21BDS0340 - Abhinav Dinesh Srivatsa

Data Mining Lab

Digital Assignment 1

In [ ]:

```
import pandas as pd
import numpy as np
```

Question 1

In [ ]:

```
data = pd.read_csv("./missing data.csv")
data
```

Out[ ]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date
0	Abbotsford	85 Turner St	2	h	1480000	S	Biggin	03-12-2016
1	Abbotsford	25 Bloomburg St	2	h	1035000	S	Biggin	04-02-2016
2	Abbotsford	5 Charles St	3	h	1465000	SP	Biggin	04-03-2017
3	Abbotsford	40 Federation La	3	h	850000	PI	Biggin	04-03-2017
4	Abbotsford	55a Park St	4	h	1600000	VB	Nelson	04-06-2016
...	...	...	...	...	...	...	...	...
13575	Wheelers Hill	12 Strada Cr	4	h	1245000	S	Barry	26-08-2017
13576	Williamstown	77 Merrett Dr	3	h	1031000	SP	Williams	26-08-2017
13577	Williamstown	83 Power St	3	h	1170000	S	Raine	26-08-2017
13578	Williamstown	96 Verdon St	4	h	2500000	PI	Sweeney	26-08-2017
13579	Yarraville	6 Agnes St	4	h	1285000	SP	Village	26-08-2017

13580 rows × 21 columns

21BDS0340 - Abhinav Dinesh Srivatsa

Data Mining Lab

Digital Assignment 1

In [ ]:

```
# part a
data.isna().sum()
```

Out[ ]:

```
Suburb              0
Address             0
Rooms               0
Type                0
Price               0
Method              0
SellerG             0
Date                0
Distance            0
Postcode            0
Bedroom2            0
Bathroom            0
Car                62
Landsize            0
BuildingArea     6450
YearBuilt        5375
CouncilArea      1369
Lattitude           0
Longtitude          0
Regionname          0
Propertycount       0
dtype: int64
```

In [ ]:

```
# part b
data["Car"].fillna(data["Car"].mode()[0])
```

Out[ ]:

```
0        1.0
1        0.0
2        0.0
3        1.0
4        2.0
        ... 
13575    2.0
13576    2.0
13577    4.0
13578    5.0
13579    1.0
Name: Car, Length: 13580, dtype: float64
```

In [ ]:

```
# part c
data["BuildingArea"].interpolate("linear")
# first Nan cannot be replaced
```

Out[ ]:
```
0          NaN
1         79.0
2        150.0
3        146.0
4        142.0
         ...  
13575    146.0
13576    133.0
13577    145.0
13578    157.0
13579    112.0
Name: BuildingArea, Length: 13580, dtype: float64
```

In [ ]:
```
data["BuildingArea"].interpolate("quadratic")
# first Nan cannot be replaced
```

Out[ ]:
```
0               NaN
1         79.000000
2        150.000000
3        148.715056
4        142.000000
            ...    
13575    159.855594
13576    133.000000
13577    148.024068
13578    157.000000
13579    112.000000
Name: BuildingArea, Length: 13580, dtype: float64
```

In [ ]:
```
# part d
data["YearBuilt"].ffill()
```

Out[ ]:
```
0           NaN
1        1900.0
2        1900.0
3        1900.0
4        2014.0
          ...  
13575    1981.0
13576    1995.0
13577    1997.0
13578    1920.0
13579    1920.0
Name: YearBuilt, Length: 13580, dtype: float64
```

In [ ]:
```
# part e
data.dropna(subset=["CouncilArea"])
```

Out[ ]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Da
0	Abbotsford	85 Turner St	2	h	1480000	S	Biggin	0 1 20
1	Abbotsford	25 Bloomburg St	2	h	1035000	S	Biggin	0 0 20
2	Abbotsford	5 Charles St	3	h	1465000	SP	Biggin	0 0 20
3	Abbotsford	40 Federation La	3	h	850000	PI	Biggin	0 0 20
4	Abbotsford	55a Park St	4	h	1600000	VB	Nelson	0 0 20
...	...	...	...	...	...	...	...	
12208	Williamstown	87 Pasco St	3	h	1285000	S	Jas	2 0 20
12209	Windsor	201/152 Peel St	2	u	560000	PI	hockingstuart	2 0 20
12210	Wollert	60 Saltlake Bvd	3	h	525300	S	Stockdale	2 0 20
12211	Yarraville	2 Adeney St	2	h	750000	SP	hockingstuart	2 0 20
12212	Yarraville	54 Pentland Pde	6	h	2450000	VB	Village	2 0 20

12211 rows × 21 columns

Question 2

In [ ]:

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

data = [
    ['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
    ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
    ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
    ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
    ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']
]

te = TransactionEncoder()
te_ary = te.fit(data).transform(data)
```

```
dataframe = pd.DataFrame(te_ary, columns=te.columns_)
dataframe
```

Out[ ]:

	Apple	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Unicorn	Yogu
0	False	False	False	True	False	True	True	True	True	False	Tr
1	False	False	True	True	False	True	False	True	True	False	Tr
2	True	False	False	True	False	True	True	False	False	False	Fal
3	False	True	False	False	False	True	True	False	False	True	Tr
4	False	True	False	True	True	True	False	False	True	False	Fal

In [ ]:

```
# part a
min_support_threshold = 2 / len(data)
frequent_itemsets = apriori(dataframe, min_support=min_support_threshold,
frequent_itemsets['support_count'] = frequent_itemsets['support'] * len(d
```

In [ ]:

```
frequent_itemsets[frequent_itemsets['support_count'] == 4]
```

Out[ ]:

	support	itemsets	support_count
1	0.8	(Eggs)	4.0
8	0.8	(Eggs, Kidney Beans)	4.0

In [ ]:

```
frequent_itemsets[frequent_itemsets['support_count'] == 2]
```

Out[ ]:

	support	itemsets	support_count
0	0.4	(Corn)	2.0
4	0.4	(Nutmeg)	2.0
7	0.4	(Kidney Beans, Corn)	2.0
9	0.4	(Milk, Eggs)	2.0
10	0.4	(Eggs, Nutmeg)	2.0
12	0.4	(Eggs, Yogurt)	2.0
14	0.4	(Kidney Beans, Nutmeg)	2.0
17	0.4	(Milk, Yogurt)	2.0
18	0.4	(Nutmeg, Onion)	2.0
19	0.4	(Yogurt, Nutmeg)	2.0
20	0.4	(Yogurt, Onion)	2.0
21	0.4	(Milk, Eggs, Kidney Beans)	2.0
22	0.4	(Eggs, Kidney Beans, Nutmeg)	2.0
24	0.4	(Eggs, Kidney Beans, Yogurt)	2.0
25	0.4	(Eggs, Nutmeg, Onion)	2.0
26	0.4	(Eggs, Yogurt, Nutmeg)	2.0
27	0.4	(Eggs, Yogurt, Onion)	2.0
28	0.4	(Milk, Kidney Beans, Yogurt)	2.0
29	0.4	(Kidney Beans, Nutmeg, Onion)	2.0
30	0.4	(Kidney Beans, Yogurt, Nutmeg)	2.0
31	0.4	(Kidney Beans, Yogurt, Onion)	2.0
32	0.4	(Yogurt, Nutmeg, Onion)	2.0
33	0.4	(Eggs, Kidney Beans, Nutmeg, Onion)	2.0
34	0.4	(Eggs, Kidney Beans, Yogurt, Nutmeg)	2.0
35	0.4	(Eggs, Kidney Beans, Yogurt, Onion)	2.0
36	0.4	(Eggs, Yogurt, Nutmeg, Onion)	2.0
37	0.4	(Kidney Beans, Yogurt, Nutmeg, Onion)	2.0
38	0.4	(Kidney Beans, Yogurt, Onion, Eggs, Nutmeg)	2.0

In [ ]:

```
# part b
rules = association_rules(frequent_itemsets, metric="support", min_thresh
rules[rules['confidence'] >= 0.5]
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
1	(Corn)	(Kidney Beans)	0.4	1.0	0.4	1.000000	1.0
2	(Eggs)	(Kidney Beans)	0.8	1.0	0.8	1.000000	1.0
3	(Kidney Beans)	(Eggs)	1.0	0.8	0.8	0.800000	1.0
4	(Milk)	(Eggs)	0.6	0.8	0.4	0.666667	0.8
5	(Eggs)	(Milk)	0.8	0.6	0.4	0.500000	0.8
...	...	...	...	...	...	...	
194	(Eggs, Nutmeg)	(Kidney Beans, Yogurt, Onion)	0.4	0.4	0.4	1.000000	2.5
196	(Yogurt)	(Eggs, Kidney Beans, Nutmeg, Onion)	0.6	0.4	0.4	0.666667	1.6
197	(Onion)	(Eggs, Kidney Beans, Yogurt, Nutmeg)	0.6	0.4	0.4	0.666667	1.6
198	(Eggs)	(Kidney Beans, Yogurt, Nutmeg, Onion)	0.8	0.4	0.4	0.500000	1.2
199	(Nutmeg)	(Eggs, Kidney Beans, Yogurt, Onion)	0.4	0.4	0.4	1.000000	2.5

186 rows × 10 columns

In [ ]:

```
rules[rules['lift'] >= 1.0]
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
0	(Kidney Beans)	(Corn)	1.0	0.4	0.4	0.400000	1.0
1	(Corn)	(Kidney Beans)	0.4	1.0	0.4	1.000000	1.0
2	(Eggs)	(Kidney Beans)	0.8	1.0	0.8	1.000000	1.0
3	(Kidney Beans)	(Eggs)	1.0	0.8	0.8	0.800000	1.0
6	(Eggs)	(Nutmeg)	0.8	0.4	0.4	0.500000	1.2
...	...	...	...	...	...	...	
195	(Kidney Beans)	(Eggs, Yogurt, Nutmeg, Onion)	1.0	0.4	0.4	0.400000	1.0
196	(Yogurt)	(Eggs, Kidney Beans, Nutmeg, Onion)	0.6	0.4	0.4	0.666667	1.6
197	(Onion)	(Eggs, Kidney Beans, Yogurt, Nutmeg)	0.6	0.4	0.4	0.666667	1.6
198	(Eggs)	(Kidney Beans, Yogurt, Nutmeg, Onion)	0.8	0.4	0.4	0.500000	1.2
199	(Nutmeg)	(Eggs, Kidney Beans, Yogurt, Onion)	0.4	0.4	0.4	1.000000	2.5

188 rows × 10 columns

Question 3

In [ ]:

```
dataset = [
    ['Milk', 'Onion', np.nan, 'Kidney Beans', 'Eggs', 'Yogurt'],
    ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],
    ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
    ['Milk', 'Unicorn', 'Corn', np.nan, 'Yogurt'],
    ['Corn', 'Onion', np.nan, 'Kidney Beans', 'Ice cream', 'Eggs']
]
dataset = [[item if pd.notna(item) else 'nan' for item in transaction] fo

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
```

```
dataframe = pd.DataFrame(te_ary, columns=te.columns_)
dataframe
```

Out[ ]:

	Apple	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Unicorn	Yogu
0	False	False	False	True	False	True	True	False	True	False	Tr
1	False	False	True	True	False	True	False	True	True	False	Tr
2	True	False	False	True	False	True	True	False	False	False	Fal
3	False	True	False	False	False	False	True	False	False	True	Tr
4	False	True	False	True	True	True	False	False	True	False	Fal

In [ ]:

```
min_support_threshold = 2 / len(dataset)
frequent_itemsets = apriori(dataframe, min_support=min_support_threshold,
frequent_itemsets['support_count'] = frequent_itemsets['support'] * len(d
```

In [ ]:

```
rules = association_rules(frequent_itemsets, metric="support", min_thresh
rules
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
0	(nan)	(Corn)	0.6	0.4	0.4	0.666667	1.6
1	(Corn)	(nan)	0.4	0.6	0.4	1.000000	1.6
2	(Eggs)	(Kidney Beans)	0.8	0.8	0.8	1.000000	1.2
3	(Kidney Beans)	(Eggs)	0.8	0.8	0.8	1.000000	1.2
4	(Milk)	(Eggs)	0.6	0.8	0.4	0.666667	0.8
...	...	...	...	...	...	...	
107	(nan, Onion)	(Eggs, Kidney Beans)	0.4	0.8	0.4	1.000000	1.2
108	(Kidney Beans)	(Eggs, nan, Onion)	0.8	0.4	0.4	0.500000	1.2
109	(Eggs)	(nan, Kidney Beans, Onion)	0.8	0.4	0.4	0.500000	1.2
110	(nan)	(Eggs, Kidney Beans, Onion)	0.6	0.6	0.4	0.666667	1
111	(Onion)	(Eggs, Kidney Beans, nan)	0.6	0.4	0.4	0.666667	1.6

112 rows × 10 columns

In [ ]:

```
rules[rules['confidence'] >= 0.5]
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
0	(nan)	(Corn)	0.6	0.4	0.4	0.666667	1.6
1	(Corn)	(nan)	0.4	0.6	0.4	1.000000	1.6
2	(Eggs)	(Kidney Beans)	0.8	0.8	0.8	1.000000	1.2
3	(Kidney Beans)	(Eggs)	0.8	0.8	0.8	1.000000	1.2
4	(Milk)	(Eggs)	0.6	0.8	0.4	0.666667	0.8
...	...	...	...	...	...	...	
107	(nan, Onion)	(Eggs, Kidney Beans)	0.4	0.8	0.4	1.000000	1.2
108	(Kidney Beans)	(Eggs, nan, Onion)	0.8	0.4	0.4	0.500000	1.2
109	(Eggs)	(nan, Kidney Beans, Onion)	0.8	0.4	0.4	0.500000	1.2
110	(nan)	(Eggs, Kidney Beans, Onion)	0.6	0.6	0.4	0.666667	1
111	(Onion)	(Eggs, Kidney Beans, nan)	0.6	0.4	0.4	0.666667	1.6

112 rows × 10 columns

In [ ]:

```
rules[rules['lift'] >= 1.0]
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
0	(nan)	(Corn)	0.6	0.4	0.4	0.666667	1.6
1	(Corn)	(nan)	0.4	0.6	0.4	1.000000	1.6
2	(Eggs)	(Kidney Beans)	0.8	0.8	0.8	1.000000	1.2
3	(Kidney Beans)	(Eggs)	0.8	0.8	0.8	1.000000	1.2
6	(Eggs)	(Onion)	0.8	0.6	0.6	0.750000	1.2
...	...	...	...	...	...	...	
107	(nan, Onion)	(Eggs, Kidney Beans)	0.4	0.8	0.4	1.000000	1.2
108	(Kidney Beans)	(Eggs, nan, Onion)	0.8	0.4	0.4	0.500000	1.2
109	(Eggs)	(nan, Kidney Beans, Onion)	0.8	0.4	0.4	0.500000	1.2
110	(nan)	(Eggs, Kidney Beans, Onion)	0.6	0.6	0.4	0.666667	1
111	(Onion)	(Eggs, Kidney Beans, nan)	0.6	0.4	0.4	0.666667	1.6

94 rows × 10 columns