Week 3 Association Rule Mining

Theory and Practice

Ying Lin, Ph.D

31 January, 2020

Overview of Topics

- Basic concepts
- Transactional dataset, itemset and association rules
- Metrics to evaluate quality of association rules
- Aprior algorithm to derive frequent itemsets and generate association rules
- Association rule mining demo in R and Python

Motivation

- Sift through large size of transactional database (a.k.a Market Basket Dataset)
  - Each record is a transaction
- Identify those items which are most likely to co-occur in an efficient way
  - A Priori principle
  - Rule generation heuristic
- Output findings as Association Rules
  - {Antecedent} -> {Consequent}
  - If…Then…

Application Examples

- Could be used for both supervised and unsupervised purpose
  - For unsupervised learning purpose, search for interesting data co-occurrence patterns
  - For supervised learning purpose, force the target classification attribute on the Right Hand Side (RHS)
- Finding behavioral patterns of credit card usage that occur in combination with fraudulent case
- Searching for interesting or frequently occurring patterns of DNA in cancer patients
- E-Commerce recommender system: recommend new items based on consumer's shopping history

Market Basket Data

- Transactional dataset examples

ID	Transactions
10001	banana, milk, beer, diaper
10002	milk, diaper, beer, chicken, apple
10003	diaper, beer, orange, cheese
10004	beer, chicken, orange
10005	cheese, orange, milk, apple, cereal, egg
...	…

Derive Association Rules from Market Basket Data

- Item, itemset, length of itemset
  - {banana, milk, beer}: length = 3
- Association rules:
  - {banana} => {milk}, {beer, egg} => {diaper}
  - How to evaluate quality of the association rules?
    - Frequency of the association
    - Strength of the association

Key Evaluation Metrics

- Support

$$support(X) = \frac{count(X)}{N}$$

- Confidence

$$confidence(X \rightarrow Y) = \frac{support(X, Y)}{support(X)}$$

- Lift

$$lift(X \rightarrow Y) = \frac{support(X, Y)}{support(X)support(Y)} = \frac{confidence(X \rightarrow Y)}{support(Y)}$$

Necessity of Lift as a Metric for Association Rules

- Example: {“Miss Homework”} => {“Get A”} has confidence = 0.7. Is it a strong rule?? Could you conclude it is better not to do your homework in order to get a good grade?
- A missing piece of information: this is an easy course and 90% of the students get A
- Therefore not doing your homework leads to a worse grade which is not captured by confidence
- A good metric should capture the real correlation between items
  - $lift = confidence/SupportLHS = 0.7/0.9 = 0.78 < 1$
  - We are only interested in those association rules with lift above 1 and higher is better which indicates a stronger positive correlation between RHS and LHS
  - lift = 1 indicates independence between RHS and RHS

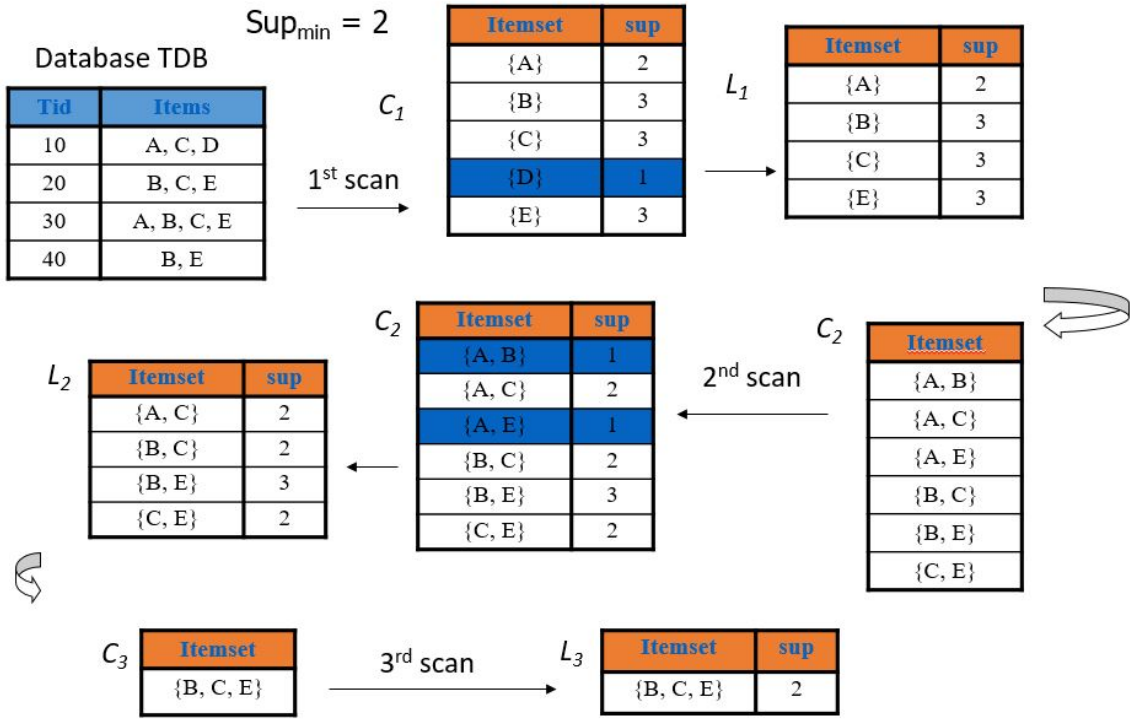
Properties of Performance Metrics

- Support: How often can the association rule be applied?
- Confidence: Strength of the rules
  - How often Y co-occurs with X
  - Asymmetric: direction of rule makes difference
- Lift: normalized confidence
  - If X and Y are independent, lift = 1
  - If X and Y tend to occur together, lift > 1
  - If X and Y tend to occur together, lift < 1
  - similarity to correlation
  - Symmetric (direction doesn't make any difference)
  - Potential issue: order between items, such as {buy iphone} → {buy earphone}
- Range and critical values of the above metrics

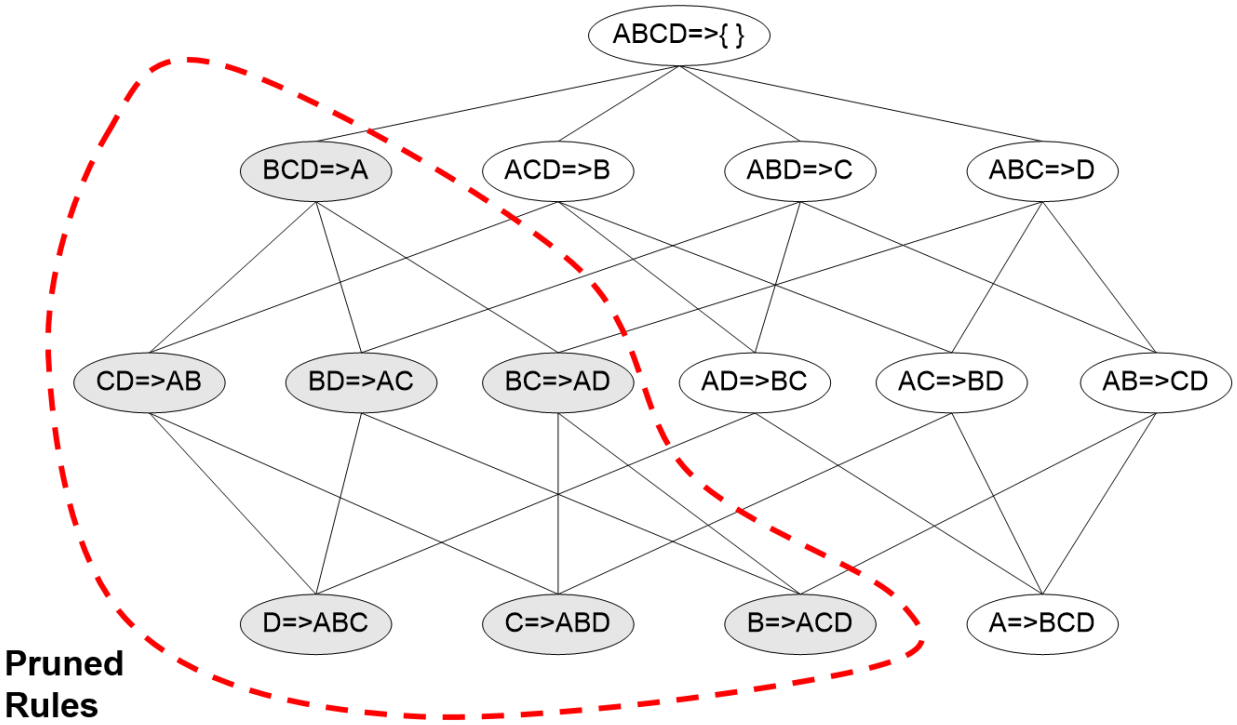
Two-Steps to Mine Association Rules

- Frequent Itemset Generation
  - Generate all itemsets with support $\geq$ min_sup
  - More computational intensive step
  - A Priori principle: if an itemset is found to be frequent, there is no need to further expand such itemset
- Rule Generation
  - Generate high confidence rules from each frequent itemset by doing binary partition of frequent itemsets
  - Heuristic: if an association rule's confidence is not high enough, no needs to further explore all other association rules from the same frequent itemset which have a larger RHS

Apriori Algorithm to Derive Frequent Itemsets



Rule Generation Heuristic



Limitations of Association Rule Mining

- Illustrate association but not causation
  - {guy, diaper} -> {beer} doesn't necessarily mean buying diaper buying beer for a guy; it only means a guy buying diaper is more likely to buy beer than a random guy
- Can only work with categorical attributes.
  - Need to discretize all numerical attributes in pre-processing
- Can't deal with categorical attributes with large number of possible values which lead to low support association rules
  - Perform data exploration to check distribution of attributes and collapse those attribute values with low counts
- Iterative model parameter tuning process
  - Too many discovered rules: increase metric cutoffs in parameter setting
  - Obtaining non interesting rules: domain knowledge

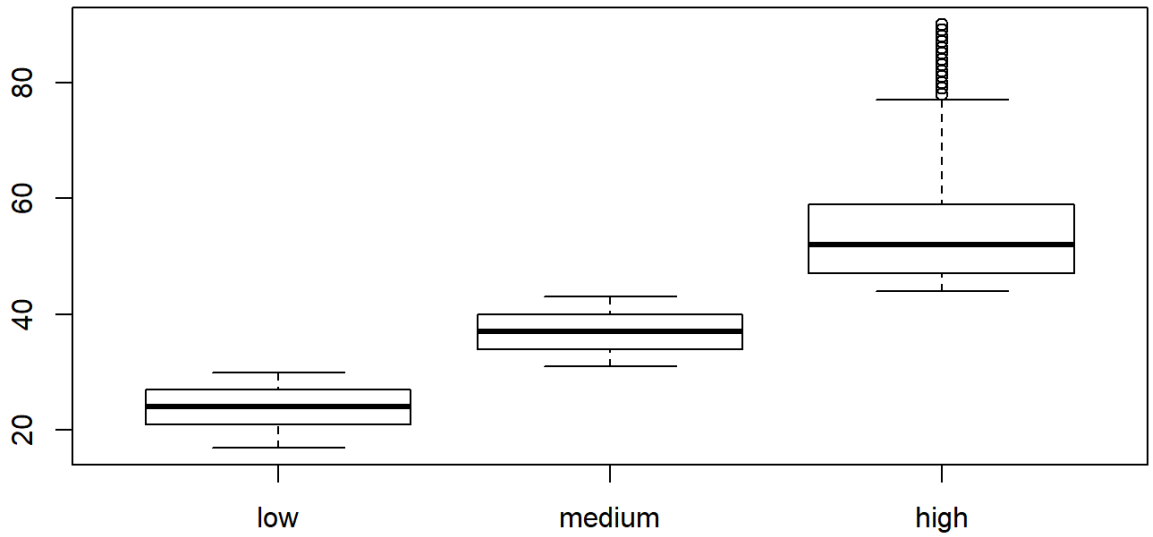
Demo Dataset: AdultUCI

```
library(arules)
library(arulesViz)
data("AdultUCI")
str(AdultUCI)
```

```
## 'data.frame':    48842 obs. of  15 variables:
##  $ age           : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ workclass     : Factor w/ 8 levels "Federal-gov",..: 7 6 4 4 4 4 4 6 4 4 ...
##  $ fnlwgt        : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 .
##  $ education     : Ord.factor w/ 16 levels "Preschool"<"1st-4th"<..: 14 14 9 7 14 15 5 9 15
##  $ education-num : int  13 13 9 7 13 14 5 9 14 13 ...
##  $ marital-status: Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 1 3 3 3 4 3 5 3
##  $ occupation    : Factor w/ 14 levels "Adm-clerical",..: 1 4 6 6 10 4 8 4 10 4 ...
##  $ relationship  : Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
##  $ race          : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
##  $ sex           : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
##  $ capital-gain  : int  2174 0 0 0 0 0 0 0 14084 5178 ...
##  $ capital-loss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hours-per-week: int  40 13 40 40 40 40 16 45 50 40 ...
##  $ native-country: Factor w/ 41 levels "Cambodia","Canada",..: 39 39 39 39 5 39 23 39 39 39
##  $ income        : Ord.factor w/ 2 levels "small"<"large": 1 1 1 1 1 1 1 2 2 2 ...
```

Data Pre-processing: Discretization

```
var_to_discretize <- c("age", "hours-per-week")
AdultUCI$age_grp <- discretize(AdultUCI$age, method = "frequency", breaks = 3,
                               labels = c("low", "medium", "high"), order = T)
boxplot(age ~ age_grp, data = AdultUCI)
```



Mining Association with Apriori

```
apriori(data, parameter = NULL, apearance = NULL, control = NULL)
```

- Parameter: named list with default settings
  - support/supp: minimal support of a rule (default: 0.1)
  - confidence/conf: minimal confidence of association rules (default: 0.8)
  - minlen: an integer for the mininal number of items per rule (default: 1)
  - maxlen: an integer for the maxiaml number of items per rule (default: 10)
- Apriori only creates rules with one item in the RHS (Consequent)
  - Need to set minlen = 2 if you don’t want a rule like {} => {beer} with empty antecedent/LHS

Approach 1: Run A Priori Against Record Dataset Directly

```
rules_record <- apriori(AdultUCI[, sapply(AdultUCI, is.factor)],
                parameter = list(support = 0.1, confidence = 0.5, minlen = 3))
```

```
inspect(head(rules_record, 5))
```

```
##     lhs                                rhs                                support confidence
## [1] {occupation=Exec-managerial,
##      race=White}                    => {native-country=United-States} 0.1059539  0.9453782 1.
## [2] {occupation=Exec-managerial,
##      native-country=United-States} => {race=White}                    0.1059539  0.9231181 1.
## [3] {occupation=Craft-repair,
##      sex=Male}                      => {race=White}                    0.1076532  0.9082743 1.
## [4] {occupation=Craft-repair,
##      race=White}                    => {sex=Male}                      0.1076532  0.9553052 1.
## [5] {occupation=Craft-repair,
##      sex=Male}                      => {native-country=United-States} 0.1068138  0.9011919 1.
```

Approach 2: Run A Priori Against Transactional Dataset: Prepare a Transactional Dataset

- First convert the record dataset to a transational dataset, then apply A Priori algorithm.
  - Note difference between the converted transactional dataset (all transactionals with the same length) and a market basket dataset.
  - Not all the attributes have presence/absence value pairs unlike a market baset

```
fac_var <- sapply(AdultUCI, is.factor)
adult <- as(AdultUCI[, fac_var], "transactions")
inspect(head(adult, 2))
```

```
##     items                                transactionID
## [1] {workclass=State-gov,
##      education=Bachelors,
##      marital-status=Never-married,
##      occupation=Adm-clerical,
##      relationship=Not-in-family,
##      race=White,
##      sex=Male,
```

Approach 2: Run A Priori Against Transactional Dataset

```
rules_transaction <- apriori(adult,
                             parameter = list(support = 0.1, confidence = 0.5, minlen = 3))
```

```
inspect(head(rules_transaction, 5))
```

```
##     lhs                                rhs                                support confidence
## [1] {occupation=Exec-managerial,
##      race=White}                    => {native-country=United-States} 0.1059539  0.9453782 1.
## [2] {occupation=Exec-managerial,
##      native-country=United-States} => {race=White}                    0.1059539  0.9231181 1.
## [3] {occupation=Craft-repair,
##      sex=Male}                      => {race=White}                    0.1076532  0.9082743 1.
## [4] {occupation=Craft-repair,
##      race=White}                    => {sex=Male}                      0.1076532  0.9553052 1.
## [5] {occupation=Craft-repair,
##      sex=Male}                      => {native-country=United-States} 0.1068138  0.9011919 1.
```

Check and Visualize the Most Frequent Items

```
frequent_items <- eclat(adult, parameter = list(support = 0.7, minlen = 2))
```

```
inspect(head(frequent_items, 2))
```

```
##     items                                       support   count
## [1] {race=White,native-country=United-States} 0.7881127 38493
```

```
itemFrequencyPlot(adult, topN = 10, type = "absolute", main = "Item frequency")
```

Sort Association Rules by Performance Metrics

```
rules <- apriori(adult, parameter = list(support = 0.1, confidence = 0.5))
```

```
quality(head(rules, 3))
```

```
##     support confidence lift count
## 1 0.5061218  0.5061218    1 24720
## 2 0.6684820  0.6684820    1 32650
## 3 0.6941976  0.6941976    1 33906
```

```
inspect(head(sort(rules, by = "lift", decreasing = T), 8))
```

```
##     lhs                                rhs                              support confidence
## [1] {marital-status=Never-married,
##      native-country=United-States,
##      age_grp=low}                   => {relationship=Own-child}       0.1123828  0.5414817 3.
## [2] {marital-status=Never-married,
##      race=White,
##      age_grp=low}                   => {relationship=Own-child}       0.1029442  0.5327964 3.
## [3] {marital-status=Never-married,
##      age_grp=low}                   => {relationship=Own-child}       0.1209410  0.5233918 3.
## [4] {relationship=Own-child,
##      race=White,
```

Remove redundant rules

- In the presence of more general rules, the specific rules are considered as redundant and should be removed
  - Example: {human} => {mammal} vs. {human, male} => {mammal}

```
subset_rules <- which(colSums(is.subset(rules, rules)) > 1)
rules <- sort(rules[-subset_rules], by = "lift", descreasing = T)
inspect(head(rules, 5))
```

```
##     lhs                              rhs                                support confidence
## [1] {relationship=Own-child}     => {marital-status=Never-married}      0.1382007  0.890383
## [2] {relationship=Own-child}     => {age_grp=low}                       0.1276975  0.822714
## [3] {income=large}               => {relationship=Husband}              0.1211662  0.754750
## [4] {income=large}               => {marital-status=Married-civ-spouse} 0.1370132  0.853462
## [5] {relationship=Not-in-family} => {marital-status=Never-married}      0.1456533  0.565366
```

Use Association Rule Mining as a Supervised Learning Method

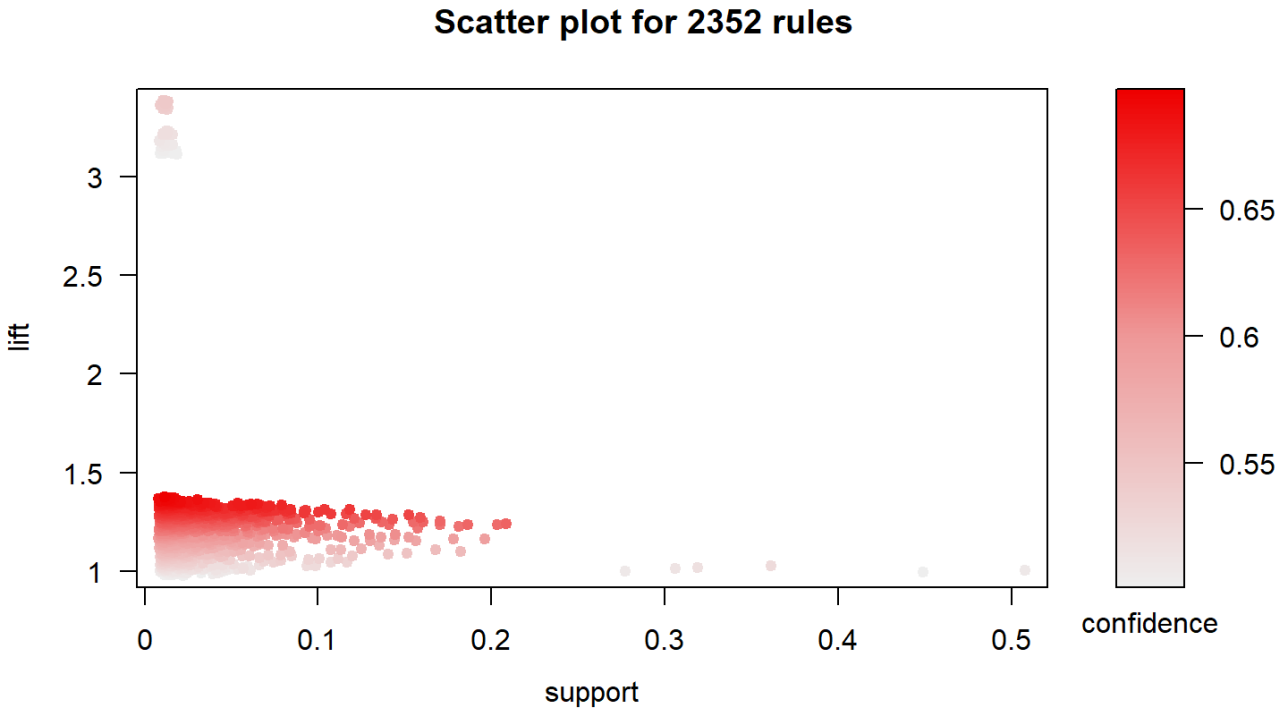
Force the target classification attribute on RHS of association rules

```
rules <- apriori(data = adult, parameter = list(supp = 0.01, conf = 0.5),
                 appearance = list(default = "lhs", rhs = c("income=small", "income=large")),
                 control = list(verbose = F))
inspect(head(sort(rules, by = "lift", descreasing = T), 3))
```

```
##     lhs                                    rhs              support confidence     lift cou
## [1] {education=Bachelors,
##      occupation=Exec-managerial,
##      relationship=Husband}              => {income=large} 0.01250973  0.5431111 3.383068   6
## [2] {education=Bachelors,
##      marital-status=Married-civ-spouse,
##      occupation=Exec-managerial,
##      relationship=Husband}              => {income=large} 0.01250973  0.5431111 3.383068   6
## [3] {education=Bachelors,
##      occupation=Exec-managerial,
##      relationship=Husband,
##      sex=Male}                          => {income=large} 0.01250973  0.5431111 3.383068   6
```

Plot Association Rules on Selected Metrics Dimensions

```
plot(rules, measure = c("support", "lift"), shading = "confidence")
```




```
plot(rules, shading = "order", control = list(main = "Two-key plot"))
```

24/38

Demo in Python: Import Libraries

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas import DataFrame, Series
import seaborn as sns
import apyori as ap
from apyori import apriori #Apriori Algorithm
import mlxtend as ml
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
from mlxtend.preprocessing import TransactionEncoder
```

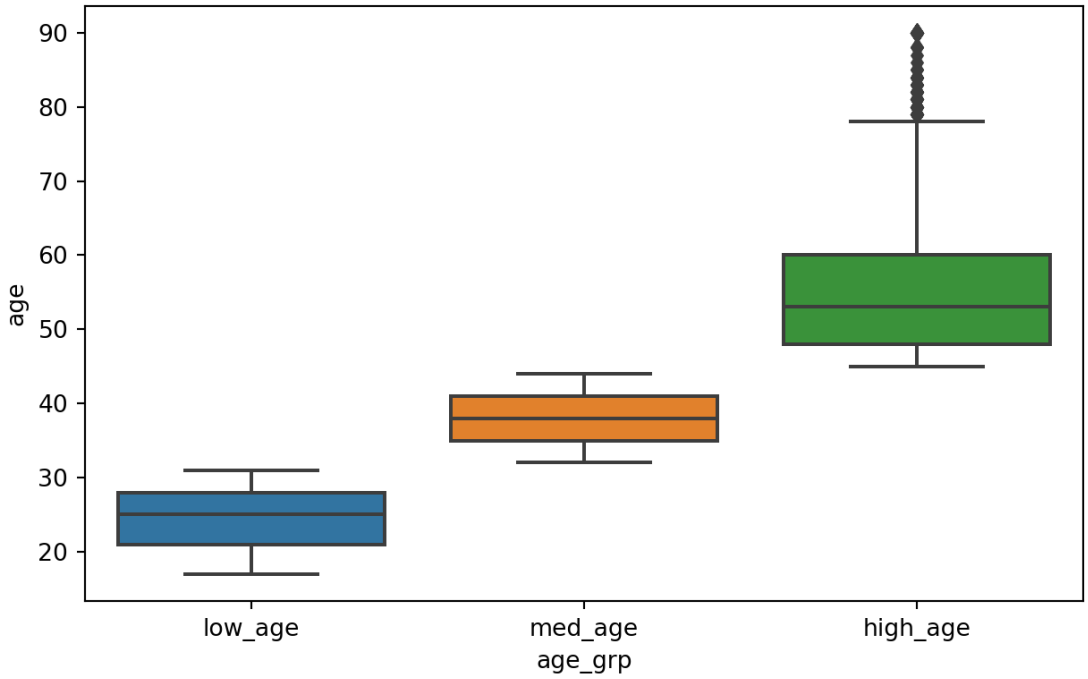
Read Data into Pandas Data Frame

```
AdultUCIdata = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adul
               sep=",", names=["age", "type_employer", "fnlwgt", "education",
               "education_num", "marital", "occupation", "relationship", "race","sex",
               "capital_gain", "capital_loss", "hr_per_week", "country", "income"])

AdultUCIdata.head()
```

```
##    age      type_employer  fnlwgt  ... hr_per_week         country  income
## 0   39          State-gov   77516  ...           40   United-States   <=50K
## 1   50   Self-emp-not-inc   83311  ...           13   United-States   <=50K
## 2   38            Private  215646  ...           40   United-States   <=50K
## 3   53            Private  234721  ...           40   United-States   <=50K
## 4   28            Private  338409  ...           40            Cuba   <=50K
##
## [5 rows x 15 columns]
```

Discretize Numerical Attributes

```
AdultUCIdata["age_grp"] = pd.qcut(AdultUCIdata.age, 3,
                                  labels = ['low_age','med_age','high_age'])
AdultUCIdata["capitalgain_grp"] = pd.cut(AdultUCIdata.capital_gain, 2,
                                         labels = ['low_gain','high_gain'])
AdultUCIdata["capitalloss_grp"] = pd.cut(AdultUCIdata.capital_loss, 2,
                                         labels = ['low_loss','high_loss'])
AdultUCIdata["hr_perweek_grp"] = pd.cut(AdultUCIdata.hr_per_week, [0, 39, 45, 80],
                                        labels = ['low_hr','normal_hr','high_hr'])

sns.boxplot(x='age_grp',y='age',data=AdultUCIdata)
```


90
80
70
60
50
40
30
20
age
low_age
med_age
high_age
age_grp

Prepare Dataset for Association Rule Mining

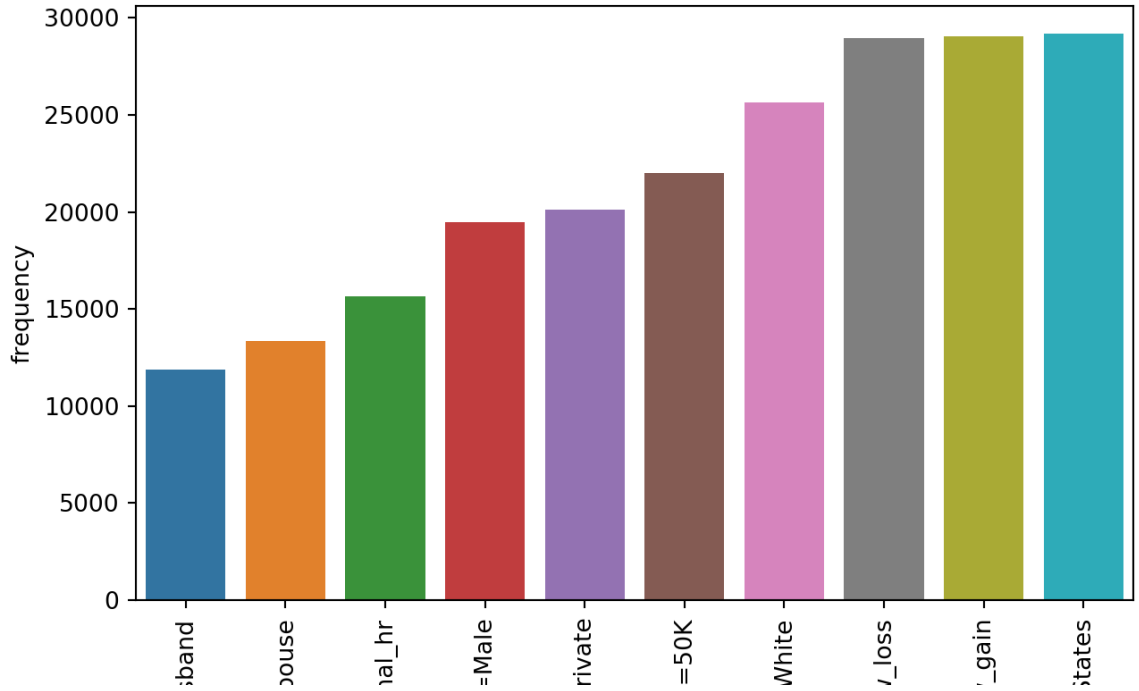
```
AdultUCIdata = AdultUCIdata.astype('str')
AdultUCIdata.loc[:,['age', 'age_grp', 'capital_gain', 'capitalgain_grp',
                    'hr_per_week', 'hr_perweek_grp']].head(5)
AdultUCIdata[AdultUCIdata.columns] = AdultUCIdata.apply(lambda x: x.str.strip())
AdultUCIdata = AdultUCIdata.astype('object')
AdultUCIdata2 = AdultUCIdata.drop(['age', 'capital_gain', 'hr_per_week',
                                   'capital_loss', 'fnlwgt'], axis=1)
AdultUCIdata3 = AdultUCIdata2[AdultUCIdata['country'] == 'United-States']
AdultUCIdata4 = pd.DataFrame({col: str(col)+'=' for col in AdultUCIdata3}, index=AdultUCIdata3.
AdultUCIdata4.head()
```

```
##   age   age_grp capital_gain capitalgain_grp hr_per_week hr_perweek_grp
## 0  39   med_age         2174        low_gain          40      normal_hr
## 1  50  high_age            0        low_gain          13         low_hr
## 2  38   med_age            0        low_gain          40      normal_hr
## 3  53  high_age            0        low_gain          40      normal_hr
## 4  28   low_age            0        low_gain          40      normal_hr

##                     type_employer  ...           hr_perweek_grp
## 0         type_employer=State-gov  ...  hr_perweek_grp=normal_hr
## 1  type_employer=Self-emp-not-inc  ...     hr_perweek_grp=low_hr
## 2           type_employer=Private  ...  hr_perweek_grp=normal_hr
## 3           type_employer=Private  ...  hr_perweek_grp=normal_hr
## 5           type_employer=Private  ...  hr_perweek_grp=normal_hr
##
## [5 rows x 14 columns]
```

Visualize Frequent Itemsets

```
melted_data = pd.melt(AdultUCIdata4)
melted_data.head()
frequency = melted_data.groupby(by=['value'])['value'].count().sort_values(ascending=True)
freq_itemset = pd.DataFrame({'item':frequency.index, 'frequency':frequency.values})
g = sns.barplot(data=freq_itemset.tail(10), x='item', y='frequency')
g.set_xticklabels(g.get_xticklabels(), rotation=90)
plt.show()
```

```
## [Text(0, 0, 'relationship=Husband'), Text(0, 0, 'marital=Married-civ-spouse'), Text(0, 0, 'h
```



32/38

Run APriori with Apyori Library

```
records = []
for i in range(0,len(AdultUCIdata4)):
    records.append([str(AdultUCIdata4.values[i,j])
    for j in range(0, len(AdultUCIdata4.columns))])
frequent_itemset = ap.apriori(records, min_support=0.8, min_confidence=0.8,
                              min_lift=1,min_length=2)
results = list(frequent_itemset)
len(results)
results[1:5]
```

```
## 11
```

```
## [RelationRecord(items=frozenset({'capitalloss_grp=low_loss'}), support=0.9930065135413095, c
```

Use Association Rules from Mlxtend Library

```
te = TransactionEncoder()
te_ary = te.fit(records).transform(records)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)
frequent_itemsets.sort_values(by='support',ascending=False).head(10)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.head()
rules[(rules['lift']>1) & (rules['confidence'] > 0.8)].head(10)
```

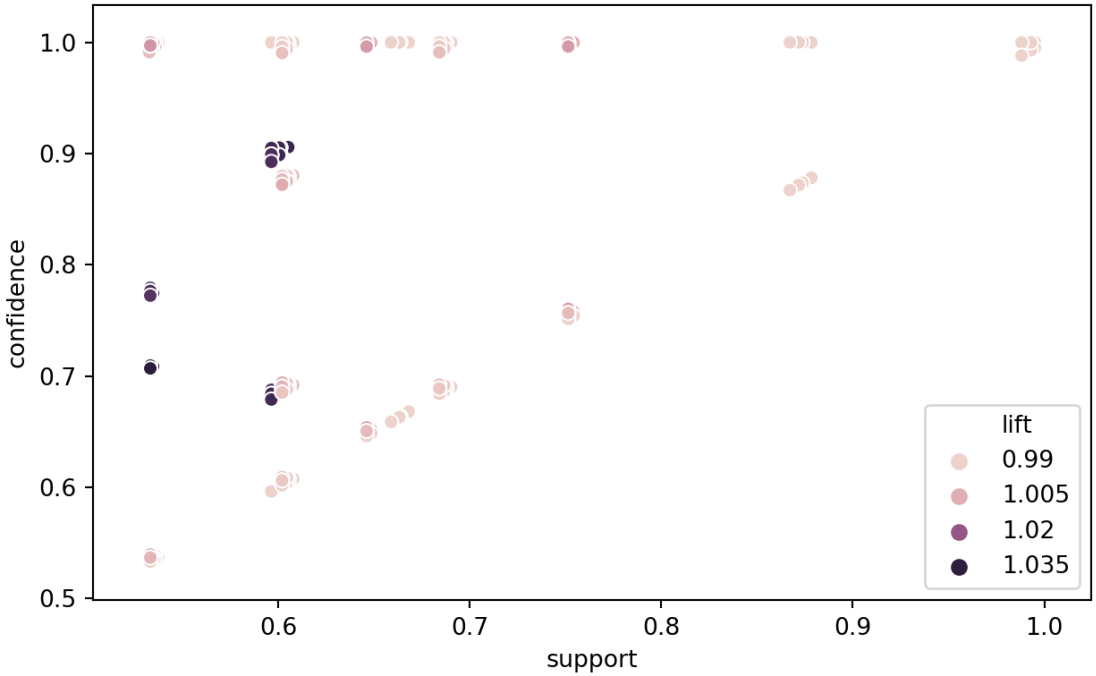
```
##      support                                              itemsets
## 2   1.000000                              (country=United-States)
## 0   0.995132                            (capitalgain_grp=low_gain)
## 9   0.995132  (capitalgain_grp=low_gain, country=United-States)
## 15  0.993007  (country=United-States, capitalloss_grp=low_loss)
## 1   0.993007                            (capitalloss_grp=low_loss)
## 30  0.988138  (capitalgain_grp=low_gain, country=United-Stat...
## 8   0.988138  (capitalgain_grp=low_gain, capitalloss_grp=low...
## 5   0.878334                                           (race=White)
## 23  0.878334                 (race=White, country=United-States)
## 12  0.873740              (race=White, capitalgain_grp=low_gain)
```

```
##                    antecedents                  consequents  ...  leverage  conviction
## 0  (capitalgain_grp=low_gain)     (country=United-States)  ...  0.000000         inf
## 1     (country=United-States)  (capitalgain_grp=low_gain)  ...  0.000000    1.000000
## 2  (capitalgain_grp=low_gain)  (hr_perweek_grp=normal_hr)  ...  0.001176    1.002560
## 3  (hr_perweek_grp=normal_hr)  (capitalgain_grp=low_gain)  ...  0.001176    1.816581
## 4  (capitalgain_grp=low_gain)              (income=<=50K)  ...  0.003671    1.015236
##
## [5 rows x 9 columns]
```

```
##                                    antecedents  ... conviction
## 3                    (hr_perweek_grp=normal_hr)  ...   1.816581
## 5                                (income=<=50K)  ...        inf
## 7                   (type_employer=Private)  ...   1.324561
```

Plot Association Rules

```
b = sns.scatterplot(data=rules,x='support',y='confidence',hue='lift')
plt.show()
```



Classification with Association Rule Mining

```
def SupervisedApriori(data,consequent,min_supp,min_conf,min_lift):
    frequent_itemsets = apriori(data, min_supp, use_colnames=True)
    rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=min_conf)
    #filter according to lift
    rules = rules[rules['lift'] > min_lift]
    sup_rules = pd.DataFrame()
    for i in consequent:
        df = rules[rules['consequents'] == {i}]
        sup_rules = sup_rules.append(df,ignore_index = True)
    return(sup_rules)

SupervisedApriori(df,consequent = ['income=>50K','income=<=50K'],
min_supp=0.04, min_conf=0.7, min_lift=2).sort_values(by='support',ascending=False).head(5)
```

```
##                                          antecedents  ...      lift
## 0   (marital=Married-civ-spouse, occupation=Prof-s...  ...  2.893750
## 3   (marital=Married-civ-spouse, occupation=Prof-s...  ...  2.893750
## 2   (marital=Married-civ-spouse, occupation=Prof-s...  ...  2.885460
## 8   (marital=Married-civ-spouse, occupation=Prof-s...  ...  2.885460
## 1   (capitalgain_grp=low_gain, marital=Married-civ...  ...  2.860188
## 6   (capitalgain_grp=low_gain, marital=Married-civ...  ...  2.860188
## 4   (race=White, marital=Married-civ-spouse, occup...  ...  2.911115
## 10  (race=White, marital=Married-civ-spouse, occup...  ...  2.911115
## 5   (capitalgain_grp=low_gain, marital=Married-civ...  ...  2.851285
## 11  (capitalgain_grp=low_gain, marital=Married-civ...  ...  2.851285
##
## [10 rows x 7 columns]
```