Co-Occurance Of Procedures:

Data Collection:

1. Kaggle DataSet: https://www.kaggle.com/datasets/cms/cms-codes

Since, The Kaggle dataset was old and the procedures were only for very few IDs. I made a new data set with the help of ChatGPT.

2. Used Dataset: https://drive.google.com/file/d/1_7UMp2_n6bf6KRwJUfWhMTpzq5DiNQSM/view?usp=sharing

Preview of dataset used:

	PatientID	TestName
0	1	Blood test
1	1	X-ray
2	1	ECG
3	1	Allergy test
4	1	Stool sample analysis

Insights of dataset:

DataTypes:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 139 entries, 0 to 138
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   PatientID  139 non-null    int64
 1   TestName   139 non-null    object
dtypes: int64(1), object(1)
memory usage: 2.3+ KB
```

Null Value check:

```
raw_data.isna().sum()
```

```
PatientID    0
TestName     0
dtype: int64
```

Algorithm Used:

Apriori Algorithm with some use cases:

The Apriori algorithm is a classic algorithm used in data mining and association rule learning. It is designed to discover frequent itemsets in a dataset and generate association rules based on those itemsets. The algorithm is primarily used for market basket analysis and can provide valuable insights into purchasing patterns, customer behavior, and product recommendations. Here's a detailed explanation of the Apriori algorithm and some of its use cases:

1. Algorithm Overview:

    - Step 1: Generate a set of frequent 1-itemsets by scanning the transaction database and counting the occurrences of each item.

- Step 2: Join the frequent 1-itemsets to form candidate 2-itemsets and scan the database again to count their occurrences.
- Step 3: Prune the candidate 2-itemsets by removing those that do not meet the minimum support threshold.
- Step 4: Repeat steps 2 and 3 to generate frequent k-itemsets until no new frequent itemsets can be found.
- Step 5: Generate association rules from the frequent itemsets and calculate their confidence.

2. Use Cases:

   a. Market Basket Analysis: One of the primary applications of the Apriori algorithm is in market basket analysis. It helps retailers and e-commerce platforms identify frequently co-occurring items in customer transactions. By discovering associations between products, retailers can optimize product placement, improve cross-selling and upselling strategies, and provide personalized product recommendations.

   b. Customer Behavior Analysis: The Apriori algorithm can be used to analyze customer behavior in various industries, such as telecommunications, banking, and insurance. By identifying patterns in customer interactions, companies can gain insights into cross-product usage, churn prediction, customer segmentation, and targeted marketing campaigns.

   c. Web Usage Mining: In web analytics, the Apriori algorithm can be applied to understand browsing behavior and discover associations between web pages or content items. This information can be used to improve website navigation, personalize content recommendations, and optimize advertising placements.

   d. Healthcare Data Analysis: The Apriori algorithm can be employed in healthcare to analyze patient records, identify co-occurring medical conditions, and discover patterns related to disease diagnosis and treatment. This can assist in medical decision-making, disease surveillance, and healthcare resource planning.

   e. Fraud Detection: By analyzing transaction data, the Apriori algorithm can help detect unusual patterns or associations that may indicate fraudulent activities. It can be used in financial institutions to identify potential fraud cases, such as credit card fraud, money laundering, or insurance fraud.

f. Supply Chain Optimization: The Apriori algorithm can analyze supply chain data to identify associations between different products, suppliers, or transportation routes. This information can be utilized to optimize inventory management, streamline logistics, and reduce costs.

Code:

Code Notebook as PDF LINK:

https://drive.google.com/file/d/1O91xDTuHROGH8qzzA-RiswX6kXoNJLkX/view?usp=sharing

1 Importing Required Libraries

[1]:
```
import pandas as pd
```

[2]:
```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

[3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

2 Loading raw data and checking for datatype and null values

[4]:
```
raw_data= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
↪Co-Occurance_of_procedures/data.csv')
```

[5]:
```
raw_data.head()
```

[5]:
```
   PatientID               TestName
0          1             Blood test
1          1                  X-ray
2          1                    ECG
3          1           Allergy test
4          1  Stool sample analysis
```

[6]:
```
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 139 entries, 0 to 138
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   PatientID  139 non-null    int64 
 1   TestName   139 non-null    object
dtypes: int64(1), object(1)
memory usage: 2.3+ KB
```

[7]:
```
raw_data.isna().sum()
```

[7]:
```
PatientID    0
TestName     0
dtype: int64
```

3 Pre-Processing the data for the association algorithm

[8]:
```
grouped_df = raw_data.groupby('PatientID')['TestName'].apply(lambda x: x.values.
  ↪tolist())
```

[9]:
```
procedures=[]
for group in grouped_df:
    procedures.append(group)

print(procedures)
```

```
[['Blood test', 'X-ray', 'ECG', 'Allergy test', 'Stool sample analysis'],
['Urine test', 'MRI scan', 'Biopsy'], ['CT scan', 'ECG', 'Colonoscopy', 'Pap
smear', 'Bone density test', 'Stool sample analysis', 'Blood test'],
['Mammogram', 'HIV test'], ['X-ray', 'Pulmonary function test', 'Biopsy', 'Urine
test'], ['HIV test'], ['ECG', 'Blood test', 'CT scan', 'X-ray', 'Colonoscopy',
'Mammogram'], ['Stool sample analysis', 'Bone density test', 'Urine test',
'ECG', 'MRI scan', 'Pap smear', 'Allergy test', 'Blood test', 'Colonoscopy'],
['Pulmonary function test', 'Biopsy', 'Mammogram'], ['Bone density test', 'MRI
scan', 'Stool sample analysis', 'Allergy test', 'X-ray'], ['ECG', 'Blood test'],
['CT scan', 'Urine test', 'Colonoscopy', 'Bone density test', 'HIV test', 'Stool
sample analysis', 'Pap smear', 'Mammogram'], ['MRI scan', 'Urine test', 'Allergy
test', 'X-ray'], ['ECG', 'Blood test', 'CT scan', 'Colonoscopy', 'Mammogram',
'Pap smear'], ['Pulmonary function test', 'Biopsy', 'HIV test'], ['CT scan',
'Pap smear', 'Bone density test', 'Mammogram'], ['X-ray', 'Blood test', 'MRI
scan', 'Allergy test', 'Colonoscopy', 'Stool sample analysis', 'ECG'], ['Urine
test', 'HIV test'], ['Pulmonary function test', 'Biopsy', 'Urine test', 'ECG',
'Pap smear', 'Mammogram'], ['CT scan', 'Bone density test', 'Blood test'], ['MRI
scan', 'Allergy test', 'X-ray', 'Colonoscopy', 'Stool sample analysis'], ['HIV
test'], ['ECG', 'Blood test', 'CT scan', 'Colonoscopy', 'Mammogram', 'Pap
smear', 'Bone density test', 'Stool sample analysis'], ['Pulmonary function
test', 'Biopsy', 'Urine test'], ['Mammogram', 'HIV test', 'X-ray', 'Allergy
test', 'CT scan'], ['Bone density test', 'MRI scan'], ['ECG', 'Blood test', 'CT
scan', 'Colonoscopy', 'Mammogram', 'Pap smear'], ['Urine test', 'Allergy test',
'X-ray', 'Stool sample analysis'], ['Pulmonary function test', 'Biopsy',
'Mammogram'], ['ECG', 'Blood test', 'CT scan', 'Colonoscopy', 'Mammogram', 'Pap
smear', 'Bone density test'], ['MRI scan', 'Urine test', 'Allergy test',
'X-ray', 'Colonoscopy']]
```

[10]:
```
# Transaction encoding
te = TransactionEncoder()
te_array = te.fit_transform(procedures)
df = pd.DataFrame(te_array, columns=te.columns_)
```

4 Implementing the algorithm

[11]:
```
# Applying Apriori algorithm
frequent_itemsets = apriori(df, min_support=0.2, use_colnames=True)
```

[12]:
```
# Generating association rules
rules = association_rules(frequent_itemsets, metric='confidence',␣
 ↪min_threshold=0.5)
```

[13]:
```
# Printing frequent itemsets
print("Frequent Itemsets:")
print(frequent_itemsets)
```

```
Frequent Itemsets:
     support                        itemsets
0   0.290323                  (Allergy test)
1   0.225806                        (Biopsy)
2   0.354839                    (Blood test)
3   0.290323             (Bone density test)
4   0.322581                       (CT scan)
5   0.354839                   (Colonoscopy)
6   0.354839                           (ECG)
7   0.225806                      (HIV test)
8   0.258065                      (MRI scan)
9   0.387097                     (Mammogram)
10  0.290323                     (Pap smear)
11  0.290323         (Stool sample analysis)
12  0.322581                    (Urine test)
13  0.322581                         (X-ray)
14  0.258065           (X-ray, Allergy test)
15  0.225806           (Blood test, CT scan)
16  0.258065       (Colonoscopy, Blood test)
17  0.322581               (Blood test, ECG)
18  0.225806          (Colonoscopy, CT scan)
19  0.258065            (Mammogram, CT scan)
20  0.225806            (CT scan, Pap smear)
21  0.258065              (Colonoscopy, ECG)
22  0.225806        (Colonoscopy, Pap smear)
23  0.225806                (ECG, Pap smear)
24  0.225806          (Mammogram, Pap smear)
25  0.258065  (Colonoscopy, Blood test, ECG)
```

[14]:
```python
# Printing association rules
print("\nAssociation Rules:")
print(rules)
```

```
Association Rules:
                  antecedents                 consequents  antecedent support  \
0                     (X-ray)              (Allergy test)            0.322581
1              (Allergy test)                     (X-ray)            0.290323
2                (Blood test)                   (CT scan)            0.354839
3                   (CT scan)                (Blood test)            0.322581
4               (Colonoscopy)                (Blood test)            0.354839
5                (Blood test)               (Colonoscopy)            0.354839
6                (Blood test)                       (ECG)            0.354839
7                       (ECG)                (Blood test)            0.354839
8               (Colonoscopy)                   (CT scan)            0.354839
9                   (CT scan)               (Colonoscopy)            0.322581
10                (Mammogram)                   (CT scan)            0.387097
11                  (CT scan)                 (Mammogram)            0.322581
12                  (CT scan)                 (Pap smear)            0.322581
13                (Pap smear)                   (CT scan)            0.290323
14              (Colonoscopy)                       (ECG)            0.354839
15                      (ECG)               (Colonoscopy)            0.354839
16              (Colonoscopy)                 (Pap smear)            0.354839
17                (Pap smear)               (Colonoscopy)            0.290323
18                      (ECG)                 (Pap smear)            0.354839
19                (Pap smear)                       (ECG)            0.290323
20                (Mammogram)                 (Pap smear)            0.387097
21                (Pap smear)                 (Mammogram)            0.290323
22  (Colonoscopy, Blood test)                       (ECG)            0.258065
23         (Colonoscopy, ECG)                (Blood test)            0.258065
24          (Blood test, ECG)               (Colonoscopy)            0.322581
25              (Colonoscopy)           (Blood test, ECG)            0.354839
26               (Blood test)          (Colonoscopy, ECG)            0.354839
27                      (ECG)   (Colonoscopy, Blood test)            0.354839

    consequent support   support  confidence      lift  leverage  conviction
0             0.290323  0.258065    0.800000  2.755556  0.164412    3.548387
1             0.322581  0.258065    0.888889  2.755556  0.164412    6.096774
2             0.322581  0.225806    0.636364  1.972727  0.111342    1.862903
3             0.354839  0.225806    0.700000  1.972727  0.111342    2.150538
4             0.354839  0.258065    0.727273  2.049587  0.132154    2.365591
5             0.354839  0.258065    0.727273  2.049587  0.132154    2.365591
6             0.354839  0.322581    0.909091  2.561983  0.196670    7.096774
7             0.354839  0.322581    0.909091  2.561983  0.196670    7.096774
8             0.322581  0.225806    0.636364  1.972727  0.111342    1.862903
9             0.354839  0.225806    0.700000  1.972727  0.111342    2.150538
```

10	0.322581	0.258065	0.666667	2.066667	0.133195	2.032258
11	0.387097	0.258065	0.800000	2.066667	0.133195	3.064516
12	0.290323	0.225806	0.700000	2.411111	0.132154	2.365591
13	0.322581	0.225806	0.777778	2.411111	0.132154	3.048387
14	0.354839	0.258065	0.727273	2.049587	0.132154	2.365591
15	0.354839	0.258065	0.727273	2.049587	0.132154	2.365591
16	0.290323	0.225806	0.636364	2.191919	0.122789	1.951613
17	0.354839	0.225806	0.777778	2.191919	0.122789	2.903226
18	0.290323	0.225806	0.636364	2.191919	0.122789	1.951613
19	0.354839	0.225806	0.777778	2.191919	0.122789	2.903226
20	0.290323	0.225806	0.583333	2.009259	0.113424	1.703226
21	0.387097	0.225806	0.777778	2.009259	0.113424	2.758065
22	0.354839	0.258065	1.000000	2.818182	0.166493	inf
23	0.354839	0.258065	1.000000	2.818182	0.166493	inf
24	0.354839	0.258065	0.800000	2.254545	0.143600	3.225806
25	0.322581	0.258065	0.727273	2.254545	0.143600	2.483871
26	0.258065	0.258065	0.727273	2.818182	0.166493	2.720430
27	0.258065	0.258065	0.727273	2.818182	0.166493	2.720430

5 Output

[15]:
```
output_df=rules[['antecedents','consequents','conviction']]
```

[16]:
```
output_df.head(27)
```

[16]:

	antecedents	consequents	conviction
0	(X-ray)	(Allergy test)	3.548387
1	(Allergy test)	(X-ray)	6.096774
2	(Blood test)	(CT scan)	1.862903
3	(CT scan)	(Blood test)	2.150538
4	(Colonoscopy)	(Blood test)	2.365591
5	(Blood test)	(Colonoscopy)	2.365591
6	(Blood test)	(ECG)	7.096774
7	(ECG)	(Blood test)	7.096774
8	(Colonoscopy)	(CT scan)	1.862903
9	(CT scan)	(Colonoscopy)	2.150538
10	(Mammogram)	(CT scan)	2.032258
11	(CT scan)	(Mammogram)	3.064516
12	(CT scan)	(Pap smear)	2.365591
13	(Pap smear)	(CT scan)	3.048387
14	(Colonoscopy)	(ECG)	2.365591
15	(ECG)	(Colonoscopy)	2.365591
16	(Colonoscopy)	(Pap smear)	1.951613
17	(Pap smear)	(Colonoscopy)	2.903226
18	(ECG)	(Pap smear)	1.951613
19	(Pap smear)	(ECG)	2.903226
20	(Mammogram)	(Pap smear)	1.703226

21	(Pap smear)	(Mammogram)	2.758065
22	(Colonoscopy, Blood test)	(ECG)	inf
23	(Colonoscopy, ECG)	(Blood test)	inf
24	(Blood test, ECG)	(Colonoscopy)	3.225806
25	(Colonoscopy)	(Blood test, ECG)	2.483871
26	(Blood test)	(Colonoscopy, ECG)	2.720430

6 Inference

[17]:
```
def tests_after(test_string):
  after_tests=''
  for i in output_df['antecedents']:
    if test_string==next(iter(i)):
      rows=output_df[output_df['antecedents']==i]
      break

  max_conviction =rows['conviction'].max()

  after_frozenset=rows['consequents'].loc[rows['conviction']==max_conviction].
↪values
  after_list=list(after_frozenset[0])
  for tests in after_list:
    after_tests+=f' {tests}'

    return (f"The patient takes {after_tests} test after {test_string}")
```

[18]:
```
def tests_before(test_string):
  before_tests=''
  for i in output_df['consequents']:
    if test_string==next(iter(i)):
      rows=output_df[output_df['consequents']==i]
      break

  max_conviction =rows['conviction'].max()

  before_frozenset=rows['antecedents'].loc[rows['conviction']==max_conviction].
↪values
  before_list=list(before_frozenset[0])
  for tests in before_list:
    before_tests+=f' {tests}'

  return (f"The patient takes {before_tests} test before {test_string}")
```

[19]:
```
def procedures_predict(test_name):
  print(f'{tests_before(test_name)}'+' and '+f'{tests_after(test_name)}')
```

```
procedures_predict('CT scan')
```

```
The patient takes  Pap smear test before CT scan and The patient takes
Mammogram test after CT scan
```

Output:

Some Outputs:

```
def procedures_predict(test_name):
  print(f'{tests_before(test_name)}'+' and '+f'{tests_after(test_name)}')


procedures_predict('CT scan')
```

```
The patient takes  Pap smear test before CT scan and The patient takes  Mammogram test after CT scan
```

```
def procedures_predict(test_name):
  print(f'{tests_before(test_name)}'+' and '+f'{tests_after(test_name)}')


procedures_predict('Colonoscopy')
```

```
The patient takes  Blood test ECG test before Colonoscopy and The patient takes  Blood test test after Colonoscopy
```

Link To the Notebook in Google Colab:

https://colab.research.google.com/drive/1_YEdhvFNZT5zks_zdAploVeHmN4ptlnG?usp=sharing

Conclusion:

The procedures are selected according to the antecedents and consequents pairs generated by the Apriori Association Algorithm. The before and after procedures from a test-taken is selected according to the max conviction value provided by the Apriori Algorithm.