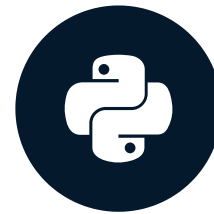# What is market basket analysis?
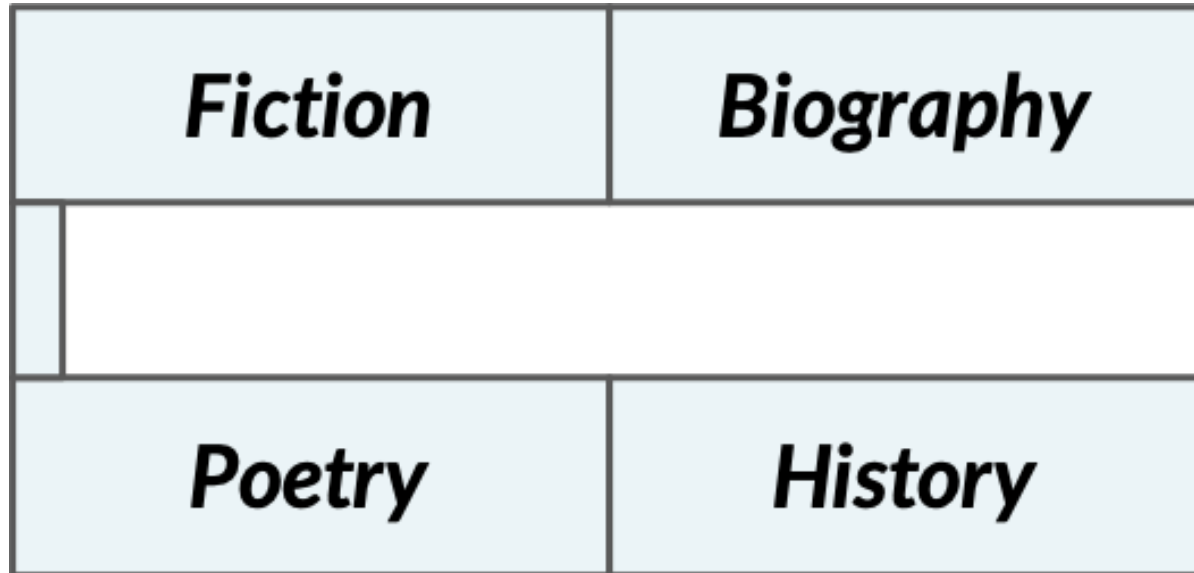
MARKET BASKET ANALYSIS IN PYTHON

Isaiah Hull
Visiting Associate Professor of Finance,
BI Norwegian Business School

Selecting a bookstore layout

Fiction	Biography
Poetry	History

Fiction	Poetry
Biography	History

Exploring transaction data

TID	Transaction
1	biography, history
2	fiction
3	biography, poetry
4	fiction, history
5	biography
...	...
75000	fiction, poetry

- **TID** = unique ID associated with each transaction.

- **Transaction** = set of unique items purchased together.

What is market basket analysis?

1. **Identify products frequently purchased together.**
   - Biography and history

   - Fiction and poetry

2. **Construct recommendations based on these findings.**
   - Place biography and history sections together.

   - Keep fiction and history apart.

The use cases of market basket analysis

1. Build Netflix-style recommendations engine.

2. Improve product recommendations on an e-commerce store.

3. Cross-sell products in a retail setting.

4. Improve inventory management.

5. Upsell products.

Using market basket analysis

TID	Transaction
11	**fiction, biography**
12	**fiction, biography**
13	history, biography
...	...
19	**fiction, biography**
20	**fiction, biography**
...	...

- **Market basket analysis**
  - Construct association rules
  - Identify items frequently purchased together
- **Association rules**
  - {antecedent} $\rightarrow$ {consequent}
    - {fiction} $\rightarrow$ {biography}

Loading the data

```python
import pandas as pd

# Load transactions from pandas.
books = pd.read_csv("datasets/bookstore.csv")


# Print the header
print(books.head(2))
```

```
TID        Transaction
0    biography, history
1              fiction
```

For a refresher, see the Pandas Cheat Sheet.

Building transactions

```python
# Split transaction strings into lists.
transactions = books['Transaction'].apply(lambda t: t.split(','))


# Convert DataFrame into list of strings.
transactions = list(transactions)
```

Counting the itemsets

```python
# Print the first transaction.
print(transactions[0])
```

```
['biography', 'history']
```

```python
# Count the number of transactions that contain biography and fiction.
transactions.count(['biography', 'fiction'])
```

```
218
```

Making a recommendation

```python
# Count the number of transactions that contain fiction and poetry.
transactions.count(['fiction', 'poetry'])
```
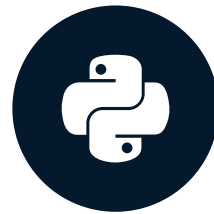
```
5357
```

Fiction	Poetry
Biography	History

Let's practice!

MARKET BASKET ANALYSIS IN PYTHON

Identifying association rules

MARKET BASKET ANALYSIS IN PYTHON

Isaiah Hull

Visiting Associate Professor of Finance,
BI Norwegian Business School

Loading and preparing data

```python
import pandas as pd


# Load transactions from pandas.
books = pd.read_csv("datasets/bookstore.csv")


# Split transaction strings into lists.
transactions = books['Transaction'].apply(lambda t: t.split(','))


# Convert DataFrame into list of strings.
transactions = list(transactions)
```

Exploring the data

```python
print(transactions[:5])
```

```
[['language', 'travel', 'humor', 'fiction'],
 ['humor', 'language'],
 ['humor', 'biography', 'cooking'],
 ['cooking', 'language'],
 ['travel']]
```

Association rules

- **Association rule**
  - Contains antecedent and consequent
    - *{health} → {cooking}*

- **Multi-antecedent rule**
  - *{humor, travel} → {language}*

- **Multi-consequent rule**
  - *{biography} → {history, language}*

Difficulty of selecting rules

- **Finding useful rules is difficult.**
  - Set of all possible rules is large.

  - Most rules are not useful.

  - Must discard most rules.

- **What if we restrict ourselves to simple rules?**
  - One antecedent and one consequent.

  - Still challenging, even for small dataset.

Generating the rules

- **fiction**

- **poetry**

- **history**

- **biography**

- **cooking**

- **health**

- **travel**

- **language**

- **humor**

Generating the rules

Fiction Rules	Poetry Rules	...	Humor Rules
fiction->poetry	poetry->fiction	...	humor->fiction
fiction->history	poetry->history	...	humor->history
fiction->biography	poetry->biography	...	humor->biography
fiction->cooking	poetry->cooking	...	humor->cooking
...	...	...	...
fiction->humor	poetry->humor	...	

Generating rules with itertools

```python
from itertools import permutations

# Extract unique items.
flattened = [item for transaction in transactions for item in transaction]
items = list(set(flattened))
```
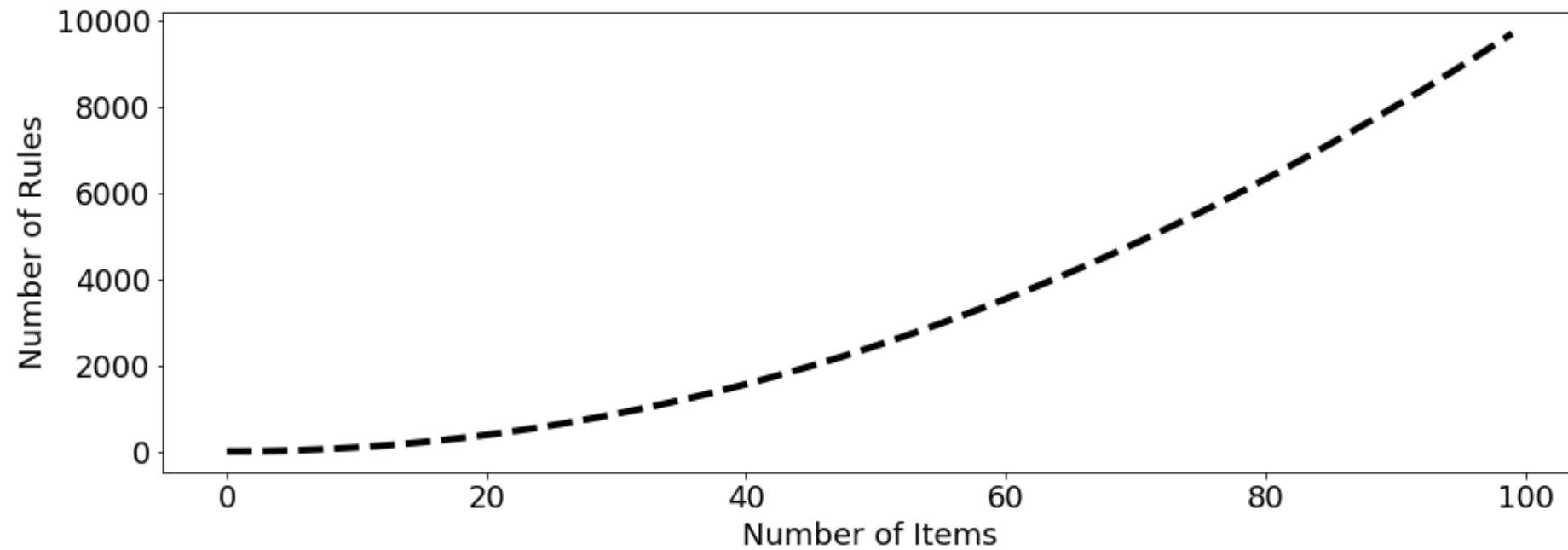
```python
# Compute and print rules.
rules = list(permutations(items, 2))
print(rules)
```

```
[('fiction', 'poetry'),
 ('fiction', 'history'),
 ...
 ('humor', 'travel'),
 ('humor', 'language')]
```

Counting the rules

```python
# Print the number of rules
print(len(rules))
```

```
72
```

Looking ahead

```python
# Import the association rules function
from mlxtend.frequent_patterns import association_rules
from mlxtend.frequent_patterns import apriori


# Compute frequent itemsets using the Apriori algorithm
frequent_itemsets = apriori(onehot, min_support = 0.001,
                                    max_len = 2, use_colnames = True)


# Compute all association rules for frequent_itemsets
rules = association_rules(frequent_itemsets,
                                metric = "lift",
                                 min_threshold = 1.0)
```
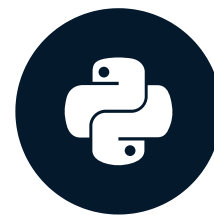
Let's practice!

MARKET BASKET ANALYSIS IN PYTHON

The simplest metric

MARKET BASKET ANALYSIS IN PYTHON

Isaiah Hull

Visiting Associate Professor of Finance,
BI Norwegian Business School

datacamp

Metrics and pruning

- A **metric** is a measure of performance for rules.
  - {humor} $\to$ {poetry}
    - 0.81

  - {fiction} $\to$ {travel}
    - 0.23

- **Pruning** is the use of metrics to discard rules.
  - Retain: {humor} $\to$ {poetry}

  - Discard: {fiction} $\to$ {travel}

The simplest metric

- The **support metric** measures the share of transactions that contain an itemset.

$$\frac{\text{number of transactions with items(s)}}{\text{number of transactions}}$$

$$\frac{\text{number of transactions with milk}}{\text{total transactions}}$$

Support for language

TID	Transaction
0	travel, humor, fiction
1	**humor, language**
2	humor, biography, cooking
3	**cooking, language**
4	travel

TID	Transaction
5	poetry, health, travel, history
6	humor
7	travel
8	poetry, fiction, humor
9	fiction, biography

Support for *{language}* = 2 / 10 = 0.2

Support for {Humor} → {Language}

TID	Transaction
0	travel,humor,fiction
1	**humor,language**
2	humor,biography,cooking
3	cooking,language
4	travel

TID	Transaction
5	poetry,health,travel,history
6	humor
7	travel
8	poetry,fiction,humor
9	fiction,biography

SUPPORT for *{language}* → *{humor}* = 0.1

Preparing the data

```python
print(transactions)
```

```
[['travel', 'humor', 'fiction'],
...
['fiction', 'biography']]
```

```python
from mlxtend.preprocessing import TransactionEncoder
```

```python
# Instantiate transaction encoder
encoder = TransactionEncoder().fit(transactions)
```

Preparing the data

```python
# One-hot encode itemsets by applying fit and transform
onehot = encoder.transform(transactions)
```

```python
# Convert one-hot encoded data to DataFrame
onehot = pd.DataFrame(onehot, columns = encoder.columns_)
print(onehot)
```

```
   biography   cooking   ...   poetry   travel
0  False       False     ...   False    True
...
9  True        False     ...   False    False
```

Computing support for single items

```python
print(onehot.mean())
```

```
biography    0.2
cooking      0.2
fiction      0.3
health       0.1
history      0.1
humor        0.5
language     0.2
poetry       0.2
travel       0.4
dtype: float64
```

Computing support for multiple items

```python
import numpy as np

# Define itemset that contains fiction and poetry
onehot['fiction+poetry'] = np.logical_and(onehot['fiction'],onehot['poetry'])

print(onehot.mean())
```

```
biography          0.2
cooking            0.2
...                ...
travel             0.4
fiction+poetry     0.1
dtype: float64
```

Let's practice!

MARKET BASKET ANALYSIS IN PYTHON