```
!CMAKE_ARGS="-DLLAMA_CUBLAS=on" FORCE_CMAKE=1 pip install llama-cpp-
python==0.2.45 --force-reinstall --no-cache-dir -q

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 36.7/36.7 MB 94.5 MB/s eta
0:00:00
ents to build wheel ... etadata (pyproject.toml) ...
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.1/62.1 kB 323.7 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.5/45.5 kB 287.1 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.9/134.9 kB 397.9 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.6/16.6 MB 335.0 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.6/44.6 kB 225.8 MB/s eta
0:00:00
a-cpp-python (pyproject.toml) ... ERROR: pip's dependency resolver
does not currently take into account all the packages that are
installed. This behaviour is the source of the following dependency
conflicts.
opencv-contrib-python 4.12.0.88 requires numpy<2.3.0,>=2;
python_version >= "3.9", but you have numpy 2.3.4 which is
incompatible.
opencv-python 4.12.0.88 requires numpy<2.3.0,>=2; python_version >=
"3.9", but you have numpy 2.3.4 which is incompatible.
tensorflow 2.19.0 requires numpy<2.2.0,>=1.26.0, but you have numpy
2.3.4 which is incompatible.
opencv-python-headless 4.12.0.88 requires numpy<2.3.0,>=2;
python_version >= "3.9", but you have numpy 2.3.4 which is
incompatible.
numba 0.60.0 requires numpy<2.1,>=1.22, but you have numpy 2.3.4 which
is incompatible.
cupy-cuda12x 13.3.0 requires numpy<2.3,>=1.22, but you have numpy
2.3.4 which is incompatible.

!pip install huggingface_hub

Requirement already satisfied: huggingface_hub in
/usr/local/lib/python3.12/dist-packages (0.36.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.12/dist-packages (from huggingface_hub)
(3.20.0)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.12/dist-packages (from huggingface_hub)
(2025.3.0)
Requirement already satisfied: packaging>=20.9 in
/usr/local/lib/python3.12/dist-packages (from huggingface_hub) (25.0)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.12/dist-packages (from huggingface_hub) (6.0.3)
Requirement already satisfied: requests in
```

```
/usr/local/lib/python3.12/dist-packages (from huggingface_hub)
(2.32.4)
Requirement already satisfied: tqdm>=4.42.1 in
/usr/local/lib/python3.12/dist-packages (from huggingface_hub)
(4.67.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface_hub)
(4.15.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface_hub) (1.2.0)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests-
>huggingface_hub) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests-
>huggingface_hub) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests-
>huggingface_hub) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests-
>huggingface_hub) (2025.10.5)
```

```
import pandas as pd
from huggingface_hub import hf_hub_download
from llama_cpp import Llama
import json

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

```
Mounted at /content/drive
```

```
data = pd.read_csv("/content/drive/MyDrive/restaurant_reviews.csv")

data.head()
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 20,\n  \"fields\":
[\n    {\n      \"column\": \"restaurant_ID\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 20,\
n        \"samples\": [\n          \"FLV202\",\n          \"PIZ555\",\
n          \"CRV333\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating_review\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 5,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          5,\n          3,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"review_full\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 20,\n        \"samples\": [\n
```

\"Totally in love with the Auro of the place, really beautiful and quite fancy at the same time. The ambience is very pure and gives a sense of positivity throughout. Outdoor and indoor interior are quite quaint and cute. Love the open kitchen idea and there whole marketplace ideology. Due to coronovirus they specifically use disposable cutlery to keep the pandemic in mind taking all the precautionary measures from the beginning of the place with the mask on their staff and using good sanitisation. The food is really amazing specially the pizza straight from the oven and the hummus and pita bread are quite delicious too. If you're looking for a classy yet soothing Italian place in Delhi,Fatjar is a go to for you!\",\n \"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75 expecting great things but we were disappointed. The place has great trip advisor reviews so we had high expectations that were sadly not met. What really ruined our nice meal was the self service. A waiter gave us (a party of 5) two tablets to look at the menu on. Unlike with proper menus, where everyone can browse at their own speed we had to share a tablet between 5. We asked the waiter for more tablets and were given one. After we had eventually all used the tablets to browse and input our orders on to one tablet the tablet then had an error message and froze. We used the other tablet to order and that one ran out of battery. We were hungry after a long day travelling this was stressful. Eventually one of the waiters came over and took details of our order. I really disliked this way of ordering food, and would not use the restaurant again because of this. The table was also dirty, there was dust and bits of fluff on the table and cloth napkins and my aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment. Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n],\n \"semantic_type\": \"\",\n

```
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```

```
data['review_full'][3]
```

```
{"type":"string"}
```

```
data.shape
```

```
(20, 3)
```

```
data.isnull().sum()
```

```
restaurant_ID    0
rating_review    0
review_full      0
dtype: int64
```

```
model_name_or_path = "TheBloke/Llama-2-13B-chat-GGUF"
model_basename = "llama-2-13b-chat.Q5_K_M.gguf"
```

```
model_path = hf_hub_download(
    repo_id=model_name_or_path,
    filename=model_basename)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```

```
{"model_id":"e4c9bcddce894d148f6d397e60cfb36d","version_major":2,"vers
ion_minor":0}
```

```
Llama_llm = Llama(
    model_path=model_path,
    n_threads=2,
    n_batch=512,
    n_gpu_layers=43,
    n_ctx=4096,
)
```

```
llama_model_loader: loaded meta data with 19 key-value pairs and 363
tensors from /root/.cache/huggingface/hub/models--TheBloke--Llama-2-
13B-chat-GGUF/snapshots/4458acc949de0a9914c3eab623904d4fe999050a/
llama-2-13b-chat.Q5_K_M.gguf (version GGUF V2)
llama_model_loader: Dumping metadata keys/values. Note: KV overrides
```

```
do not apply in this output.
llama_model_loader: - kv   0:
general.architecture str              = llama
llama_model_loader: - kv   1:
general.name str              = LLaMA v2
llama_model_loader: - kv   2:
llama.context_length u32              = 4096
llama_model_loader: - kv   3:
llama.embedding_length u32              = 5120
llama_model_loader: - kv   4:
llama.block_count u32              = 40
llama_model_loader: - kv   5:
llama.feed_forward_length u32              = 13824
llama_model_loader: - kv   6:
llama.rope.dimension_count u32              = 128
llama_model_loader: - kv   7:
llama.attention.head_count u32              = 40
llama_model_loader: - kv   8:
llama.attention.head_count_kv u32              = 40
llama_model_loader: - kv   9:
llama.attention.layer_norm_rms_epsilon f32              = 0.000010
llama_model_loader: - kv  10:
general.file_type u32              = 17
llama_model_loader: - kv  11:
tokenizer.ggml.model str              = llama
llama_model_loader: - kv  12:
tokenizer.ggml.tokens arr[str,32000]   = ["<unk>", "<s>", "</s>",
"<0x00>", "<...
llama_model_loader: - kv  13:
tokenizer.ggml.scores arr[f32,32000]   = [0.000000, 0.000000,
0.000000, 0.0000...
llama_model_loader: - kv  14:
tokenizer.ggml.token_type arr[i32,32000]   = [2, 3, 3, 6, 6, 6, 6, 6,
6, 6, 6, 6, ...
llama_model_loader: - kv  15:
tokenizer.ggml.bos_token_id u32              = 1
llama_model_loader: - kv  16:
tokenizer.ggml.eos_token_id u32              = 2
llama_model_loader: - kv  17:
tokenizer.ggml.unknown_token_id u32              = 0
llama_model_loader: - kv  18:
general.quantization_version u32              = 2
llama_model_loader: - type  f32:   81 tensors
llama_model_loader: - type q5_K:  241 tensors
llama_model_loader: - type q6_K:   41 tensors
llm_load_vocab: special tokens definition check successful ( 259/32000
).
llm_load_print_meta: format           = GGUF V2
llm_load_print_meta: arch             = llama
```

```
llm_load_print_meta: vocab type       = SPM
llm_load_print_meta: n_vocab          = 32000
llm_load_print_meta: n_merges         = 0
llm_load_print_meta: n_ctx_train      = 4096
llm_load_print_meta: n_embd           = 5120
llm_load_print_meta: n_head           = 40
llm_load_print_meta: n_head_kv        = 40
llm_load_print_meta: n_layer          = 40
llm_load_print_meta: n_rot            = 128
llm_load_print_meta: n_embd_head_k    = 128
llm_load_print_meta: n_embd_head_v    = 128
llm_load_print_meta: n_gqa            = 1
llm_load_print_meta: n_embd_k_gqa     = 5120
llm_load_print_meta: n_embd_v_gqa     = 5120
llm_load_print_meta: f_norm_eps       = 0.0e+00
llm_load_print_meta: f_norm_rms_eps   = 1.0e-05
llm_load_print_meta: f_clamp_kqv      = 0.0e+00
llm_load_print_meta: f_max_alibi_bias = 0.0e+00
llm_load_print_meta: n_ff             = 13824
llm_load_print_meta: n_expert         = 0
llm_load_print_meta: n_expert_used    = 0
llm_load_print_meta: rope scaling     = linear
llm_load_print_meta: freq_base_train  = 10000.0
llm_load_print_meta: freq_scale_train = 1
llm_load_print_meta: n_yarn_orig_ctx  = 4096
llm_load_print_meta: rope_finetuned   = unknown
llm_load_print_meta: model type       = 13B
llm_load_print_meta: model ftype      = Q5_K - Medium
llm_load_print_meta: model params     = 13.02 B
llm_load_print_meta: model size       = 8.60 GiB (5.67 BPW)
llm_load_print_meta: general.name     = LLaMA v2
llm_load_print_meta: BOS token        = 1 '<s>'
llm_load_print_meta: EOS token        = 2 '</s>'
llm_load_print_meta: UNK token        = 0 '<unk>'
llm_load_print_meta: LF token         = 13 '<0x0A>'
llm_load_tensors: ggml ctx size =    0.28 MiB
llm_load_tensors: offloading 40 repeating layers to GPU
llm_load_tensors: offloading non-repeating layers to GPU
llm_load_tensors: offloaded 41/41 layers to GPU
llm_load_tensors:        CPU buffer size =   107.42 MiB
llm_load_tensors:      CUDA0 buffer size =  8694.21 MiB
.....................................................................
.............................
llama_new_context_with_model: n_ctx      = 4096
llama_new_context_with_model: freq_base  = 10000.0
llama_new_context_with_model: freq_scale = 1
llama_kv_cache_init:      CUDA0 KV buffer size =  3200.00 MiB
llama_new_context_with_model: KV self size  = 3200.00 MiB, K (f16):
1600.00 MiB, V (f16): 1600.00 MiB
```

```
llama_new_context_with_model:  CUDA_Host input buffer size   =
19.04 MiB
llama_new_context_with_model:      CUDA0 compute buffer size =
368.02 MiB
llama_new_context_with_model:  CUDA_Host compute buffer size =
10.00 MiB
llama_new_context_with_model: graph splits (measure): 3
AVX = 1 | AVX_VNNI = 0 | AVX2 = 1 | AVX512 = 1 | AVX512_VBMI = 0 |
AVX512_VNNI = 0 | FMA = 1 | NEON = 0 | ARM_FMA = 0 | F16C = 1 |
FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 1 | SSE3 = 1 | SSSE3 = 1 | VSX =
0 | MATMUL_INT8 = 0 |
Model metadata: {'tokenizer.ggml.unknown_token_id': '0',
'tokenizer.ggml.eos_token_id': '2', 'general.architecture': 'llama',
'llama.context_length': '4096', 'general.name': 'LLaMA v2',
'llama.embedding_length': '5120', 'llama.feed_forward_length':
'13824', 'llama.attention.layer_norm_rms_epsilon': '0.000010',
'llama.rope.dimension_count': '128', 'llama.attention.head_count':
'40', 'tokenizer.ggml.bos_token_id': '1', 'llama.block_count': '40',
'llama.attention.head_count_kv': '40', 'general.quantization_version':
'2', 'tokenizer.ggml.model': 'llama', 'general.file_type': '17'}
```

```
model_name_or_path = "TheBloke/Mistral-7B-Instruct-v0.2-GGUF"
model_basename = "mistral-7b-instruct-v0.2.Q6_K.gguf"

model_path = hf_hub_download(
    repo_id=model_name_or_path,
    filename=model_basename
)
```


```
{"model_id":"80c9001ac3fc4c9bb71fa2c8dcc97e4d","version_major":2,"vers
ion_minor":0}
```


```
mistral_llm = Llama(
    model_path=model_path,n_ctx=1024)
```

```
llama_model_loader: loaded meta data with 24 key-value pairs and 291
tensors from /root/.cache/huggingface/hub/models--TheBloke--Mistral-
7B-Instruct-v0.2-GGUF/snapshots/
3a6fbf4a41a1d52e415a4958cde6856d34b2db93/mistral-7b-instruct-
v0.2.Q6_K.gguf (version GGUF V3 (latest))
llama_model_loader: Dumping metadata keys/values. Note: KV overrides
do not apply in this output.
llama_model_loader: - kv   0:
general.architecture str              = llama
llama_model_loader: - kv   1:
general.name str              = mistralai_mistral-7b-instruct-v0.2
llama_model_loader: - kv   2:
llama.context_length u32              = 32768
llama_model_loader: - kv   3:
llama.embedding_length u32              = 4096
```

```
llama_model_loader: - kv   4:
llama.block_count u32              = 32
llama_model_loader: - kv   5:
llama.feed_forward_length u32              = 14336
llama_model_loader: - kv   6:
llama.rope.dimension_count u32              = 128
llama_model_loader: - kv   7:
llama.attention.head_count u32              = 32
llama_model_loader: - kv   8:
llama.attention.head_count_kv u32              = 8
llama_model_loader: - kv   9:
llama.attention.layer_norm_rms_epsilon f32              = 0.000010
llama_model_loader: - kv  10:
llama.rope.freq_base f32              = 1000000.000000
llama_model_loader: - kv  11:
general.file_type u32              = 18
llama_model_loader: - kv  12:
tokenizer.ggml.model str              = llama
llama_model_loader: - kv  13:
tokenizer.ggml.tokens arr[str,32000]   = ["<unk>", "<s>", "</s>",
"<0x00>", "<...
llama_model_loader: - kv  14:
tokenizer.ggml.scores arr[f32,32000]   = [0.000000, 0.000000,
0.000000, 0.0000...
llama_model_loader: - kv  15:
tokenizer.ggml.token_type arr[i32,32000]   = [2, 3, 3, 6, 6, 6, 6, 6,
6, 6, 6, 6, ...
llama_model_loader: - kv  16:
tokenizer.ggml.bos_token_id u32              = 1
llama_model_loader: - kv  17:
tokenizer.ggml.eos_token_id u32              = 2
llama_model_loader: - kv  18:
tokenizer.ggml.unknown_token_id u32              = 0
llama_model_loader: - kv  19:
tokenizer.ggml.padding_token_id u32              = 0
llama_model_loader: - kv  20:
tokenizer.ggml.add_bos_token bool             = true
llama_model_loader: - kv  21:
tokenizer.ggml.add_eos_token bool             = false
llama_model_loader: - kv  22:
tokenizer.chat_template str              = {{ bos_token }}{% for
message in mess...
llama_model_loader: - kv  23:
general.quantization_version u32              = 2
llama_model_loader: - type  f32:   65 tensors
llama_model_loader: - type q6_K:  226 tensors
llm_load_vocab: special tokens definition check successful ( 259/32000
).
llm_load_print_meta: format           = GGUF V3 (latest)
```

```
llm_load_print_meta: arch              = llama
llm_load_print_meta: vocab type        = SPM
llm_load_print_meta: n_vocab           = 32000
llm_load_print_meta: n_merges          = 0
llm_load_print_meta: n_ctx_train       = 32768
llm_load_print_meta: n_embd            = 4096
llm_load_print_meta: n_head            = 32
llm_load_print_meta: n_head_kv         = 8
llm_load_print_meta: n_layer           = 32
llm_load_print_meta: n_rot             = 128
llm_load_print_meta: n_embd_head_k     = 128
llm_load_print_meta: n_embd_head_v     = 128
llm_load_print_meta: n_gqa             = 4
llm_load_print_meta: n_embd_k_gqa      = 1024
llm_load_print_meta: n_embd_v_gqa      = 1024
llm_load_print_meta: f_norm_eps        = 0.0e+00
llm_load_print_meta: f_norm_rms_eps    = 1.0e-05
llm_load_print_meta: f_clamp_kqv       = 0.0e+00
llm_load_print_meta: f_max_alibi_bias = 0.0e+00
llm_load_print_meta: n_ff              = 14336
llm_load_print_meta: n_expert          = 0
llm_load_print_meta: n_expert_used     = 0
llm_load_print_meta: rope scaling      = linear
llm_load_print_meta: freq_base_train  = 1000000.0
llm_load_print_meta: freq_scale_train = 1
llm_load_print_meta: n_yarn_orig_ctx  = 32768
llm_load_print_meta: rope_finetuned   = unknown
llm_load_print_meta: model type        = 7B
llm_load_print_meta: model ftype       = Q6_K
llm_load_print_meta: model params      = 7.24 B
llm_load_print_meta: model size        = 5.53 GiB (6.56 BPW)
llm_load_print_meta: general.name      = mistralai_mistral-7b-instruct-
v0.2
llm_load_print_meta: BOS token         = 1 '<s>'
llm_load_print_meta: EOS token         = 2 '</s>'
llm_load_print_meta: UNK token         = 0 '<unk>'
llm_load_print_meta: PAD token         = 0 '<unk>'
llm_load_print_meta: LF token          = 13 '<0x0A>'
llm_load_tensors: ggml ctx size =    0.11 MiB
llm_load_tensors: offloading 0 repeating layers to GPU
llm_load_tensors: offloaded 0/33 layers to GPU
llm_load_tensors:        CPU buffer size =  5666.09 MiB
.....................................................................
.............................
llama_new_context_with_model: n_ctx      = 1024
llama_new_context_with_model: freq_base  = 1000000.0
llama_new_context_with_model: freq_scale = 1
llama_kv_cache_init:  CUDA_Host KV buffer size =   128.00 MiB
llama_new_context_with_model: KV self size  =  128.00 MiB, K (f16):
```

```
64.00 MiB, V (f16):   64.00 MiB
llama_new_context_with_model:  CUDA_Host input buffer size   =
11.01 MiB
llama_new_context_with_model:  CUDA_Host compute buffer size =
96.00 MiB
llama_new_context_with_model: graph splits (measure): 1
AVX = 1 | AVX_VNNI = 0 | AVX2 = 1 | AVX512 = 1 | AVX512_VBMI = 0 |
AVX512_VNNI = 0 | FMA = 1 | NEON = 0 | ARM_FMA = 0 | F16C = 1 |
FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 1 | SSE3 = 1 | SSSE3 = 1 | VSX =
0 | MATMUL_INT8 = 0 |
Model metadata: {'tokenizer.chat_template': "{{ bos_token }}{% for
message in messages %}{% if (message['role'] == 'user') !=
(loop.index0 % 2 == 0) %}{{ raise_exception('Conversation roles must
alternate user/assistant/user/assistant/...') }}{% endif %}{% if
message['role'] == 'user' %}{{ '[INST] ' + message['content'] + '
[/INST]' }}{% elif message['role'] == 'assistant' %}
{{ message['content'] + eos_token}}{% else %}{{ raise_exception('Only
user and assistant roles are supported!') }}{% endif %}{% endfor %}",
'tokenizer.ggml.add_eos_token': 'false',
'tokenizer.ggml.padding_token_id': '0',
'tokenizer.ggml.unknown_token_id': '0', 'tokenizer.ggml.eos_token_id':
'2', 'general.architecture': 'llama', 'llama.rope.freq_base':
'1000000.000000', 'llama.context_length': '32768', 'general.name':
'mistralai_mistral-7b-instruct-v0.2', 'tokenizer.ggml.add_bos_token':
'true', 'llama.embedding_length': '4096', 'llama.feed_forward_length':
'14336', 'llama.attention.layer_norm_rms_epsilon': '0.000010',
'llama.rope.dimension_count': '128', 'tokenizer.ggml.bos_token_id':
'1', 'llama.attention.head_count': '32', 'llama.block_count': '32',
'llama.attention.head_count_kv': '8', 'general.quantization_version':
'2', 'tokenizer.ggml.model': 'llama', 'general.file_type': '18'}
Using chat template: {{ bos_token }}{% for message in messages %}{% if
(message['role'] == 'user') != (loop.index0 % 2 == 0) %}
{{ raise_exception('Conversation roles must alternate
user/assistant/user/assistant/...') }}{% endif %}{% if message['role']
== 'user' %}{{ '[INST] ' + message['content'] + ' [/INST]' }}{% elif
message['role'] == 'assistant' %}{{ message['content'] + eos_token}}{%
else %}{{ raise_exception('Only user and assistant roles are
supported!') }}{% endif %}{% endfor %}
Using chat eos_token:
Using chat bos_token:

def generate_llama_response(instruction, review):
    system_message = """
        [INST]<<SYS>>
        {}
        <</SYS>>[/INST]
    """.format(instruction)
    prompt = f"{review}\n{system_message}"
    response = Llama_llm(
        prompt=prompt,
```

```
        max_tokens=1024,
        temperature=0,
        top_p=0.95,
        repeat_penalty=1.2,
        top_k=50,
        stop=['INST'],
        echo=False,
        seed=42,
    )
    response_text = response["choices"][0]["text"]
    return response_text

def extract_json_data(json_str):
    try:
        json_start = json_str.find('{')
        json_end = json_str.rfind('}')

        if json_start != -1 and json_end != -1:
            extracted_sentiment = json_str[json_start:json_end + 1]
            data_dict = json.loads(extracted_sentiment)
            return data_dict
        else:
            print(f"Warning: JSON object not found in response:
{json_str}")
            return {}
    except json.JSONDecodeError as e:
        print(f"Error parsing JSON: {e}")
        return {}

data_1 = data.copy()

instruction_1 = """
    You are an AI analyzing restaurant reviews. Classify the sentiment
of the provided review into the following categories:
    - Positive
    - Negative
    - Neutral
"""

data_1['model_response'] = data_1['review_full'].apply(lambda x:
generate_llama_response(instruction_1, x))


llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     163.98 ms /   151 runs   (
1.09 ms per token,   920.85 tokens per second)
llama_print_timings: prompt eval time =    1016.32 ms /   237 tokens (
4.29 ms per token,   233.20 tokens per second)
llama_print_timings:        eval time =    8510.52 ms /   150 runs   (
56.74 ms per token,    17.63 tokens per second)
llama_print_timings:       total time =   10820.55 ms /   387 tokens
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      52.83 ms /    98 runs   (
0.54 ms per token,  1854.87 tokens per second)
llama_print_timings: prompt eval time =     803.26 ms /   307 tokens (
2.62 ms per token,   382.19 tokens per second)
llama_print_timings:        eval time =    5860.15 ms /    97 runs   (
60.41 ms per token,    16.55 tokens per second)
llama_print_timings:       total time =    7009.01 ms /   404 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      64.85 ms /   104 runs   (
0.62 ms per token,  1603.70 tokens per second)
llama_print_timings: prompt eval time =     415.00 ms /    97 tokens (
4.28 ms per token,   233.74 tokens per second)
llama_print_timings:        eval time =    6079.59 ms /   103 runs   (
59.03 ms per token,    16.94 tokens per second)
llama_print_timings:       total time =    6983.00 ms /   200 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      71.02 ms /   119 runs   (
0.60 ms per token,  1675.63 tokens per second)
llama_print_timings: prompt eval time =     596.30 ms /   207 tokens (
2.88 ms per token,   347.14 tokens per second)
llama_print_timings:        eval time =    7668.47 ms /   118 runs   (
64.99 ms per token,    15.39 tokens per second)
llama_print_timings:       total time =    8758.04 ms /   325 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      63.24 ms /   116 runs   (
0.55 ms per token,  1834.25 tokens per second)
llama_print_timings: prompt eval time =     634.03 ms /   244 tokens (
2.60 ms per token,   384.84 tokens per second)
llama_print_timings:        eval time =    7986.03 ms /   115 runs   (
69.44 ms per token,    14.40 tokens per second)
llama_print_timings:       total time =    9023.08 ms /   359 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      93.70 ms /   146 runs   (
0.64 ms per token,  1558.25 tokens per second)
llama_print_timings: prompt eval time =     840.69 ms /   287 tokens (
2.93 ms per token,   341.39 tokens per second)
llama_print_timings:        eval time =   10361.53 ms /   145 runs   (
71.46 ms per token,    13.99 tokens per second)
llama_print_timings:       total time =   11811.84 ms /   432 tokens
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      72.58 ms /   120 runs   (
0.60 ms per token,  1653.35 tokens per second)
llama_print_timings: prompt eval time =     841.06 ms /   293 tokens (
2.87 ms per token,   348.37 tokens per second)
llama_print_timings:        eval time =    8677.42 ms /   119 runs   (
72.92 ms per token,    13.71 tokens per second)
llama_print_timings:       total time =   10022.42 ms /   412 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      76.13 ms /   119 runs   (
0.64 ms per token,  1563.12 tokens per second)
llama_print_timings: prompt eval time =     576.54 ms /   132 tokens (
4.37 ms per token,   228.95 tokens per second)
llama_print_timings:        eval time =    8271.52 ms /   118 runs   (
70.10 ms per token,    14.27 tokens per second)
llama_print_timings:       total time =    9360.02 ms /   250 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      76.57 ms /   134 runs   (
0.57 ms per token,  1749.96 tokens per second)
llama_print_timings: prompt eval time =     561.74 ms /   134 tokens (
4.19 ms per token,   238.55 tokens per second)
llama_print_timings:        eval time =    9113.52 ms /   133 runs   (
68.52 ms per token,    14.59 tokens per second)
llama_print_timings:       total time =   10213.85 ms /   267 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      69.87 ms /   124 runs   (
0.56 ms per token,  1774.72 tokens per second)
llama_print_timings: prompt eval time =     555.14 ms /   133 tokens (
4.17 ms per token,   239.58 tokens per second)
llama_print_timings:        eval time =    8255.13 ms /   123 runs   (
67.11 ms per token,    14.90 tokens per second)
llama_print_timings:       total time =    9283.44 ms /   256 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      59.29 ms /    95 runs   (
0.62 ms per token,  1602.19 tokens per second)
llama_print_timings: prompt eval time =     545.10 ms /   133 tokens (
4.10 ms per token,   243.99 tokens per second)
llama_print_timings:        eval time =    6137.29 ms /    94 runs   (
65.29 ms per token,    15.32 tokens per second)
llama_print_timings:       total time =    7111.63 ms /   227 tokens
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      83.93 ms /   148 runs   (
0.57 ms per token,  1763.33 tokens per second)
llama_print_timings: prompt eval time =     540.51 ms /   133 tokens (
4.06 ms per token,   246.06 tokens per second)
llama_print_timings:        eval time =    9703.98 ms /   147 runs   (
66.01 ms per token,    15.15 tokens per second)
llama_print_timings:       total time =   10827.64 ms /   280 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     106.80 ms /   181 runs   (
0.59 ms per token,  1694.82 tokens per second)
llama_print_timings: prompt eval time =     536.80 ms /   129 tokens (
4.16 ms per token,   240.31 tokens per second)
llama_print_timings:        eval time =   12004.62 ms /   180 runs   (
66.69 ms per token,    14.99 tokens per second)
llama_print_timings:       total time =   13271.95 ms /   309 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      63.50 ms /   114 runs   (
0.56 ms per token,  1795.19 tokens per second)
llama_print_timings: prompt eval time =     990.27 ms /   435 tokens (
2.28 ms per token,   439.28 tokens per second)
llama_print_timings:        eval time =    7883.51 ms /   113 runs   (
69.77 ms per token,    14.33 tokens per second)
llama_print_timings:       total time =    9294.03 ms /   548 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      64.78 ms /   103 runs   (
0.63 ms per token,  1590.07 tokens per second)
llama_print_timings: prompt eval time =    1547.84 ms /   606 tokens (
2.55 ms per token,   391.51 tokens per second)
llama_print_timings:        eval time =    7171.38 ms /   102 runs   (
70.31 ms per token,    14.22 tokens per second)
llama_print_timings:       total time =    9206.38 ms /   708 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      93.78 ms /   149 runs   (
0.63 ms per token,  1588.82 tokens per second)
llama_print_timings: prompt eval time =     842.12 ms /   347 tokens (
2.43 ms per token,   412.06 tokens per second)
llama_print_timings:        eval time =   10309.36 ms /   148 runs   (
69.66 ms per token,    14.36 tokens per second)
llama_print_timings:       total time =   11776.21 ms /   495 tokens
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      52.93 ms /    93 runs   (
0.57 ms per token,  1757.20 tokens per second)
llama_print_timings: prompt eval time =     845.68 ms /   351 tokens (
2.41 ms per token,   415.05 tokens per second)
llama_print_timings:        eval time =    6393.69 ms /    92 runs   (
69.50 ms per token,    14.39 tokens per second)
llama_print_timings:       total time =    7597.09 ms /   443 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      79.15 ms /   134 runs   (
0.59 ms per token,  1692.99 tokens per second)
llama_print_timings: prompt eval time =     952.31 ms /   415 tokens (
2.29 ms per token,   435.78 tokens per second)
llama_print_timings:        eval time =    9261.30 ms /   133 runs   (
69.63 ms per token,    14.36 tokens per second)
llama_print_timings:       total time =   10741.93 ms /   548 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      70.83 ms /   113 runs   (
0.63 ms per token,  1595.44 tokens per second)
llama_print_timings: prompt eval time =    1446.03 ms /   545 tokens (
2.65 ms per token,   376.90 tokens per second)
llama_print_timings:        eval time =    7769.85 ms /   112 runs   (
69.37 ms per token,    14.41 tokens per second)
llama_print_timings:       total time =    9718.94 ms /   657 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     106.34 ms /   171 runs   (
0.62 ms per token,  1608.11 tokens per second)
llama_print_timings: prompt eval time =     856.12 ms /   368 tokens (
2.33 ms per token,   429.85 tokens per second)
llama_print_timings:        eval time =   11618.91 ms /   170 runs   (
68.35 ms per token,    14.63 tokens per second)
llama_print_timings:       total time =   13190.27 ms /   538 tokens

data_1['model_response'].head()

0     Sure! Here's the sentiment analysis of the re...
1     Sure! Here's the sentiment analysis of the pr...
2     Sure, I can help you with that! Based on the ...
3     Sure! Here's the classification of the review...
4     Sure! Here's the sentiment analysis of the re...
Name: model_response, dtype: object
```

```
data.shape
```

```
(20, 3)
```

```
i = 2
print(data_1.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had
given us a great service. One of the best experience.
```

```
print(data_1.loc[i, 'model_response'])
```

```
 Sure, I can help you with that! Based on the review you provided, I
would classify it as:

Positive

Reason: The reviewer enjoyed their experience at the restaurant,
mentioning that it has "excellent taste" and "awesome decorum." They
also appreciated the "great service" provided by Subham Barnwal, which
suggests that the staff was attentive and helpful. Overall, the review
expresses a positive opinion of the restaurant.
```

```
def extract_sentiment(model_response):
    if 'positive' in model_response.lower():
        return 'Positive'
    elif 'negative' in model_response.lower():
        return 'Negative'
    elif 'neutral' in model_response.lower():
        return 'Neutral'

data_1['sentiment'] =
data_1['model_response'].apply(extract_sentiment)
data_1['sentiment'].head()
```

```
0    Positive
1    Positive
2    Positive
3    Positive
4    Positive
Name: sentiment, dtype: object
```

```
data_1['sentiment'].value_counts()
```

```
sentiment
Positive    16
Negative     3
Neutral      1
Name: count, dtype: int64
```

```
final_data_1 = data_1.drop(['model_response'], axis=1)
final_data_1.head()
```

{"summary":"{\n \"name\": \"final_data_1\",\n \"rows\": 20,\n \"fields\": [\n   {\n   \"column\": \"restaurant_ID\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 20,\n   \"samples\": [\n   \"FLV202\",\n   \"PIZ555\",\n   \"CRV333\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"rating_review\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 1,\n   \"max\": 5,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   5,\n   3,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"review_full\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 20,\n   \"samples\": [\n   \"Totally in love with the Auro of the place, really beautiful and quite fancy at the same time. The ambience is very pure and gives a sense of positivity throughout. Outdoor and indoor interior are quite quaint and cute. Love the open kitchen idea and there whole marketplace ideology. Due to coronovirus they specifically use disposable cutlery to keep the pandemic in mind taking all the precautionary measures from the beginning of the place with the mask on their staff and using good sanitisation. The food is really amazing specially the pizza straight from the oven and the hummus and pita bread are quite delicious too. If you're looking for a classy yet soothing Italian place in Delhi,Fatjar is a go to for you!\",\n   \"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75 expecting great things but we were disappointed. The place has great trip advisor reviews so we had high expectations that were sadly not met. What really ruined our nice meal was the self service. A waiter gave us (a party of 5) two tablets to look at the menu on. Unlike with proper menus, where everyone can browse at their own speed we had to share a tablet between 5. We asked the waiter for more tablets and were given one. After we had eventually all used the tablets to browse and input our orders on to one tablet the tablet then had an error message and froze. We used the other tablet to order and that one ran out of battery. We were hungry after a long day travelling this was stressful. Eventually one of the waiters came over and took details of our order. I really disliked this way of ordering food, and would not use the restaurant again because of this. The table was also dirty, there was dust and bits of fluff on the table and cloth napkins and my aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment. Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n   \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back

```
to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"sentiment\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          \"Positive\",\n          \"Neutral\",\n          \"Negative\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"final_data_1"}
```

```
data_1 = data.copy()
```

```
def response_1(prompt,review):
    model_output = mistral_llm(
      f"""
      Q: {prompt}
      Review: {review}
      A:
      """,
      max_tokens=32,
      stop=["Q:", "\n"],
      temperature=0.01,
      echo=False,
    )

    temp_output = model_output["choices"][0]["text"]

    return temp_output

instruction_1 = """
    You are an AI analyzing restaurant reviews. Classify the sentiment of the provided review into the following categories:
    - Positive
    - Negative
    - Neutral
"""

data_1['model_response'] = data_1['review_full'].apply(lambda x: response_1(instruction_1, x))
```

```
llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.74 ms /    32 runs   (
0.55 ms per token,  1803.93 tokens per second)
llama_print_timings: prompt eval time =    3330.46 ms /   218 tokens (
15.28 ms per token,    65.46 tokens per second)
llama_print_timings:        eval time =   23966.36 ms /    31 runs   (
773.11 ms per token,     1.29 tokens per second)
llama_print_timings:       total time =   27447.96 ms /   249 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.87 ms /    32 runs   (
0.56 ms per token,  1790.51 tokens per second)
llama_print_timings: prompt eval time =    2738.96 ms /   235 tokens (
11.66 ms per token,    85.80 tokens per second)
llama_print_timings:        eval time =   24528.54 ms /    31 runs   (
791.24 ms per token,     1.26 tokens per second)
llama_print_timings:       total time =   27419.21 ms /   266 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.39 ms /    32 runs   (
0.54 ms per token,  1839.82 tokens per second)
llama_print_timings: prompt eval time =    1654.25 ms /    34 tokens (
48.65 ms per token,    20.55 tokens per second)
llama_print_timings:        eval time =   24012.85 ms /    31 runs   (
774.61 ms per token,     1.29 tokens per second)
llama_print_timings:       total time =   25810.04 ms /    65 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      18.05 ms /    32 runs   (
0.56 ms per token,  1773.05 tokens per second)
llama_print_timings: prompt eval time =    2294.60 ms /   143 tokens (
16.05 ms per token,    62.32 tokens per second)
llama_print_timings:        eval time =   24286.28 ms /    31 runs   (
783.43 ms per token,     1.28 tokens per second)
llama_print_timings:       total time =   26726.90 ms /   174 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.33 ms /    32 runs   (
0.54 ms per token,  1846.30 tokens per second)
llama_print_timings: prompt eval time =    2373.38 ms /   169 tokens (
14.04 ms per token,    71.21 tokens per second)
llama_print_timings:        eval time =   24194.88 ms /    31 runs   (
780.48 ms per token,     1.28 tokens per second)
llama_print_timings:       total time =   26704.66 ms /   200 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.90 ms /    32 runs   (
0.56 ms per token,  1787.81 tokens per second)
llama_print_timings: prompt eval time =    2603.50 ms /   217 tokens (
12.00 ms per token,    83.35 tokens per second)
llama_print_timings:        eval time =   24435.75 ms /    31 runs   (
788.25 ms per token,     1.27 tokens per second)
llama_print_timings:       total time =   27195.74 ms /   248 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.37 ms /    32 runs   (
0.54 ms per token,  1842.57 tokens per second)
llama_print_timings: prompt eval time =    2432.18 ms /   215 tokens (
11.31 ms per token,    88.40 tokens per second)
llama_print_timings:        eval time =   24592.29 ms /    31 runs   (
793.30 ms per token,     1.26 tokens per second)
llama_print_timings:       total time =   27161.21 ms /   246 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.50 ms /    32 runs   (
0.55 ms per token,  1828.47 tokens per second)
llama_print_timings: prompt eval time =    1722.78 ms /    69 tokens (
24.97 ms per token,    40.05 tokens per second)
llama_print_timings:        eval time =   24206.02 ms /    31 runs   (
780.84 ms per token,     1.28 tokens per second)
llama_print_timings:       total time =   26077.02 ms /   100 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      18.60 ms /    32 runs   (
0.58 ms per token,  1720.25 tokens per second)
llama_print_timings: prompt eval time =    1709.09 ms /    67 tokens (
25.51 ms per token,    39.20 tokens per second)
llama_print_timings:        eval time =   24127.57 ms /    31 runs   (
778.31 ms per token,     1.28 tokens per second)
llama_print_timings:       total time =   25980.35 ms /    98 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.57 ms /    32 runs   (
0.55 ms per token,  1821.18 tokens per second)
llama_print_timings: prompt eval time =    1766.08 ms /    68 tokens (
25.97 ms per token,    38.50 tokens per second)
llama_print_timings:        eval time =   24230.61 ms /    31 runs   (
781.63 ms per token,     1.28 tokens per second)
llama_print_timings:       total time =   26136.74 ms /    99 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      18.08 ms /    32 runs   (
0.57 ms per token,  1769.91 tokens per second)
llama_print_timings: prompt eval time =    1823.56 ms /    65 tokens (
28.05 ms per token,    35.64 tokens per second)
llama_print_timings:        eval time =   24371.22 ms /    31 runs   (
786.17 ms per token,     1.27 tokens per second)
llama_print_timings:       total time =   26341.41 ms /    96 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.71 ms /    32 runs   (
0.55 ms per token,  1807.30 tokens per second)
llama_print_timings: prompt eval time =    1882.68 ms /    71 tokens (
26.52 ms per token,    37.71 tokens per second)
llama_print_timings:        eval time =   24287.68 ms /    31 runs   (
783.47 ms per token,     1.28 tokens per second)
llama_print_timings:       total time =   26308.69 ms /   102 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =       2.02 ms /     4 runs   (
0.50 ms per token,  1981.18 tokens per second)
llama_print_timings: prompt eval time =    1864.81 ms /    64 tokens (
29.14 ms per token,    34.32 tokens per second)
llama_print_timings:        eval time =    2248.20 ms /     3 runs   (
749.40 ms per token,     1.33 tokens per second)
llama_print_timings:       total time =    4128.35 ms /    67 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      19.50 ms /    32 runs   (
0.61 ms per token,  1640.69 tokens per second)
llama_print_timings: prompt eval time =    3467.14 ms /   354 tokens (
9.79 ms per token,   102.10 tokens per second)
llama_print_timings:        eval time =   24909.35 ms /    31 runs   (
803.53 ms per token,     1.24 tokens per second)
llama_print_timings:       total time =   28527.69 ms /   385 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =       1.52 ms /     3 runs   (
0.51 ms per token,  1973.68 tokens per second)
llama_print_timings: prompt eval time =   12553.24 ms /   526 tokens (
23.87 ms per token,    41.90 tokens per second)
llama_print_timings:        eval time =    1535.87 ms /     2 runs   (
767.94 ms per token,     1.30 tokens per second)
llama_print_timings:       total time =   14105.35 ms /   528 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.63 ms /    32 runs   (
0.55 ms per token,  1814.88 tokens per second)
llama_print_timings: prompt eval time =    2898.27 ms /   274 tokens (
10.58 ms per token,    94.54 tokens per second)
llama_print_timings:        eval time =   24850.30 ms /    31 runs   (
801.62 ms per token,     1.25 tokens per second)
llama_print_timings:       total time =   27898.34 ms /   305 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.90 ms /    32 runs   (
0.56 ms per token,  1787.91 tokens per second)
llama_print_timings: prompt eval time =    2909.68 ms /   284 tokens (
10.25 ms per token,    97.61 tokens per second)
llama_print_timings:        eval time =   24959.22 ms /    31 runs   (
805.14 ms per token,     1.24 tokens per second)
llama_print_timings:       total time =   28010.44 ms /   315 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.61 ms /    32 runs   (
0.55 ms per token,  1816.94 tokens per second)
llama_print_timings: prompt eval time =    3729.71 ms /   335 tokens (
11.13 ms per token,    89.82 tokens per second)
llama_print_timings:        eval time =   24321.40 ms /    31 runs   (
784.56 ms per token,     1.27 tokens per second)
llama_print_timings:       total time =   28197.28 ms /   366 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.55 ms /    32 runs   (
0.55 ms per token,  1822.95 tokens per second)
llama_print_timings: prompt eval time =    4927.53 ms /   462 tokens (
10.67 ms per token,    93.76 tokens per second)
llama_print_timings:        eval time =   24761.06 ms /    31 runs   (
798.74 ms per token,     1.25 tokens per second)
llama_print_timings:       total time =   29838.77 ms /   493 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.33 ms /    32 runs   (
0.54 ms per token,  1846.40 tokens per second)
llama_print_timings: prompt eval time =    2952.18 ms /   297 tokens (
9.94 ms per token,   100.60 tokens per second)
llama_print_timings:        eval time =   24444.33 ms /    31 runs   (
788.53 ms per token,     1.27 tokens per second)
llama_print_timings:       total time =   27536.96 ms /   328 tokens
```

```
data_1['model_response'].head()
```

```
0     This review expresses a very positive sentime...
1     Based on the given review, the sentiment can ...
2     The sentiment expressed in this review is pos...
3     Based on the given review, it can be classifi...
4     Based on the provided review, the sentiment c...
Name: model_response, dtype: object
```

```
i = 2
print(data_1.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had
given us a great service. One of the best experience.
```

```
print(data_1.loc[i, 'model_response'])
```

```
 The sentiment expressed in this review is positive. The words
"excellent taste," "awesome decorum," "must visit," and "great service
```

```
def extract_sentiment(model_response):
    if 'positive' in model_response.lower():
        return 'Positive'
    elif 'negative' in model_response.lower():
        return 'Negative'
    elif 'neutral' in model_response.lower():
        return 'Neutral'

data_1['sentiment'] =
data_1['model_response'].apply(extract_sentiment)
data_1['sentiment'].head()
```

```
0    Positive
1    Positive
2    Positive
3    Positive
4    Positive
Name: sentiment, dtype: object
```

```
data_1['sentiment'].value_counts()
```

```
sentiment
Positive    10
Negative     7
Neutral      3
Name: count, dtype: int64
```

```
final_data_1 = data_1.drop(['model_response'], axis=1)
final_data_1.head()
```


```
{"summary":"{\n  \"name\": \"final_data_1\",\n  \"rows\": 20,\n
\"fields\": [\n    {\n      \"column\": \"restaurant_ID\",\n
```

\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 20,\n   \"samples\": [\n
\"FLV202\",\n   \"PIZ555\",\n   \"CRV333\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"rating_review\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1,\n   \"min\": 1,\n   \"max\": 5,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   5,\n
3,\n   1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"review_full\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"num_unique_values\": 20,\n   \"samples\":
[\n   \"Totally in love with the Auro of the place, really
beautiful and quite fancy at the same time. The ambience is very pure
and gives a sense of positivity throughout. Outdoor and indoor
interior are quite quaint and cute. Love the open kitchen idea and
there whole marketplace ideology. Due to coronovirus they specifically
use disposable cutlery to keep the pandemic in mind taking all the
precautionary measures from the beginning of the place with the mask
on their staff and using good sanitisation. The food is really amazing
specially the pizza straight from the oven and the hummus and pita
bread are quite delicious too. If you're looking for a classy yet
soothing Italian place in Delhi,Fatjar is a go to for you!\",\n
\"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75
expecting great things but we were disappointed. The place has great
trip advisor reviews so we had high expectations that were sadly not
met. What really ruined our nice meal was the self service. A waiter
gave us (a party of 5) two tablets to look at the menu on. Unlike with
proper menus, where everyone can browse at their own speed we had to
share a tablet between 5. We asked the waiter for more tablets and
were given one. After we had eventually all used the tablets to browse
and input our orders on to one tablet the tablet then had an error
message and froze. We used the other tablet to order and that one ran
out of battery. We were hungry after a long day travelling this was
stressful. Eventually one of the waiters came over and took details of
our order. I really disliked this way of ordering food, and would not
use the restaurant again because of this. The table was also dirty,
there was dust and bits of fluff on the table and cloth napkins and my
aunt was given a chipped glass which she discovered upon taking a
drink. We told the waiter and he apologised and got us a new glass but
again, this just really made our experience a disappointment. Very
disappointed as seems to have good reviews, maybe we came on a bad day
but we didn't like the menu/way of ordering, the table was dirty and a
member of our party was given a chipped glass.\",\n   \"SERVICE
HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough
(maybe because I was soooo hungry after a hectic day out) well
presented but the service put me off completely. 12 servers on the
floor including the manager and not one person came over to check back
to make sure all was okay with our meals. All hanging around talking.

```
And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"sentiment\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          \"Positive\",\n          \"Neutral\",\n          \"Negative\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"final_data_1"}
```

```
data_2 = data.copy()
```

```
instruction_2 = """
    You are an AI analyzing restaurant reviews. Classify the sentiment of the provided review into the following categories:
    - Positive
    - Negative
    - Neutral

    Format the output as a JSON object with a single key-value pair as shown below:
    {"sentiment": "your_sentiment_prediction"}
"""

data_2['model_response'] = data_2['review_full'].apply(lambda x: generate_llama_response(instruction_2, x))
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      96.40 ms /   161 runs   (    0.60 ms per token,  1670.12 tokens per second)
llama_print_timings: prompt eval time =     801.70 ms /   272 tokens (    2.95 ms per token,   339.28 tokens per second)
llama_print_timings:        eval time =   11779.72 ms /   160 runs   (   73.62 ms per token,    13.58 tokens per second)
llama_print_timings:       total time =   13241.32 ms /   432 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
```

```
llama_print_timings:      sample time =     129.66 ms /   225 runs   (
0.58 ms per token,  1735.32 tokens per second)
llama_print_timings: prompt eval time =     892.88 ms /   343 tokens (
2.60 ms per token,   384.15 tokens per second)
llama_print_timings:        eval time =   17104.71 ms /   224 runs   (
76.36 ms per token,    13.10 tokens per second)
llama_print_timings:       total time =   18929.88 ms /   567 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      31.39 ms /    48 runs   (
0.65 ms per token,  1528.95 tokens per second)
llama_print_timings: prompt eval time =     579.20 ms /   133 tokens (
4.35 ms per token,   229.63 tokens per second)
llama_print_timings:        eval time =    3319.10 ms /    47 runs   (
70.62 ms per token,    14.16 tokens per second)
llama_print_timings:       total time =    4104.61 ms /   180 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      60.88 ms /   109 runs   (
0.56 ms per token,  1790.38 tokens per second)
llama_print_timings: prompt eval time =     649.47 ms /   243 tokens (
2.67 ms per token,   374.15 tokens per second)
llama_print_timings:        eval time =    7559.87 ms /   108 runs   (
70.00 ms per token,    14.29 tokens per second)
llama_print_timings:       total time =    8614.14 ms /   351 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     143.36 ms /   238 runs   (
0.60 ms per token,  1660.17 tokens per second)
llama_print_timings: prompt eval time =     795.39 ms /   280 tokens (
2.84 ms per token,   352.03 tokens per second)
llama_print_timings:        eval time =   15786.46 ms /   237 runs   (
66.61 ms per token,    15.01 tokens per second)
llama_print_timings:       total time =   17629.96 ms /   517 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      77.96 ms /   130 runs   (
0.60 ms per token,  1667.52 tokens per second)
llama_print_timings: prompt eval time =     813.43 ms /   323 tokens (
2.52 ms per token,   397.08 tokens per second)
llama_print_timings:        eval time =    8546.84 ms /   129 runs   (
66.25 ms per token,    15.09 tokens per second)
llama_print_timings:       total time =    9888.18 ms /   452 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
```

```
llama_print_timings:      sample time =     150.51 ms /   259 runs   (
0.58 ms per token,  1720.78 tokens per second)
llama_print_timings: prompt eval time =     810.68 ms /   329 tokens (
2.46 ms per token,   405.83 tokens per second)
llama_print_timings:        eval time =   17256.86 ms /   258 runs   (
66.89 ms per token,    14.95 tokens per second)
llama_print_timings:       total time =   19185.00 ms /   587 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      54.26 ms /    90 runs   (
0.60 ms per token,  1658.74 tokens per second)
llama_print_timings: prompt eval time =     591.47 ms /   168 tokens (
3.52 ms per token,   284.04 tokens per second)
llama_print_timings:        eval time =    6015.08 ms /    89 runs   (
67.59 ms per token,    14.80 tokens per second)
llama_print_timings:       total time =    6997.91 ms /   257 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      77.34 ms /   140 runs   (
0.55 ms per token,  1810.17 tokens per second)
llama_print_timings: prompt eval time =     594.22 ms /   170 tokens (
3.50 ms per token,   286.09 tokens per second)
llama_print_timings:        eval time =    9744.10 ms /   139 runs   (
70.10 ms per token,    14.27 tokens per second)
llama_print_timings:       total time =   10853.98 ms /   309 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      47.95 ms /    78 runs   (
0.61 ms per token,  1626.83 tokens per second)
llama_print_timings: prompt eval time =     593.34 ms /   169 tokens (
3.51 ms per token,   284.83 tokens per second)
llama_print_timings:        eval time =    5350.57 ms /    77 runs   (
69.49 ms per token,    14.39 tokens per second)
llama_print_timings:       total time =    6286.35 ms /   246 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      79.16 ms /   134 runs   (
0.59 ms per token,  1692.84 tokens per second)
llama_print_timings: prompt eval time =     596.99 ms /   169 tokens (
3.53 ms per token,   283.09 tokens per second)
llama_print_timings:        eval time =    9298.78 ms /   133 runs   (
69.92 ms per token,    14.30 tokens per second)
llama_print_timings:       total time =   10456.96 ms /   302 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
```

```
llama_print_timings:      sample time =      42.31 ms /    80 runs   (
0.53 ms per token,  1890.63 tokens per second)
llama_print_timings: prompt eval time =     602.22 ms /   169 tokens (
3.56 ms per token,   280.63 tokens per second)
llama_print_timings:        eval time =    5614.47 ms /    79 runs   (
71.07 ms per token,    14.07 tokens per second)
llama_print_timings:       total time =    6488.57 ms /   248 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      50.75 ms /    78 runs   (
0.65 ms per token,  1536.85 tokens per second)
llama_print_timings: prompt eval time =     596.91 ms /   165 tokens (
3.62 ms per token,   276.42 tokens per second)
llama_print_timings:        eval time =    5224.18 ms /    77 runs   (
67.85 ms per token,    14.74 tokens per second)
llama_print_timings:       total time =    6195.72 ms /   242 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      73.35 ms /   127 runs   (
0.58 ms per token,  1731.45 tokens per second)
llama_print_timings: prompt eval time =    1006.42 ms /   471 tokens (
2.14 ms per token,   468.00 tokens per second)
llama_print_timings:        eval time =    8890.14 ms /   126 runs   (
70.56 ms per token,    14.17 tokens per second)
llama_print_timings:       total time =   10380.93 ms /   597 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      65.31 ms /   115 runs   (
0.57 ms per token,  1760.97 tokens per second)
llama_print_timings: prompt eval time =    1645.69 ms /   642 tokens (
2.56 ms per token,   390.11 tokens per second)
llama_print_timings:        eval time =    8092.12 ms /   114 runs   (
70.98 ms per token,    14.09 tokens per second)
llama_print_timings:       total time =   10177.86 ms /   756 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     106.45 ms /   172 runs   (
0.62 ms per token,  1615.86 tokens per second)
llama_print_timings: prompt eval time =     882.04 ms /   383 tokens (
2.30 ms per token,   434.22 tokens per second)
llama_print_timings:        eval time =   11704.29 ms /   171 runs   (
68.45 ms per token,    14.61 tokens per second)
llama_print_timings:       total time =   13299.23 ms /   554 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
```

```
llama_print_timings:      sample time =      69.98 ms /   118 runs   (
0.59 ms per token,  1686.29 tokens per second)
llama_print_timings: prompt eval time =     915.35 ms /   387 tokens (
2.37 ms per token,   422.79 tokens per second)
llama_print_timings:        eval time =    8019.15 ms /   117 runs   (
68.54 ms per token,    14.59 tokens per second)
llama_print_timings:       total time =    9444.14 ms /   504 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     105.48 ms /   172 runs   (
0.61 ms per token,  1630.56 tokens per second)
llama_print_timings: prompt eval time =     976.09 ms /   451 tokens (
2.16 ms per token,   462.05 tokens per second)
llama_print_timings:        eval time =   11862.17 ms /   171 runs   (
69.37 ms per token,    14.42 tokens per second)
llama_print_timings:       total time =   13567.82 ms /   622 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      90.57 ms /   155 runs   (
0.58 ms per token,  1711.42 tokens per second)
llama_print_timings: prompt eval time =    1515.04 ms /   581 tokens (
2.61 ms per token,   383.49 tokens per second)
llama_print_timings:        eval time =   10863.24 ms /   154 runs   (
70.54 ms per token,    14.18 tokens per second)
llama_print_timings:       total time =   13034.34 ms /   735 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     159.99 ms /   250 runs   (
0.64 ms per token,  1562.57 tokens per second)
llama_print_timings: prompt eval time =     936.16 ms /   404 tokens (
2.32 ms per token,   431.55 tokens per second)
llama_print_timings:        eval time =   17238.11 ms /   249 runs   (
69.23 ms per token,    14.44 tokens per second)
llama_print_timings:       total time =   19316.41 ms /   653 tokens
```

```
data_2['model_response'].head()
```

```
0     Sure! Here's the sentiment analysis of the re...
1     Sure! Here's my analysis of the review you pr...
2     Sure, I can help you with that! Based on the ...
3     Sure! Here's the sentiment analysis of the re...
4     Sure! Here's the sentiment analysis of the re...
Name: model_response, dtype: object
```

```
i = 2
print(data_2.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had given us a great service. One of the best experience.

print(data_2.loc[i, 'model_response'])

 Sure, I can help you with that! Based on the review you provided, I would classify the sentiment as Positive. Here's the JSON object with the sentiment prediction:

{"sentiment": "Positive"}

data_2['model_response_parsed'] =
data_2['model_response'].apply(extract_json_data)
data_2['model_response_parsed'].head()

0    {'sentiment': 'Positive'}
1    {'sentiment': 'Positive'}
2    {'sentiment': 'Positive'}
3    {'sentiment': 'Positive'}
4    {'sentiment': 'Positive'}
Name: model_response_parsed, dtype: object

model_response_parsed_df_2 =
pd.json_normalize(data_2['model_response_parsed'])
model_response_parsed_df_2.head()

{"summary":"{\n  \"name\": \"model_response_parsed_df_2\",\n
\"rows\": 20,\n  \"fields\": [\n    {\n      \"column\":
\"sentiment\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"Positive\",\n          \"Neutral\",\n
\"Negative\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"model_response_parsed_df_2"}

data_with_parsed_model_output_2 = pd.concat([data_2,
model_response_parsed_df_2], axis=1)
data_with_parsed_model_output_2.head()

{"summary":"{\n  \"name\": \"data_with_parsed_model_output_2\",\n
\"rows\": 20,\n  \"fields\": [\n    {\n      \"column\":
\"restaurant_ID\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 20,\n        \"samples\":
[\n          \"FLV202\",\n          \"PIZ555\",\n          \"CRV333\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating_review\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 5,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          5,\n          3,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

n   },\n   {\n   \"column\": \"review_full\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 20,\n   \"samples\": [\n
\"Totally in love with the Auro of the place, really beautiful and quite fancy at the same time. The ambience is very pure and gives a sense of positivity throughout. Outdoor and indoor interior are quite quaint and cute. Love the open kitchen idea and there whole marketplace ideology. Due to coronovirus they specifically use disposable cutlery to keep the pandemic in mind taking all the precautionary measures from the beginning of the place with the mask on their staff and using good sanitisation. The food is really amazing specially the pizza straight from the oven and the hummus and pita bread are quite delicious too. If you're looking for a classy yet soothing Italian place in Delhi,Fatjar is a go to for you!\",\n
\"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75 expecting great things but we were disappointed. The place has great trip advisor reviews so we had high expectations that were sadly not met. What really ruined our nice meal was the self service. A waiter gave us (a party of 5) two tablets to look at the menu on. Unlike with proper menus, where everyone can browse at their own speed we had to share a tablet between 5. We asked the waiter for more tablets and were given one. After we had eventually all used the tablets to browse and input our orders on to one tablet the tablet then had an error message and froze. We used the other tablet to order and that one ran out of battery. We were hungry after a long day travelling this was stressful. Eventually one of the waiters came over and took details of our order. I really disliked this way of ordering food, and would not use the restaurant again because of this. The table was also dirty, there was dust and bits of fluff on the table and cloth napkins and my aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment. Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n   \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two

```
sandwiches, bottled water and terrible service! Certainly not worth
it. I reckon the street food stalls would give a better service then
here \"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"model_response\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 20,\n        \"samples\":
[\n          \" Sure! Here's the sentiment analysis of the review you
provided:\\n\\n    {\\n    \\\"sentiment\\\": \\\"Positive\\\"\\
n    }\\n\\nHere's my reasoning:\\n\\n* The reviewer expresses their
love for the restaurant's ambiance, food, and service.\\n* They use
positive adjectives such as \\\"beautiful\\\", \\\"quite fancy\\\",
\\\"quaint\\\", \\\"delicious\\\", and \\\"classy\\\".\\n* They
mention that the restaurant takes precautionary measures against
COVID-19, which suggests that they value cleanliness and safety.\\n*
There are no negative comments or complaints about the restaurant.\\
n\\nOverall, the review expresses a positive sentiment towards Fatjar
Italian Restaurant.\",\n          \" Sure, I can help you with that!
Based on the review provided, I would classify the sentiment as
follows:\\n\\n{\\\"sentiment\\\": \\\"Negative\\\"}\\n\\nHere's a
breakdown of my analysis:\\n\\n* The reviewer expresses disappointment
with their experience at Pavilion 75, indicating a negative
sentiment.\\n* They mention specific issues such as the self-service
ordering system, dirty tables, and a chipped glass, which contribute
to a negative overall assessment of the restaurant.\\n* Although they
mention good trip advisor reviews, this does not seem to have an
impact on their own negative experience, further reinforcing a
negative sentiment.\\n\\nI hope this helps! Let me know if you have
any other questions or if you'd like me to analyze any other
reviews.\",\n          \" Sure, I can help you with that! Based on the
review you provided, here's my sentiment analysis:\\n\\
n{\\\"sentiment\\\": \\\"Negative\\\"}\\n\\nThe review expresses
disappointment with the service, stating that it was
\\\"horrendous\\\" and that the servers were not attentive or
responsive. The reviewer also mentions that they had to ask multiple
times for cold water, which took a long time to arrive, and that the
water that arrived was only \\\"coldish.\\\" Additionally, the
reviewer mentions that the sandwich was \\\"decent\\\" but that the
presentation was poor, which suggests that the food may not have been
up to their expectations either. Overall, the review expresses a
negative sentiment towards the restaurant, indicating that it may not
be worth visiting based on their experience.\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"model_response_parsed\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"sentiment\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Positive\",\n          \"Neutral\",\n          \"Negative\"\n
```

```
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\
n}","type":"dataframe","variable_name":"data_with_parsed_model_output_
2"}
```

```
final_data_2 =
data_with_parsed_model_output_2.drop(['model_response','model_response
_parsed'], axis=1)
final_data_2.head()
```


```
{"summary":"{\n  \"name\": \"final_data_2\",\n  \"rows\": 20,\n
\"fields\": [\n    {\n      \"column\": \"restaurant_ID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 20,\n        \"samples\": [\n
\"FLV202\",\n          \"PIZ555\",\n          \"CRV333\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"rating_review\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 1,\n        \"max\": 5,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          5,\n
3,\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"review_full\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 20,\n        \"samples\":
[\n          \"Totally in love with the Auro of the place, really
beautiful and quite fancy at the same time. The ambience is very pure
and gives a sense of positivity throughout. Outdoor and indoor
interior are quite quaint and cute. Love the open kitchen idea and
there whole marketplace ideology. Due to coronovirus they specifically
use disposable cutlery to keep the pandemic in mind taking all the
precautionary measures from the beginning of the place with the mask
on their staff and using good sanitisation. The food is really amazing
specially the pizza straight from the oven and the hummus and pita
bread are quite delicious too. If you're looking for a classy yet
soothing Italian place in Delhi,Fatjar is a go to for you!\",\n
\"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75
expecting great things but we were disappointed. The place has great
trip advisor reviews so we had high expectations that were sadly not
met.  What really ruined our nice meal was the self service. A waiter
gave us (a party of 5) two tablets to look at the menu on. Unlike with
proper menus, where everyone can browse at their own speed we had to
share a tablet between 5. We asked the waiter for more tablets and
were given one. After we had eventually all used the tablets to browse
and input our orders on to one tablet the tablet then had an error
message and froze. We used the other tablet to order and that one ran
out of battery. We were hungry after a long day travelling this was
stressful. Eventually one of the waiters came over and took details of
our order. I really disliked this way of ordering food, and would not
use the restaurant again because of this.  The table was also dirty,
there was dust and bits of fluff on the table and cloth napkins and my
```

aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment. Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n   \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"sentiment\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"Positive\",\n   \"Neutral\",\n   \"Negative\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"final_data_2"}

```
final_data_2['sentiment'].value_counts()
```

```
sentiment
Neutral     7
Negative    7
Positive    6
Name: count, dtype: int64
```

```
data_3 = data.copy()

instruction_3 = """
    You are an AI analyzing restaurant reviews. Classify the overall sentiment of the provided review into the following categories:
    - "Positive"
    - "Negative"
    - "Neutral"

    Once that is done, check for a mention of the following aspects in the review and classify the sentiment of each aspect as "Positive",
```

```
"Negative", or "Neutral":
    1. "Food Quality"
    2. "Service"
    3. "Ambience"

    Output the overall sentiment and sentiment for each category in a
JSON format with the following keys:
    {
        "Overall": "your_sentiment_prediction",
        "Food Quality": "your_sentiment_prediction",
        "Service": "your_sentiment_prediction",
        "Ambience": "your_sentiment_prediction"
    }

    In case one of the three aspects is not mentioned in the review,
set "Not Applicable" (including quotes) for the corresponding JSON key
value.
    Only return the JSON, do not return any other information.
"""

data_3['model_response'] = data_3['review_full'].apply(lambda x:
generate_llama_response(instruction_3, x))
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     611.15 ms /  1024 runs   (
0.60 ms per token,  1675.52 tokens per second)
llama_print_timings: prompt eval time =     967.50 ms /   451 tokens (
2.15 ms per token,   466.15 tokens per second)
llama_print_timings:        eval time =   73871.52 ms /  1023 runs   (
72.21 ms per token,    13.85 tokens per second)
llama_print_timings:       total time =   79993.10 ms /  1474 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     117.74 ms /   190 runs   (
0.62 ms per token,  1613.70 tokens per second)
llama_print_timings: prompt eval time =    1349.10 ms /   522 tokens (
2.58 ms per token,   386.92 tokens per second)
llama_print_timings:        eval time =   12843.74 ms /   189 runs   (
67.96 ms per token,    14.72 tokens per second)
llama_print_timings:       total time =   15110.46 ms /   711 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      35.62 ms /    47 runs   (
0.76 ms per token,  1319.37 tokens per second)
llama_print_timings: prompt eval time =     812.48 ms /   312 tokens (
2.60 ms per token,   384.01 tokens per second)
```

```
llama_print_timings:        eval time =    3006.16 ms /    46 runs   (
65.35 ms per token,    15.30 tokens per second)
llama_print_timings:       total time =    4098.52 ms /   358 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      25.93 ms /    47 runs   (
0.55 ms per token,  1812.50 tokens per second)
llama_print_timings: prompt eval time =     954.27 ms /   422 tokens (
2.26 ms per token,   442.22 tokens per second)
llama_print_timings:        eval time =    3234.40 ms /    46 runs   (
70.31 ms per token,    14.22 tokens per second)
llama_print_timings:       total time =    4362.62 ms /   468 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     597.93 ms /  1024 runs   (
0.58 ms per token,  1712.58 tokens per second)
llama_print_timings: prompt eval time =     994.22 ms /   459 tokens (
2.17 ms per token,   461.67 tokens per second)
llama_print_timings:        eval time =   73072.32 ms /  1023 runs   (
71.43 ms per token,    14.00 tokens per second)
llama_print_timings:       total time =   79249.53 ms /  1482 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      28.76 ms /    49 runs   (
0.59 ms per token,  1703.58 tokens per second)
llama_print_timings: prompt eval time =    1042.89 ms /   502 tokens (
2.08 ms per token,   481.36 tokens per second)
llama_print_timings:        eval time =    3346.72 ms /    48 runs   (
69.72 ms per token,    14.34 tokens per second)
llama_print_timings:       total time =    4585.35 ms /   550 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     606.96 ms /  1024 runs   (
0.59 ms per token,  1687.09 tokens per second)
llama_print_timings: prompt eval time =    1047.43 ms /   508 tokens (
2.06 ms per token,   485.00 tokens per second)
llama_print_timings:        eval time =   73503.67 ms /  1023 runs   (
71.85 ms per token,    13.92 tokens per second)
llama_print_timings:       total time =   79700.48 ms /  1531 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      28.08 ms /    52 runs   (
0.54 ms per token,  1852.12 tokens per second)
llama_print_timings: prompt eval time =     822.24 ms /   347 tokens (
2.37 ms per token,   422.02 tokens per second)
```

```
llama_print_timings:        eval time =    3614.50 ms /    51 runs   (
70.87 ms per token,    14.11 tokens per second)
llama_print_timings:       total time =    4615.49 ms /   398 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      28.37 ms /    51 runs   (
0.56 ms per token,  1797.67 tokens per second)
llama_print_timings: prompt eval time =     836.93 ms /   349 tokens (
2.40 ms per token,   417.00 tokens per second)
llama_print_timings:        eval time =    3483.18 ms /    50 runs   (
69.66 ms per token,    14.35 tokens per second)
llama_print_timings:       total time =    4506.87 ms /   399 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      35.34 ms /    52 runs   (
0.68 ms per token,  1471.46 tokens per second)
llama_print_timings: prompt eval time =     840.32 ms /   348 tokens (
2.41 ms per token,   414.13 tokens per second)
llama_print_timings:        eval time =    3464.58 ms /    51 runs   (
67.93 ms per token,    14.72 tokens per second)
llama_print_timings:       total time =    4569.75 ms /   399 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      27.92 ms /    52 runs   (
0.54 ms per token,  1862.53 tokens per second)
llama_print_timings: prompt eval time =     815.96 ms /   348 tokens (
2.34 ms per token,   426.49 tokens per second)
llama_print_timings:        eval time =    3583.54 ms /    51 runs   (
70.27 ms per token,    14.23 tokens per second)
llama_print_timings:       total time =    4584.47 ms /   399 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      27.49 ms /    52 runs   (
0.53 ms per token,  1891.39 tokens per second)
llama_print_timings: prompt eval time =     834.75 ms /   348 tokens (
2.40 ms per token,   416.89 tokens per second)
llama_print_timings:        eval time =    3599.10 ms /    51 runs   (
70.57 ms per token,    14.17 tokens per second)
llama_print_timings:       total time =    4609.75 ms /   399 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      35.78 ms /    51 runs   (
0.70 ms per token,  1425.26 tokens per second)
llama_print_timings: prompt eval time =     827.16 ms /   344 tokens (
2.40 ms per token,   415.88 tokens per second)
```

```
llama_print_timings:        eval time =    3371.34 ms /    50 runs   (
67.43 ms per token,    14.83 tokens per second)
llama_print_timings:       total time =    4459.10 ms /   394 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      29.38 ms /    52 runs   (
0.57 ms per token,  1769.79 tokens per second)
llama_print_timings: prompt eval time =    1653.20 ms /   650 tokens (
2.54 ms per token,   393.18 tokens per second)
llama_print_timings:        eval time =    3643.80 ms /    51 runs   (
71.45 ms per token,    14.00 tokens per second)
llama_print_timings:       total time =    5488.23 ms /   701 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     618.25 ms /  1024 runs   (
0.60 ms per token,  1656.30 tokens per second)
llama_print_timings: prompt eval time =    1908.62 ms /   821 tokens (
2.32 ms per token,   430.15 tokens per second)
llama_print_timings:        eval time =   74888.54 ms /  1023 runs   (
73.20 ms per token,    13.66 tokens per second)
llama_print_timings:       total time =   82074.61 ms /  1844 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      32.01 ms /    52 runs   (
0.62 ms per token,  1624.54 tokens per second)
llama_print_timings: prompt eval time =    1466.29 ms /   562 tokens (
2.61 ms per token,   383.28 tokens per second)
llama_print_timings:        eval time =    3596.16 ms /    51 runs   (
70.51 ms per token,    14.18 tokens per second)
llama_print_timings:       total time =    5282.96 ms /   613 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      28.89 ms /    53 runs   (
0.55 ms per token,  1834.61 tokens per second)
llama_print_timings: prompt eval time =    1468.04 ms /   566 tokens (
2.59 ms per token,   385.55 tokens per second)
llama_print_timings:        eval time =    3726.91 ms /    52 runs   (
71.67 ms per token,    13.95 tokens per second)
llama_print_timings:       total time =    5391.44 ms /   618 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     186.26 ms /   294 runs   (
0.63 ms per token,  1578.47 tokens per second)
llama_print_timings: prompt eval time =    1569.70 ms /   630 tokens (
2.49 ms per token,   401.35 tokens per second)
```

```
llama_print_timings:        eval time =   20662.43 ms /   293 runs   (
70.52 ms per token,    14.18 tokens per second)
llama_print_timings:       total time =   23690.68 ms /   923 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      32.02 ms /    51 runs   (
0.63 ms per token,  1592.51 tokens per second)
llama_print_timings: prompt eval time =    1748.47 ms /   760 tokens (
2.30 ms per token,   434.67 tokens per second)
llama_print_timings:        eval time =    3522.23 ms /    50 runs   (
70.44 ms per token,    14.20 tokens per second)
llama_print_timings:       total time =    5529.26 ms /   810 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      29.55 ms /    51 runs   (
0.58 ms per token,  1725.83 tokens per second)
llama_print_timings: prompt eval time =    1511.55 ms /   583 tokens (
2.59 ms per token,   385.70 tokens per second)
llama_print_timings:        eval time =    3563.12 ms /    50 runs   (
71.26 ms per token,    14.03 tokens per second)
llama_print_timings:       total time =    5284.87 ms /   633 tokens
```

```
data_3['model_response'].head()
```

```
0    \n    {\n        "Overall": "Positive",\n     ...
1    \n    {\n        "Overall": "Positive",\n     ...
2     {\n            "Overall": "Positive",\n      ...
3     {\n        "Overall": "Positive",\n        "F...
4    \n    {\n        "Overall": "Positive",\n     ...
Name: model_response, dtype: object
```

```
i = 2
print(data_3.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had
given us a great service. One of the best experience.
```

```
print(data_3.loc[i, 'model_response'])
```

```
 {
            "Overall": "Positive",
            "Food Quality": "Positive",
            "Service": "Positive",
            "Ambience": "Positive"
          }
```

```
data_3['model_response_parsed'] =
data_3['model_response'].apply(extract_json_data)
data_3['model_response_parsed'].head()
```

```
0    {'Overall': 'Positive', 'Food Quality': 'Posit...
1    {'Overall': 'Positive', 'Food Quality': 'Posit...
2    {'Overall': 'Positive', 'Food Quality': 'Posit...
3    {'Overall': 'Positive', 'Food Quality': 'Posit...
4    {'Overall': 'Positive', 'Food Quality': 'Posit...
Name: model_response_parsed, dtype: object
```

```
model_response_parsed_df_3 =
pd.json_normalize(data_3['model_response_parsed'])
model_response_parsed_df_3.head()
```


```
{"summary":"{\n  \"name\": \"model_response_parsed_df_3\",\n
\"rows\": 20,\n  \"fields\": [\n    {\n      \"column\": \"Overall\",\
n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Positive\",\n          \"Neutral\",\n          \"Negative\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Food Quality\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Negative\",\n          \"Not Applicable\",\n          \"Mixed\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Service\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Neutral\",\n          \"Negative\",\n          \"Slow\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Ambience\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          \"Not
Applicable\",\n          \"Negative\",\n          \"Positive\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\
n}","type":"dataframe","variable_name":"model_response_parsed_df_3"}
```


```
data_with_parsed_model_output_3 = pd.concat([data_3,
model_response_parsed_df_3], axis=1)
data_with_parsed_model_output_3.head()
```


```
{"summary":"{\n  \"name\": \"data_with_parsed_model_output_3\",\n
\"rows\": 20,\n  \"fields\": [\n    {\n      \"column\":
\"restaurant_ID\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 20,\n        \"samples\":
[\n          \"FLV202\",\n          \"PIZ555\",\n          \"CRV333\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating_review\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 5,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          5,\n          3,\n          1\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"review_full\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 20,\n   \"samples\": [\n
\"Totally in love with the Auro of the place, really beautiful and quite fancy at the same time. The ambience is very pure and gives a sense of positivity throughout. Outdoor and indoor interior are quite quaint and cute. Love the open kitchen idea and there whole marketplace ideology. Due to coronovirus they specifically use disposable cutlery to keep the pandemic in mind taking all the precautionary measures from the beginning of the place with the mask on their staff and using good sanitisation. The food is really amazing specially the pizza straight from the oven and the hummus and pita bread are quite delicious too. If you're looking for a classy yet soothing Italian place in Delhi,Fatjar is a go to for you!\",\n
\"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75 expecting great things but we were disappointed. The place has great trip advisor reviews so we had high expectations that were sadly not met. What really ruined our nice meal was the self service. A waiter gave us (a party of 5) two tablets to look at the menu on. Unlike with proper menus, where everyone can browse at their own speed we had to share a tablet between 5. We asked the waiter for more tablets and were given one. After we had eventually all used the tablets to browse and input our orders on to one tablet the tablet then had an error message and froze. We used the other tablet to order and that one ran out of battery. We were hungry after a long day travelling this was stressful. Eventually one of the waiters came over and took details of our order. I really disliked this way of ordering food, and would not use the restaurant again because of this. The table was also dirty, there was dust and bits of fluff on the table and cloth napkins and my aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment. Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n   \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a

preference, cold water or temperature. 3700 rupees \\u20ac46 for two
sandwiches, bottled water and terrible service! Certainly not worth
it. I reckon the street food stalls would give a better service then
here \"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"model_response\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"num_unique_values\": 17,\n   \"samples\":
[\n   \"\\n   {\\n   \\\"Overall\\\": \\\"Positive\\\",\\
n   \\\"Food Quality\\\": \\\"Positive\\\",\\n
\\\"Service\\\": \\\"Positive\\\",\\n   \\\"Ambience\\\":
\\\"Positive\\\"\\n   }\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\

```
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\",\n
\"\\n    {\\n        \\\"Overall\\\": \\\"Positive\\\",\\n
\\\"Food Quality\\\": \\\"Positive\\\",\\n        \\\"Service\\\":
\\\"Neutral\\\",\\n        \\\"Ambience\\\": \\\"Positive\\\"\\
n    }\\n\\nHere's my analysis of the review:\\n\\n* Overall
sentiment: The review expresses a positive sentiment towards the
restaurant overall.\\n* Food Quality: The reviewer enjoyed the food
quality, specifically mentioning that the Margherita pizza was
\\\"exquisite.\\\" Sentiment: Positive.\\n* Service: The reviewer
found the service to be neutral, mentioning that the food was prepared
freshly in front of them but not commenting further on the service
quality. Sentiment: Neutral.\\n* Ambience: The reviewer appreciated
the peaceful ambience of the restaurant, enhanced by plants.
Sentiment: Positive.\",\n           \"\\n    {\\n
\\\"Overall\\\": \\\"Neutral\\\",\\n        \\\"Food Quality\\\":
\\\"Negative\\\",\\n        \\\"Service\\\": \\\"Positive\\\",\\n
\\\"Ambience\\\": \\\"Positive\\\"\\n    }\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
```

```
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\
n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\\n\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"model_response_parsed\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Overall\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"Positive\",\n          \"Neutral\",\n
\"Negative\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Food Quality\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \"Negative\",\n          \"Not Applicable\",\n
\"Mixed\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Service\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \"Neutral\",\n          \"Negative\",\n
\"Slow\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ambience\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"Not Applicable\",\n          \"Negative\",\n
\"Positive\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data_with_parsed_model_output_
3"}
```


```
final_data_3 =
data_with_parsed_model_output_3.drop(['model_response','model_response
_parsed'], axis=1)
final_data_3.head()
```


```
{"summary":"{\n  \"name\": \"final_data_3\",\n  \"rows\": 20,\n
\"fields\": [\n    {\n      \"column\": \"restaurant_ID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 20,\n        \"samples\": [\n
```

\"FLV202\",\n   \"PIZ555\",\n   \"CRV333\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"rating_review\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1,\n   \"min\": 1,\n   \"max\": 5,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   5,\n
3,\n   1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"review_full\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"num_unique_values\": 20,\n   \"samples\":
[\n   \"Totally in love with the Auro of the place, really
beautiful and quite fancy at the same time. The ambience is very pure
and gives a sense of positivity throughout. Outdoor and indoor
interior are quite quaint and cute. Love the open kitchen idea and
there whole marketplace ideology. Due to coronovirus they specifically
use disposable cutlery to keep the pandemic in mind taking all the
precautionary measures from the beginning of the place with the mask
on their staff and using good sanitisation. The food is really amazing
specially the pizza straight from the oven and the hummus and pita
bread are quite delicious too. If you're looking for a classy yet
soothing Italian place in Delhi,Fatjar is a go to for you!\",\n
\"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75
expecting great things but we were disappointed. The place has great
trip advisor reviews so we had high expectations that were sadly not
met. What really ruined our nice meal was the self service. A waiter
gave us (a party of 5) two tablets to look at the menu on. Unlike with
proper menus, where everyone can browse at their own speed we had to
share a tablet between 5. We asked the waiter for more tablets and
were given one. After we had eventually all used the tablets to browse
and input our orders on to one tablet the tablet then had an error
message and froze. We used the other tablet to order and that one ran
out of battery. We were hungry after a long day travelling this was
stressful. Eventually one of the waiters came over and took details of
our order. I really disliked this way of ordering food, and would not
use the restaurant again because of this. The table was also dirty,
there was dust and bits of fluff on the table and cloth napkins and my
aunt was given a chipped glass which she discovered upon taking a
drink. We told the waiter and he apologised and got us a new glass but
again, this just really made our experience a disappointment. Very
disappointed as seems to have good reviews, maybe we came on a bad day
but we didn't like the menu/way of ordering, the table was dirty and a
member of our party was given a chipped glass.\",\n   \"SERVICE
HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough
(maybe because I was soooo hungry after a hectic day out) well
presented but the service put me off completely. 12 servers on the
floor including the manager and not one person came over to check back
to make sure all was okay with our meals. All hanging around talking.
And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star
service. And certainly not a 5 star hotel. We did ask for bottled

water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Overall\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"Positive\",\n   \"Neutral\",\n   \"Negative\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Food Quality\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 5,\n   \"samples\": [\n   \"Negative\",\n   \"Not Applicable\",\n   \"Mixed\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Service\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 5,\n   \"samples\": [\n   \"Neutral\",\n   \"Negative\",\n   \"Slow\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Ambience\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 4,\n   \"samples\": [\n   \"Not Applicable\",\n   \"Negative\",\n   \"Positive\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"final_data_3"}

```
final_data_3['Overall'].value_counts()
```

```
Overall
Neutral     7
Negative    7
Positive    6
Name: count, dtype: int64
```

```
final_data_3['Overall'].value_counts()
```

```
Overall
Neutral     7
Negative    7
Positive    6
Name: count, dtype: int64
```

```
final_data_3['Food Quality'].value_counts()
```

```
Food Quality
Positive          7
Neutral           7
Negative          3
Mixed             2
Not Applicable    1
Name: count, dtype: int64
```

```
final_data_3['Service'].value_counts()
```

```
Service
Negative        9
Positive        8
Neutral         1
Slow            1
Inconsistent    1
Name: count, dtype: int64
```

```
final_data_3['Ambience'].value_counts()
```

```
Ambience
Positive          9
Not Applicable    6
Neutral           4
Negative          1
Name: count, dtype: int64
```

```
data_3 = data.copy()
```

```
def response_2(prompt,review,sentiment):
    model_output = mistral_llm(
      f"""
      Q: {prompt}
      review: {review}
      sentiment: {sentiment}
      A:
      """,
      max_tokens=64,
      stop=["Q:", "\n"],
      temperature=0.01,
      echo=False,
    )

    temp_output = model_output["choices"][0]["text"]
    final_output = temp_output[temp_output.index('{'):]

    return final_output

instruction_3 = """
    You are provided a review and it's sentiment.
```

```
    Instructions:
    Classify the sentiment of each aspect as either of "Positive",
"Negative", or "Neutral" only and not any other for the given review:
    1. "Food Quality"
    2. "Service"
    3. "Ambience"
    In case one of the three aspects is not mentioned in the review,
return "Not Applicable" (including quotes) for the corresponding JSON
key value.
    Return the output in the format {"Overall": given sentiment
input,"Food Quality": "your_sentiment_prediction","Service":
"your_sentiment_prediction","Ambience": "your_sentiment_prediction"}

"""

data_3['model_response'] =
final_data_1[['review_full','sentiment']].apply(lambda x:
response_2(instruction_3, x[0],x[1]),axis=1)
```

```
/tmp/ipython-input-399174068.py:1: FutureWarning: Series.__getitem__
treating keys as positions is deprecated. In a future version, integer
keys will always be treated as labels (consistent with DataFrame
behavior). To access a value by position, use `ser.iloc[pos]`
  data_3['model_response'] =
final_data_1[['review_full','sentiment']].apply(lambda x:
response_2(instruction_3, x[0],x[1]),axis=1)
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.97 ms /    31 runs   (
0.58 ms per token,  1725.29 tokens per second)
llama_print_timings: prompt eval time =    2333.30 ms /   172 tokens (
13.57 ms per token,    73.72 tokens per second)
llama_print_timings:        eval time =   23882.27 ms /    30 runs   (
796.08 ms per token,     1.26 tokens per second)
llama_print_timings:       total time =   26351.60 ms /   202 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      18.32 ms /    33 runs   (
0.56 ms per token,  1801.80 tokens per second)
llama_print_timings: prompt eval time =    2818.41 ms /   241 tokens (
11.69 ms per token,    85.51 tokens per second)
llama_print_timings:        eval time =   25676.21 ms /    32 runs   (
802.38 ms per token,     1.25 tokens per second)
llama_print_timings:       total time =   28637.19 ms /   273 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      19.17 ms /    31 runs   (
```

```
0.62 ms per token,  1616.94 tokens per second)
llama_print_timings: prompt eval time =    1649.31 ms /    40 tokens (
41.23 ms per token,    24.25 tokens per second)
llama_print_timings:        eval time =   24290.30 ms /    30 runs   (
809.68 ms per token,     1.24 tokens per second)
llama_print_timings:       total time =   26072.88 ms /    70 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      19.08 ms /    35 runs   (
0.55 ms per token,  1834.29 tokens per second)
llama_print_timings: prompt eval time =    2202.72 ms /   149 tokens (
14.78 ms per token,    67.64 tokens per second)
llama_print_timings:        eval time =   27158.05 ms /    34 runs   (
798.77 ms per token,     1.25 tokens per second)
llama_print_timings:       total time =   29509.64 ms /   183 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      16.81 ms /    31 runs   (
0.54 ms per token,  1844.25 tokens per second)
llama_print_timings: prompt eval time =    2661.19 ms /   175 tokens (
15.21 ms per token,    65.76 tokens per second)
llama_print_timings:        eval time =   23837.81 ms /    30 runs   (
794.59 ms per token,     1.26 tokens per second)
llama_print_timings:       total time =   26628.67 ms /   205 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      16.68 ms /    31 runs   (
0.54 ms per token,  1859.07 tokens per second)
llama_print_timings: prompt eval time =    3279.34 ms /   223 tokens (
14.71 ms per token,    68.00 tokens per second)
llama_print_timings:        eval time =   23819.87 ms /    30 runs   (
794.00 ms per token,     1.26 tokens per second)
llama_print_timings:       total time =   27230.34 ms /   253 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.74 ms /    32 runs   (
0.55 ms per token,  1803.43 tokens per second)
llama_print_timings: prompt eval time =    3030.41 ms /   221 tokens (
13.71 ms per token,    72.93 tokens per second)
llama_print_timings:        eval time =   24871.52 ms /    31 runs   (
802.31 ms per token,     1.25 tokens per second)
llama_print_timings:       total time =   28043.94 ms /   252 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      18.89 ms /    34 runs   (
```

```
0.56 ms per token,  1800.28 tokens per second)
llama_print_timings: prompt eval time =    1820.71 ms /    75 tokens (
24.28 ms per token,    41.19 tokens per second)
llama_print_timings:        eval time =   26169.50 ms /    33 runs   (
793.02 ms per token,     1.26 tokens per second)
llama_print_timings:       total time =   28137.41 ms /   108 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      19.80 ms /    32 runs   (
0.62 ms per token,  1616.49 tokens per second)
llama_print_timings: prompt eval time =    1798.03 ms /    73 tokens (
24.63 ms per token,    40.60 tokens per second)
llama_print_timings:        eval time =   24548.18 ms /    31 runs   (
791.88 ms per token,     1.26 tokens per second)
llama_print_timings:       total time =   26483.03 ms /   104 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      19.25 ms /    35 runs   (
0.55 ms per token,  1818.56 tokens per second)
llama_print_timings: prompt eval time =    1794.95 ms /    75 tokens (
23.93 ms per token,    41.78 tokens per second)
llama_print_timings:        eval time =   26633.68 ms /    34 runs   (
783.34 ms per token,     1.28 tokens per second)
llama_print_timings:       total time =   28577.79 ms /   109 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      18.12 ms /    33 runs   (
0.55 ms per token,  1821.09 tokens per second)
llama_print_timings: prompt eval time =    1783.49 ms /    72 tokens (
24.77 ms per token,    40.37 tokens per second)
llama_print_timings:        eval time =   25445.64 ms /    32 runs   (
795.18 ms per token,     1.26 tokens per second)
llama_print_timings:       total time =   27369.44 ms /   104 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.21 ms /    32 runs   (
0.54 ms per token,  1859.28 tokens per second)
llama_print_timings: prompt eval time =    1805.79 ms /    77 tokens (
23.45 ms per token,    42.64 tokens per second)
llama_print_timings:        eval time =   24379.27 ms /    31 runs   (
786.43 ms per token,     1.27 tokens per second)
llama_print_timings:       total time =   26321.27 ms /   108 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.49 ms /    32 runs   (
```

```
0.55 ms per token,  1829.41 tokens per second)
llama_print_timings: prompt eval time =    1764.23 ms /    71 tokens (
24.85 ms per token,    40.24 tokens per second)
llama_print_timings:        eval time =   24479.34 ms /    31 runs   (
789.66 ms per token,     1.27 tokens per second)
llama_print_timings:       total time =   26379.51 ms /   102 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      18.68 ms /    34 runs   (
0.55 ms per token,  1819.64 tokens per second)
llama_print_timings: prompt eval time =    3957.56 ms /   360 tokens (
10.99 ms per token,    90.97 tokens per second)
llama_print_timings:        eval time =   26662.34 ms /    33 runs   (
807.95 ms per token,     1.24 tokens per second)
llama_print_timings:       total time =   30766.34 ms /   393 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.62 ms /    33 runs   (
0.53 ms per token,  1872.77 tokens per second)
llama_print_timings: prompt eval time =   16068.44 ms /   532 tokens (
30.20 ms per token,    33.11 tokens per second)
llama_print_timings:        eval time =   25686.12 ms /    32 runs   (
802.69 ms per token,     1.25 tokens per second)
llama_print_timings:       total time =   41897.81 ms /   564 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      18.72 ms /    34 runs   (
0.55 ms per token,  1816.34 tokens per second)
llama_print_timings: prompt eval time =    3091.80 ms /   280 tokens (
11.04 ms per token,    90.56 tokens per second)
llama_print_timings:        eval time =   26309.20 ms /    33 runs   (
797.25 ms per token,     1.25 tokens per second)
llama_print_timings:       total time =   29549.25 ms /   313 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.80 ms /    33 runs   (
0.54 ms per token,  1853.93 tokens per second)
llama_print_timings: prompt eval time =    3218.51 ms /   290 tokens (
11.10 ms per token,    90.10 tokens per second)
llama_print_timings:        eval time =   25392.71 ms /    32 runs   (
793.52 ms per token,     1.26 tokens per second)
llama_print_timings:       total time =   28752.00 ms /   322 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.63 ms /    33 runs   (
```

```
0.53 ms per token,  1871.92 tokens per second)
llama_print_timings: prompt eval time =    3972.87 ms /   341 tokens (
11.65 ms per token,    85.83 tokens per second)
llama_print_timings:        eval time =   25221.13 ms /    32 runs   (
788.16 ms per token,     1.27 tokens per second)
llama_print_timings:       total time =   29334.72 ms /   373 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.13 ms /    31 runs   (
0.55 ms per token,  1809.37 tokens per second)
llama_print_timings: prompt eval time =    4945.93 ms /   468 tokens (
10.57 ms per token,    94.62 tokens per second)
llama_print_timings:        eval time =   24175.99 ms /    30 runs   (
805.87 ms per token,     1.24 tokens per second)
llama_print_timings:       total time =   29262.73 ms /   498 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    3330.86 ms
llama_print_timings:      sample time =      17.43 ms /    32 runs   (
0.54 ms per token,  1835.92 tokens per second)
llama_print_timings: prompt eval time =    3255.73 ms /   303 tokens (
10.74 ms per token,    93.07 tokens per second)
llama_print_timings:        eval time =   24711.13 ms /    31 runs   (
797.13 ms per token,     1.25 tokens per second)
llama_print_timings:       total time =   28107.28 ms /   334 tokens
```

```
data_3['model_response'].values
```

```
array(['{"Overall": "Positive","Food Quality": "Positive","Service":
"Positive","Ambience": "Positive"}',
       '{"Overall": "Positive","Food Quality": "Positive","Service":
"Not Applicable","Ambience": "Positive"}',
       '{"Overall": "Positive","Food Quality": "Positive","Service":
"Positive","Ambience": "Positive"}',
       '{"Overall": "Positive","Food Quality": "Not
Applicable","Service": "Positive","Ambience": "Not Applicable"}',
       '{"Overall": "Positive","Food Quality": "Positive","Service":
"Positive","Ambience": "Positive"}',
       '{"Overall": "Positive","Food Quality": "Positive","Service":
"Positive","Ambience": "Positive"}',
       '{"Overall": "Positive","Food Quality": "Neutral","Service":
"Positive","Ambience": "Positive"}',
       '{"Overall": "Positive","Food Quality": "Neutral","Service":
"Negative","Ambience": "Not Applicable"}',
       '{"Overall": "Positive","Food Quality": "Neutral","Service":
"Negative","Ambience": "Positive"}',
       '{"Overall": "Neutral","Food Quality": "If not
exceptional","Service": "Positive","Ambience": "Not Applicable"}',
       '{"Overall": "Neutral","Food Quality": "Decent","Service":
```

```
"Prompt","Ambience": "Neutral"}',
       '{"Overall": "Positive","Food Quality": "Neutral","Service":
"Negative","Ambience": "Positive"}',
       '{"Overall": "Neutral","Food Quality": "Negative","Service":
"Positive","Ambience": "Positive"}',
       '{"Overall": "Negative","Food Quality": "Not
Applicable","Service": "Negative","Ambience": "Neutral"}',
       '{"Overall": "Negative","Food Quality": "Negative","Service":
"Negative","Ambience": "Not Applicable"}',
       '{"Overall": "Negative","Food Quality": "Neutral","Service":
"Negative","Ambience": "Not Applicable"}',
       '{"Overall": "Negative","Food Quality": "Not
Applicable","Service": "Negative","Ambience": "Negative"}',
       '{"Overall": "Negative","Food Quality": "Not
Applicable","Service": "Negative","Ambience": "Negative"}',
       '{"Overall": "Negative","Food Quality": "Negative","Service":
"Negative","Ambience": "Negative"}',
       '{"Overall": "Negative","Food Quality": "Positive","Service":
"Negative","Ambience": "Neutral"}'],
      dtype=object)

i = 2
print(data_3.loc[i, 'review_full'])

Excellent taste and awesome decorum. Must visit. Subham Barnwal had
given us a great service. One of the best experience.

print(data_3.loc[i, 'model_response'])

{"Overall": "Positive","Food Quality": "Positive","Service":
"Positive","Ambience": "Positive"}

data_3['model_response_parsed'] =
data_3['model_response'].apply(extract_json_data)
data_3['model_response_parsed']

0     {'Overall': 'Positive', 'Food Quality': 'Posit...
1     {'Overall': 'Positive', 'Food Quality': 'Posit...
2     {'Overall': 'Positive', 'Food Quality': 'Posit...
3     {'Overall': 'Positive', 'Food Quality': 'Not A...
4     {'Overall': 'Positive', 'Food Quality': 'Posit...
5     {'Overall': 'Positive', 'Food Quality': 'Posit...
6     {'Overall': 'Positive', 'Food Quality': 'Neutr...
7     {'Overall': 'Positive', 'Food Quality': 'Neutr...
8     {'Overall': 'Positive', 'Food Quality': 'Neutr...
9     {'Overall': 'Neutral', 'Food Quality': 'If not...
10    {'Overall': 'Neutral', 'Food Quality': 'Decent...
11    {'Overall': 'Positive', 'Food Quality': 'Neutr...
12    {'Overall': 'Neutral', 'Food Quality': 'Negati...
13    {'Overall': 'Negative', 'Food Quality': 'Not A...
14    {'Overall': 'Negative', 'Food Quality': 'Negat...
```

```
15    {'Overall': 'Negative', 'Food Quality': 'Neutr...
16    {'Overall': 'Negative', 'Food Quality': 'Not A...
17    {'Overall': 'Negative', 'Food Quality': 'Not A...
18    {'Overall': 'Negative', 'Food Quality': 'Negat...
19    {'Overall': 'Negative', 'Food Quality': 'Posit...
Name: model_response_parsed, dtype: object
```

```
model_response_parsed_df_3 =
pd.json_normalize(data_3['model_response_parsed'])
model_response_parsed_df_3
```

```
{"summary":"{\n  \"name\": \"model_response_parsed_df_3\",\n
\"rows\": 20,\n  \"fields\": [\n    {\n      \"column\": \"Overall\",\
n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Positive\",\n          \"Neutral\",\n          \"Negative\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Food Quality\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n
\"Positive\",\n          \"Not Applicable\",\n          \"Negative\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Service\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n          \"Not
Applicable\",\n          \"Prompt\",\n          \"Positive\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ambience\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"Not Applicable\",\n          \"Negative\",\n
\"Positive\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"model_response_parsed_df_3"}
```

```
model_response_parsed_df_3 = model_response_parsed_df_3.apply(lambda
x: x.astype(str).str.lower())
```

```
data_with_parsed_model_output_3 = pd.concat([data_3,
model_response_parsed_df_3], axis=1)
data_with_parsed_model_output_3.head()
```

```
{"summary":"{\n  \"name\": \"data_with_parsed_model_output_3\",\n
\"rows\": 20,\n  \"fields\": [\n    {\n      \"column\":
\"restaurant_ID\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 20,\n        \"samples\":
[\n          \"FLV202\",\n          \"PIZ555\",\n          \"CRV333\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating_review\",\n      \"properties\": {\n        \"dtype\":
```

\"number\",\n   \"std\": 1,\n   \"min\": 1,\n
\"max\": 5,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   5,\n   3,\n   1\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"review_full\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 20,\n   \"samples\": [\n
\"Totally in love with the Auro of the place, really beautiful and quite fancy at the same time. The ambience is very pure and gives a sense of positivity throughout. Outdoor and indoor interior are quite quaint and cute. Love the open kitchen idea and there whole marketplace ideology. Due to coronovirus they specifically use disposable cutlery to keep the pandemic in mind taking all the precautionary measures from the beginning of the place with the mask on their staff and using good sanitisation. The food is really amazing specially the pizza straight from the oven and the hummus and pita bread are quite delicious too. If you're looking for a classy yet soothing Italian place in Delhi,Fatjar is a go to for you!\",\n
\"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75 expecting great things but we were disappointed. The place has great trip advisor reviews so we had high expectations that were sadly not met. What really ruined our nice meal was the self service. A waiter gave us (a party of 5) two tablets to look at the menu on. Unlike with proper menus, where everyone can browse at their own speed we had to share a tablet between 5. We asked the waiter for more tablets and were given one. After we had eventually all used the tablets to browse and input our orders on to one tablet the tablet then had an error message and froze. We used the other tablet to order and that one ran out of battery. We were hungry after a long day travelling this was stressful. Eventually one of the waiters came over and took details of our order. I really disliked this way of ordering food, and would not use the restaurant again because of this. The table was also dirty, there was dust and bits of fluff on the table and cloth napkins and my aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment. Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n   \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained

```
that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"model_response\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 15,\n        \"samples\": [\n          \"{\\\"Overall\\\": \\\"Negative\\\",\\\"Food Quality\\\": \\\"Not Applicable\\\",\\\"Service\\\": \\\"Negative\\\",\\\"Ambience\\\": \\\"Neutral\\\"}\",\n          \"{\\\"Overall\\\": \\\"Negative\\\",\\\"Food Quality\\\": \\\"Neutral\\\",\\\"Service\\\": \\\"Negative\\\",\\\"Ambience\\\": \\\"Not Applicable\\\"}\",\n          \"{\\\"Overall\\\": \\\"Positive\\\",\\\"Food Quality\\\": \\\"Positive\\\",\\\"Service\\\": \\\"Positive\\\",\\\"Ambience\\\": \\\"Positive\\\"}\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"model_response_parsed\",\n      \"properties\": {\n        \"dtype\": \"object\",\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Overall\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          \"positive\",\n          \"neutral\",\n          \"negative\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Food Quality\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 6,\n        \"samples\": [\n          \"positive\",\n          \"not applicable\",\n          \"negative\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Service\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"not applicable\",\n          \"prompt\",\n          \"positive\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Ambience\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"not applicable\",\n          \"negative\",\n          \"positive\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"data_with_parsed_model_output_3"}
```


```
final_data_3 = data_with_parsed_model_output_3.drop(['model_response','model_response_parsed'], axis=1)
final_data_3.head()
```

{"summary":"{\n \"name\": \"final_data_3\",\n \"rows\": 20,\n \"fields\": [\n   {\n   \"column\": \"restaurant_ID\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 20,\n   \"samples\": [\n   \"FLV202\",\n   \"PIZ555\",\n   \"CRV333\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"rating_review\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 1,\n   \"max\": 5,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   5,\n   3,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"review_full\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 20,\n   \"samples\": [\n   \"Totally in love with the Auro of the place, really beautiful and quite fancy at the same time. The ambience is very pure and gives a sense of positivity throughout. Outdoor and indoor interior are quite quaint and cute. Love the open kitchen idea and there whole marketplace ideology. Due to coronovirus they specifically use disposable cutlery to keep the pandemic in mind taking all the precautionary measures from the beginning of the place with the mask on their staff and using good sanitisation. The food is really amazing specially the pizza straight from the oven and the hummus and pita bread are quite delicious too. If you're looking for a classy yet soothing Italian place in Delhi,Fatjar is a go to for you!\",\n   \"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75 expecting great things but we were disappointed. The place has great trip advisor reviews so we had high expectations that were sadly not met. What really ruined our nice meal was the self service. A waiter gave us (a party of 5) two tablets to look at the menu on. Unlike with proper menus, where everyone can browse at their own speed we had to share a tablet between 5. We asked the waiter for more tablets and were given one. After we had eventually all used the tablets to browse and input our orders on to one tablet the tablet then had an error message and froze. We used the other tablet to order and that one ran out of battery. We were hungry after a long day travelling this was stressful. Eventually one of the waiters came over and took details of our order. I really disliked this way of ordering food, and would not use the restaurant again because of this. The table was also dirty, there was dust and bits of fluff on the table and cloth napkins and my aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment. Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n   \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back

```
to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Overall\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          \"positive\",\n          \"neutral\",\n          \"negative\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Food Quality\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 6,\n        \"samples\": [\n          \"positive\",\n          \"not applicable\",\n          \"negative\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Service\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"not applicable\",\n          \"prompt\",\n          \"positive\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Ambience\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"not applicable\",\n          \"negative\",\n          \"positive\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"final_data_3"}
```


```
final_data_3['Overall'].value_counts()
```

```
Overall
positive    10
negative     7
neutral      3
Name: count, dtype: int64
```

```
final_data_3['Food Quality'].value_counts()
```

```
Food Quality
positive              6
neutral               5
not applicable        4
```

```
negative               3
if not exceptional     1
decent                 1
Name: count, dtype: int64
```

```
final_data_3['Service'].value_counts()
```

```
Service
negative          10
positive           8
not applicable     1
prompt             1
Name: count, dtype: int64
```

```
final_data_3['Ambience'].value_counts()
```

```
Ambience
positive          9
not applicable    5
neutral           3
negative          3
Name: count, dtype: int64
```

```
data_4 = data.copy()
```

```
instruction_4 = """
    You are an AI tasked with analyzing restaurant reviews. Your goal
is to classify the overall sentiment of the provided review into the
following categories:
        - Positive
        - Negative
        - Neutral

    Subsequently, assess the sentiment of specific aspects mentioned
in the review, namely:
        1. Food quality
        2. Service
        3. Ambience

    Further, identify liked and/or disliked features associated with
each aspect in the review.

    Return the output in the specified JSON format, ensuring
consistency and handling missing values appropriately:

    {
        "Overall": "your_sentiment_prediction",
        "Food Quality": "your_sentiment_prediction",
        "Service": "your_sentiment_prediction",
        "Ambience": "your_sentiment_prediction",
        "Food Quality Features": ["liked/disliked features"],
```

```
        "Service Features": ["liked/disliked features"],
        "Ambience Features": ["liked/disliked features"]
    }

    The sentiment prediction for Overall, Food Quality, Service, and
Ambience should be one of "Positive", "Negative", or "Neutral" only.
    In case one of the three aspects is not mentioned in the review,
set "Not Applicable" (including quotes) in the corresponding JSON key
value for the sentiment.
    In case there are no liked/disliked features for a particular
aspect, assign an empty list in the corresponding JSON key value for
the aspect.
    Only return the JSON, do NOT return any other text or information.
"""

data_4['model_response'] = data_4['review_full'].apply(lambda x:
generate_llama_response(instruction_4, x).replace('\n', ''))
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     169.21 ms /   294 runs   (
0.58 ms per token,  1737.49 tokens per second)
llama_print_timings: prompt eval time =    1485.31 ms /   574 tokens (
2.59 ms per token,   386.45 tokens per second)
llama_print_timings:        eval time =   21874.41 ms /   293 runs   (
74.66 ms per token,    13.39 tokens per second)
llama_print_timings:       total time =   24554.05 ms /   867 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     140.31 ms /   246 runs   (
0.57 ms per token,  1753.25 tokens per second)
llama_print_timings: prompt eval time =    1705.54 ms /   645 tokens (
2.64 ms per token,   378.18 tokens per second)
llama_print_timings:        eval time =   18264.50 ms /   245 runs   (
74.55 ms per token,    13.41 tokens per second)
llama_print_timings:       total time =   20957.52 ms /   890 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     173.44 ms /   281 runs   (
0.62 ms per token,  1620.17 tokens per second)
llama_print_timings: prompt eval time =     971.37 ms /   435 tokens (
2.23 ms per token,   447.82 tokens per second)
llama_print_timings:        eval time =   19296.35 ms /   280 runs   (
68.92 ms per token,    14.51 tokens per second)
llama_print_timings:       total time =   21463.88 ms /   715 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     203.83 ms /   345 runs   (
0.59 ms per token,  1692.63 tokens per second)
llama_print_timings: prompt eval time =    1428.48 ms /   545 tokens (
2.62 ms per token,   381.52 tokens per second)
llama_print_timings:        eval time =   23816.55 ms /   344 runs   (
69.23 ms per token,    14.44 tokens per second)
llama_print_timings:       total time =   26718.64 ms /   889 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     127.90 ms /   227 runs   (
0.56 ms per token,  1774.78 tokens per second)
llama_print_timings: prompt eval time =    1508.57 ms /   582 tokens (
2.59 ms per token,   385.80 tokens per second)
llama_print_timings:        eval time =   16156.11 ms /   226 runs   (
71.49 ms per token,    13.99 tokens per second)
llama_print_timings:       total time =   18566.81 ms /   808 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     165.61 ms /   274 runs   (
0.60 ms per token,  1654.53 tokens per second)
llama_print_timings: prompt eval time =    1557.28 ms /   625 tokens (
2.49 ms per token,   401.34 tokens per second)
llama_print_timings:        eval time =   19737.91 ms /   273 runs   (
72.30 ms per token,    13.83 tokens per second)
llama_print_timings:       total time =   22443.45 ms /   898 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     194.77 ms /   335 runs   (
0.58 ms per token,  1719.94 tokens per second)
llama_print_timings: prompt eval time =    1562.55 ms /   631 tokens (
2.48 ms per token,   403.83 tokens per second)
llama_print_timings:        eval time =   23810.08 ms /   334 runs   (
71.29 ms per token,    14.03 tokens per second)
llama_print_timings:       total time =   26790.82 ms /   965 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     163.45 ms /   274 runs   (
0.60 ms per token,  1676.39 tokens per second)
llama_print_timings: prompt eval time =     998.11 ms /   470 tokens (
2.12 ms per token,   470.89 tokens per second)
llama_print_timings:        eval time =   18900.81 ms /   273 runs   (
69.23 ms per token,    14.44 tokens per second)
llama_print_timings:       total time =   21066.01 ms /   743 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     174.99 ms /   298 runs   (
0.59 ms per token,  1702.97 tokens per second)
llama_print_timings: prompt eval time =     987.59 ms /   472 tokens (
2.09 ms per token,   477.93 tokens per second)
llama_print_timings:        eval time =   20737.16 ms /   297 runs   (
69.82 ms per token,    14.32 tokens per second)
llama_print_timings:       total time =   22989.89 ms /   769 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     217.81 ms /   383 runs   (
0.57 ms per token,  1758.44 tokens per second)
llama_print_timings: prompt eval time =     990.80 ms /   471 tokens (
2.10 ms per token,   475.38 tokens per second)
llama_print_timings:        eval time =   26716.50 ms /   382 runs   (
69.94 ms per token,    14.30 tokens per second)
llama_print_timings:       total time =   29325.35 ms /   853 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     181.87 ms /   316 runs   (
0.58 ms per token,  1737.50 tokens per second)
llama_print_timings: prompt eval time =     994.92 ms /   471 tokens (
2.11 ms per token,   473.41 tokens per second)
llama_print_timings:        eval time =   21978.08 ms /   315 runs   (
69.77 ms per token,    14.33 tokens per second)
llama_print_timings:       total time =   24296.27 ms /   786 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     154.55 ms /   269 runs   (
0.57 ms per token,  1740.55 tokens per second)
llama_print_timings: prompt eval time =     994.52 ms /   471 tokens (
2.11 ms per token,   473.60 tokens per second)
llama_print_timings:        eval time =   18689.95 ms /   268 runs   (
69.74 ms per token,    14.34 tokens per second)
llama_print_timings:       total time =   20791.61 ms /   739 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     111.33 ms /   177 runs   (
0.63 ms per token,  1589.93 tokens per second)
llama_print_timings: prompt eval time =     989.67 ms /   467 tokens (
2.12 ms per token,   471.88 tokens per second)
llama_print_timings:        eval time =   12148.17 ms /   176 runs   (
69.02 ms per token,    14.49 tokens per second)
llama_print_timings:       total time =   13917.30 ms /   643 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     192.46 ms /   316 runs   (
0.61 ms per token,  1641.94 tokens per second)
llama_print_timings: prompt eval time =    1882.81 ms /   773 tokens (
2.44 ms per token,   410.56 tokens per second)
llama_print_timings:        eval time =   22547.63 ms /   315 runs   (
71.58 ms per token,    13.97 tokens per second)
llama_print_timings:       total time =   25845.22 ms /  1088 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     210.68 ms /   355 runs   (
0.59 ms per token,  1684.99 tokens per second)
llama_print_timings: prompt eval time =    2086.30 ms /   944 tokens (
2.21 ms per token,   452.48 tokens per second)
llama_print_timings:        eval time =   25780.08 ms /   354 runs   (
72.83 ms per token,    13.73 tokens per second)
llama_print_timings:       total time =   29525.96 ms /  1298 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     239.62 ms /   388 runs   (
0.62 ms per token,  1619.20 tokens per second)
llama_print_timings: prompt eval time =    1674.61 ms /   685 tokens (
2.44 ms per token,   409.05 tokens per second)
llama_print_timings:        eval time =   27701.65 ms /   387 runs   (
71.58 ms per token,    13.97 tokens per second)
llama_print_timings:       total time =   31165.77 ms /  1072 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     128.31 ms /   215 runs   (
0.60 ms per token,  1675.69 tokens per second)
llama_print_timings: prompt eval time =    1695.07 ms /   689 tokens (
2.46 ms per token,   406.47 tokens per second)
llama_print_timings:        eval time =   15219.72 ms /   214 runs   (
71.12 ms per token,    14.06 tokens per second)
llama_print_timings:       total time =   17925.18 ms /   903 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     190.99 ms /   308 runs   (
0.62 ms per token,  1612.66 tokens per second)
llama_print_timings: prompt eval time =    1744.21 ms /   753 tokens (
2.32 ms per token,   431.71 tokens per second)
llama_print_timings:        eval time =   22042.92 ms /   307 runs   (
71.80 ms per token,    13.93 tokens per second)
llama_print_timings:       total time =   25217.03 ms /  1060 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     172.16 ms /   295 runs   (
0.58 ms per token,  1713.55 tokens per second)
llama_print_timings: prompt eval time =    1966.07 ms /   883 tokens (
2.23 ms per token,   449.12 tokens per second)
llama_print_timings:        eval time =   21343.90 ms /   294 runs   (
72.60 ms per token,    13.77 tokens per second)
llama_print_timings:       total time =   24629.92 ms /  1177 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     160.73 ms /   266 runs   (
0.60 ms per token,  1654.97 tokens per second)
llama_print_timings: prompt eval time =    1695.78 ms /   706 tokens (
2.40 ms per token,   416.33 tokens per second)
llama_print_timings:        eval time =   19062.92 ms /   265 runs   (
71.94 ms per token,    13.90 tokens per second)
llama_print_timings:       total time =   21835.33 ms /   971 tokens
```

```
i = 2
print(data_4.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had
given us a great service. One of the best experience.
```

```
print(data_4.loc[i, 'model_response'])
```

```
 Sure, I can assist you in that! Here's my analysis of your provided
review: {"Overall": "Positive", "Food Quality": "Positive", "Service":
"Positive", "Ambience": "Positive", "Food Quality Features":
["Excellent taste"], "Service Features": ["Great service"], "Ambience
Features": ["Awesome decorum"]} Based on the review, I have determined
that: * The overall sentiment is positive, as mentioned in the
review's opening sentence. * The food quality is also positive, as
described as "Excellent taste." * The service is positive, as
described as "Great." * The ambience is positive, as described as
"Awesome decorum." * There were no mentioned liked/disliked features
for any aspect, so I have assigned empty lists for those values. * I
have set "Not Applicable" for the sentiment prediction for "Ambience"
since it was not mentioned in the review. Please note that I have
handled missing values appropriately by assigning appropriate values
based on the context of the review.
```

```
data_4['model_response_parsed'] =
data_4['model_response'].apply(extract_json_data)
data_4['model_response_parsed'].head()
```

```
0    {'Overall': 'Positive', 'Food Quality': 'Posit...
1    {'Overall': 'Positive', 'Food Quality': 'Posit...
2    {'Overall': 'Positive', 'Food Quality': 'Posit...
3    {'Overall': 'Positive', 'Food Quality': 'Posit...
```

```
4    {'Overall': 'Positive', 'Food Quality': 'Posit...
Name: model_response_parsed, dtype: object
```

```
data_4[data_4.model_response_parsed == {}]
```

```
{"repr_error":"Out of range float values are not JSON compliant:
nan","type":"dataframe"}
```

```
print(data_4.loc[3, 'model_response'])
```

```
 Sure! Here's the JSON output based on your review:{    "Overall":
"Positive",    "Food Quality": "Positive",    "Service": "Positive",
"Ambience": "Neutral",    "Food Quality Features": ["superb", "first-
class"],    "Service Features": ["superb", "handling"],    "Ambience
Features": [""]}Here's how I analyzed your review:* Overall: You
mentioned that you had a "superb" experience at JW Marriott Hotel
Aerocity, Delhi, which suggests a positive sentiment.* Food Quality:
You praised the "first-class" food quality, which also suggests a
positive sentiment.* Service: You appreciated the "superb" service you
received, which further supports a positive sentiment.* Ambience: You
did not mention anything specific about the ambience, so I have marked
it as "Neutral".For each aspect, I've listed the liked features (in
quotes) and disliked features (empty list), based on your review:*
Food Quality Features: ["superb", "first-class"] (liked)* Service
Features: ["superb", "handling"] (liked)* Ambience Features: [] (no
disliked features mentioned)Note that if you had mentioned any
disliked features or aspects, I would have included them in the
respective lists.
```

```
print(data_4.loc[6, 'model_response'])
```

```
    Sure! Here's my analysis of the review you provided:
{        "Overall": "Neutral",        "Food Quality": "Mixed",
"Service": "Positive",        "Ambience": "Positive",        "Food
Quality Features": ["pullled duck was prepared and seasoned well but
the meat had been marinated with too much balsamic vinegar"],
"Service Features": ["friendly and not intimidating"],
"Ambience Features": ["cozy feel"]    }Here's my analysis:* Overall:
Neutral - The reviewer had mixed feelings about their experience at
the restaurant.* Food Quality: Mixed - While the pulled duck was
prepared and seasoned well, the meat was marinated too much in
balsamic vinegar.* Service: Positive - The service was described as
friendly and not intimidating.* Ambience: Positive - The reviewer
enjoyed the cozy feel of the restaurant's ambience.Liked features:*
Food Quality: prepared and seasoned well* Service: friendly* Ambience:
cozy feelDisliked features:* Food Quality: too much balsamic vinegarI
hope that helps! Let me know if you have any further questions or if
you'd like me to analyze another review.
```

```
print(data_4.loc[7, 'model_response'])
```

```
    Sure! Here's the JSON output for the review you provided:
{    "Overall": "Neutral",    "Food Quality": "Mixed",    "Service":
"Slow",    "Ambience": "Not Applicable",    "Food Quality Features":
["mixed"],    "Service Features": ["slow"],    "Ambience Features":
["Not Applicable"]}Here's the breakdown:* Overall: Neutral - Based on
the review, the reviewer had a neutral impression of their visit to
Green Bites.* Food Quality: Mixed - While the reviewer mentions that
the food quality is "mixed", they do not provide any specific liked or
disliked features for this aspect.* Service: Slow - The reviewer notes
that the service was slow, which contributes to the neutral sentiment
for this aspect.* Ambience: Not Applicable - As the reviewer does not
mention the ambience at all, this aspect is not applicable for this
review.I hope this helps! Let me know if you have any further
questions or if you'd like me to analyze any other reviews.

upd_val_1 = {
    "Overall": "Positive",
    "Food Quality": "Positive",
    "Service": "Positive",
    "Ambience": "Not Applicable",
    "Food Quality Features": [],
    "Service Features": ["excellent service"],
    "Ambience Features": []
}

upd_val_2 = {
    "Overall": "Neutral",
    "Food Quality": "Neutral",
    "Service": "Neutral",
    "Ambience": "Not Applicable",
    "Food Quality Features": ["well prepared"],
    "Service Features": ["slow and inattentive"],
    "Ambience Features": ["interior is friendly", "not intimidating"]
}

upd_val_3 = {
    "Overall": "Neutral",
    "Food Quality": "Positive",
    "Service": "Negative",
    "Ambience": "Positive",
    "Food Quality Features": ["Some tasty, others average"],
    "Service Features": ["Attentive staff", "Slow service"],
    "Ambience Features": []
}

idx_list = [3,6,7]
data_4.loc[idx_list, 'model_response_parsed'] = [upd_val_1, upd_val_2,
upd_val_3]
```

```
model_response_parsed_df_4 =
pd.json_normalize(data_4['model_response_parsed'])
model_response_parsed_df_4.head()
```

```
{"summary":"{\n  \"name\": \"model_response_parsed_df_4\",\n
\"rows\": 20,\n  \"fields\": [\n    {\n      \"column\": \"Overall\",\
n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n
\"Neutral\",\n          \"Negative\",\n          \"Positive\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Food Quality\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n
\"Positive\",\n          \"Neutral\",\n          \"Negative\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Service\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Neutral\",\n          \"your_sentiment_prediction\",\n
\"Negative\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ambience\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \"Neutral\",\n          \"Negative\",\n          \"Not
Applicable\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Food Quality Features\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Service Features\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ambience Features\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"model_response_parsed_df_4"}
```

```
data_with_parsed_model_output_4 = pd.concat([data_4,
model_response_parsed_df_4], axis=1)
data_with_parsed_model_output_4.head()
```

```
{"summary":"{\n  \"name\": \"data_with_parsed_model_output_4\",\n
\"rows\": 20,\n  \"fields\": [\n    {\n      \"column\":
\"restaurant_ID\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 20,\n        \"samples\":
[\n          \"FLV202\",\n          \"PIZ555\",\n          \"CRV333\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating_review\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
```

\"max\": 5,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   5,\n   3,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"review_full\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 20,\n   \"samples\": [\n   \"Totally in love with the Auro of the place, really beautiful and quite fancy at the same time. The ambience is very pure and gives a sense of positivity throughout. Outdoor and indoor interior are quite quaint and cute. Love the open kitchen idea and there whole marketplace ideology. Due to coronovirus they specifically use disposable cutlery to keep the pandemic in mind taking all the precautionary measures from the beginning of the place with the mask on their staff and using good sanitisation. The food is really amazing specially the pizza straight from the oven and the hummus and pita bread are quite delicious too. If you're looking for a classy yet soothing Italian place in Delhi,Fatjar is a go to for you!\",\n   \"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75 expecting great things but we were disappointed. The place has great trip advisor reviews so we had high expectations that were sadly not met. What really ruined our nice meal was the self service. A waiter gave us (a party of 5) two tablets to look at the menu on. Unlike with proper menus, where everyone can browse at their own speed we had to share a tablet between 5. We asked the waiter for more tablets and were given one. After we had eventually all used the tablets to browse and input our orders on to one tablet the tablet then had an error message and froze. We used the other tablet to order and that one ran out of battery. We were hungry after a long day travelling this was stressful. Eventually one of the waiters came over and took details of our order. I really disliked this way of ordering food, and would not use the restaurant again because of this. The table was also dirty, there was dust and bits of fluff on the table and cloth napkins and my aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment. Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n   \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a

new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"model_response\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 20,\n   \"samples\": [\n   \"   {   \\\"Overall\\\": \\\"Positive\\\",   \\\"Food Quality\\\": \\\"Positive\\\",   \\\"Service\\\": \\\"Positive\\\",   \\\"Ambience\\\": \\\"Positive\\\",   \\\"Food Quality Features\\\": [\\\"straight from the oven\\\"],   \\\"Service Features\\\": [\\\"staff wearing masks\\\"],   \\\"Ambience Features\\\": [\\\"quite fancy\\\"]   }Here's my analysis of the review:* Overall: Positive - The reviewer loves the place and finds it to be classy yet soothing.* Food Quality: Positive - The reviewer enjoyed the food, specifically mentioning the pizza straight from the oven and hummus and pita bread as delicious.* Service: Positive - The reviewer appreciated the service, mentioning that the staff wears masks as a precautionary measure against COVID-19.* Ambience: Positive - The reviewer found the ambience to be quite fancy and positive.There are no disliked features mentioned in the review for any aspect. Liked features for Food Quality include \\\"straight from the oven.\\\" Liked features for Service include \\\"staff wearing masks.\\\" Liked features for Ambience include \\\"quite fancy.\\\"\",\n   \"   {   \\\"Overall\\\": \\\"Neutral\\\",   \\\"Food Quality\\\": \\\"Neutral\\\",   \\\"Service\\\": \\\"Negative\\\",   \\\"Ambience\\\": \\\"Neutral\\\",   \\\"Food Quality Features\\\": [\\\"\\\"],   \\\"Service Features\\\": [\\\"self-service\\\"],   \\\"Ambience Features\\\": [\\\"dirty table\\\"]   }Here's my analysis of your review:* Overall: Based on your experience, I classify your sentiment as Neutral, as you had mixed feelings about your dining experience.* Food Quality: You did not mention anything specific about the food quality, so I classify this aspect as Neutral as well.* Service: You mentioned that the self-service aspect of ordering food was a negative experience, hence I classify this aspect as Negative.* Ambience: You mentioned that the table was dirty, which suggests a Neutral sentiment for this aspect.Regarding liked/disliked features for each aspect:* Food Quality: No liked/disliked features mentioned.* Service: Disliked feature: self-service.* Ambience: Disliked feature: dirty table.Note: I have handled missing values by assigning an empty list for each aspect when there are no liked/disliked features mentioned in the review.\",\n   \"   {   \\\"Overall\\\": \\\"Negative\\\",   \\\"Food Quality\\\": \\\"Neutral\\\",   \\\"Service\\\": \\\"Negative\\\",   \\\"Ambience\\\": \\\"Not Applicable\\\",   \\\"Food Quality Features\\\": [\\\"well-presented\\\"],   \\\"Service Features\\\": [\\\"poor\\\"],

\\\"Ambience Features\\\": []   }Here's my analysis of the review:Overall: The reviewer had a negative experience at the restaurant, mentioning that the service was poor and the ambience was not good.Food Quality: The reviewer thought that the food quality was neutral, mentioning that the sandwich was \\\"well-presented\\\" but not commenting on the taste or quality of the ingredients.Service: The reviewer had a negative experience with the service, mentioning that there were 12 servers on the floor but none of them checked back to ensure that everything was okay with their meals.Ambience: The reviewer did not mention anything about the ambience, so it is not applicable in this case.Food Quality Features: The only feature mentioned for food quality is \\\"well-presented\\\", which suggests that the reviewer liked the presentation of their sandwich.Service Features: The only feature mentioned for service is \\\"poor\\\", which suggests that the reviewer did not like the service they received.Ambience Features: There are no features mentioned for ambience, as it is not applicable in this case.Overall, based on the review, it seems that the reviewer had a negative experience at the restaurant due to poor service and an unpleasant ambience, despite finding the food quality to be neutral.\"\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"model_response_parsed\",\n \"properties\": {\n   \"dtype\": \"object\",\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Overall\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 4,\n   \"samples\": [\n   \"Neutral\",\n \"Negative\",\n   \"Positive\"\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Food Quality\",\n \"properties\": {\n   \"dtype\": \"category\",\n \"num_unique_values\": 6,\n   \"samples\": [\n \"Positive\",\n   \"Neutral\",\n   \"Negative\"\n],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n }\n   },\n   {\n   \"column\": \"Service\",\n \"properties\": {\n   \"dtype\": \"category\",\n \"num_unique_values\": 5,\n   \"samples\": [\n \"Neutral\",\n   \"your_sentiment_prediction\",\n \"Negative\"\n   ],\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Ambience\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 5,\n   \"samples\": [\n   \"Neutral\",\n   \"Negative\",\n   \"Not Applicable\"\n   ],\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Food Quality Features\",\n   \"properties\": {\n \"dtype\": \"object\",\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Service Features\",\n   \"properties\": {\n   \"dtype\":

\"object\",\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Ambience Features\",\n   \"properties\": {\n   \"dtype\":
\"object\",\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"data_with_parsed_model_output_
4"}

```
final_data_4 =
data_with_parsed_model_output_4.drop(['model_response','model_response
_parsed'], axis=1)
final_data_4.head()
```

{"summary":"{\n \"name\": \"final_data_4\",\n \"rows\": 20,\n
\"fields\": [\n   {\n   \"column\": \"restaurant_ID\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 20,\n   \"samples\": [\n
\"FLV202\",\n   \"PIZ555\",\n   \"CRV333\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"rating_review\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1,\n   \"min\": 1,\n   \"max\": 5,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   5,\n
3,\n   1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"review_full\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"num_unique_values\": 20,\n   \"samples\":
[\n   \"Totally in love with the Auro of the place, really
beautiful and quite fancy at the same time. The ambience is very pure
and gives a sense of positivity throughout. Outdoor and indoor
interior are quite quaint and cute. Love the open kitchen idea and
there whole marketplace ideology. Due to coronovirus they specifically
use disposable cutlery to keep the pandemic in mind taking all the
precautionary measures from the beginning of the place with the mask
on their staff and using good sanitisation. The food is really amazing
specially the pizza straight from the oven and the hummus and pita
bread are quite delicious too. If you're looking for a classy yet
soothing Italian place in Delhi,Fatjar is a go to for you!\",\n
\"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75
expecting great things but we were disappointed. The place has great
trip advisor reviews so we had high expectations that were sadly not
met. What really ruined our nice meal was the self service. A waiter
gave us (a party of 5) two tablets to look at the menu on. Unlike with
proper menus, where everyone can browse at their own speed we had to
share a tablet between 5. We asked the waiter for more tablets and
were given one. After we had eventually all used the tablets to browse
and input our orders on to one tablet the tablet then had an error
message and froze. We used the other tablet to order and that one ran
out of battery. We were hungry after a long day travelling this was
stressful. Eventually one of the waiters came over and took details of

```
our order. I really disliked this way of ordering food, and would not use the restaurant again because of this.  The table was also dirty, there was dust and bits of fluff on the table and cloth napkins and my aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment.  Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n          \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Overall\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"Neutral\",\n          \"Negative\",\n          \"Positive\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Food Quality\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 6,\n        \"samples\": [\n          \"Positive\",\n          \"Neutral\",\n          \"Negative\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Service\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 5,\n        \"samples\": [\n          \"Neutral\",\n          \"your_sentiment_prediction\",\n          \"Negative\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Ambience\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 5,\n        \"samples\": [\n          \"Neutral\",\n          \"Negative\",\n          \"Not Applicable\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Food Quality Features\",\n      \"properties\": {\n        \"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Service Features\",\n   \"properties\": {\n   \"dtype\":
\"object\",\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Ambience Features\",\n   \"properties\": {\n   \"dtype\":
\"object\",\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"final_data_4"}

```
final_data_4['Overall'].value_counts()
```

```
Overall
Neutral                      7
Positive                     6
Negative                     6
your_sentiment_prediction    1
Name: count, dtype: int64
```

```
final_data_4['Food Quality'].value_counts()
```

```
Food Quality
Positive                     8
Neutral                      7
Not Applicable               2
Mixed                        1
your_sentiment_prediction    1
Negative                     1
Name: count, dtype: int64
```

```
final_data_4['Service'].value_counts()
```

```
Service
Negative                     9
Positive                     8
Neutral                      1
Slow                         1
your_sentiment_prediction    1
Name: count, dtype: int64
```

```
final_data_4['Ambience'].value_counts()
```

```
Ambience
Positive                     6
Neutral                      6
Not Applicable               6
your_sentiment_prediction    1
Negative                     1
Name: count, dtype: int64
```

```
data_5 = data.copy()
```

```
instruction_5 = """
    You are an AI analyzing restaurant reviews. Classify the overall
sentiment of the provided review into the following categories:
    - "Positive"
    - "Negative"
    - "Neutral"

    Once that is done, check for a mention of the following aspects in
the review and clasify the sentiment of each aspect as positive,
negative, or neutral:
    1. Food quality
    2. Service
    3. Ambience

    Once that is done, look for liked and/or disliked features
mentioned against each of the above aspects in the review and extract
them.

    Finally, draft a response for the customer based on the review.
Start out with a thank you note and then add on to it as per the
following:
    1. If the review is positive, mention that it would be great to
have them again
    2. If the review is neutral, ask them for what the restaurant
could have done better
    3. If the review is negative, apologive for the inconvenience and
mention that we'll be looking into the points raised

    Return the output in the specified JSON format, ensuring
consistency and handling missing values appropriately Ensure that all
values in the JSON are formatted as strings, and each element within
the lists should be enclosed in double quotes:

    {
        "Overall": "your_sentiment_prediction",
        "Food Quality": "your_sentiment_prediction",
        "Service": "your_sentiment_prediction",
        "Ambience": "your_sentiment_prediction",
        "Food Quality Features": ["liked/disliked features"],
        "Service Features": ["liked/disliked features"],
        "Ambience Features": ["liked/disliked features"],
        "Response": "your_response_to_the_customer_review",
    }

    The sentiment prediction for Overall, Food Quality, Service, and
Ambience should be one of "Positive", "Negative", or "Neutral" only.
    In case one of the three aspects is not mentioned in the review,
set "Not Applicable" (including quotes) in the corresponding JSON key
value for the sentiment.
    In case there are no liked/disliked features for a particular
```

```
aspect, assign an empty list in the corresponding JSON key value for
the aspect.
    Be polite and empathetic in the response to the customer review.
    Only return the JSON, do NOT return any other text or information.
"""

data_5['model_response'] = data_5['review_full'].apply(lambda x:
generate_llama_response(instruction_5, x))
```

```
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     216.30 ms /   370 runs   (
0.58 ms per token,  1710.56 tokens per second)
llama_print_timings: prompt eval time =    1740.23 ms /   764 tokens (
2.28 ms per token,   439.02 tokens per second)
llama_print_timings:        eval time =   27904.41 ms /   369 runs   (
75.62 ms per token,    13.22 tokens per second)
llama_print_timings:       total time =   31325.63 ms /  1133 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     207.73 ms /   360 runs   (
0.58 ms per token,  1733.01 tokens per second)
llama_print_timings: prompt eval time =    1957.54 ms /   835 tokens (
2.34 ms per token,   426.56 tokens per second)
llama_print_timings:        eval time =   25963.33 ms /   359 runs   (
72.32 ms per token,    13.83 tokens per second)
llama_print_timings:       total time =   29503.86 ms /  1194 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      87.33 ms /   152 runs   (
0.57 ms per token,  1740.54 tokens per second)
llama_print_timings: prompt eval time =    1518.14 ms /   625 tokens (
2.43 ms per token,   411.69 tokens per second)
llama_print_timings:        eval time =   10350.13 ms /   151 runs   (
68.54 ms per token,    14.59 tokens per second)
llama_print_timings:       total time =   12455.46 ms /   776 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     108.92 ms /   187 runs   (
0.58 ms per token,  1716.94 tokens per second)
llama_print_timings: prompt eval time =    1677.91 ms /   735 tokens (
2.28 ms per token,   438.05 tokens per second)
llama_print_timings:        eval time =   13073.76 ms /   186 runs   (
70.29 ms per token,    14.23 tokens per second)
llama_print_timings:       total time =   15540.23 ms /   921 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     240.39 ms /   424 runs   (
0.57 ms per token,  1763.79 tokens per second)
llama_print_timings: prompt eval time =    1881.92 ms /   772 tokens (
2.44 ms per token,   410.22 tokens per second)
llama_print_timings:        eval time =   31227.18 ms /   423 runs   (
73.82 ms per token,    13.55 tokens per second)
llama_print_timings:       total time =   34909.26 ms /  1195 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     312.03 ms /   511 runs   (
0.61 ms per token,  1637.68 tokens per second)
llama_print_timings: prompt eval time =    1923.02 ms /   815 tokens (
2.36 ms per token,   423.81 tokens per second)
llama_print_timings:        eval time =   36805.49 ms /   510 runs   (
72.17 ms per token,    13.86 tokens per second)
llama_print_timings:       total time =   41063.57 ms /  1325 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     210.41 ms /   359 runs   (
0.59 ms per token,  1706.19 tokens per second)
llama_print_timings: prompt eval time =    1887.08 ms /   821 tokens (
2.30 ms per token,   435.06 tokens per second)
llama_print_timings:        eval time =   25784.04 ms /   358 runs   (
72.02 ms per token,    13.88 tokens per second)
llama_print_timings:       total time =   29217.54 ms /  1179 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     174.42 ms /   292 runs   (
0.60 ms per token,  1674.13 tokens per second)
llama_print_timings: prompt eval time =    1647.31 ms /   660 tokens (
2.50 ms per token,   400.65 tokens per second)
llama_print_timings:        eval time =   20684.33 ms /   291 runs   (
71.08 ms per token,    14.07 tokens per second)
llama_print_timings:       total time =   23594.45 ms /   951 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     248.88 ms /   429 runs   (
0.58 ms per token,  1723.75 tokens per second)
llama_print_timings: prompt eval time =    1663.13 ms /   662 tokens (
2.51 ms per token,   398.04 tokens per second)
llama_print_timings:        eval time =   30775.64 ms /   428 runs   (
71.91 ms per token,    13.91 tokens per second)
llama_print_timings:       total time =   34282.54 ms /  1090 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     184.97 ms /   327 runs   (
0.57 ms per token,  1767.90 tokens per second)
llama_print_timings: prompt eval time =    1656.02 ms /   661 tokens (
2.51 ms per token,   399.15 tokens per second)
llama_print_timings:        eval time =   23358.20 ms /   326 runs   (
71.65 ms per token,    13.96 tokens per second)
llama_print_timings:       total time =   26384.71 ms /   987 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     300.04 ms /   520 runs   (
0.58 ms per token,  1733.10 tokens per second)
llama_print_timings: prompt eval time =    1668.51 ms /   661 tokens (
2.52 ms per token,   396.16 tokens per second)
llama_print_timings:        eval time =   37357.39 ms /   519 runs   (
71.98 ms per token,    13.89 tokens per second)
llama_print_timings:       total time =   41347.50 ms /  1180 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     272.85 ms /   477 runs   (
0.57 ms per token,  1748.23 tokens per second)
llama_print_timings: prompt eval time =    1668.50 ms /   661 tokens (
2.52 ms per token,   396.16 tokens per second)
llama_print_timings:        eval time =   34152.08 ms /   476 runs   (
71.75 ms per token,    13.94 tokens per second)
llama_print_timings:       total time =   38031.67 ms /  1137 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      79.95 ms /   130 runs   (
0.62 ms per token,  1626.00 tokens per second)
llama_print_timings: prompt eval time =    1664.35 ms /   657 tokens (
2.53 ms per token,   394.75 tokens per second)
llama_print_timings:        eval time =    9073.61 ms /   129 runs   (
70.34 ms per token,    14.22 tokens per second)
llama_print_timings:       total time =   11329.37 ms /   786 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     244.35 ms /   412 runs   (
0.59 ms per token,  1686.13 tokens per second)
llama_print_timings: prompt eval time =    2096.87 ms /   963 tokens (
2.18 ms per token,   459.26 tokens per second)
llama_print_timings:        eval time =   30359.44 ms /   411 runs   (
73.87 ms per token,    13.54 tokens per second)
llama_print_timings:       total time =   34281.17 ms /  1374 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     142.94 ms /   244 runs   (
0.59 ms per token,  1706.96 tokens per second)
llama_print_timings: prompt eval time =    2653.44 ms /  1134 tokens (
2.34 ms per token,   427.37 tokens per second)
llama_print_timings:        eval time =   18014.22 ms /   243 runs   (
74.13 ms per token,    13.49 tokens per second)
llama_print_timings:       total time =   21731.00 ms /  1377 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     295.88 ms /   472 runs   (
0.63 ms per token,  1595.26 tokens per second)
llama_print_timings: prompt eval time =    1945.31 ms /   875 tokens (
2.22 ms per token,   449.80 tokens per second)
llama_print_timings:        eval time =   34109.86 ms /   471 runs   (
72.42 ms per token,    13.81 tokens per second)
llama_print_timings:       total time =   38212.40 ms /  1346 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     184.61 ms /   311 runs   (
0.59 ms per token,  1684.62 tokens per second)
llama_print_timings: prompt eval time =    1936.91 ms /   879 tokens (
2.20 ms per token,   453.82 tokens per second)
llama_print_timings:        eval time =   22219.51 ms /   310 runs   (
71.68 ms per token,    13.95 tokens per second)
llama_print_timings:       total time =   25535.54 ms /  1189 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     261.52 ms /   423 runs   (
0.62 ms per token,  1617.44 tokens per second)
llama_print_timings: prompt eval time =    2068.81 ms /   943 tokens (
2.19 ms per token,   455.82 tokens per second)
llama_print_timings:        eval time =   30710.97 ms /   422 runs   (
72.77 ms per token,    13.74 tokens per second)
llama_print_timings:       total time =   34722.37 ms /  1365 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =     158.24 ms /   271 runs   (
0.58 ms per token,  1712.62 tokens per second)
llama_print_timings: prompt eval time =    2573.38 ms /  1073 tokens (
2.40 ms per token,   416.96 tokens per second)
llama_print_timings:        eval time =   19951.39 ms /   270 runs   (
73.89 ms per token,    13.53 tokens per second)
llama_print_timings:       total time =   23737.93 ms /  1343 tokens
Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1019.94 ms
llama_print_timings:      sample time =      97.17 ms /   156 runs   (
0.62 ms per token,  1605.52 tokens per second)
llama_print_timings: prompt eval time =    2010.97 ms /   896 tokens (
2.24 ms per token,   445.56 tokens per second)
llama_print_timings:        eval time =   11211.15 ms /   155 runs   (
72.33 ms per token,    13.83 tokens per second)
llama_print_timings:       total time =   13894.08 ms /  1051 tokens
```

```
i = 2
print(data_5.loc[i, 'review_full'])
```

```
Excellent taste and awesome decorum. Must visit. Subham Barnwal had
given us a great service. One of the best experience.
```

```
print(data_5.loc[i, 'model_response'])
```

```
 {
"Overall": "Positive",
"Food Quality": "Positive",
"Service": "Positive",
"Ambience": "Positive",
"Food Quality Features": ["Excellent taste"],
"Service Features": ["Great service"],
"Ambience Features": ["Awesome decorum"],
"Response": "Thank you for taking the time to share your positive
review with us! We're thrilled to hear that you enjoyed our food
quality, service, and ambience. We're constantly striving to improve,
so your feedback is invaluable to us. We hope to have you back again
soon!"]
}
```

```
data_5['model_response_parsed'] =
data_5['model_response'].apply(extract_json_data)
data_5['model_response_parsed'].head()
```

```
Error parsing JSON: Expecting property name enclosed in double quotes:
line 10 column 5 (char 539)
Error parsing JSON: Expecting ',' delimiter: line 9 column 277 (char
503)
Error parsing JSON: Expecting property name enclosed in double quotes:
line 10 column 5 (char 579)
Error parsing JSON: Expecting property name enclosed in double quotes:
line 10 column 5 (char 669)
```

```
0    {'Overall': 'Positive', 'Food Quality': 'Posit...
1                                                   {}
2                                                   {}
3    {'Overall': 'Positive', 'Food Quality': 'Posit...
```

```
4                                                    {}
Name: model_response_parsed, dtype: object

model_response_parsed_df_5 =
pd.json_normalize(data_5['model_response_parsed'])
model_response_parsed_df_5.head()
```


```
{"summary":"{\n  \"name\": \"model_response_parsed_df_5\",\n
\"rows\": 20,\n  \"fields\": [\n    {\n      \"column\": \"Overall\",\
n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Positive\",\n          \"Neutral\",\n          \"Negative\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Food Quality\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"Mixed\",\n          \"Not Applicable\",\n          \"Neutral\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Service\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Positive\",\n          \"Neutral\",\n          \"Negative\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Ambience\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n        \"samples\": [\n
\"Neutral\",\n          \"Negative\",\n          \"Positive\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Food Quality Features\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Service Features\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Ambience Features\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Response\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 16,\
n        \"samples\": [\n          \"Thank you for taking the time to
review us! We're thrilled to hear that you enjoyed our food quality,
service, and ambience. We're especially glad that you appreciated our
open kitchen concept and our use of disposable cutlery to ensure your
safety during these challenging times. We hope to have you back again
soon! If you have any further suggestions or feedback, please don't
hesitate to reach out to us.\",\n          \"Thank you so much for
taking the time to review your recent visit to JW Marriott Hotel at
Atrocity, Delhi! We are thrilled to hear that you had a positive
experience with us. Your kind words about our food quality, service,
and ambience are truly appreciated. We would be delighted to have you
```

```
again! If there's anything we could have done better, please don't
hesitate to let us know.\",\n          \"Thank you for taking the time
to review your recent visit to our restaurant! We're glad to hear that
our service was positive, but we're disappointed to hear that our food
quality and ambience didn't quite meet your expectations. We'll be
looking into these points and making improvements where we can.\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\
n}","type":"dataframe","variable_name":"model_response_parsed_df_5"}
```


```
model_response_parsed_df_5['Response'][0]
```


```
{"type":"string"}
```


```
data_with_parsed_model_output_5 = pd.concat([data_5,
model_response_parsed_df_5], axis=1)
data_with_parsed_model_output_5.head()
```


```
{"summary":"{\n  \"name\": \"data_with_parsed_model_output_5\",\n
\"rows\": 20,\n  \"fields\": [\n    {\n      \"column\":
\"restaurant_ID\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 20,\n        \"samples\":
[\n          \"FLV202\",\n          \"PIZ555\",\n          \"CRV333\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating_review\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 5,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          5,\n          3,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"review_full\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 20,\n        \"samples\": [\n
\"Totally in love with the Auro of the place, really beautiful and
quite fancy at the same time. The ambience is very pure and gives a
sense of positivity throughout. Outdoor and indoor interior are quite
quaint and cute. Love the open kitchen idea and there whole
marketplace ideology. Due to coronovirus they specifically use
disposable cutlery to keep the pandemic in mind taking all the
precautionary measures from the beginning of the place with the mask
on their staff and using good sanitisation. The food is really amazing
specially the pizza straight from the oven and the hummus and pita
bread are quite delicious too. If you're looking for a classy yet
soothing Italian place in Delhi,Fatjar is a go to for you!\",\n
\"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75
expecting great things but we were disappointed. The place has great
trip advisor reviews so we had high expectations that were sadly not
met.  What really ruined our nice meal was the self service. A waiter
gave us (a party of 5) two tablets to look at the menu on. Unlike with
proper menus, where everyone can browse at their own speed we had to
```

share a tablet between 5. We asked the waiter for more tablets and were given one. After we had eventually all used the tablets to browse and input our orders on to one tablet the tablet then had an error message and froze. We used the other tablet to order and that one ran out of battery. We were hungry after a long day travelling this was stressful. Eventually one of the waiters came over and took details of our order. I really disliked this way of ordering food, and would not use the restaurant again because of this. The table was also dirty, there was dust and bits of fluff on the table and cloth napkins and my aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment. Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n   \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"model_response\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 20,\n   \"samples\": [\n   \"\\n   {\\n   \\\"Overall\\\": \\\"Positive\\\",\\n   \\\"Food Quality\\\": \\\"Positive\\\",\\n   \\\"Service\\\": \\\"Positive\\\",\\n   \\\"Ambience\\\": \\\"Positive\\\",\\n   \\\"Food Quality Features\\\": [\\\"straight from the oven\\\"],\\n   \\\"Service Features\\\": [\\\"quite delicious\\\"],\\n   \\\"Ambience Features\\\": [\\\"quite quaint and cute\\\"],\\n   \\\"Response\\\": \\\"Thank you for taking the time to review us! We're thrilled to hear that you enjoyed our food quality, service, and ambience. We're especially glad that you appreciated our open kitchen concept and our use of disposable cutlery to ensure your safety during these challenging times. We hope to have you back again soon! If you have any further suggestions or feedback, please don't hesitate to reach out to us.\\\"\\n   }\\n\\nHere's how I analyzed the review:\\n\\n1. Overall

sentiment: Positive\\n2. Food Quality sentiment: Positive (mentioned as \\\"really amazing\\\")\\n3. Service sentiment: Positive (mentioned as \\\"quite delicious\\\")\\n4. Ambience sentiment: Positive (mentioned as \\\"quite quaint and cute\\\")\\n5. Food Quality features:\\n\\t* Liked: straight from the oven\\n6. Service features:\\n\\t* Liked: quite delicious\\n7. Ambience features:\\n\\t* Liked: quite quaint and cute\\n8. Response: Thank you note and an invitation to come back again, along with a mention of the restaurant's commitment to safety during the pandemic.\",\n \"\\n   {\\n   \\\"Overall\\\": \\\"Neutral\\\",\\n \\\"Food Quality\\\": \\\"Neutral\\\",\\n   \\\"Service\\\": \\\"Negative\\\",\\n   \\\"Ambience\\\": \\\"Neutral\\\",\\n \\\"Food Quality Features\\\": [\\\"\\\"],\\n   \\\"Service Features\\\": [\\\"dirty table, chipped glass\\\"],\\n \\\"Ambience Features\\\": [\\\"\\\"],\\n   \\\"Response\\\": \\\"Thank you for taking the time to share your feedback with us. We apologize for any inconvenience you experienced during your visit, particularly with the self-service ordering system and dirty table. We will look into these issues and make necessary improvements to ensure a better dining experience for all our guests. Your feedback is greatly appreciated and helps us to improve.\\\"\\n   }\\n\\nHere's how I analyzed the review:\\n\\n1. Overall sentiment: The review expresses a mix of positive and negative experiences, so I classify the overall sentiment as Neutral.\\n2. Food quality: Not mentioned in the review, so I set \\\"Not Applicable\\\" for this aspect.\\n3. Service: The review mentions \\\"dirty table\\\" and \\\"chipped glass\\\", which are negative experiences, so I classify the sentiment for service as Negative.\\n4. Ambience: Not mentioned in the review, so I set \\\"Not Applicable\\\" for this aspect.\\n\\nFor each aspect, I also extracted the liked/disliked features mentioned in the review and included them in the corresponding JSON key values. In case there were no liked/disliked features mentioned, I assigned an empty list to ensure consistency in the JSON format.\\n\\nFinally, I drafted a response to the customer based on their review, thanking them for their feedback and apologizing for any inconvenience they experienced. I also mentioned that their feedback is greatly appreciated and helps us to improve, which is a polite and empathetic way to address their concerns.\",\n   \"\\n   {\\n   \\\"Overall\\\": \\\"Negative\\\",\\n   \\\"Food Quality\\\": \\\"Neutral\\\",\\n \\\"Service\\\": \\\"Negative\\\",\\n   \\\"Ambience\\\": \\\"Neutral\\\",\\n   \\\"Food Quality Features\\\": [\\\"well-presented sandwiches\\\"],\\n   \\\"Service Features\\\": [\\\"poor service, no check-backs\\\"],\\n   \\\"Ambience Features\\\": [\\\"not a 5-star hotel\\\"],\\n \\\"Response\\\": \\\"Thank you for taking the time to share your review with us! We're sorry to hear that you did not enjoy your experience with us, particularly with regards to our service. We take these comments very seriously and will be looking into improving our service standards. If you would be willing, we would love to have you

```
back again to see how we have improved.\\\"\\n    }\\n\\nHere's how I
arrived at this output:\\n\\nOverall Sentiment:\\nThe overall
sentiment of the review is negative, as the customer expresses
disappointment with their experience at the restaurant, specifically
with the service.\\n\\nFood Quality Sentiment:\\nThe sentiment for
food quality is neutral, as the customer mentions that the sandwiches
were \\\"well-presented\\\" but does not provide any further details
or opinions about the taste or quality of the food.\\n\\nService
Sentiment:\\nThe sentiment for service is negative, as the customer
expresses frustration with the lack of check-backs from the server
staff during their meal.\\n\\nAmbience Sentiment:\\nThe sentiment for
ambience is neutral, as the customer does not provide any specific
comments or opinions about the atmosphere of the restaurant.\\n\\nFood
Quality Features:\\nThe only feature mentioned for food quality is
\\\"well-presented sandwiches\\\".\\n\\nService Features:\\nThe only
feature mentioned for service is \\\"poor service, no check-
backs\\\".\\n\\nAmbience Features:\\nNo features were mentioned for
ambience.\\n\\nResponse:\\nBased on the negative sentiment for
service, I would suggest apologizing for any inconvenience caused and
inviting the customer back again to see how we have improved our
service standards.\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"model_response_parsed\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Overall\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"Positive\",\n          \"Neutral\",\n
\"Negative\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Food Quality\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \"Mixed\",\n          \"Not Applicable\",\n
\"Neutral\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Service\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"Positive\",\n          \"Neutral\",\n
\"Negative\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ambience\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"Neutral\",\n          \"Negative\",\n
\"Positive\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Food Quality Features\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Service Features\",\n      \"properties\": {\n        \"dtype\":
```

```
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ambience Features\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Response\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 16,\n        \"samples\":
[\n          \"Thank you for taking the time to review us! We're
thrilled to hear that you enjoyed our food quality, service, and
ambience. We're especially glad that you appreciated our open kitchen
concept and our use of disposable cutlery to ensure your safety during
these challenging times. We hope to have you back again soon! If you
have any further suggestions or feedback, please don't hesitate to
reach out to us.\",\n          \"Thank you so much for taking the time
to review your recent visit to JW Marriott Hotel at Atrocity, Delhi!
We are thrilled to hear that you had a positive experience with us.
Your kind words about our food quality, service, and ambience are
truly appreciated. We would be delighted to have you again! If there's
anything we could have done better, please don't hesitate to let us
know.\",\n          \"Thank you for taking the time to review your
recent visit to our restaurant! We're glad to hear that our service
was positive, but we're disappointed to hear that our food quality and
ambience didn't quite meet your expectations. We'll be looking into
these points and making improvements where we can.\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\
n}","type":"dataframe","variable_name":"data_with_parsed_model_output_
5"}

final_data_5 =
data_with_parsed_model_output_5.drop(['model_response','model_response
_parsed'], axis=1)
final_data_5.head()

{"summary":"{\n  \"name\": \"final_data_5\",\n  \"rows\": 20,\n
\"fields\": [\n    {\n      \"column\": \"restaurant_ID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 20,\n        \"samples\": [\n
\"FLV202\",\n          \"PIZ555\",\n          \"CRV333\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"rating_review\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 1,\n        \"max\": 5,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          5,\n
3,\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"review_full\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 20,\n        \"samples\":
[\n          \"Totally in love with the Auro of the place, really
beautiful and quite fancy at the same time. The ambience is very pure
```

and gives a sense of positivity throughout. Outdoor and indoor interior are quite quaint and cute. Love the open kitchen idea and there whole marketplace ideology. Due to coronovirus they specifically use disposable cutlery to keep the pandemic in mind taking all the precautionary measures from the beginning of the place with the mask on their staff and using good sanitisation. The food is really amazing specially the pizza straight from the oven and the hummus and pita bread are quite delicious too. If you're looking for a classy yet soothing Italian place in Delhi,Fatjar is a go to for you!\",\n \"Whilst staying at the Welcolmhotel Dwarka we visited Pavillion 75 expecting great things but we were disappointed. The place has great trip advisor reviews so we had high expectations that were sadly not met. What really ruined our nice meal was the self service. A waiter gave us (a party of 5) two tablets to look at the menu on. Unlike with proper menus, where everyone can browse at their own speed we had to share a tablet between 5. We asked the waiter for more tablets and were given one. After we had eventually all used the tablets to browse and input our orders on to one tablet the tablet then had an error message and froze. We used the other tablet to order and that one ran out of battery. We were hungry after a long day travelling this was stressful. Eventually one of the waiters came over and took details of our order. I really disliked this way of ordering food, and would not use the restaurant again because of this. The table was also dirty, there was dust and bits of fluff on the table and cloth napkins and my aunt was given a chipped glass which she discovered upon taking a drink. We told the waiter and he apologised and got us a new glass but again, this just really made our experience a disappointment. Very disappointed as seems to have good reviews, maybe we came on a bad day but we didn't like the menu/way of ordering, the table was dirty and a member of our party was given a chipped glass.\",\n \"SERVICE HORRENDOUS!!!!! Listen to the reviews. Sandwich was decent enough (maybe because I was soooo hungry after a hectic day out) well presented but the service put me off completely. 12 servers on the floor including the manager and not one person came over to check back to make sure all was okay with our meals. All hanging around talking. And when I say 12 that\\u2019s not even an exaggeration. Not a 5 star service. And certainly not a 5 star hotel. We did ask for bottled water and when poured in to a glass you could see tiny bubbles forming and microscopic imperfections on the glass because it was so warm. I asked for cold water and I was told they did not have any? I explained that I will not be drinking the warm water after waiting 15 minutes a new bottle came out...coldish.. 5 star ? And they do not have cold water! Ridiculous. Maybe for further tourist this could be a preference, cold water or temperature. 3700 rupees \\u20ac46 for two sandwiches, bottled water and terrible service! Certainly not worth it. I reckon the street food stalls would give a better service then here \"\n   ],\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Overall\",\n   \"properties\": {\n   \"dtype\":

```
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"Positive\",\n          \"Neutral\",\n
\"Negative\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Food Quality\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \"Mixed\",\n          \"Not Applicable\",\n
\"Neutral\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Service\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"Positive\",\n          \"Neutral\",\n
\"Negative\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ambience\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"Neutral\",\n          \"Negative\",\n
\"Positive\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Food Quality Features\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Service Features\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ambience Features\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Response\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 16,\n        \"samples\":
[\n          \"Thank you for taking the time to review us! We're
thrilled to hear that you enjoyed our food quality, service, and
ambience. We're especially glad that you appreciated our open kitchen
concept and our use of disposable cutlery to ensure your safety during
these challenging times. We hope to have you back again soon! If you
have any further suggestions or feedback, please don't hesitate to
reach out to us.\",\n          \"Thank you so much for taking the time
to review your recent visit to JW Marriott Hotel at Atrocity, Delhi!
We are thrilled to hear that you had a positive experience with us.
Your kind words about our food quality, service, and ambience are
truly appreciated. We would be delighted to have you again! If there's
anything we could have done better, please don't hesitate to let us
know.\",\n          \"Thank you for taking the time to review your
recent visit to our restaurant! We're glad to hear that our service
was positive, but we're disappointed to hear that our food quality and
ambience didn't quite meet your expectations. We'll be looking into
these points and making improvements where we can.\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"final_data_5"}
```

```
final_data_5['Overall'].value_counts()
```

```
Overall
Neutral     9
Negative    5
Positive    2
Name: count, dtype: int64
```

```
final_data_5['Food Quality'].value_counts()
```

```
Food Quality
Neutral           9
Positive          3
Negative          2
Mixed             1
Not Applicable    1
Name: count, dtype: int64
```

```
final_data_5['Service'].value_counts()
```

```
Service
Negative    9
Positive    5
Neutral     2
Name: count, dtype: int64
```

```
final_data_5['Ambience'].value_counts()
```

```
Ambience
Neutral           10
Positive           4
Not Applicable     1
Negative           1
Name: count, dtype: int64
```