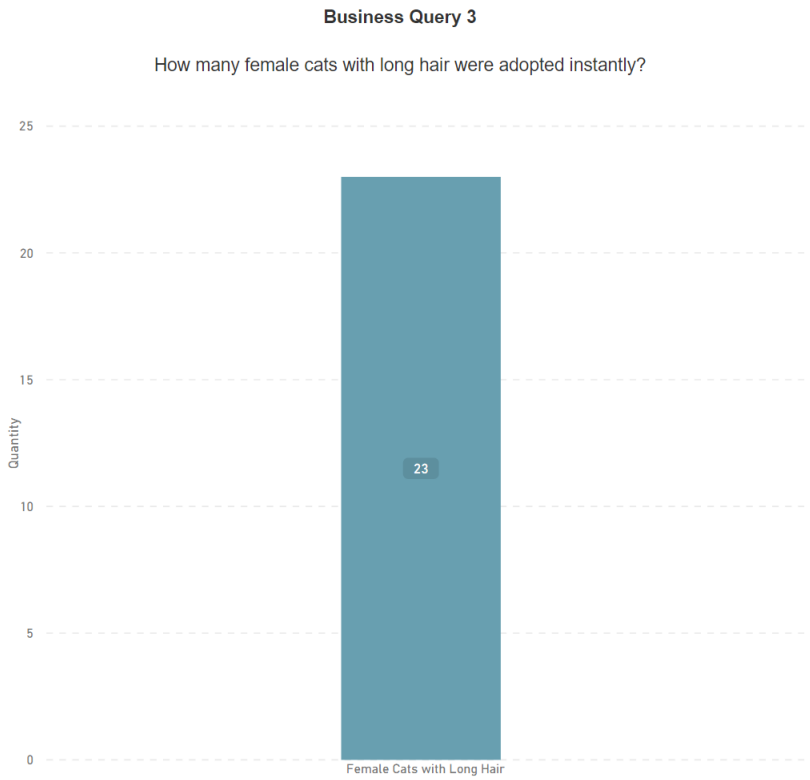


Figure 10: Business Query 3 Visualisation

1.6.4 Business Query 4

This query can be visualised by filtering the *AdoptionSpeed* attributes for all values except "Not Adopted", and grouping by *State*.

The below query was executed in SSDT to validate the visualisation results.

```
-- Business Query 4
SELECT s.State State, SUM(Quantity) Total
FROM FactPetAdoptionSpeed f, DimLocation l, DimState s, DimAdoptionSpeed a
WHERE f.LocationID = l.LocationID
AND f.AdoptionSpeedID = a.AdoptionSpeedID
AND l.State = s.StateID
AND a.AdoptionSpeedID != 4
GROUP BY s.State
ORDER BY Total DESC;
```

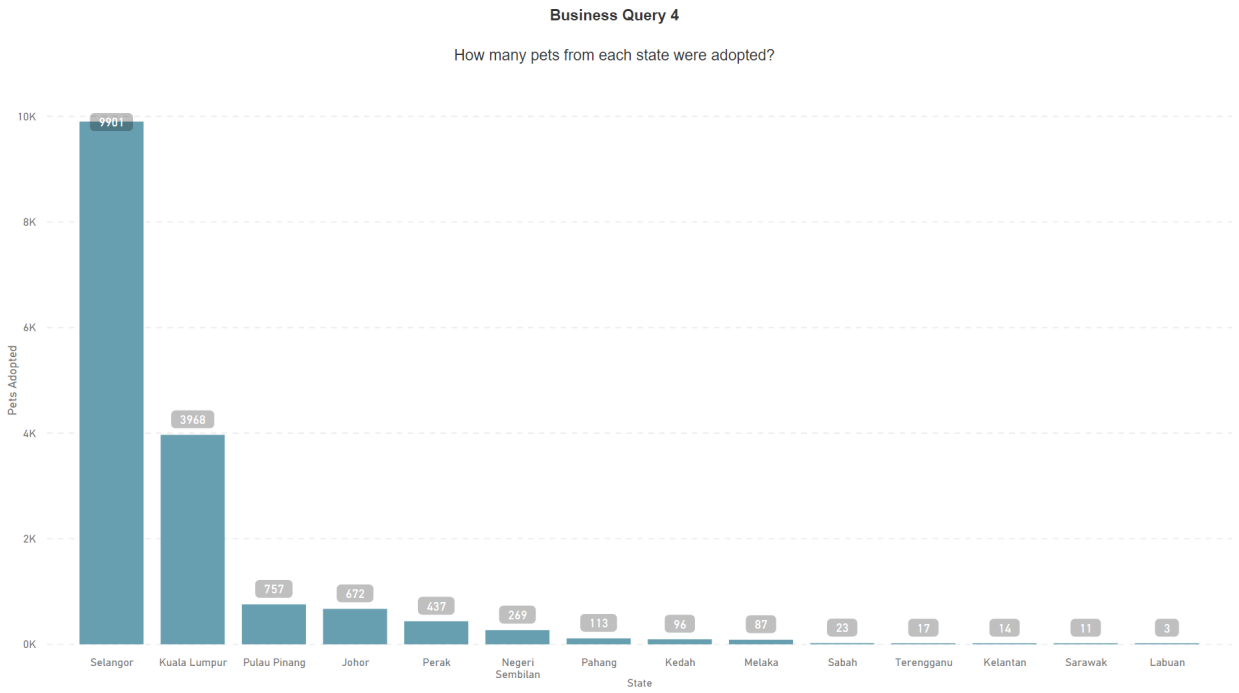


Figure 11: Business Query 4 Visualisation

1.6.5 Business Query 5

Figure 12 illustrates the use of the *DimAdoptionSpeed* dimension and its *AdoptionSpeed* column. This query is another example of a low level query that can easily visualised with our data.

This query can be visualised by filtering the AdoptionSpeed attribute for the value "Slowly" and grouping by "BreedPurity".

The below query was executed in SSDT to validate the visualisation results.

```
-- Business Query 5
SELECT BreedPurity, SUM(Quantity) Total
FROM FactPetAdoptionSpeed f, DimHealth h, DimAdoptionSpeed a, DimBreed b
WHERE f.HealthID = h.HealthID
AND f.AdoptionSpeedID = a.AdoptionSpeedID
AND f.BreedID = b.BreedID
AND h.Health = 'Healthy'
AND a.AdoptionSpeed = 'Slow'
GROUP BY b.BreedPurity;
```

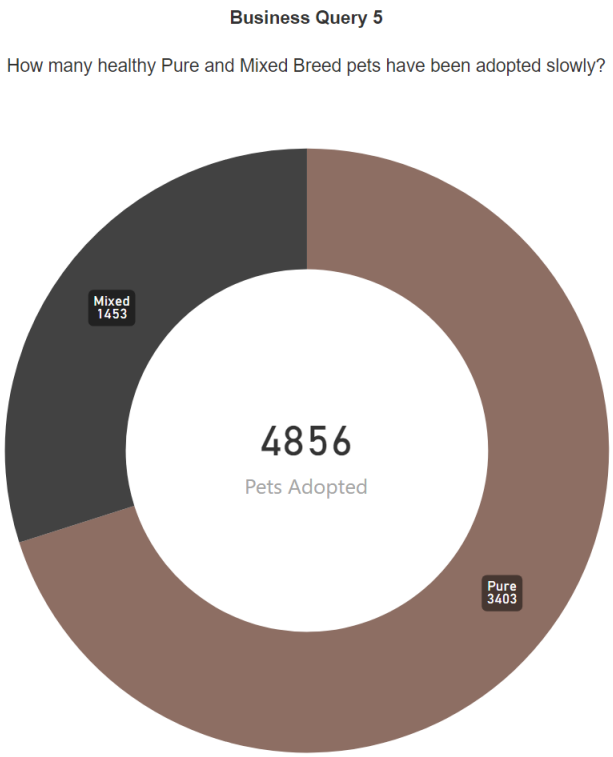


Figure 12: Business Query 5 Visualisation

2 Usage

For the usage of the data, *train.csv* needed to be modified so that each column was no longer numerical data, but categorical, so that Weka could process it sufficiently. This is done through *build_associations.py*, writing each column into *associations.csv*. Additionally, to make columns such as *Fee*, *Age*, *PhotoAmt* and *VideoAmt* into categorical data, we chose groups, such as "Not Free" and "Free", or ages "0-2" and "2-5".

```
def get_type(row):
    pet_type = row[0]
    if pet_type == "2":
        return "Cat"
    return "Dog"
```

This replaces all instances of "2" with "Cat", allowing for categorical data.

2.1 Association Rules

Association Rule Mining is a category of unsupervised learning, allowing patterns in data to be found without a target variable.

Using Weka, association rules are able to be mined using the Apriori algorithm. In PetAdoption, Apriori identifies the individual variables in the data set, and extends them into larger and larger item sets, as long as the item sets is frequent in the data set.

```
Apriori
=======

Minimum support: 0.2 (2999 instances)
Minimum metric <confidence>: 0.1
Number of cycles performed: 16

Generated sets of large itemsets:

Size of set of large itemsets L(1): 13

Size of set of large itemsets L(2): 9

Size of set of large itemsets L(3): 2

Best rules found:

 1. Sterilized=No 10077 ==> AdoptionSpeed=Average 3022    conf:(0.3)
 2. BreedPurity=Pure VideoAmt=None 10337 ==> AdoptionSpeed=Not Adopted 3033    conf:(0.29)
 3. BreedPurity=Pure 10762 ==> AdoptionSpeed=Not Adopted 3124    conf:(0.29)
 4. VideoAmt=None 14419 ==> AdoptionSpeed=Not Adopted 4076    conf:(0.28)
 5. Health=Healthy VideoAmt=None 13921 ==> AdoptionSpeed=Not Adopted 3896    conf:(0.28)
 6. Health=Healthy 14478 ==> AdoptionSpeed=Not Adopted 4012    conf:(0.28)
 7. Fee=Free VideoAmt=None 12186 ==> AdoptionSpeed=Not Adopted 3374    conf:(0.28)
 8. Health=Healthy Quantity=Single 11123 ==> AdoptionSpeed=Average 3068    conf:(0.28)
 9. Fee=Free 12663 ==> AdoptionSpeed=Not Adopted 3476    conf:(0.27)
10. Health=Healthy Fee=Free VideoAmt=None 11755 ==> AdoptionSpeed=Not Adopted 3220    conf:(0.27)
11. Quantity=Single 11565 ==> AdoptionSpeed=Average 3163    conf:(0.27)
12. Health=Healthy Fee=Free 12218 ==> AdoptionSpeed=Average 3333    conf:(0.27)
13. Health=Healthy Fee=Free 12218 ==> AdoptionSpeed=Not Adopted 3318    conf:(0.27)
14. Quantity=Single VideoAmt=None 11123 ==> AdoptionSpeed=Average 3019    conf:(0.27)
15. Health=Healthy Fee=Free VideoAmt=None 11755 ==> AdoptionSpeed=Average 3188    conf:(0.27)
16. Health=Healthy 14478 ==> AdoptionSpeed=Average 3925    conf:(0.27)
17. Fee=Free 12663 ==> AdoptionSpeed=Average 3430    conf:(0.27)
18. Health=Healthy VideoAmt=None 13921 ==> AdoptionSpeed=Average 3754    conf:(0.27)
19. Fee=Free VideoAmt=None 12186 ==> AdoptionSpeed=Average 3278    conf:(0.27)
20. VideoAmt=None 14419 ==> AdoptionSpeed=Average 3859    conf:(0.27)
```

Figure 13: Association Rules mined with Apriori

Apriori calculates the confidence of each association rule, and orders them highest to lowest, the highest being the strongest association rule.

In PetAdoption, ordering by confidence displays many rules that are ranked highly, however not interesting and helpful to our understanding of the data set. This is because it is highly dependent on the probabilities of the right hand side of the rule. In other words, the higher frequency right hand side will have a higher chance of being predicted according to the confidence rule.

An alternative to the confidence rule is using Lift. Weka's Class Based Association Rule is limited, not allowing Lift. Lift would allow for a better ranking in terms of dependence of the left hand side and the right hand side, producing a better more meaningful result.

The following 5 rules are chosen as the most meaningful and interesting results:

1. Non sterilized pets are adopted at an average rate (0.3)
2. If a pet does not have an associated video, it is not adopted (0.28)
3. Free, healthy pets are adopted at an average rate (0.27)
4. Free pets are adopted at an average rate (0.27)
5. Single pets are adopted at an average rate (0.27)

2.2 Attribute Selection

Attribute Selection selects the attributes that are best for a Decision Tree. PetAdoption uses Information Gain to select the best attributes for the decision tree, based on Adoption Speed as the class. The highest information gain variables are analysed first, as they provide the most information about the class. Entropy is a measure of uncertainty associated with a variable. To calculate the Information Gain for variable A:

$$Gain(A) = Info(D) - Info_A(D)$$

Where D is an arbitrary tuple that belongs to class C.
Weka uses InfoGainAttributeEval to evaluate the Information Gain for each attribute in the dataset.

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 17 AdoptionSpeed):
        Information Gain Ranking Filter

Ranked attributes:
 0.044185   2 Age
 0.038198  10 Sterilized
 0.018709  16 PhotoAmt
 0.014879  14 State
 0.013705   8 Vaccinated
 0.007897   7 FurLength
 0.007801   1 Type
 0.007201   6 MaturitySize
 0.006985   9 Dewormed
 0.003721   4 Gender
 0.002269  12 Quantity
 0.002025   3 BreedPurity
 0.001272  11 Health
 0.001249  15 VideoAmt
 0.000764   5 ColorPurity
 0.0007    13 Fee

Selected attributes: 2,10,16,14,8,7,1,6,9,4,12,3,11,15,5,13 : 16
```

Figure 14: Attribute Selection using Information Gain

As an example, the calculation of Information Gain for *Age* is included in Appendices, as it verifies the solution provided by Weka.

2.3 Decision Tree Model

Using every single attribute from the data set creates a huge amount of branches for a decision tree. For PetAdoption, this is the decision tree without attribute selection. The benefit to having no attribute selection is that the effect of every variable can be seen in the tree structure. However, as shown in the figure, the tree becomes extremely cluttered, and unreadable.

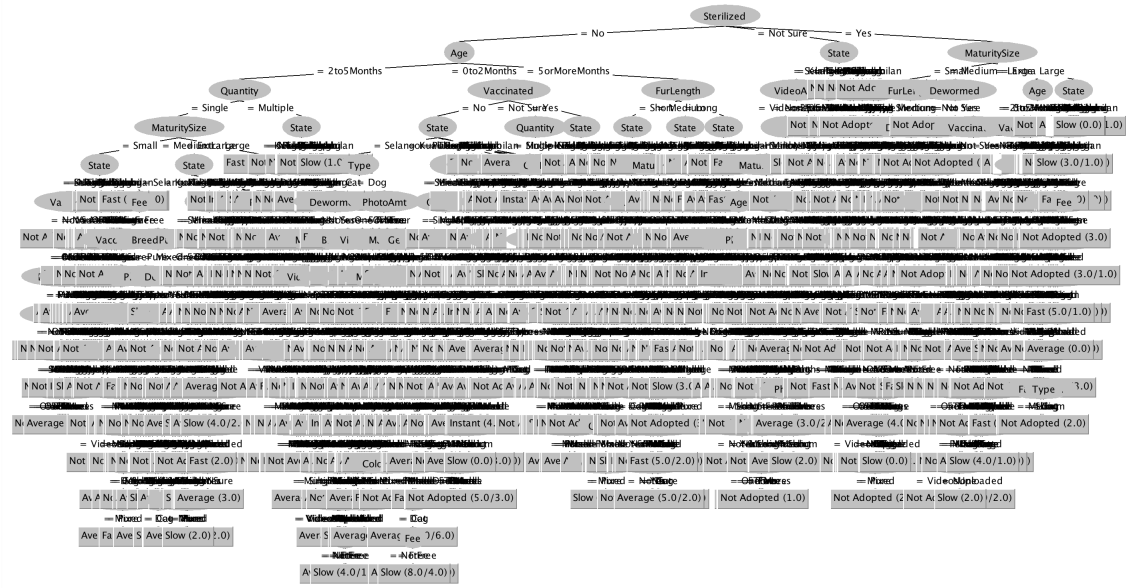


Figure 15: Decision Tree without Attribute Selection

Precision	Recall
0.315	0.333
0.091	0.037
0.296	0.231
0.307	0.271
0.434	0.548
0.337	0.350

Figure 16: Precision and Recall for No Attribute Selection

Due to using Information Gain, the higher the information gain, the more desirable the value is to the decision tree. Additionally, Information Gain heavily favours attributes with multiple variables. In PetAdoption, there are two attributes that have a large amount of variables. These are *State* and *FurLength*. By removing these attributes, and limiting the amount of attributes to the 5 highest Information Gain attributes (essentially selecting the right attributes to use in our decision tree), we are able to create a much more readable decision tree.

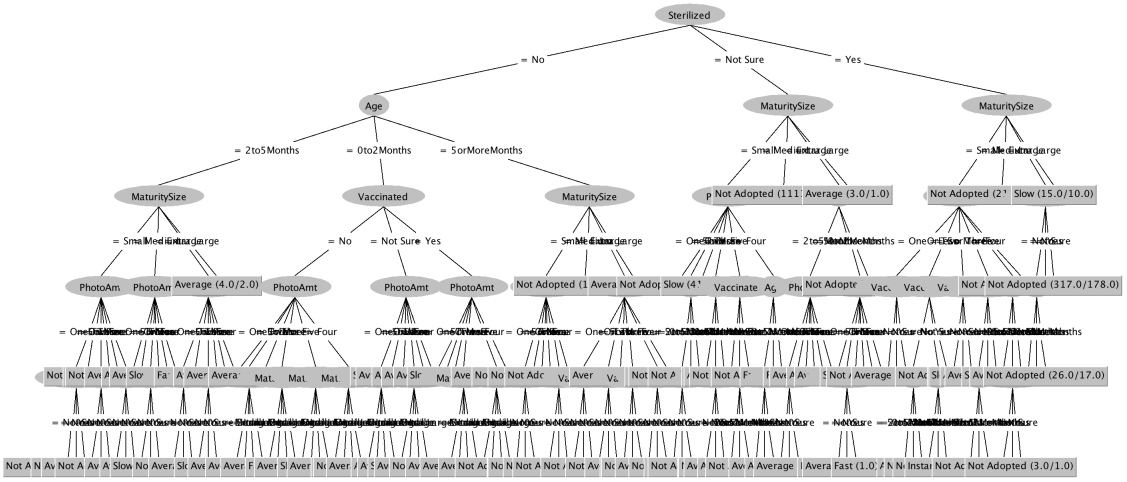


Figure 17: Decision Tree with Attribute Selection

Precision	Recall
0.323	0.482
0.250	0.005
0.376	0.120
0.276	0.132
0.416	0.638
0.349	0.362

Figure 18: Precision and Recall for Attribute Selection

Furthermore, the precision and recall for the attribute selection tree is very similar to the no attribute selection tree. This means that the attribute selection tree provides a very similar read as the no attribute selection tree, with the benefit of being a lot more readable and concise.

2.4 Clustering

Clustering allows the user to detect the intrinsic grouping of the data. However, even if there are groups in the data, this does not necessarily mean that it corresponds to a target variable.

Weka allows the grouping of data between attributes, however clustering visualization with multiple attributes is not meaningful. For PetAdoption, it is not possible to visualize all 17 attributes and get meaningful information, therefore as an example, two attributes have been chosen, *Sterilized* and *Vaccinated*. Using SimpleKMeans with two clusters, Weka produces the following graph.

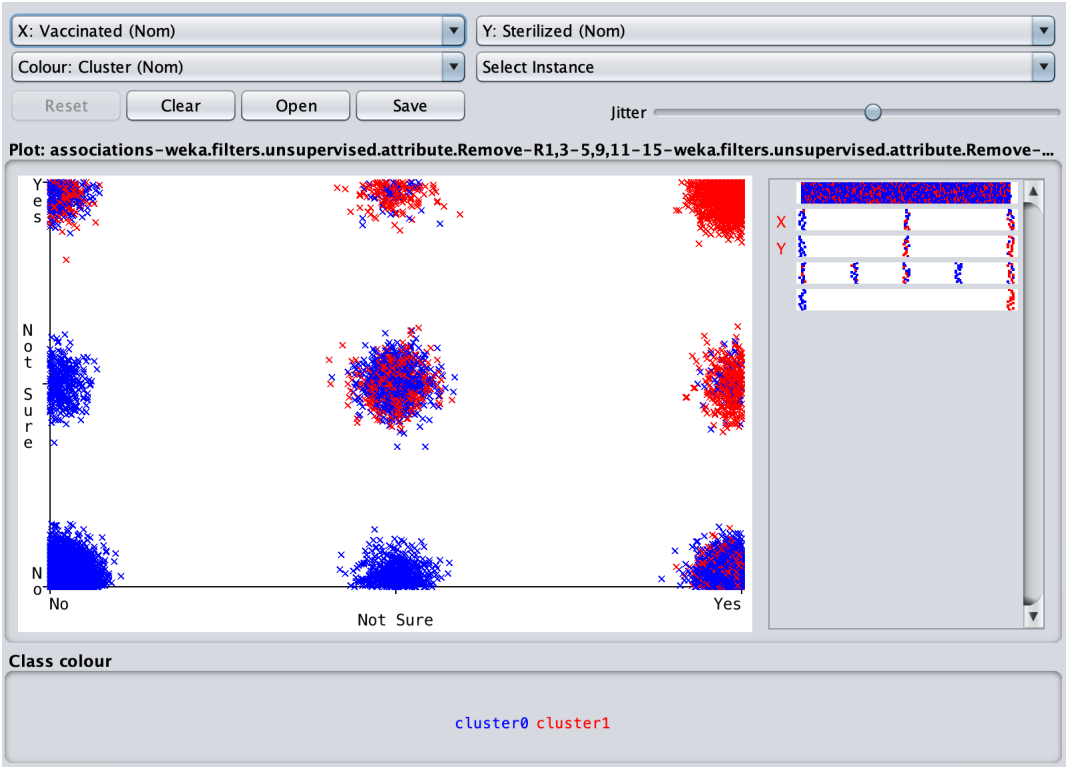


Figure 19: Vaccinated vs Sterilized Clusters

From these two clusters, two natural groups can be visualized. Firstly, non sterilized and non vaccinated cluster, and sterilized and vaccinated cluster. However, this cannot be used against the target variable, *AdoptionSpeed*, as it does not correspond.

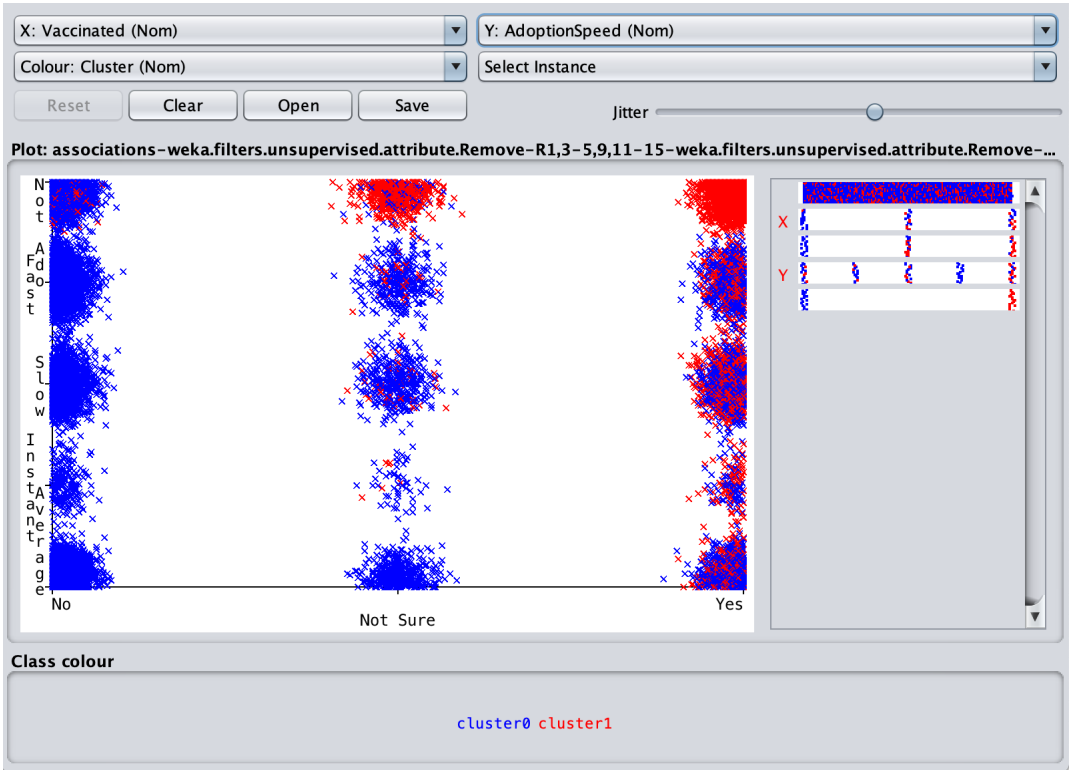


Figure 20: Vaccinated vs Adoption Speed Clusters

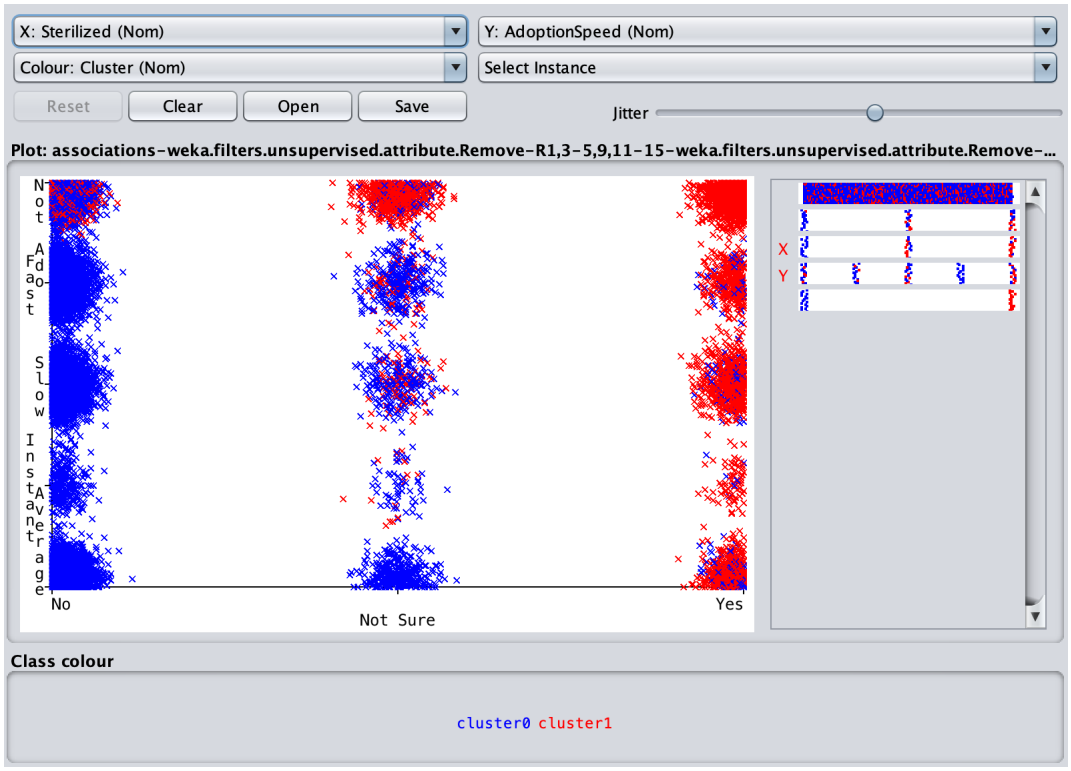


Figure 21: Sterilized vs Adoption Speed Clusters

Even though the visualization does not make sense, the clustering exercise is still carried out on the entire data set, and due to having five different adoption speeds, we have five different clusters.

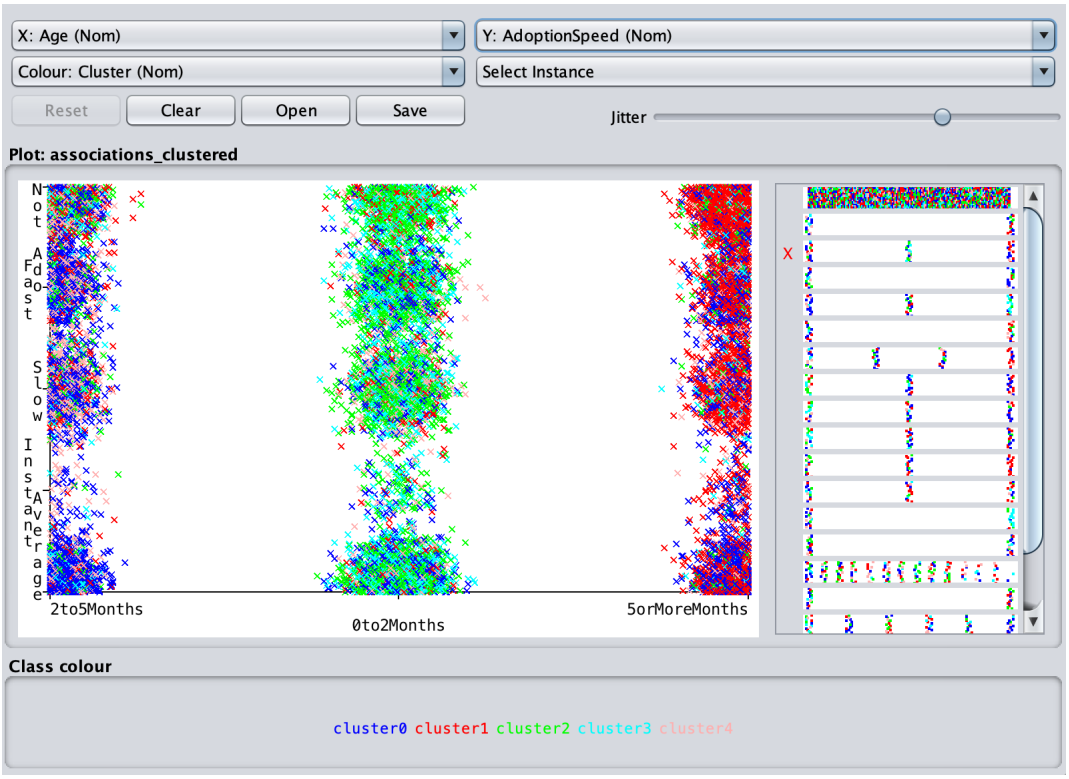


Figure 22: Entire data set cluster

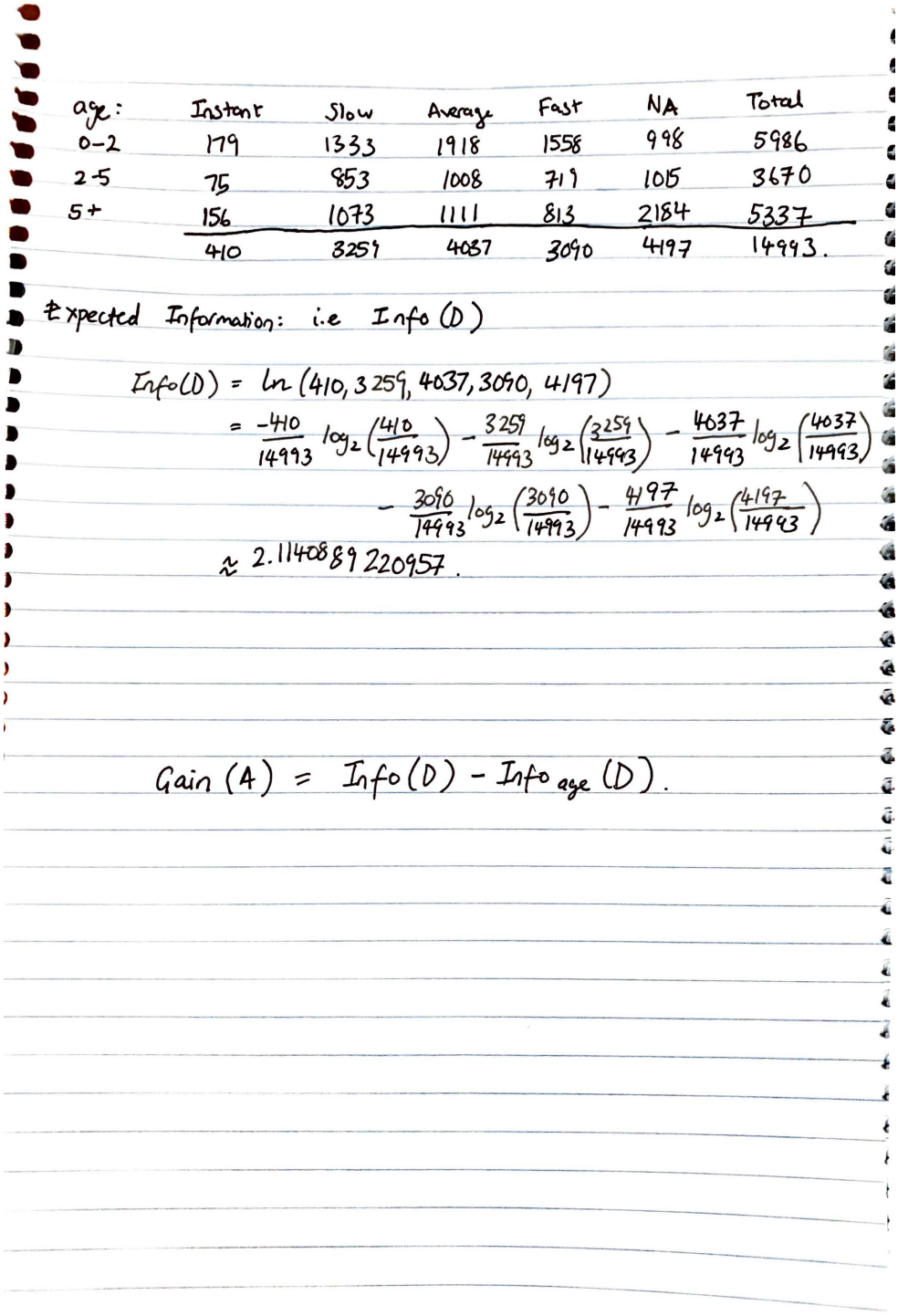
2.5 Learning Relation Context

Association rule mining and clustering fall into the category of unsupervised learning. Unsupervised learning finds patterns in a given data set, without a target variable. PetAdoption is able to find association rules using Apriori, which is independent of the target variable. Additionally, clustering is done through SimpleKMeans, again, independent of the target variable, as shown by the example of Sterilized and Vaccinated.

Supervised learning requires a target variable to train the model on. PetAdoption uses *AdoptionSpeed* as the target variable, and the decision tree, using J48, is able to eventually arrive at the adoption speed at each leaf node.

3 Appendix

3.1 Information Gain Calculation for Age



age:	Instant	Slow	Average	Fast	NA	Total
0-2	179	1333	1918	1558	998	5986
2-5	75	853	1008	719	105	3670
5+	156	1073	1111	813	2184	5337
	410	3259	4037	3090	4197	14993.

Expected Information: i.e Info (D)

$$Info(D) = ln(410, 3259, 4037, 3090, 4197)$$

$$= \frac{-410}{14993} \log_2\left(\frac{410}{14993}\right) - \frac{3259}{14993} \log_2\left(\frac{3259}{14993}\right) - \frac{4037}{14993} \log_2\left(\frac{4037}{14993}\right) - \frac{3090}{14993} \log_2\left(\frac{3090}{14993}\right) - \frac{4197}{14993} \log_2\left(\frac{4197}{14993}\right)$$

$$\approx 2.114088 9220957.$$

$$Gain(A) = Info(D) - Info_{age}(D).$$

Information needed after using age:

$= I(179, 1383, 1918, 1558, 998)$ . . . . . .

$\frac{5986}{14993}$ $\quad Info_{age}(D) = \frac{5986}{14993} I(0\text{-}2) + \frac{3670}{14993} I(2\text{-}5) + \frac{5337}{14993} I(5+)$.

$I(0\text{-}2)$:

$I(179, 1333, 1918, 1558, 998)$

$= -\frac{179}{5986}\log_2\left(\frac{179}{5986}\right) - \frac{1333}{5986}\log_2\left(\frac{1333}{5986}\right) - \frac{1918}{5986}\log_2\left(\frac{1918}{5986}\right)$

$\quad - \frac{1558}{5986}\log_2\left(\frac{1558}{5986}\right) - \frac{998}{5986}\log_2\left(\frac{998}{5986}\right)$

$= 2.096391397 78$

$I(2\text{-}5)$:

$I(75, 853, 1008, 719, 1015)$

$= -\frac{75}{3670}\log_2\left(\frac{75}{3670}\right) - \frac{853}{3670}\log_2\left(\frac{853}{3670}\right) - \frac{1008}{3670}\log_2\left(\frac{1008}{3670}\right)$

$\quad - \frac{719}{3670}\log_2\left(\frac{719}{3670}\right) - \frac{1015}{3670}\log_2\left(\frac{1015}{3670}\right)$

$= 2.089606836 0$

$I(5+)$:

$I(156, 1073, 1111, 813, 2184)$

$= -\frac{156}{5337}\log_2\left(\frac{156}{5337}\right) - \frac{1073}{5337}\log_2\left(\frac{1073}{5337}\right) - \frac{1111}{5337}\log_2\left(\frac{1111}{5337}\right)$

$\quad - \frac{813}{5337}\log_2\left(\frac{813}{5337}\right) - \frac{2184}{5337}\log_2\left(\frac{2184}{5337}\right)$.

$= 2.026646614 2$

$$\text{Info}_{age}(D) = \frac{5986}{14993} \times 2.096391397\,78 + \frac{3670}{14993} \times 2.089606836\,0 + \frac{537}{14993} \times 2.026646614\,2.$$

$$= 2.069903887.$$

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_{age}(D)$$

$$\therefore \text{Info gain}_{(A)} = 2.114088922\,0957 - 2.069903887$$

$$= 0.044185$$

= Exactly what Weka gives!