Lab Report: **03**

Subject: CSE - 477

Section: 1

Submitted By:

Syed Musayedul Hussain

ID: 2021-2-60-015

Submitted to:

Amit Mandal
Lecturer

Department of Computer Science and Engineering ,East West University

In this lab, I applied frequent itemset mining and association rule learning to analyze textual data from comments and captions. The steps were as follows:

I. **Data Cleaning & Preparation**
   - Loaded cleaned_comments.csv and cleaned_captions.csv, ensuring the cleaned_tokens column contained valid token lists.
   - Removed empty or short baskets (fewer than 3 tokens) and optionally deduplicated tokens within baskets.
   - Applied lemmatization first, and later experimented with **stemming** for comparison.
   - Filtered out short tokens (under 4 characters) to reduce noise.

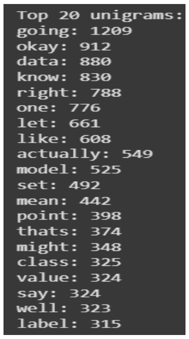
II. **Transaction Encoding & Apriori Mining**
   - Converted the cleaned baskets into a **one-hot encoded DataFrame** using TransactionEncoder.
   - Ran the **Apriori algorithm** with multiple min_support thresholds (0.3, 0.2, 0.15, 0.1, 0.05) to extract frequent itemsets.
   - Filtered itemsets by length (2 or 3 items) for clearer analysis.

III. **Association Rule Generation & Filtering**
   - Used mlxtend's association_rules() to compute confidence and lift.
   - Filtered rules with confidence ≥ 0.6 and lift ≥ 1.2 to retain meaningful patterns.

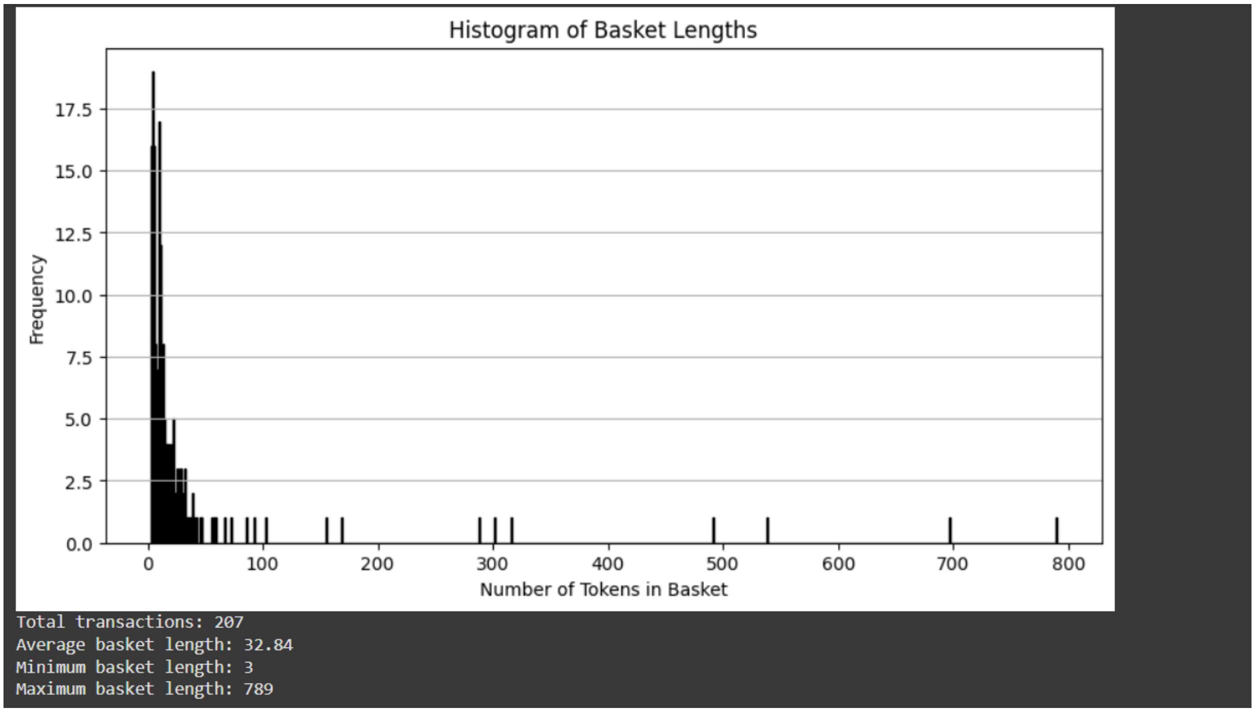
IV. **Visualization & Insights**
   - Plotted:
     - Top 2- and 3-itemsets by support and confidence.
     - Word cloud of frequent tokens.
     - Scatter plot of support vs confidence (colored by lift).
     - Network graph of top co-occurring pairs.
   - Compared patterns found in **comments vs captions**, and also explored a **merged dataset** to reveal broader associations.

V. **Reflections & Conclusions**
   - Documented three key insights from discovered patterns, annotated with support and confidence.
   - Saved cleaned transactions, itemsets, rules, and all plots for reuse and submission.

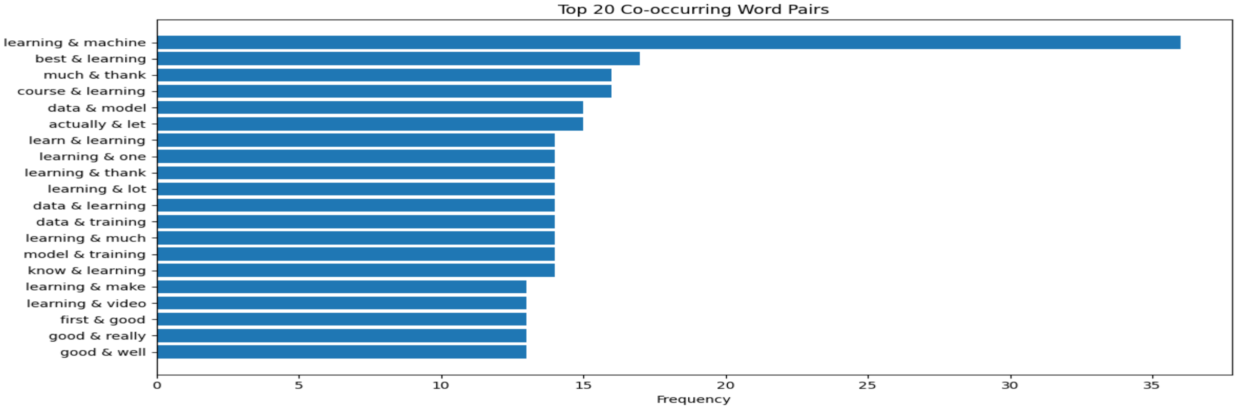
Screenshots of my every working outputs:

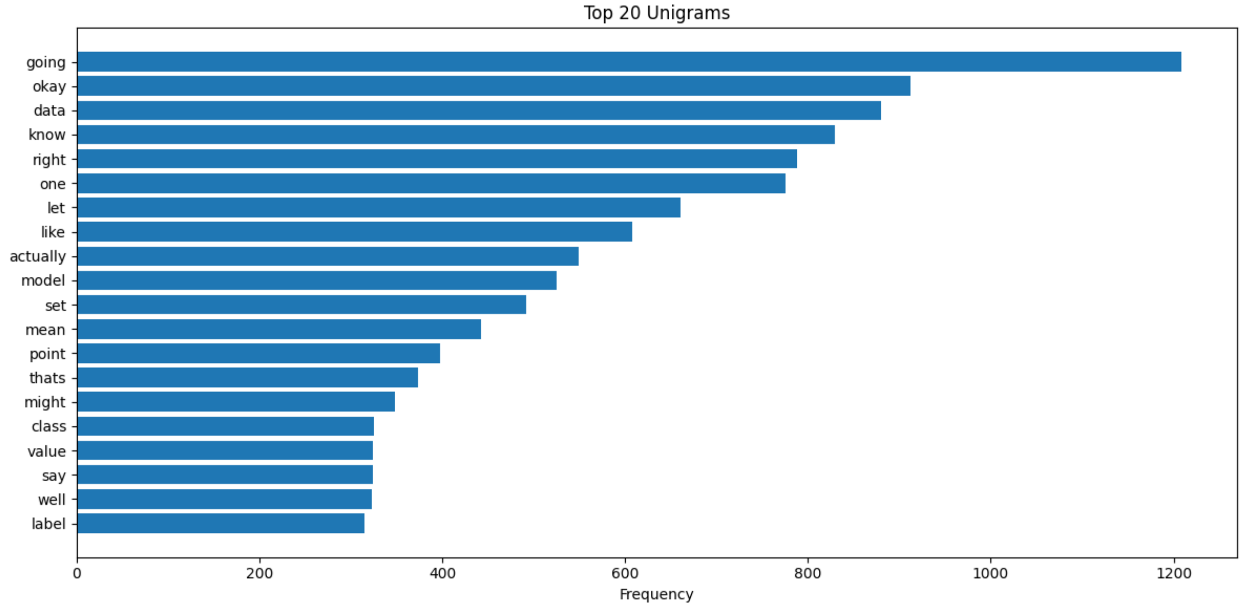


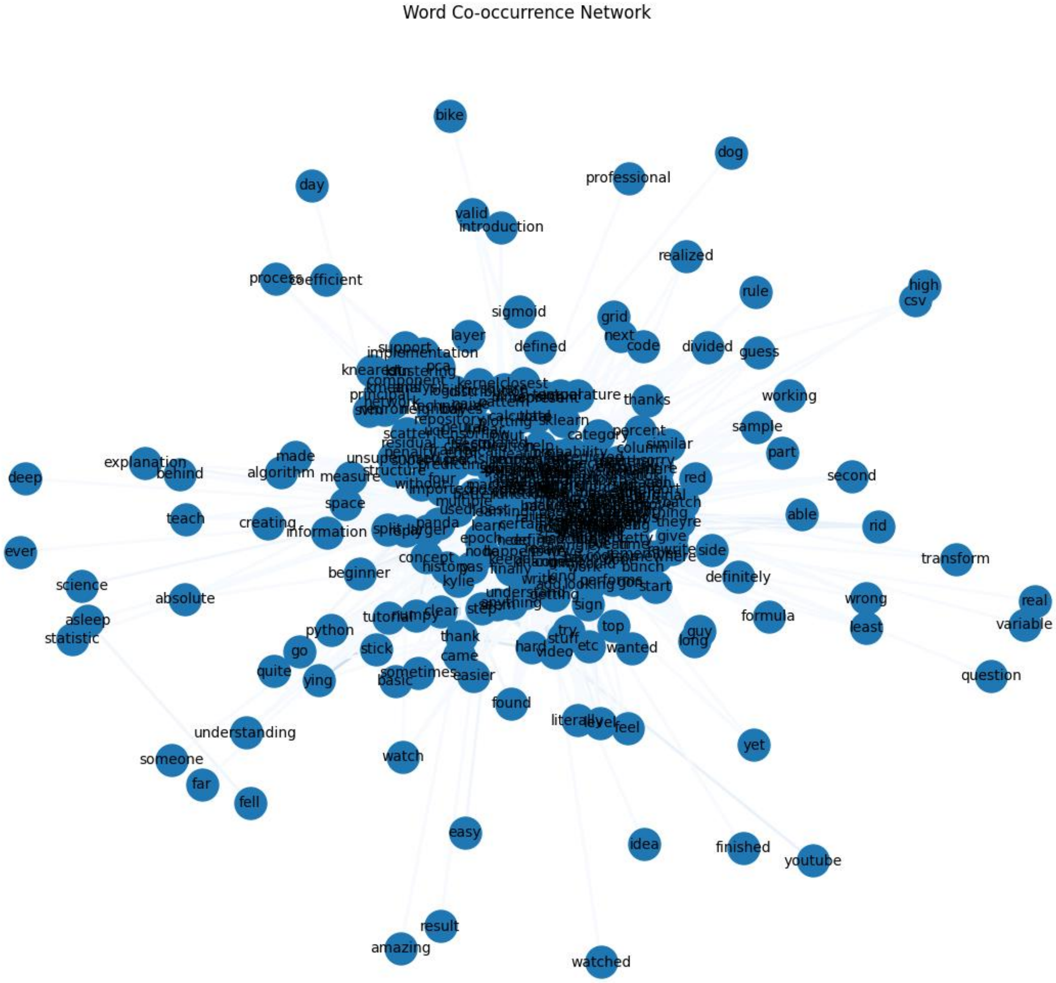
```
Top 20 unigrams:
going: 1209
okay: 912
data: 880
know: 830
right: 788
one: 776
let: 661
like: 608
actually: 549
model: 525
set: 492
mean: 442
point: 398
thats: 374
might: 348
class: 325
value: 324
say: 324
well: 323
label: 315
```

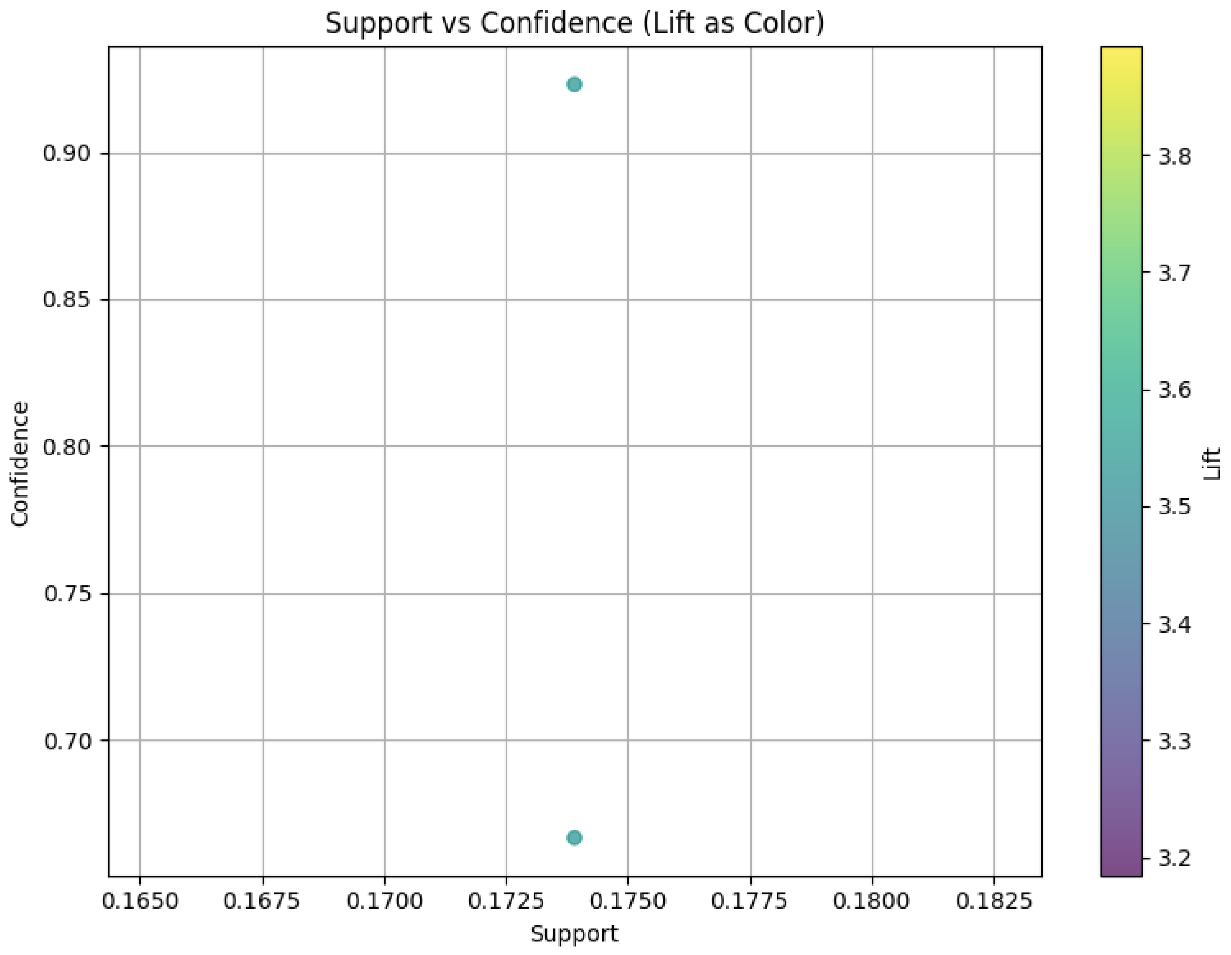


```
Total transactions: 207
Average basket length: 32.84
Minimum basket length: 3
Maximum basket length: 789
```

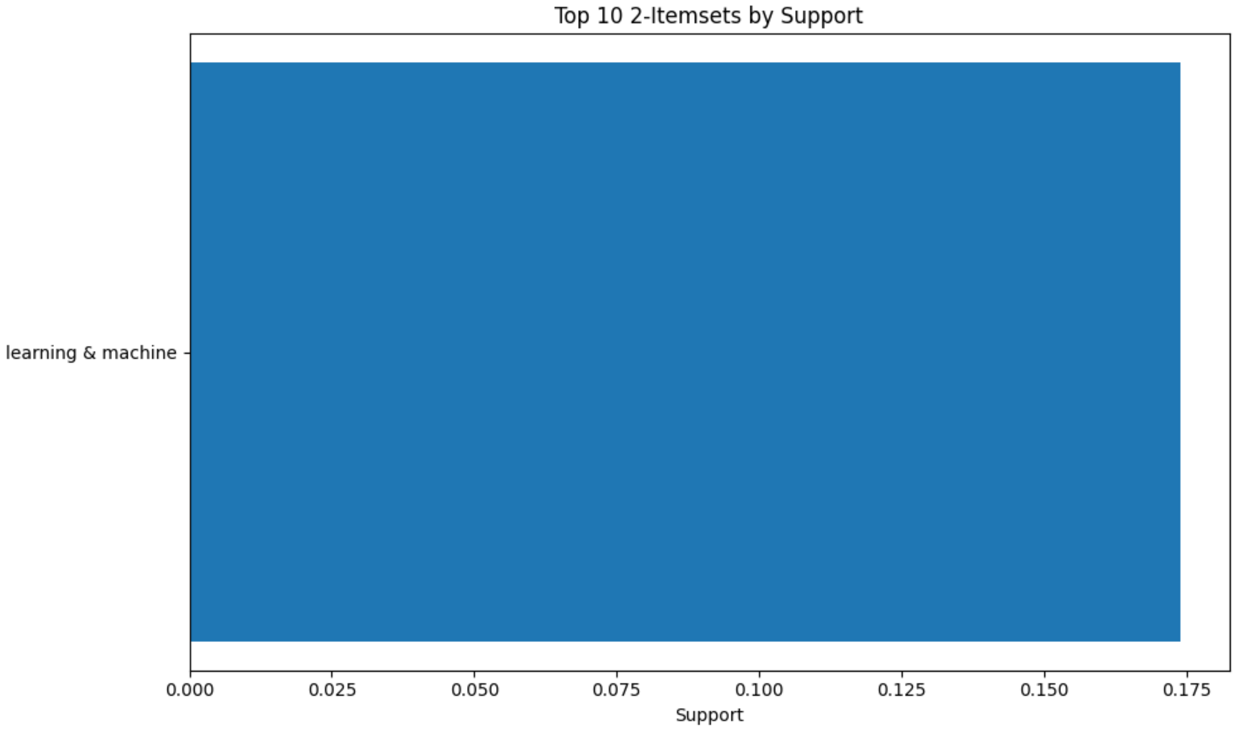



Top 20 Unigrams
going
okay
data
know
right
one
let
like
actually
model
set
mean
point
thats
might
class
value
say
well
label
0
200
400
600
800
1000
1200
Frequency

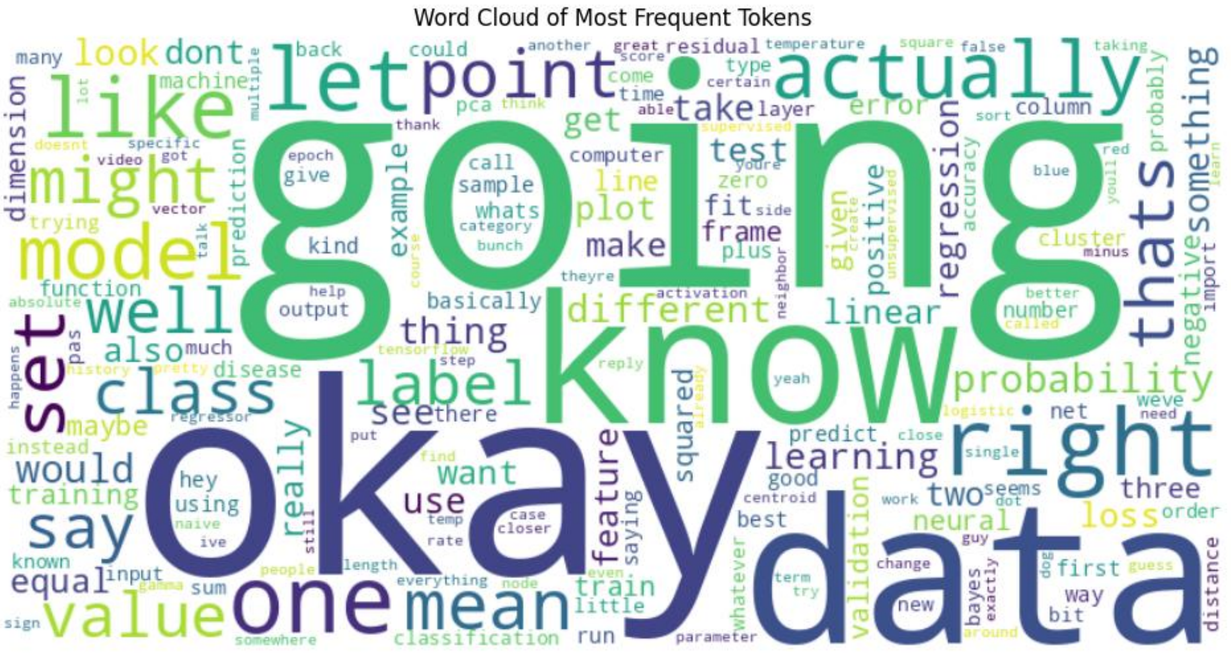

Word Co-occurrence Network
bike
dog
day
professional
valid
introduction
realized
process
coefficient
rule
high
csv
sigmoid
grid
layer
next
defined
code
divided
guess
support
implementation
pca
knearest
clustering
component
kernel
closest
temperature
thanks
working
principal
sample
scatter
category
percent
similar
part
residual
column
explanation
made
deep
behind
algorithm
measure
unsupervised
structure
red
second
space
teach
creating
information
split
panda
epoch
learn
able
rid
ever
theyre
give
transform
concept
side
science
beginner
history
kylie
finally
definitely
absolute
wrong
real
asleep
statistic
tutorial
numpy
clear
step
anything
sign
formula
variable
least
python
thank
try
stuff
top
guy
go
stick
hard
video
etc
wanted
long
question
came
quite
sometimes
basic
ying
easier
found
literally
level
feel
understanding
yet
someone
far
watch
fell
easy
idea
finished
youtube
result
amazing
watched

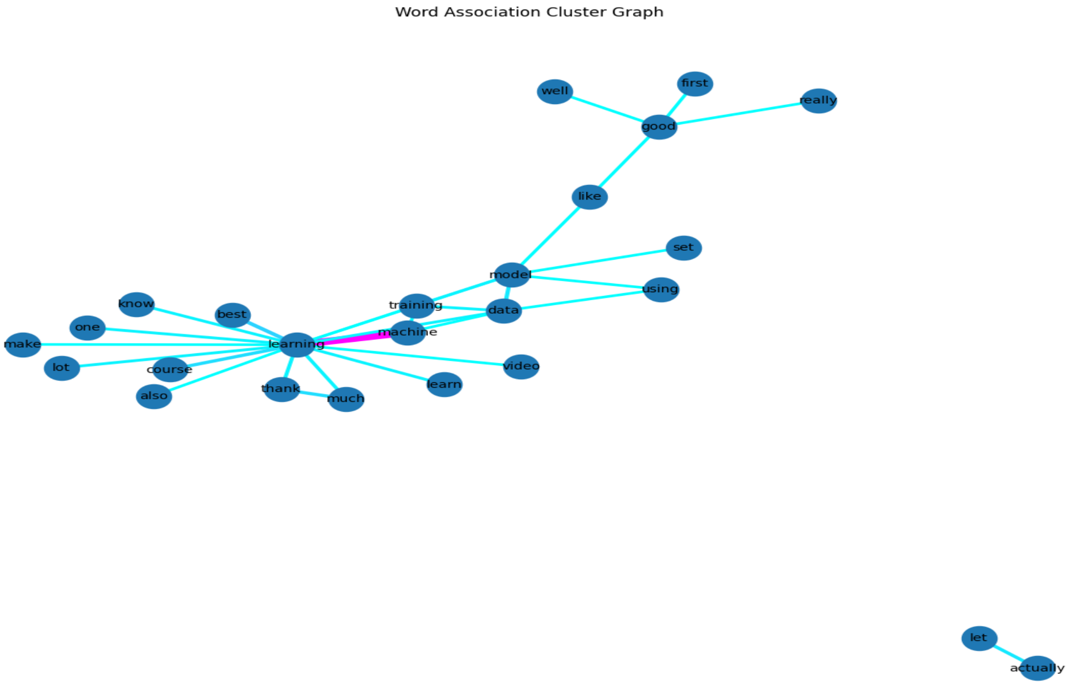


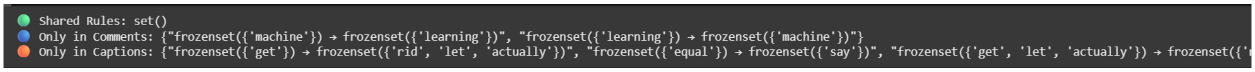
Support vs Confidence (Lift as Color)

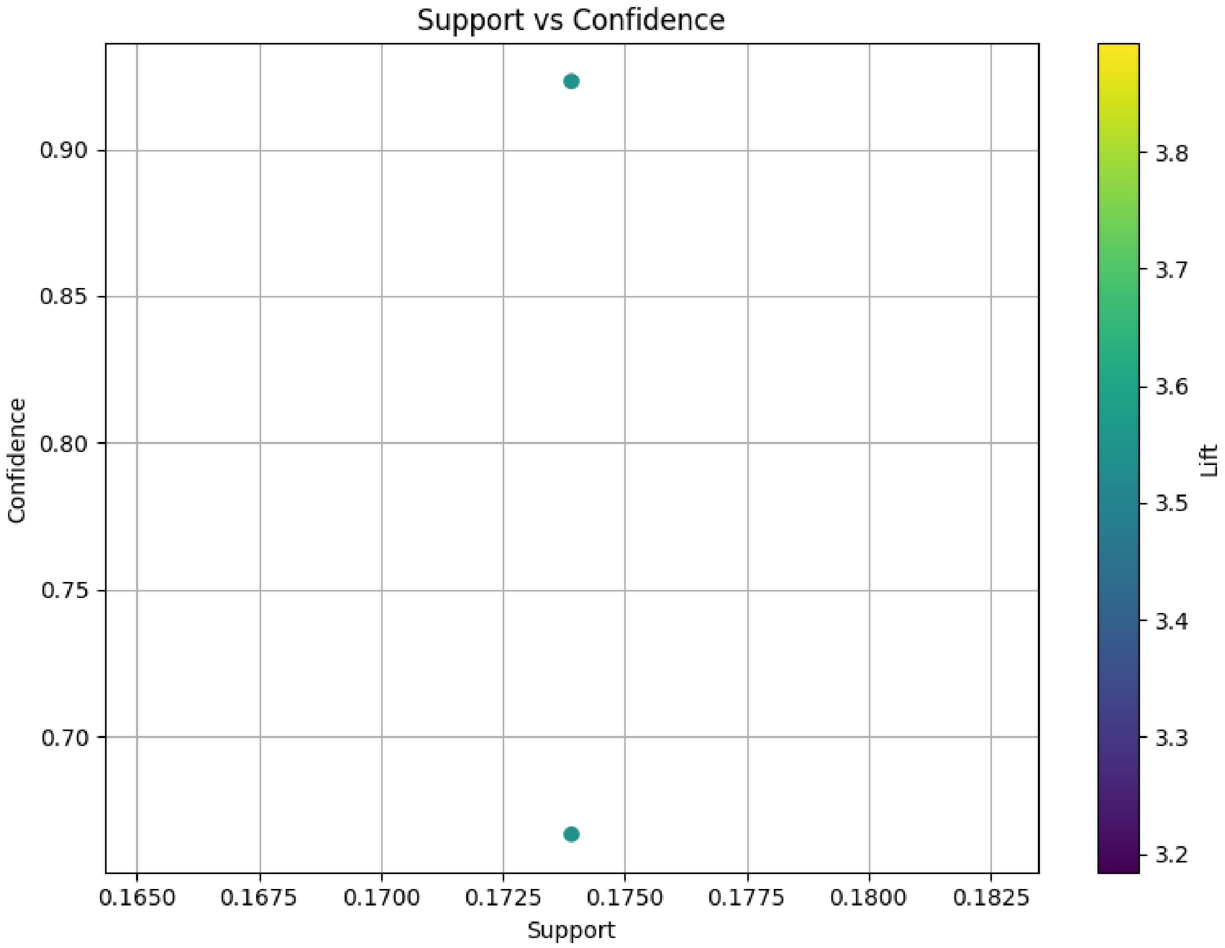


Top 10 2-Itemsets by Support






```
● Shared Rules: set()
● Only in Comments: {"frozenset({'machine'}) → frozenset({'learning'})", "frozenset({'learning'}) → frozenset({'machine'})"}
● Only in Captions: {"frozenset({'get'}) → frozenset({'rid', 'let', 'actually'})", "frozenset({'equal'}) → frozenset({'say'})", "frozenset({'get', 'let', 'actually'}) → frozenset({'
```



Challenge Faced: Runtime Error during Apriori Processing

While applying the Apriori algorithm on the cleaned caption tokens, I encountered a runtime error. The error was caused by the format of the cleaned_tokens column, which was initially stored as string representations of Python lists. Attempting to encode this column without converting it into actual list objects caused issues during the transformation process.

Resolution:

To resolve this, I used ast.literal_eval() to safely convert the string representations into proper Python list objects. I also ensured that each token list had at least 3 unique items by converting them to sets and filtering short lists. After these preprocessing steps, the data was properly encoded using TransactionEncoder, and the Apriori algorithm ran successfully without further errors.