3_assignment

November 11, 2023

1 I will use a late submission token for this assignment

2 Q4

First, we generate length 1 frequent itemsets from local DB. The number of frequent itemsets generated in each DB should be in proportion to the number of transactions in that DB. After the generation, local DBs then submit their length 1 frequent itemsets and corresponding support to the central server.

Second, the central server aggregates all submitted local supports by each itemsets. If the total supports exceed the threshold , which is 0.1 million (1% of the total transactions), that itemset is frequet.

Third, the central server would pass the frequent itemsets generated in central server to local servers. Then the local servers would use those frequent itemsets from the central server to generate new supersets and then pass those supersets to the central server.

Finally, repeat steps above and stop the process until no frequent itemsets are generated in the central server.

3 Q5-1

I use mlxtend to meet the question requirement. My methodology is after generated all frequent itemsets, filtered those don't have the "Outcome" attribute so that all frequent pattern will only contain "Outcome".

[83]:
```
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
import pandas as pd

#1. discretize the data
df = pd.read_csv("diabetes.csv")
cols = ["Pregnancies", "Glucose", "BloodPressure", "Age", 'Outcome']
df = df[cols]
df['Glucose'] = pd.cut(df['Glucose'], bins=range(0, 200, 10), right=False,␣
 ↪labels=[f"Glucose_{i}_{i+9}" for i in range(0, 190, 10)])
df['BloodPressure'] = pd.cut(df['BloodPressure'], bins=range(0, 200, 10),␣
 ↪right=False, labels=[f"BP_{i}_{i+9}" for i in range(0, 190, 10)])
```

```
df['Age'] = pd.cut(df['Age'], bins=range(0, 100, 10), right=False,␣
 ↪labels=[f"Age_{i}_{i+9}" for i in range(0, 90, 10)])
df['Outcome'] = df['Outcome'].apply(lambda x: 'Diabetes' if x == 1 else␣
 ↪'No_Diabetes')
```

[84]:
```
#2. Convert all columns to transactions
df = df.astype(str)
transactions = df.values.tolist()

# 3. Apply one-hot-encoding to the dataset
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df = pd.DataFrame(te_ary, columns=te.columns_)

# 4. Apply the apriori algorithm
frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)

# 5. Filter frequent itemsets to include those contain 'Outcome' attribute
filtered_frequent_itemsets = frequent_itemsets.copy()
filtered_frequent_itemsets['contains_outcome'] =␣
 ↪filtered_frequent_itemsets['itemsets'].apply(lambda x: 'Diabetes' in x or␣
 ↪'No_Diabetes' in x)
filtered_frequent_itemsets =␣
 ↪filtered_frequent_itemsets[filtered_frequent_itemsets['contains_outcome']]

# 6. Find top 100 most frequent itemsets.
top_100_itemsets = filtered_frequent_itemsets.sort_values(by='support',␣
 ↪ascending=False).head(100)
```

4 Q5-2

[85]:
```
for index, row in top_100_itemsets.iterrows():
    print(f"Row {index}: {row.to_dict()}")
```

```
Row 40: {'support': 0.6510416666666666, 'itemsets': frozenset({'No_Diabetes'}),
'contains_outcome': True}
Row 167: {'support': 0.40625, 'itemsets': frozenset({'No_Diabetes',
'Age_20_29'}), 'contains_outcome': True}
Row 27: {'support': 0.3489583333333333, 'itemsets': frozenset({'Diabetes'}),
'contains_outcome': True}
Row 237: {'support': 0.19921875, 'itemsets': frozenset({'No_Diabetes',
'BP_70_79'}), 'contains_outcome': True}
Row 225: {'support': 0.18619791666666666, 'itemsets': frozenset({'No_Diabetes',
'BP_60_69'}), 'contains_outcome': True}
Row 403: {'support': 0.140625, 'itemsets': frozenset({'No_Diabetes', 'BP_60_69',
'Age_20_29'}), 'contains_outcome': True}
Row 71: {'support': 0.13802083333333334, 'itemsets': frozenset({'No_Diabetes',
```

'1'}), 'contains_outcome': True}
Row 301: {'support': 0.12239583333333333, 'itemsets': frozenset({'No_Diabetes',
'Age_20_29', '1'}), 'contains_outcome': True}
Row 226: {'support': 0.11979166666666667, 'itemsets': frozenset({'BP_70_79',
'Diabetes'}), 'contains_outcome': True}
Row 184: {'support': 0.11588541666666667, 'itemsets': frozenset({'No_Diabetes',
'Age_30_39'}), 'contains_outcome': True}
Row 262: {'support': 0.11588541666666667, 'itemsets': frozenset({'No_Diabetes',
'Glucose_100_109'}), 'contains_outcome': True}
Row 155: {'support': 0.109375, 'itemsets': frozenset({'Age_20_29', 'Diabetes'}),
'contains_outcome': True}
Row 87: {'support': 0.109375, 'itemsets': frozenset({'No_Diabetes', '2'}),
'contains_outcome': True}
Row 247: {'support': 0.109375, 'itemsets': frozenset({'No_Diabetes',
'BP_80_89'}), 'contains_outcome': True}
Row 270: {'support': 0.109375, 'itemsets': frozenset({'No_Diabetes',
'Glucose_90_99'}), 'contains_outcome': True}
Row 411: {'support': 0.10026041666666667, 'itemsets': frozenset({'No_Diabetes',
'Age_20_29', 'BP_70_79'}), 'contains_outcome': True}
Row 174: {'support': 0.09895833333333333, 'itemsets': frozenset({'Diabetes',
'Age_30_39'}), 'contains_outcome': True}
Row 328: {'support': 0.09765625, 'itemsets': frozenset({'No_Diabetes', '2',
'Age_20_29'}), 'contains_outcome': True}
Row 54: {'support': 0.09505208333333333, 'itemsets': frozenset({'No_Diabetes',
'0'}), 'contains_outcome': True}
Row 263: {'support': 0.08854166666666667, 'itemsets': frozenset({'No_Diabetes',
'Glucose_110_119'}), 'contains_outcome': True}
Row 264: {'support': 0.0859375, 'itemsets': frozenset({'No_Diabetes',
'Glucose_120_129'}), 'contains_outcome': True}
Row 189: {'support': 0.08463541666666667, 'itemsets': frozenset({'Age_40_49',
'Diabetes'}), 'contains_outcome': True}
Row 280: {'support': 0.07942708333333333, 'itemsets': frozenset({'No_Diabetes',
'0', 'Age_20_29'}), 'contains_outcome': True}
Row 239: {'support': 0.07942708333333333, 'itemsets': frozenset({'Diabetes',
'BP_80_89'}), 'contains_outcome': True}
Row 432: {'support': 0.07942708333333333, 'itemsets': frozenset({'No_Diabetes',
'Glucose_90_99', 'Age_20_29'}), 'contains_outcome': True}
Row 269: {'support': 0.07682291666666667, 'itemsets': frozenset({'No_Diabetes',
'Glucose_80_89'}), 'contains_outcome': True}
Row 424: {'support': 0.07682291666666667, 'itemsets': frozenset({'No_Diabetes',
'Age_20_29', 'Glucose_100_109'}), 'contains_outcome': True}
Row 215: {'support': 0.07421875, 'itemsets': frozenset({'No_Diabetes',
'BP_50_59'}), 'contains_outcome': True}
Row 216: {'support': 0.0703125, 'itemsets': frozenset({'BP_60_69', 'Diabetes'}),
'contains_outcome': True}
Row 196: {'support': 0.06901041666666667, 'itemsets': frozenset({'No_Diabetes',
'Age_40_49'}), 'contains_outcome': True}
Row 99: {'support': 0.0625, 'itemsets': frozenset({'3', 'No_Diabetes'}),

'contains_outcome': True}
Row 396: {'support': 0.0625, 'itemsets': frozenset({'No_Diabetes', 'BP_50_59',
'Age_20_29'}), 'contains_outcome': True}
Row 425: {'support': 0.059895833333333336, 'itemsets': frozenset({'No_Diabetes',
'Age_20_29', 'Glucose_110_119'}), 'contains_outcome': True}
Row 110: {'support': 0.05859375, 'itemsets': frozenset({'4', 'No_Diabetes'}),
'contains_outcome': True}
Row 416: {'support': 0.057291666666666664, 'itemsets': frozenset({'No_Diabetes',
'Age_20_29', 'BP_80_89'}), 'contains_outcome': True}
Row 431: {'support': 0.0546875, 'itemsets': frozenset({'No_Diabetes',
'Glucose_80_89', 'Age_20_29'}), 'contains_outcome': True}
Row 48: {'support': 0.049479166666666664, 'itemsets': frozenset({'0',
'Diabetes'}), 'contains_outcome': True}
Row 253: {'support': 0.046875, 'itemsets': frozenset({'Glucose_120_129',
'Diabetes'}), 'contains_outcome': True}
Row 426: {'support': 0.046875, 'itemsets': frozenset({'No_Diabetes',
'Glucose_120_129', 'Age_20_29'}), 'contains_outcome': True}
Row 119: {'support': 0.046875, 'itemsets': frozenset({'No_Diabetes', '5'}),
'contains_outcome': True}
Row 308: {'support': 0.046875, 'itemsets': frozenset({'No_Diabetes', 'BP_60_69',
'1'}), 'contains_outcome': True}
Row 439: {'support': 0.045572916666666664, 'itemsets': frozenset({'No_Diabetes',
'BP_70_79', 'Age_30_39'}), 'contains_outcome': True}
Row 344: {'support': 0.044270833333333336, 'itemsets': frozenset({'3',
'Age_20_29', 'No_Diabetes'}), 'contains_outcome': True}
Row 128: {'support': 0.044270833333333336, 'itemsets': frozenset({'No_Diabetes',
'6'}), 'contains_outcome': True}
Row 199: {'support': 0.044270833333333336, 'itemsets': frozenset({'Age_50_59',
'Diabetes'}), 'contains_outcome': True}
Row 265: {'support': 0.044270833333333336, 'itemsets': frozenset({'No_Diabetes',
'Glucose_130_139'}), 'contains_outcome': True}
Row 516: {'support': 0.04296875, 'itemsets': frozenset({'No_Diabetes',
'BP_60_69', '1', 'Age_20_29'}), 'contains_outcome': True}
Row 479: {'support': 0.04296875, 'itemsets': frozenset({'No_Diabetes',
'Glucose_90_99', 'BP_60_69'}), 'contains_outcome': True}
Row 436: {'support': 0.040364583333333336, 'itemsets': frozenset({'BP_70_79',
'Age_30_39', 'Diabetes'}), 'contains_outcome': True}
Row 310: {'support': 0.0390625, 'itemsets': frozenset({'No_Diabetes',
'BP_70_79', '1'}), 'contains_outcome': True}
Row 255: {'support': 0.037760416666666664, 'itemsets': frozenset({'Diabetes',
'Glucose_140_149'}), 'contains_outcome': True}
Row 62: {'support': 0.037760416666666664, 'itemsets': frozenset({'Diabetes',
'1'}), 'contains_outcome': True}
Row 487: {'support': 0.037760416666666664, 'itemsets': frozenset({'No_Diabetes',
'BP_70_79', 'Glucose_100_109'}), 'contains_outcome': True}
Row 251: {'support': 0.036458333333333336, 'itemsets':
frozenset({'Glucose_100_109', 'Diabetes'}), 'contains_outcome': True}
Row 331: {'support': 0.03515625, 'itemsets': frozenset({'No_Diabetes', '2',

'BP_60_69'}), 'contains_outcome': True}
Row 254: {'support': 0.03515625, 'itemsets': frozenset({'Diabetes',
'Glucose_130_139'}), 'contains_outcome': True}
Row 94: {'support': 0.03515625, 'itemsets': frozenset({'3', 'Diabetes'}),
'contains_outcome': True}
Row 274: {'support': 0.033854166666666664, 'itemsets': frozenset({'0',
'Diabetes', 'Age_20_29'}), 'contains_outcome': True}
Row 517: {'support': 0.033854166666666664, 'itemsets': frozenset({'No_Diabetes',
'Age_20_29', '1', 'BP_70_79'}), 'contains_outcome': True}
Row 489: {'support': 0.033854166666666664, 'itemsets': frozenset({'No_Diabetes',
'Glucose_120_129', 'BP_70_79'}), 'contains_outcome': True}
Row 547: {'support': 0.033854166666666664, 'itemsets': frozenset({'No_Diabetes',
'Glucose_90_99', 'BP_60_69', 'Age_20_29'}), 'contains_outcome': True}
Row 474: {'support': 0.033854166666666664, 'itemsets': frozenset({'No_Diabetes',
'BP_60_69', 'Glucose_100_109'}), 'contains_outcome': True}
Row 252: {'support': 0.033854166666666664, 'itemsets': frozenset({'Diabetes',
'Glucose_110_119'}), 'contains_outcome': True}
Row 284: {'support': 0.032552083333333336, 'itemsets': frozenset({'No_Diabetes',
'0', 'BP_60_69'}), 'contains_outcome': True}
Row 134: {'support': 0.032552083333333336, 'itemsets': frozenset({'Diabetes',
'7'}), 'contains_outcome': True}
Row 266: {'support': 0.032552083333333336, 'itemsets': frozenset({'No_Diabetes',
'Glucose_140_149'}), 'contains_outcome': True}
Row 357: {'support': 0.032552083333333336, 'itemsets': frozenset({'4',
'Age_20_29', 'No_Diabetes'}), 'contains_outcome': True}
Row 259: {'support': 0.032552083333333336, 'itemsets': frozenset({'Diabetes',
'Glucose_180_189'}), 'contains_outcome': True}
Row 268: {'support': 0.03125, 'itemsets': frozenset({'No_Diabetes',
'Glucose_70_79'}), 'contains_outcome': True}
Row 257: {'support': 0.03125, 'itemsets': frozenset({'Glucose_160_169',
'Diabetes'}), 'contains_outcome': True}
Row 250: {'support': 0.03125, 'itemsets': frozenset({'BP_90_99',
'No_Diabetes'}), 'contains_outcome': True}
Row 528: {'support': 0.03125, 'itemsets': frozenset({'No_Diabetes', '2',
'BP_60_69', 'Age_20_29'}), 'contains_outcome': True}
Row 404: {'support': 0.029947916666666668, 'itemsets': frozenset({'Age_20_29',
'Diabetes', 'BP_70_79'}), 'contains_outcome': True}
Row 105: {'support': 0.029947916666666668, 'itemsets': frozenset({'4',
'Diabetes'}), 'contains_outcome': True}
Row 201: {'support': 0.029947916666666668, 'itemsets': frozenset({'Age_50_59',
'No_Diabetes'}), 'contains_outcome': True}
Row 256: {'support': 0.028645833333333332, 'itemsets': frozenset({'Diabetes',
'Glucose_150_159'}), 'contains_outcome': True}
Row 332: {'support': 0.028645833333333332, 'itemsets': frozenset({'No_Diabetes',
'2', 'BP_70_79'}), 'contains_outcome': True}
Row 506: {'support': 0.028645833333333332, 'itemsets': frozenset({'No_Diabetes',
'0', 'BP_60_69', 'Age_20_29'}), 'contains_outcome': True}
Row 141: {'support': 0.028645833333333332, 'itemsets': frozenset({'8',

'Diabetes'}), 'contains_outcome': True}
Row 457: {'support': 0.028645833333333332, 'itemsets': frozenset({'No_Diabetes',
'Age_40_49', 'BP_70_79'}), 'contains_outcome': True}
Row 397: {'support': 0.028645833333333332, 'itemsets': frozenset({'BP_60_69',
'Diabetes', 'Age_20_29'}), 'contains_outcome': True}
Row 435: {'support': 0.02734375, 'itemsets': frozenset({'No_Diabetes',
'BP_60_69', 'Age_30_39'}), 'contains_outcome': True}
Row 494: {'support': 0.02734375, 'itemsets': frozenset({'No_Diabetes',
'Glucose_90_99', 'BP_70_79'}), 'contains_outcome': True}
Row 478: {'support': 0.02734375, 'itemsets': frozenset({'No_Diabetes',
'Glucose_80_89', 'BP_60_69'}), 'contains_outcome': True}
Row 475: {'support': 0.02734375, 'itemsets': frozenset({'No_Diabetes',
'BP_60_69', 'Glucose_110_119'}), 'contains_outcome': True}
Row 205: {'support': 0.02734375, 'itemsets': frozenset({'No_Diabetes',
'Age_60_69'}), 'contains_outcome': True}
Row 117: {'support': 0.02734375, 'itemsets': frozenset({'Diabetes', '5'}),
'contains_outcome': True}
Row 258: {'support': 0.02734375, 'itemsets': frozenset({'Diabetes',
'Glucose_170_179'}), 'contains_outcome': True}
Row 543: {'support': 0.02734375, 'itemsets': frozenset({'No_Diabetes',
'BP_60_69', 'Glucose_100_109', 'Age_20_29'}), 'contains_outcome': True}
Row 529: {'support': 0.026041666666666668, 'itemsets': frozenset({'No_Diabetes',
'2', 'Age_20_29', 'BP_70_79'}), 'contains_outcome': True}
Row 286: {'support': 0.026041666666666668, 'itemsets': frozenset({'No_Diabetes',
'0', 'BP_80_89'}), 'contains_outcome': True}
Row 248: {'support': 0.026041666666666668, 'itemsets': frozenset({'BP_90_99',
'Diabetes'}), 'contains_outcome': True}
Row 287: {'support': 0.026041666666666668, 'itemsets': frozenset({'No_Diabetes',
'0', 'Glucose_100_109'}), 'contains_outcome': True}
Row 136: {'support': 0.026041666666666668, 'itemsets': frozenset({'No_Diabetes',
'7'}), 'contains_outcome': True}
Row 317: {'support': 0.026041666666666668, 'itemsets': frozenset({'No_Diabetes',
'Glucose_90_99', '1'}), 'contains_outcome': True}
Row 454: {'support': 0.026041666666666668, 'itemsets': frozenset({'Age_40_49',
'BP_70_79', 'Diabetes'}), 'contains_outcome': True}
Row 312: {'support': 0.024739583333333332, 'itemsets': frozenset({'No_Diabetes',
'Glucose_100_109', '1'}), 'contains_outcome': True}
Row 81: {'support': 0.024739583333333332, 'itemsets': frozenset({'2',
'Diabetes'}), 'contains_outcome': True}
Row 208: {'support': 0.024739583333333332, 'itemsets': frozenset({'No_Diabetes',
'BP_0_9'}), 'contains_outcome': True}
Row 267: {'support': 0.024739583333333332, 'itemsets': frozenset({'No_Diabetes',
'Glucose_150_159'}), 'contains_outcome': True}

5 Q5-3

[86]:
```
# 7. Display the rules
rules = association_rules(frequent_itemsets, metric="confidence",␣
 ↪min_threshold=0.01)

# 8. filter out those consequents is not the outcome
filtered_rules = rules[rules['consequents'].apply(lambda x: 'Diabetes' in x or␣
 ↪'No_Diabetes' in x)]

filtered_rules_sorted = filtered_rules.sort_values(by='confidence',␣
 ↪ascending=False)
filtered_rules_sorted.head(5)
```

[86]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
2097	(Age_20_29, Glucose_70_79, 1)	(No_Diabetes)	0.010417	0.651042	0.010417	1.0	1.536	0.003635	inf	0.352632
2363	(Glucose_100_109, BP_50_59, Age_20_29)	(No_Diabetes)	0.016927	0.651042	0.016927	1.0	1.536	0.005907	inf	0.354967
850	(2, Glucose_80_89)	(No_Diabetes)	0.016927	0.651042	0.016927	1.0	1.536	0.005907	inf	0.354967
2251	(2, Glucose_120_129, Age_20_29)	(No_Diabetes)	0.016927	0.651042	0.016927	1.0	1.536	0.005907	inf	0.354967
2139	(Glucose_80_89, BP_60_69, 1)	(No_Diabetes)	0.011719	0.651042	0.011719	1.0	1.536	0.004089	inf	0.353096

Q1(1) ∵ itemset $X \in t_{i1}, t_{i2}, \ldots t_{in}$ and $X \neq \emptyset$

∴ $sup(X) \geq n$

∵ $n \geq \tau$

∴ $sup(X) \geq \tau$, X must be a frequent itemset

Q1(2) X is a closed frequent itemset

Assumption: Assume we can find a superset S for X, where $sup(S) = sup(X) = n$

Proof: Let S exsists in $TDB_S = \{t_{j1}, t_{j2}, \ldots t_{jn}\}$

∵ X is the subset of S

∴ X also exsists in $TDB_S = \{t_{j1}, t_{j2}, \ldots t_{jn}\}$

We also know X exsists in $TDB_X = \{t_{i1}, t_{i2}, \ldots t_{in}\}$, so $TDB_S = TDB_X$ and therefore $S = X$.

∵ $S = X$ is in contradiction to our assumption that is S is X's superset

∴ We can't find a superset S for X, where $sup(S) = sup(X)$

∴ X is a colsed frequent itemset.

Q2. If Y is not one of $X_1, X_2 \ldots X_k$, then $sup(Y) \leq min\{sup(X_1), sup(X_2), \ldots sup(X_k)\}$, since $X_1, X_2, \ldots X_k$ are the K most frequent items in T

If Y is one of $X_1, X_2, \ldots X_k$,

∵ Y is length-k itemset and sup(Y) is less than or equal to its subset Z_{k-1}

∴ $sup(Y) \leq sup(Z_{k-1}) \leq sup(Z_{k-2}) \leq \ldots \leq sup(Z_1)$ (Z_{k-1} is length-k-1 itemset)

There must be k-1 itemsets whose supports larger or equal than Y

Therefore, Y can only be X_k, which means $sup(Y) \leq min\{sup(X_1), sup(X_2), \ldots sup(X_k)\}$ ave

Q3.

t_1	abc
t_2	abc
t_3	abc

$X = abc$, $sup(X) = 1$

$Y = abc$, $sup(Y) = 1$

$X \cap Y = abc$, $sup(X \cap Y) = 1$

$X \cup Y = abc$, $sup(X \cup Y) = 1$

The relationship should be $sup(X \cap Y) = sup(X \cup Y)$

Proof:

∵ $sup(X) = sup(Y) = sup(X \cap Y)$

∴ transactions in X are also in $X \cap Y$ and Y, $X = Y = X \cap Y$

∴ $X \cup Y = X = Y = X \cap Y$

∴ $sup(X \cup Y) = sup(X \cap Y)$