Methodology

```
from google.colab import auth
auth.authenticate_user()
```

```
from google.cloud import storage

project_id = "sharp-matter-449521-u2"
!gcloud config set project {project_id}
```

```
Updated property [core/project].
```

```
!wget -P /usr/lib/spark/jars/ https://storage.googleapis.com/hadoop-lib/gcs/gcs-connector-hadoop3-latest.jar
```

```
--2025-04-24 03:11:14--  https://storage.googleapis.com/hadoop-lib/gcs/gcs-connector-hadoop3-latest.jar
Resolving storage.googleapis.com (storage.googleapis.com)... 142.250.98.207, 142.251.107.207, 74.125.196.207, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|142.250.98.207|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 40713341 (39M) [application/java-archive]
Saving to: ‘/usr/lib/spark/jars/gcs-connector-hadoop3-latest.jar.1’

gcs-connector-hadoo 100%[===================>]  38.83M   142MB/s    in 0.3s

2025-04-24 03:11:15 (142 MB/s) - ‘/usr/lib/spark/jars/gcs-connector-hadoop3-latest.jar.1’ saved [40713341/40713341]
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder \
    .appName("BigDataProcessing") \
    .config("spark.jars", "/usr/lib/spark/jars/gcs-connector-hadoop3-latest.jar") \
    .config("spark.hadoop.fs.gs.impl", "com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystem") \
    .config("spark.hadoop.fs.gs.auth.service.account.enable", "true") \
    .getOrCreate()
```

```
spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
v3.5.5
Master
local[*]
AppName
BigDataProcessing

```
df_reviews = spark.read.parquet('gs://final_dataset_dat490/dat490_final_dataset_cleaned.parquet', headers=True, inferSchema=True)
```

```
df_reviews.columns
```

```
['gmap_id',
 'customer_name',
 'rating',
 'reviews',
 'time',
 'avg_rating',
 'category',
 'latitude',
 'longitude',
 'business_name',
 'num_of_reviews',
 'state',
 'standard_category',
 'Monday',
 'Tuesday',
```

```
        'Wednesday',
        'Thursday',
        'Friday',
        'Saturday',
        'Sunday',
        'timestamp',
        'week',
        'month',
        'year',
        'time_seconds']
```

VADER

```
from pyspark.sql.functions import udf
from pyspark.sql.types import FloatType
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
nltk.download("vader_lexicon")

# Initializing VADER
sia = SentimentIntensityAnalyzer()

def vader_sentiment(text):
    if text:
        return float(sia.polarity_scores(text)["compound"])
    else:
        return 0.0

vader_udf = udf(vader_sentiment, FloatType())

df_sentiment = df_reviews.withColumn("sentiment_score", vader_udf("reviews"))
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

```
from pyspark.sql.functions import when

df_sentiment = df_sentiment.withColumn(
    "sentiment_label",
    when(df_sentiment["sentiment_score"] > 0.2, "Positive")
    .when(df_sentiment["sentiment_score"] < -0.2, "Negative")
    .otherwise("Neutral")
)
```

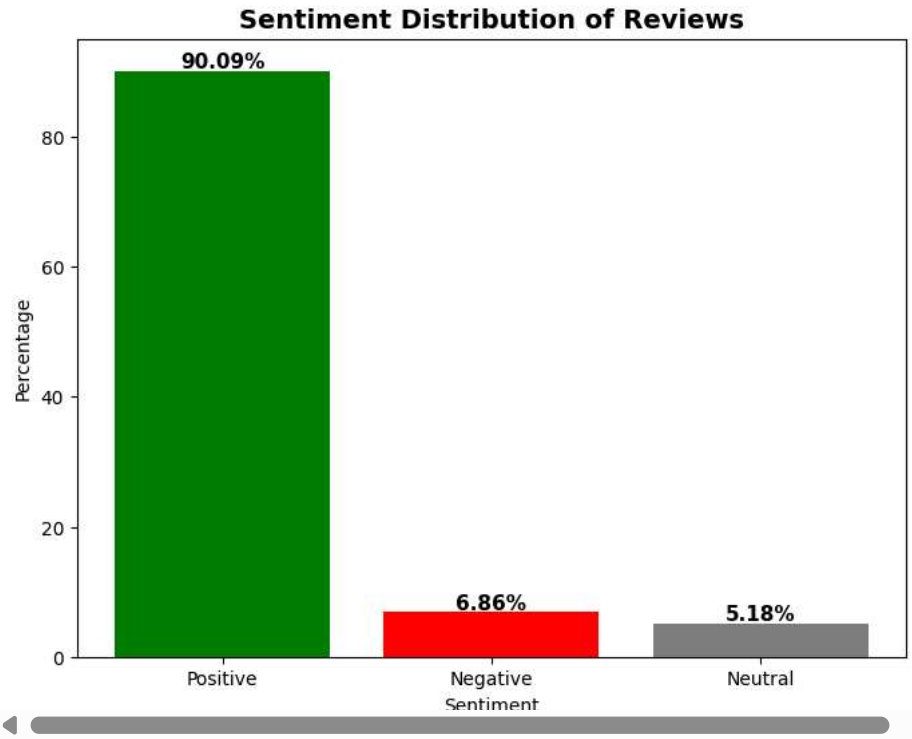
```
from pyspark.sql.functions import approx_count_distinct

df_sentiment.groupBy("sentiment_label").agg(approx_count_distinct("gmap_id").alias("approx_count")).show()
```

```
+---------------+------------+
|sentiment_label|approx_count|
+---------------+------------+
|       Positive|     2598785|
|        Neutral|      149531|
|       Negative|      197929|
+---------------+------------+
```

```
from pyspark.sql.functions import col, round
df_sentiment_counts = df_sentiment.groupBy("sentiment_label").agg(approx_count_distinct("gmap_id").alias("approx_count"))
df_percentages = df_sentiment_counts.withColumn(
    "percentage", round((col("approx_count") / 2884722) * 100, 2)
)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-4d9398626ea2> in <cell line: 0>()
      1 from pyspark.sql.functions import col, round
----> 2 df_sentiment_counts = df_sentiment.groupBy("sentiment_label").agg(approx_count_distinct("gmap_id").alias("approx_count"))
      3 df_percentages = df_sentiment_counts.withColumn(
      4     "percentage", round((col("approx_count") / 2884722) * 100, 2)
      5 )

NameError: name 'approx_count_distinct' is not defined
```

```
df_percentages_pd = df_percentages.toPandas()

import matplotlib.pyplot as plt

df_percentages_pd = df_percentages_pd.sort_values("percentage", ascending=False)

plt.figure(figsize=(8, 6))
bars = plt.bar(df_percentages_pd["sentiment_label"], df_percentages_pd["percentage"], color=["green", "red", "gray"])

for bar in bars:
    height = bar.get_height()
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        height + 0.5,
        f"{height:.2f}%",
        ha="center",
        fontsize=11,
        fontweight="bold"
    )

plt.title("Sentiment Distribution of Reviews", fontsize=14, fontweight="bold")
plt.ylabel("Percentage")
plt.xlabel("Sentiment")
plt.ylim(0, df_percentages_pd["percentage"].max() + 5)

plt.show()
```




```
from pyspark.sql.functions import when

df_sentiment = df_sentiment.withColumn(
    "sentiment_label",
    when(df_sentiment["sentiment_score"] > 0.05, "Positive")
    .when(df_sentiment["sentiment_score"] < -0.05, "Negative")
    .otherwise("Neutral")
)


df_sentiment.select("reviews", "sentiment_score", "sentiment_label").show(10)
```

```
+--------------------+---------------+---------------+
|             reviews|sentiment_score|sentiment_label|
+--------------------+---------------+---------------+
|The pizza, steak ...|         0.6114|       Positive|
|Great way to get ...|         0.6249|       Positive|
|Went there for a ...|         0.7392|       Positive|
|Friendly and know...|         0.4939|       Positive|
|My personal choic...|         0.8308|       Positive|
|This store has be...|        -0.6324|       Negative|
```

```
|I am disappointed...|          0.975|       Positive|
|AUAF oversee a va...|         0.9042|       Positive|
|Great neighborhoo...|         0.6249|       Positive|
|This is place is ...|         0.8807|       Positive|
+--------------------+---------------+---------------+
only showing top 10 rows
```

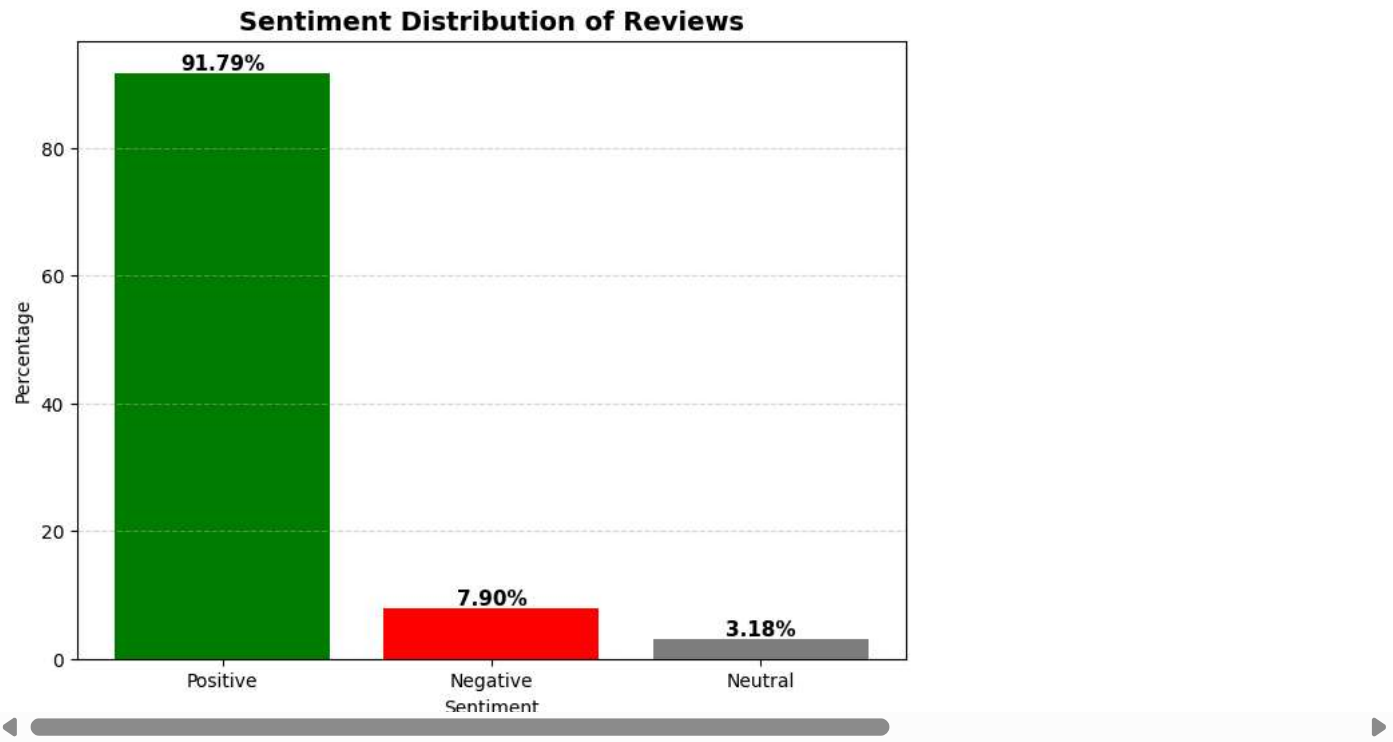
```
from pyspark.sql.functions import approx_count_distinct

df_sentiment_counts = df_sentiment.groupBy("sentiment_label").agg(approx_count_distinct("gmap_id").alias("approx_count"))
```

```
from pyspark.sql.functions import approx_count_distinct

df_sentiment.groupBy("sentiment_label").agg(approx_count_distinct("gmap_id").alias("approx_count")).show()
```

```
+---------------+------------+
|sentiment_label|approx_count|
+---------------+------------+
|       Positive|     2647752|
|        Neutral|       91804|
|       Negative|      227790|
+---------------+------------+
```

```
from pyspark.sql.functions import col, round

df_percentages = df_sentiment_counts.withColumn(
    "percentage", round((col("approx_count") / 2884722) * 100, 2)
)
```

```
df_percentages_pd = df_percentages.toPandas()

import matplotlib.pyplot as plt

df_percentages_pd = df_percentages_pd.sort_values("percentage", ascending=False)

plt.figure(figsize=(8, 6))
bars = plt.bar(df_percentages_pd["sentiment_label"], df_percentages_pd["percentage"], color=["green", "red", "gray"])

for bar in bars:
    height = bar.get_height()
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        height + 0.5,
        f"{height:.2f}%",
        ha="center",
        fontsize=11,
        fontweight="bold"
    )

plt.title("Sentiment Distribution of Reviews", fontsize=14, fontweight="bold")
plt.ylabel("Percentage")
plt.xlabel("Sentiment")
plt.ylim(0, df_percentages_pd["percentage"].max() + 5)

plt.show()
```



TextBlob

```
from pyspark.sql.functions import col, when, regexp_replace
from textblob import TextBlob
from pyspark.sql.functions import udf
from pyspark.sql.types import StructType, StructField, DoubleType, StringType

def get_textblob_sentiment(text):
    if text is not None and isinstance(text, str):
        blob = TextBlob(text)
        return float(blob.sentiment.polarity), float(blob.sentiment.subjectivity)
    else:
        return 0.0, 0.0

schema = StructType([
    StructField("polarity", DoubleType(), True),
    StructField("subjectivity", DoubleType(), True)
])

sentiment_udf = udf(get_textblob_sentiment, schema)

df_reviews = df_reviews.withColumn("sentiment", sentiment_udf("reviews"))
df_reviews = df_reviews.withColumn("polarity", col("sentiment.polarity"))
df_reviews = df_reviews.withColumn("subjectivity", col("sentiment.subjectivity"))


from pyspark.sql.functions import when

# Putting all the reviews into 3 categories based on their polarity
df_reviews = df_reviews.withColumn(
    "sentiment_label",
    when(col("polarity") <= -0.2, "Negative").
    when(col("polarity") <= 0.2, "Neutral").
    otherwise("Positive")
)


df_reviews.select('reviews', 'polarity', 'sentiment_label').show(10, truncate=False)
```

```
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
|reviews                                                                                                                                                                |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------+
|The pizza, steak street tacos and the portobello truffle fries and were all delicious!                                                                                 |
|Great way to get around town, and very affordable.                                                                                                                     |
|Went there for a volleyball tournament. The facilities are nice. They have a lot of bleacher  seating. They offered guest wifi but I di                                |
|Friendly and knowledgeable. I was able to get a copy of my pets shots.
```

```
|My personal choice for somewhat upscale dining in Woodstock. The burgers are delicious, but my go to item is the Lobster Cobb Salad. Yo
|This store has been featured on the Instagram account of “overpriced bourbon”. Tells you all you need to know. Along with the miserable
|I am disappointed to see negative reviews for this establishment. I am very frequent here and always have a pleasant experience. The be
|AUAF oversee a variety  programs. I work as a Home Care Aide through them. As an employee of AUAF Home Care program I was treated with
|Great neighborhood bar
|This is place is solid. The prices are reasonable and the portions large. However the food is not stellar. Some things are pretty good,
+--------------------------------------------------------------------------------------------------------------------------------------
only showing top 10 rows
```

```
sentiment_label_count = df_reviews.groupby('sentiment_label').count()


import seaborn as sns
import matplotlib.pyplot as plt

sentiment_label_count_pd = sentiment_label_count.toPandas()

sentiment_label_count_pd["percentage"] = (sentiment_label_count_pd["count"] /
                                          sentiment_label_count_pd["count"].sum()) * 100

plt.figure(figsize=(10,6))
ax = sns.barplot(data=sentiment_label_count_pd, x="sentiment_label", y="count", hue="sentiment_label", palette="mako")

for p, perc in zip(ax.patches, sentiment_label_count_pd["percentage"]):
    height = p.get_height()
    ax.text(p.get_x() + p.get_width() / 2, height + 0.1, f'{perc:.1f}%',
            ha="center", fontsize=10, fontweight="bold")

plt.title("Average Business Reviews Lifespan by Region", fontsize=14)
plt.xlabel("Sentiment", fontsize=12)
plt.ylabel("Count of each Sentiment", fontsize=12)

plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-14-43cdcab99499> in <cell line: 0>()
      2 import matplotlib.pyplot as plt
      3
----> 4 sentiment_label_count_pd = sentiment_label_count.toPandas()
      5
      6 sentiment_label_count_pd["percentage"] = (sentiment_label_count_pd["count"] /

NameError: name 'sentiment_label_count' is not defined
```

```
from pyspark.sql.functions import col, round
df_sentiment_all = df_reviews.withColumn("sentiment_score", vader_udf("reviews"))
df_sentiment_all = df_sentiment_all.withColumn("polarity", col("sentiment.polarity"))
df_sentiment_all.columns
```

```
['gmap_id',
 'customer_name',
 'rating',
 'reviews',
 'time',
 'avg_rating',
 'category',
 'latitude',
 'longitude',
 'business_name',
 'num_of_reviews',
 'state',
 'standard_category',
 'Monday',
 'Tuesday',
 'Wednesday',
 'Thursday',
 'Friday',
 'Saturday',
 'Sunday',
 'timestamp',
 'week',
 'month',
 'year',
 'time_seconds',
 'sentiment',
 'polarity',
```

```
        'subjectivity',
        'sentiment_label',
        'sentiment_score']


from pyspark.sql.functions import when, length, col
from pyspark.sql import functions as F

df_labeled = df_sentiment_all.withColumn(
    "vader_label",
    when(col("sentiment_score") >= 0.05, "Positive")
    .when(col("sentiment_score") <= -0.05, "Negative")
    .otherwise("Neutral")
)
df_labeled = df_labeled.withColumn(
    "textblob_label",
    when(col("polarity") >= 0.05, "Positive")
    .when(col("polarity") <= -0.05, "Negative")
    .otherwise("Neutral")
)
df_labeled = df_labeled.withColumn("review_length", length(col("reviews")))

df_labeled = df_labeled.withColumn(
    "length_bucket",
    when(col("review_length") < 100, "Short")
    .when(col("review_length") <= 300, "Medium")
    .otherwise("Long")
)

top_categories = [row['standard_category'] for row in df_labeled.groupBy("standard_category")
                  .count().orderBy(F.desc("count")).limit(5).collect()]

df_filtered = df_labeled.filter(col("standard_category").isin(top_categories))

samples = []
for category in top_categories:
    for label in ["Positive", "Negative", "Neutral"]:
        for length_group in ["Short", "Medium", "Long"]:
            subset = (
                df_filtered.filter(
                    (col("standard_category") == category) &
                    (col("vader_label") == label) &
                    (col("length_bucket") == length_group)
                ).orderBy(F.rand()).limit(1)
            )
            samples.append(subset)

df_sample_30 = samples[0]
for i in range(1, len(samples)):
    df_sample_30 = df_sample_30.union(samples[i])

df_final = df_sample_30.select(
    "reviews", "standard_category", "review_length",
    "vader_label", "sentiment", "textblob_label", "polarity"
)

df_final.show(30, truncate=False)



df_final_30 = df_final.limit(30)


df_final = df_final.toPandas()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-fda9e568de0b> in <cell line: 0>()
----> 1 df_final = df_final.toPandas()

NameError: name 'df_final' is not defined
```

```
df_sentiment_all.select('sentiment').printSchema()
```

```
root
 |-- sentiment: struct (nullable = true)
 |    |-- polarity: double (nullable = true)
```

```
 |    |-- subjectivity: double (nullable = true)
```

```
from transformers import pipeline

# Load zero-shot classification model
zero_shot_classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")
```

```
df_final.printSchema()
```

```
candidate_labels = ["positive", "neutral", "negative"]

# Predict
zs_results = []
for review in df_final:
    result = zero_shot_classifier(review, candidate_labels)
    zs_results.append({
        "label": result["labels"][0],
        "confidence": result["scores"][0]
    })

# Attach to DataFrame
df_final["zs_label"] = [r["label"] for r in zs_results]
df_final["zs_confidence"] = [r["confidence"] for r in zs_results]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-c7af8377e037> in <cell line: 0>()
      3 # Predict
      4 zs_results = []
----> 5 for review in df_final:
      6     result = zero_shot_classifier(review, candidate_labels)
      7     zs_results.append({

NameError: name 'df_final' is not defined
```

```
from pyspark.sql.functions import when, length, col

df_bucketed = df_reviews.withColumn("review_length", length(col("reviews")))

df_bucketed = df_bucketed.withColumn(
    "length_bucket",
    when(col("review_length") < 100, "Short")
    .when(col("review_length") <= 300, "Medium")
    .otherwise("Long")
)
```

```
from pyspark.sql import functions as F

# Group count
group_counts = df_bucketed.groupBy("standard_category", "length_bucket").count()

# Total rows
total_count = df_bucketed.count()

# Target sample size
sample_target = 10000

# Compute fraction per group
group_fractions = group_counts.withColumn(
    "fraction", (F.col("count") / total_count) * sample_target
).withColumn(
    "sample_size", F.round("fraction").cast("int")
)
```

```
sampled_dfs = []

for row in group_fractions.collect():
    cat = row['standard_category']
    bucket = row['length_bucket']
    n = row['sample_size']
```

```
        if n > 0:
            subset = (
                df_bucketed.filter(
                    (col("standard_category") == cat) &
                    (col("length_bucket") == bucket)
                )
                .orderBy(F.rand())
                .limit(n)
            )
            sampled_dfs.append(subset)



df_sample_10k = sampled_dfs[0]
for sdf in sampled_dfs[1:]:
    df_sample_10k = df_sample_10k.union(sdf)



output_path = "gs://final_dataset_dat490/sample_reviews_stratified_10k.parquet"


df_sample_10k.write.mode("overwrite").parquet(output_path)


import pandas as pd
df_sample_pandas = pd.read_parquet(output_path)
df_sample_pandas = df_sample_pandas.sort_values(["gmap_id", "timestamp"]).reset_index(drop=True)
df_sample_pandas.columns
```

```
Index(['gmap_id', 'customer_name', 'rating', 'reviews', 'time', 'avg_rating',
       'category', 'latitude', 'longitude', 'business_name', 'num_of_reviews',
       'state', 'standard_category', 'Monday', 'Tuesday', 'Wednesday',
       'Thursday', 'Friday', 'Saturday', 'Sunday', 'timestamp', 'week',
       'month', 'year', 'time_seconds', 'review_length', 'length_bucket'],
      dtype='object')
```

```
df_sample_pandas[['reviews']]
```

	reviews
0	Nice clean place and very friendly staff. Only...
1	This company helped us buy our land 15 years a...
2	Great place for kids and families to do sports...
3	He was able to fit me in at the last minute. A...
4	I have had good experiences here for the past ...
...	...
9996	Store was disorganized and sales representativ...
9997	Treated as a number, not a patient. Rushed by ...
9998	This is such a great place to grab dinner. The...
9999	Dividers and spots clearly marked on floor, pl...
10000	Travelers, this is the cape cod thrift store e...

10001 rows × 1 columns

```
# pip install pandas transformers openpyxl

# from google.colab import drive
# drive.mount('/content/drive')

# import pandas as pd
# df_reviews_30 = pd.read_excel('/content/drive/MyDrive/Reviews std DAT490.xlsx')
# df_reviews_30.head()

# from transformers import pipeline

# classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")
# labels = ["positive", "neutral", "negative"]

# score = []
# for review in df_reviews_30['Reviews']:
```

```
#   prediction = classifier(review, candidate_labels=labels)
#   top_label = prediction["labels"][0]
#   score.append(top_label)

# df_reviews_30['Sentiment'] = score
# df_reviews_30

# df_reviews_30.to_excel('/content/drive/MyDrive/Reviews std DAT490 labelled.xlsx')


!pip install vaderSentiment
```

```
Collecting vaderSentiment
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl.metadata (572 bytes)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from vaderSentiment) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->vaderSentiment) (3.4.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->vaderSentiment) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->vaderSentiment) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->vaderSentiment) (2025.1.31)
Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 126.0/126.0 kB 4.0 MB/s eta 0:00:00
Installing collected packages: vaderSentiment
Successfully installed vaderSentiment-3.3.2
```

```
from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from tqdm import tqdm

tqdm.pandas()  # Progress bar for apply



analyzer = SentimentIntensityAnalyzer()

def get_textblob_polarity(text):
    try:
        return TextBlob(text).sentiment.polarity
    except:
        return None

def get_vader_compound(text):
    try:
        return analyzer.polarity_scores(text)["compound"]
    except:
        return None



# Assuming df["reviews"] contains your text
df_sample_pandas["vader_polarity"] = df_sample_pandas["reviews"].progress_apply(get_textblob_polarity)
df_sample_pandas["textblob_sentiment_score"] = df_sample_pandas["reviews"].progress_apply(get_vader_compound)
```

```
100%|██████████| 10001/10001 [00:03<00:00, 2803.51it/s]
100%|██████████| 10001/10001 [00:03<00:00, 2917.04it/s]
```

```
def label_textblob(p):
    if p >= 0.05:
        return "Positive"
    elif p <= -0.05:
        return "Negative"
    else:
        return "Neutral"

def label_vader(s):
    if s >= 0.05:
        return "Positive"
    elif s <= -0.05:
        return "Negative"
    else:
        return "Neutral"

# 🔁 Updated column names
df_sample_pandas["textblob_label"] = df_sample_pandas["vader_polarity"].apply(label_textblob)
df_sample_pandas["vader_label"] = df_sample_pandas["textblob_sentiment_score"].apply(label_vader)
df_sample_pandas.head()
```

	gmap_id	customer_name	rating	reviews	time	avg_rating	category	latitude	longitude
0	0x0:0xde4ab363e58baf8	Glen Sikorski	5	Nice clean place and very friendly staff. Only...	1619963856471	4.2	Restaurant, Bar, Caterer, Event venue	45.120739	-91.448845
1	0x145e95d513a77c99:0x7aad3c9a54c17e9f	stephen travers	5	This company helped us buy our land 15 years a...	1621983091409	3.4	Property management company, Real estate agency	29.730017	-99.075182
2	0x14e037302ebfe6bd:0x483c80e39ebb0ab7	Shawn Bebej	5	Great place for kids and families to do sports...	1578142093772	4.9	Non-profit organization	34.233393	-118.590456
3	0x14e3db41cf753ebd:0x1d6536e7c20051ef	L C	5	He was able to fit me in at the last minute. A...	1616604264474	4.8	Window tinting service	33.546954	-112.202525
4	0x1520f8e750be33d7:0xc5e501b57143e755	Lorelei Flaherty	4	I have had good experiences here for the past ...	1477072334301	1.6	Mental health clinic	35.131101	-106.510093

5 rows × 31 columns

```
import torch
print(torch.cuda.is_available())
```

```
True
```

```
from transformers import pipeline
from tqdm import tqdm

# Enable progress bar
tqdm.pandas()
```

```
config.json: 100%  1.15k/1.15k [00:00<00:00, 74.2kB/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For better perfo
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to r
model.safetensors: 100%  1.63G/1.63G [00:10<00:00, 244MB/s]
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 2.94kB/s]
vocab.json: 100%  899k/899k [00:00<00:00, 21.7MB/s]
merges.txt: 100%  456k/456k [00:01<00:00, 335kB/s]
tokenizer.json: 100%  1.36M/1.36M [00:00<00:00, 30.7MB/s]
Device set to use cuda:0
```

```
# Load zero-shot classification pipeline with BART
zero_shot_classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli", device=0)


candidate_labels = ["positive", "neutral", "negative"]

def classify_bart(text):
    try:
        result = zero_shot_classifier(text, candidate_labels)
        return result["labels"][0].capitalize()  # Most likely label
    except:
        return None  # Handle any errors gracefully



df_sample_pandas["bart_label"] = df_sample_pandas["reviews"].progress_apply(classify_bart)
```

```
  0%|          | 10/10001 [00:02<22:20,  7.45it/s]You seem to be using the pipelines sequentially on GPU. In order to maximize efficienc
100%|██████████| 10001/10001 [14:47<00:00, 11.27it/s]
```

```
df_sample_pandas.head()
```

	gmap_id	customer_name	rating	reviews	time	avg_rating	category	latitude	longitude
0	0x0:0xde4ab363e58baf8	Glen Sikorski	5	Nice clean place and very friendly staff. Only...	1619963856471	4.2	Restaurant, Bar, Caterer, Event venue	45.120739	-91.448845
1	0x145e95d513a77c99:0x7aad3c9a54c17e9f	stephen travers	5	This company helped us buy our land 15 years a...	1621983091409	3.4	Property management company, Real estate agency	29.730017	-99.075182
2	0x14e037302ebfe6bd:0x483c80e39ebb0ab7	Shawn Bebej	5	Great place for kids and families to do sports...	1578142093772	4.9	Non-profit organization	34.233393	-118.590456
3	0x14e3db41cf753ebd:0x1d6536e7c20051ef	L C	5	He was able to fit me in at the last minute. A...	1616604264474	4.8	Window tinting service	33.546954	-112.202525
4	0x1520f8e750be33d7:0xc5e501b57143e755	Lorelei Flaherty	4	I have had good experiences here for the past ...	1477072334301	1.6	Mental health clinic	35.131101	-106.510093

5 rows × 32 columns

```
import pandas as pd

# Count label frequencies per method
label_counts = pd.DataFrame({
    "TextBlob": df_sample_pandas["textblob_label"].value_counts(),
    "VADER": df_sample_pandas["vader_label"].value_counts(),
    "BART": df_sample_pandas["bart_label"].value_counts()
}).fillna(0).astype(int)

# Reorder rows for consistency
label_counts = label_counts.reindex(["Positive", "Neutral", "Negative"])



# Melt into long format
df_melted = label_counts.T.reset_index().melt(
    id_vars="index", var_name="Sentiment", value_name="Count"
)
df_melted.rename(columns={"index": "Method"}, inplace=True)



import seaborn as sns
import matplotlib.pyplot as plt

# Set seaborn style
sns.set(style="whitegrid")

# Define consistent color palette
palette = {
    "Positive": "#4CAF50",
    "Neutral": "#FFC107",
    "Negative": "#F44336"
}

# Create grouped bar plot
plt.figure(figsize=(10, 6))
sns.barplot(
    data=df_melted,
```

```
    x="Method", y="Count", hue="Sentiment",
    palette=palette
)

# Customize the chart
plt.title("Sentiment Label Distribution (10,000 Reviews)", fontsize=14)
plt.xlabel("Sentiment Analysis Method")
plt.ylabel("Number of Reviews")
plt.legend(title="Sentiment")
plt.tight_layout()
plt.show()
```

Similar Businesses Recommendation System - California Restaurant Businesses

```
import pandas as pd
df_ca_pd = pd.read_csv('gs://final_dataset_dat490/df_ca_pd.csv')
df_ca_pd
```

	gmap_id	customer_name	rating	reviews	time	avg_rating	category	latitude	l
0	0x100277924bd075cf:0x580587e75a2e5930	pdangelo4 D	5	Great food, great price for the location, grea...	1499130935024	4.2	Greek restaurant, Cafe	32.709388	-1
1	0x14e035bf1d6e56af:0x4b5a1833b3b81a05	Bill Taylor	5	The spot is first-rate and spacious, the meals...	1616517663786	4.5	Thai restaurant, Asian restaurant, Southeast A...	34.179015	-1
2	0x14e17823cf45b877:0x51beb951786095fc	Nora Elfassi	5	We were visiting long beach and it was mother'...	1561056772859	3.6	Southern restaurant (US), Bakery, Cafe, Wine bar	33.770632	-1
3	0x14e1783b3ea4ac25:0xf80967990e6cfae4	James Whyte	4	Nice guys. Excellent crust. Good cheese. I wi...	1619848194853	4.5	Pizza restaurant, Italian restaurant, Delivery...	33.766830	-1
4	0x14e17940e6ecd36f:0xf82c7160bbe14c8f	Mike Blanchard	5	This place is awesome. Great place to ride, bo...	1602873670030	4.7	Ranch	33.727337	-1
...	...	...	...	...	...	...	...	...	
306380	0x89c254c644e96bcd:0x1639f94a5ddba5e6	Renee Ray	4	This office is fairly easy in and great locati...	1581464356371	2.7	Social security office	38.729066	-1
306381	0x89c2984884ddacbb:0x2f86b4fc0b9352f4	Julie Hsieh	5	So sad to see iconic American store being boug...	1592275561488	4.3	Jewelry store, Gift shop, Jewelry designer, Je...	34.058235	-1
306382	0x89c624a51b8bc7b7:0xa131bcf9dc1ed36a	Haley Hill	5	As a photographer, I've gotten to work pretty ...	1586989353540	4.2	Magazine store, Advertising agency, Newspaper ...	32.715567	-1
306383	0x89d312304c2eba19:0x1f0feb9ba12e4a1e	L Scott	5	The food is amazing! Like seriously if you're ...	1547411188107	4.2	Seafood restaurant, Restaurant	33.989595	-1
306384	0x89e4bb7c470ffaf5:0xb86a6e5f2b6d1ec9	Halima Crenshaw	1	I got really good deals but I gave this a 2 be...	1573798325698	4.3	Women's clothing store, Clothing store, Jeans ...	38.767505	-1

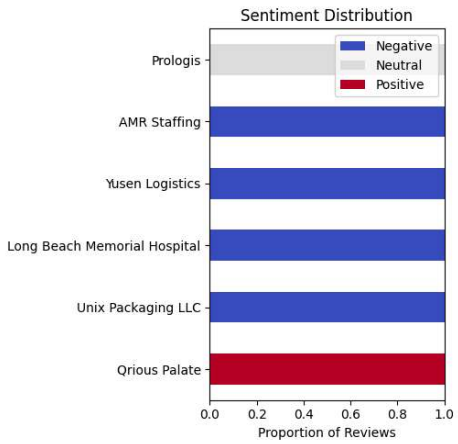
306385 rows × 27 columns

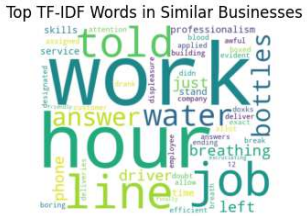
```
# Full KNN Pipeline with Geographic and POS-Aware Recommendations

import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.neighbors import NearestNeighbors
from scipy.sparse import hstack
import numpy as np
import spacy
from math import radians, cos, sin, asin, sqrt

nlp = spacy.load("en_core_web_sm")
```

```
# Step 1: Aggregate business-level data with sentiment distribution
def sentiment_distribution(group):
    counts = group['label'].value_counts(normalize=True)
    return pd.Series({
        'label_neg': counts.get(0, 0.0),
        'label_neu': counts.get(1, 0.0),
        'label_pos': counts.get(2, 0.0),
        'avg_label': group['label'].mean() or 0.0,
        'avg_rating': group['avg_rating'].mean() or 0.0,
        'num_of_reviews': group['num_of_reviews'].mean() or 0.0,
        'business_name': group['business_name'].iloc[0],
        'category': group['category'].iloc[0],
        'state': group['state'].iloc[0],
        'standard_category': group['standard_category'].iloc[0],
        'latitude': group['latitude'].iloc[0],
        'longitude': group['longitude'].iloc[0],
        'reviews': ' '.join(group['reviews']),
        'Monday': group['Monday'].iloc[0] or 0,
        'Tuesday': group['Tuesday'].iloc[0] or 0,
        'Wednesday': group['Wednesday'].iloc[0] or 0,
        'Thursday': group['Thursday'].iloc[0] or 0,
        'Friday': group['Friday'].iloc[0] or 0,
        'Saturday': group['Saturday'].iloc[0] or 0,
        'Sunday': group['Sunday'].iloc[0] or 0,
    })

day_cols = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
df_ca_pd[day_cols] = df_ca_pd[day_cols].fillna(0)
grouped = df_ca_pd.groupby('gmap_id', group_keys=False).apply(sentiment_distribution).reset_index()

# Step 2: TF-IDF on reviews
tfidf = TfidfVectorizer(max_features=500, stop_words='english')
tfidf_matrix = tfidf.fit_transform(grouped['reviews'])

# Step 3: Metadata feature extraction
preprocessor = ColumnTransformer([
    ('num', StandardScaler(), ['avg_rating', 'avg_label', 'num_of_reviews']),
    ('cat', OneHotEncoder(handle_unknown='ignore'), ['standard_category', 'state']),
    ('days', 'passthrough', day_cols)
])

meta_matrix = preprocessor.fit_transform(grouped)

# Step 4: Combine all features
X_combined = hstack([meta_matrix, tfidf_matrix]).tocsr()

# Step 5: Fit KNN
knn = NearestNeighbors(n_neighbors=6, metric='cosine')
knn.fit(X_combined)

# Step 6: Haversine distance helper
def haversine(lon1, lat1, lon2, lat2):
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371  # Radius of Earth in km
    return c * r

# Step 7: POS-aware improvement tool with geo-filter
def evaluate_and_recommend_local(idx, business_df, X_csr, knn, top_n=5, radius_km=25):
    from sklearn.feature_extraction.text import CountVectorizer

    target = business_df.iloc[idx]
    lat1, lon1 = target['latitude'], target['longitude']

    # Filter businesses within radius
    business_df['distance'] = business_df.apply(lambda row: haversine(lon1, lat1, row['longitude'], row['latitude']), axis=1)
    nearby_df = business_df[business_df['distance'] <= radius_km].reset_index(drop=True)

    if nearby_df.shape[0] < top_n + 1:
        print("Not enough nearby businesses for comparison.")
        return

    tfidf_matrix_local = tfidf.transform(nearby_df['reviews'])
```

```
meta_matrix_local = preprocessor.transform(nearby_df)
X_local = hstack([meta_matrix_local, tfidf_matrix_local]).tocsr()

knn_local = NearestNeighbors(n_neighbors=top_n+1, metric='cosine')
knn_local.fit(X_local)

query_vector = X_local[0]  # Target is first row of nearby_df
distances, indices = knn_local.kneighbors(query_vector)

neighbors = nearby_df.iloc[indices[0][1:top_n+1]].copy()

# Keyword comparison with POS tagging
target_reviews = target['reviews']
neighbor_reviews = neighbors['reviews'].str.cat(sep=' ')

vec = CountVectorizer(stop_words='english', max_features=30)
neighbor_vec = vec.fit_transform([neighbor_reviews])
target_vec = vec.transform([target_reviews])

keywords = vec.get_feature_names_out()
neighbor_counts = neighbor_vec.toarray()[0]
target_counts = target_vec.toarray()[0]

missing_keywords = [word for word, n_c, t_c in zip(keywords, neighbor_counts, target_counts) if n_c > 1 and t_c == 0]

# Use spaCy to filter meaningful keywords
actionable_keywords = []
for word in missing_keywords:
    doc = nlp(word)
    if doc and doc[0].pos_ in {"NOUN", "ADJ", "VERB"}:
        actionable_keywords.append(word)

# Simple suggestions from actionable keywords
recommendations = [f"Consider improving around: '{k}'" for k in actionable_keywords]

# Plot ratings, sentiment, word cloud
names = [target['business_name']] + neighbors['business_name'].tolist()
ratings = [target['avg_rating']] + neighbors['avg_rating'].tolist()
negs = [target['label_neg']] + neighbors['label_neg'].tolist()
neus = [target['label_neu']] + neighbors['label_neu'].tolist()
poss = [target['label_pos']] + neighbors['label_pos'].tolist()

plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.barh(names, ratings)
plt.title("Avg Ratings Comparison")
plt.xlabel("Rating")
plt.xlim(0, 5)

plt.subplot(1, 3, 2)
bar_data = pd.DataFrame({'Negative': negs, 'Neutral': neus, 'Positive': poss}, index=names)
bar_data.plot(kind='barh', stacked=True, ax=plt.gca(), colormap='coolwarm')
plt.title("Sentiment Distribution")
plt.xlabel("Proportion of Reviews")
plt.xlim(0, 1)

# TF-IDF-based word cloud (better focus)
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_kw = TfidfVectorizer(stop_words='english', max_features=50)
X_kw = tfidf_kw.fit_transform([neighbor_reviews])
scores = X_kw.toarray().flatten()
keywords = tfidf_kw.get_feature_names_out()
word_scores = dict(zip(keywords, scores))

wordcloud = WordCloud(width=500, height=400, background_color='white').generate_from_frequencies(word_scores)
plt.subplot(1, 3, 3)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title("Top TF-IDF Words in Similar Businesses")


plt.tight_layout()
plt.show()

print("\n📝 Sample Review Snippets from Similar Businesses:")
for review in neighbors['reviews'].sample(min(5, len(neighbors))):
```

```
        print(f"- {review[:200]}...")


# Usage:
evaluate_and_recommend_local(idx=41, business_df=grouped, X_csr=X_combined, knn=knn, radius_km=25)
```






```
📝 Sample Review Snippets from Similar Businesses:
- Long Beach Memorial Hospital is AWFUL. The nurses there have ZERO people skills. I was thete over 3 hrs in excruciating pain and blood
- There's a no kissing policy...
- I work as a driver and make Deliveries to Yusen. They are without doubt the least efficient and the least driver friendly company I ha
- Came in through Partners Personnel on a Monday. Started work at 7 and was sent to work on a line that simply boxed a never ending line
- Worst customer service ever! the staffing has bad professionalism. They can never answer the phone and whenever they do answer the phc
```

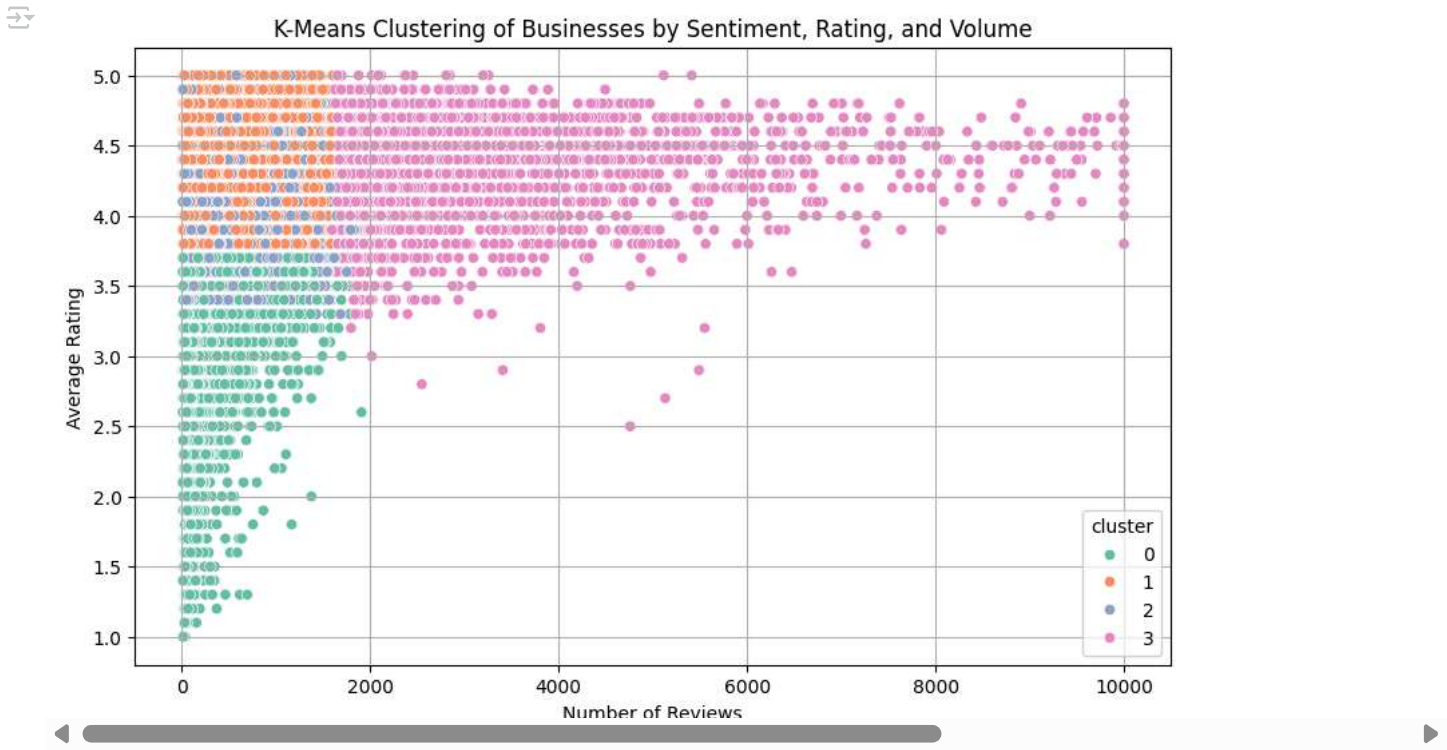
K-means Clustering

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Define features for clustering
features = ["avg_rating", "num_of_reviews", "label"]

# Drop rows with missing data
df_kmeans = df_ca_pd.dropna(subset=features).copy()

# Scale features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df_kmeans[features])

# Fit K-Means
kmeans = KMeans(n_clusters=4, random_state=42)
df_kmeans["cluster"] = kmeans.fit_predict(scaled_features)


df_kmeans.groupby("cluster")[features].mean().round(2)
```

cluster	avg_rating	num_of_reviews	label
0	3.26	131.66	1.36
1	4.43	181.28	2.00
2	4.23	201.05	0.63
3	4.36	3003.97	1.65

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
```

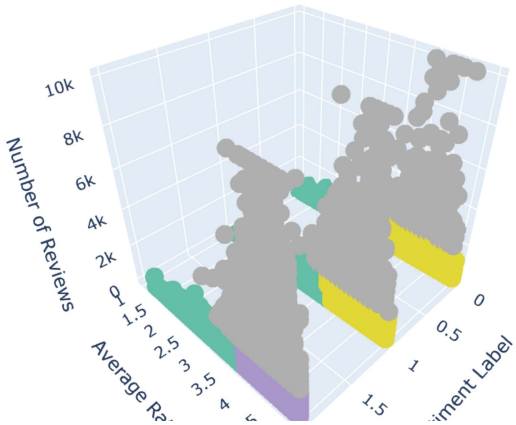
```
sns.scatterplot(
    data=df_kmeans,
    x="num_of_reviews",
    y="avg_rating",
    hue="cluster",
    palette="Set2"
)
plt.title("K-Means Clustering of Businesses by Sentiment, Rating, and Volume")
plt.xlabel("Number of Reviews")
plt.ylabel("Average Rating")
plt.grid(True)
plt.show()
```




```
import plotly.express as px
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Prepare data
df_cluster = df_ca_pd[["label", "avg_rating", "num_of_reviews"]].dropna().copy()

# Scale features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df_cluster)

# Apply KMeans clustering
kmeans = KMeans(n_clusters=4, random_state=42)
df_cluster["cluster"] = kmeans.fit_predict(scaled_features)

# Create interactive 3D plot
fig = px.scatter_3d(
    df_cluster,
    x="label",
    y="avg_rating",
    z="num_of_reviews",
    color="cluster",
    title="📊 3D Clustering by Sentiment, Rating, and Review Volume",
    labels={
        "label": "Sentiment Label",
        "avg_rating": "Average Rating",
        "num_of_reviews": "Number of Reviews",
        "cluster": "Cluster"
    },
    color_continuous_scale=px.colors.qualitative.Set2
)

fig.update_layout(
```

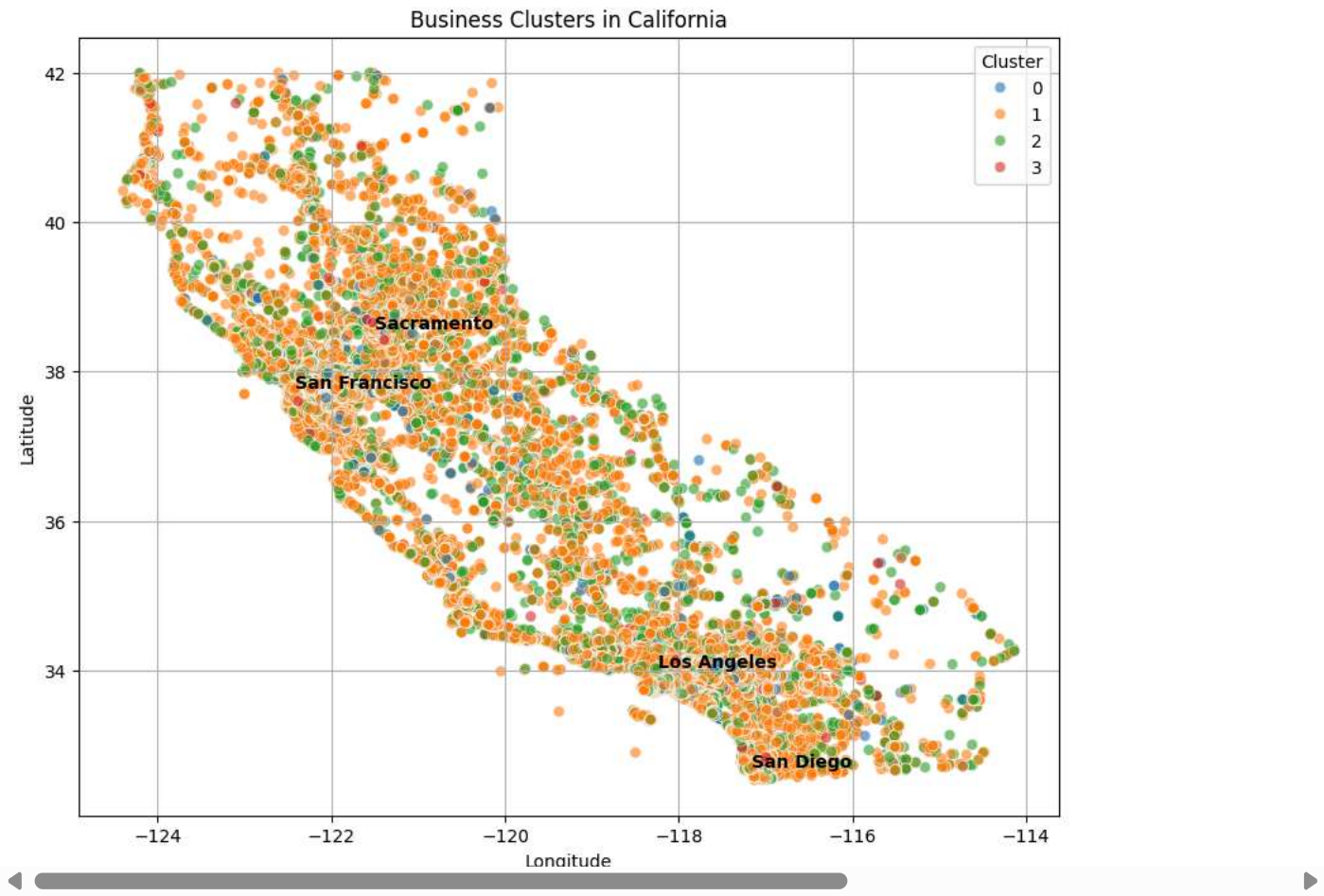
```
    legend_title_text='Cluster',
    scene=dict(
        xaxis_title='Sentiment Label',
        yaxis_title='Average Rating',
        zaxis_title='Number of Reviews'
    )
)

fig.show()
```

3D Clustering by Sentiment, Rating, and Review Volume



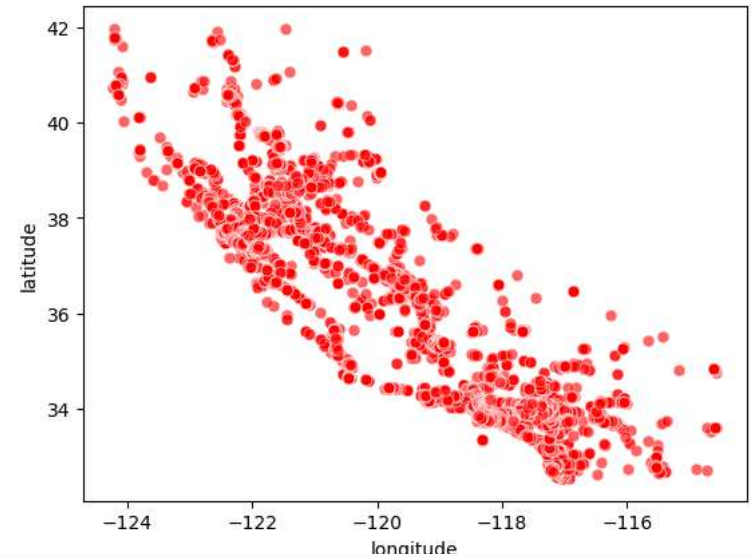

```
import matplotlib.pyplot as plt
import seaborn as sns

# Drop outliers
df_kmeans = df_kmeans[
    (df_kmeans["longitude"] >= -125) & (df_kmeans["longitude"] <= -113) &
    (df_kmeans["latitude"] >= 32) & (df_kmeans["latitude"] <= 43)
]

# City labels
city_coords_ca = {
    "Los Angeles": (-118.2437, 34.0522),
    "San Francisco": (-122.4194, 37.7749),
    "San Diego": (-117.1611, 32.7157),
    "Sacramento": (-121.4944, 38.5816)
}

# Plot
plt.figure(figsize=(10, 8))
palette = sns.color_palette("tab10", n_colors=df_kmeans["cluster"].nunique())

sns.scatterplot(
    data=df_kmeans,
    x="longitude",
    y="latitude",
    hue="cluster",
    palette=palette,
    alpha=0.6,
    s=40
)

# Add city labels
for city, (lon, lat) in city_coords_ca.items():
    plt.text(lon, lat, city, fontsize=10, weight="bold")

plt.title("Business Clusters in California")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
```

```
plt.grid(True)
plt.legend(title="Cluster", loc="upper right")
plt.show()
```



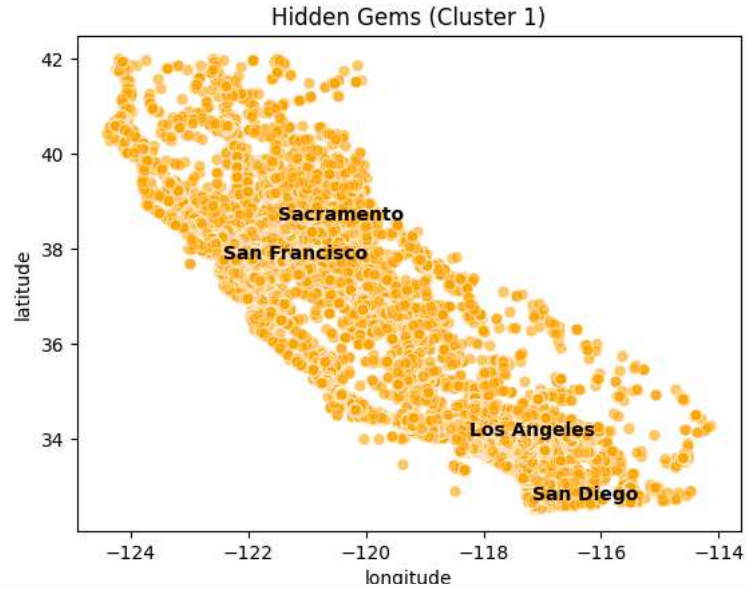

```
import matplotlib.pyplot as plt
import seaborn as sns

df_kmeans = df_kmeans[
    (df_kmeans["longitude"] >= -125) & (df_kmeans["longitude"] <= -113) &
    (df_kmeans["latitude"] >= 32) & (df_kmeans["latitude"] <= 43)
]
top_cluster = df_kmeans[df_kmeans["cluster"] == 0]

sns.scatterplot(
    data=top_cluster,
    x="longitude",
    y="latitude",
    color="red",
    alpha=0.6,
    s=40
)
for city, (lon, lat) in city_coords_ca.items():
    plt.text(lon, lat, city, fontsize=10, weight="bold")

plt.title("Struggling businesses (Cluster 0)")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-28-b1fa8c4c7de9> in <cell line: 0>()
     16     s=40
     17 )
---> 18 for city, (lon, lat) in city_coords_ca.items():
     19     plt.text(lon, lat, city, fontsize=10, weight="bold")
     20

NameError: name 'city_coords_ca' is not defined
```



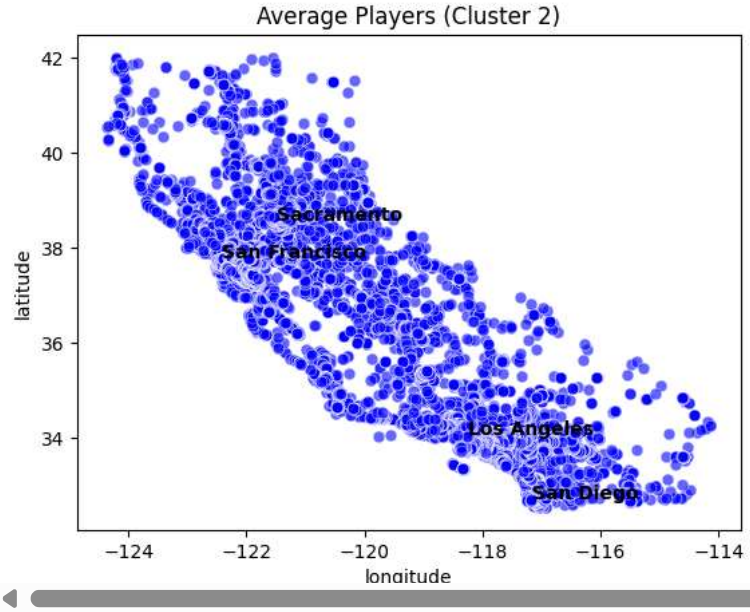

```
import matplotlib.pyplot as plt
import seaborn as sns

df_kmeans = df_kmeans[
    (df_kmeans["longitude"] >= -125) & (df_kmeans["longitude"] <= -113) &
    (df_kmeans["latitude"] >= 32) & (df_kmeans["latitude"] <= 43)
]
top_cluster = df_kmeans[df_kmeans["cluster"] == 1]

sns.scatterplot(
    data=top_cluster,
    x="longitude",
    y="latitude",
    color="orange",
    alpha=0.6,
    s=40
)
for city, (lon, lat) in city_coords_ca.items():
    plt.text(lon, lat, city, fontsize=10, weight="bold")

plt.title("Hidden Gems (Cluster 1)")
```

```
Text(0.5, 1.0, 'Hidden Gems (Cluster 1)')
```



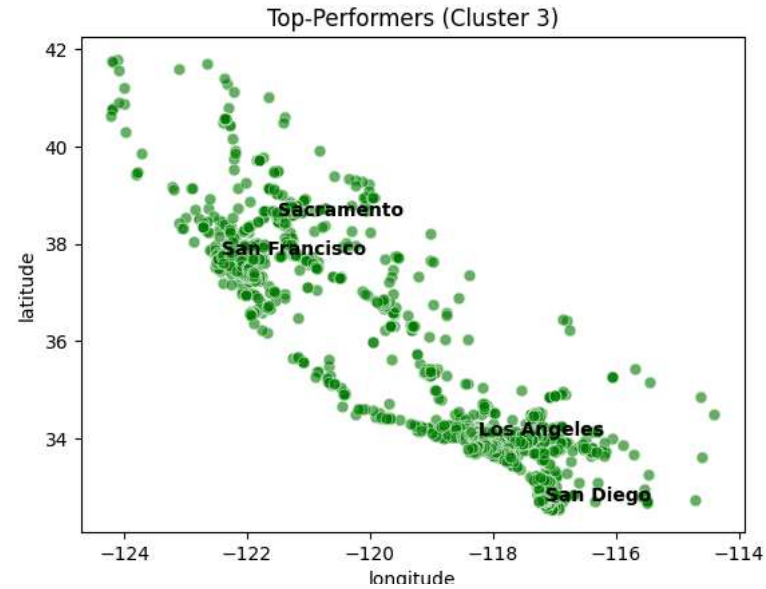

```
import matplotlib.pyplot as plt
import seaborn as sns

df_kmeans = df_kmeans[
    (df_kmeans["longitude"] >= -125) & (df_kmeans["longitude"] <= -113) &
    (df_kmeans["latitude"] >= 32) & (df_kmeans["latitude"] <= 43)
]
top_cluster = df_kmeans[df_kmeans["cluster"] == 2]

sns.scatterplot(
    data=top_cluster,
    x="longitude",
    y="latitude",
    color="blue",
    alpha=0.6,
    s=40
)
for city, (lon, lat) in city_coords_ca.items():
    plt.text(lon, lat, city, fontsize=10, weight="bold")

plt.title("Average Players (Cluster 2)")
```

```
Text(0.5, 1.0, 'Average Players (Cluster 2)')
```



```
import matplotlib.pyplot as plt
import seaborn as sns

df_kmeans = df_kmeans[
    (df_kmeans["longitude"] >= -125) & (df_kmeans["longitude"] <= -113) &
    (df_kmeans["latitude"] >= 32) & (df_kmeans["latitude"] <= 43)
]
top_cluster = df_kmeans[df_kmeans["cluster"] == 3]

sns.scatterplot(
    data=top_cluster,
    x="longitude",
    y="latitude",
    color="green",
    alpha=0.6,
    s=40
)
for city, (lon, lat) in city_coords_ca.items():
    plt.text(lon, lat, city, fontsize=10, weight="bold")

plt.title("Top-Performers (Cluster 3)")
```

```
Text(0.5, 1.0, 'Top-Performers (Cluster 3)')
```




```
top_in_cluster = df_kmeans[df_kmeans['cluster'] == 3].sort_values(by='num_of_reviews', ascending=False).head()
top_in_cluster
```

	gmap_id	customer_name	rating	reviews	time	avg_rating	category	latitude	long
303349	0x80deab043d0c63e5:0xa61214169a0cdb0a	Daniel Lipton	3	As a piece of history, this is a fascinating s...	1616523208295	4.7	Community center, Art center, Art gallery, Art...	32.737601	-117.2
108796	0x808580668e8f3159:0x6cf7a313d6a53ec7	Michael Saxby	5	Great views of the Bay and quite a few interes...	1617203383685	4.6	Historical landmark, Ferry service, Market	37.795838	-122.3
149326	0x80dd1ffeb7628365:0x1f6a4c1815205025	Aley Eldin Tohamy	4	Amazing place, nice weather but a lot of peopl...	1617040488281	4.6	Fishing pier, Public beach, Tourist attraction	33.607468	-117.9
191543	0x80d951fb495fe7f1:0xf67bc8a99bedc791	Natasha Valdez	5	I love this mall! They had absolutely	1617690834583	4.5	Shopping mall, Clothing store, Event venue, Corporate	32.655904	-117.0

```
!pip install geopandas contextily
```

5488	0x808586d510144681:0xe960db8a581aacf5	Marija Dordevic	5	must see if you are visiti...	1620943816541	4.7	Corporate entertainment service, ...	37.802931	-122.4

5 rows × 28 columns

```
Requirement already satisfied: geopandas in /usr/local/lib/python3.11/dist-packages (1.0.1)
Collecting contextily
  Downloading contextily-1.6.2-py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: numpy>=1.22 in /usr/local/lib/python3.11/dist-packages (from geopandas) (2.0.2)
Requirement already satisfied: pyogrio>=0.7.2 in /usr/local/lib/python3.11/dist-packages (from geopandas) (0.10.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from geopandas) (24.2)
Requirement already satisfied: pandas>=1.4.0 in /usr/local/lib/python3.11/dist-packages (from geopandas) (2.2.2)
Requirement already satisfied: pyproj>=3.3.0 in /usr/local/lib/python3.11/dist-packages (from geopandas) (3.7.1)
Requirement already satisfied: shapely>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from geopandas) (2.1.0)
Requirement already satisfied: geopy in /usr/local/lib/python3.11/dist-packages (from contextily) (2.4.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from contextily) (3.10.0)
Collecting mercantile (from contextily)
  Downloading mercantile-1.2.1-py3-none-any.whl.metadata (4.8 kB)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from contextily) (11.1.0)
Collecting rasterio (from contextily)
  Downloading rasterio-1.4.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.1 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from contextily) (2.32.3)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from contextily) (1.4.2)
Requirement already satisfied: xyzservices in /usr/local/lib/python3.11/dist-packages (from contextily) (2025.1.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.4.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.4.0->geopandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.4.0->geopandas) (2025.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from pyogrio>=0.7.2->geopandas) (2025.1.31)
Requirement already satisfied: geographiclib<3,>=1.52 in /usr/local/lib/python3.11/dist-packages (from geopy->contextily) (2.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->contextily) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->contextily) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->contextily) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->contextily) (1.4.8)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->contextily) (3.2.3)
Requirement already satisfied: click>=3.0 in /usr/local/lib/python3.11/dist-packages (from mercantile->contextily) (8.1.8)
Collecting affine (from rasterio->contextily)
  Downloading affine-2.4.0-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: attrs in /usr/local/lib/python3.11/dist-packages (from rasterio->contextily) (25.3.0)
Collecting cligj>=0.5 (from rasterio->contextily)
```