Homework 3

1. Weaviate

Connect to weaviate and create the schema

```
import weaviate

client = weaviate.Client(
    url="http://localhost:8080"
)
```

```
schema_class = {
    "class": "Recipe",
    "vectorizer": "text2vec-contextionary",
    "properties": [
        {
            "name": "recipe_id",
            "dataType": ["string"],
        },
        {
            "name": "description",
            "dataType": ["string"],
        }
    ]
}

client.schema.create_class(schema_class)
```

Read and process input data

```
import pandas as pd

df = pd.read_csv('part1/RAW_recipes.csv')

null_count = df.isnull().sum()
print(null_count)
```

```
name                 1
id                   0
minutes              0
contributor_id       0
submitted            0
tags                 0
nutrition            0
n_steps              0
steps                0
description       4979
ingredients          0
n_ingredients        0
dtype: int64
```

Plots

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from scipy import stats
```

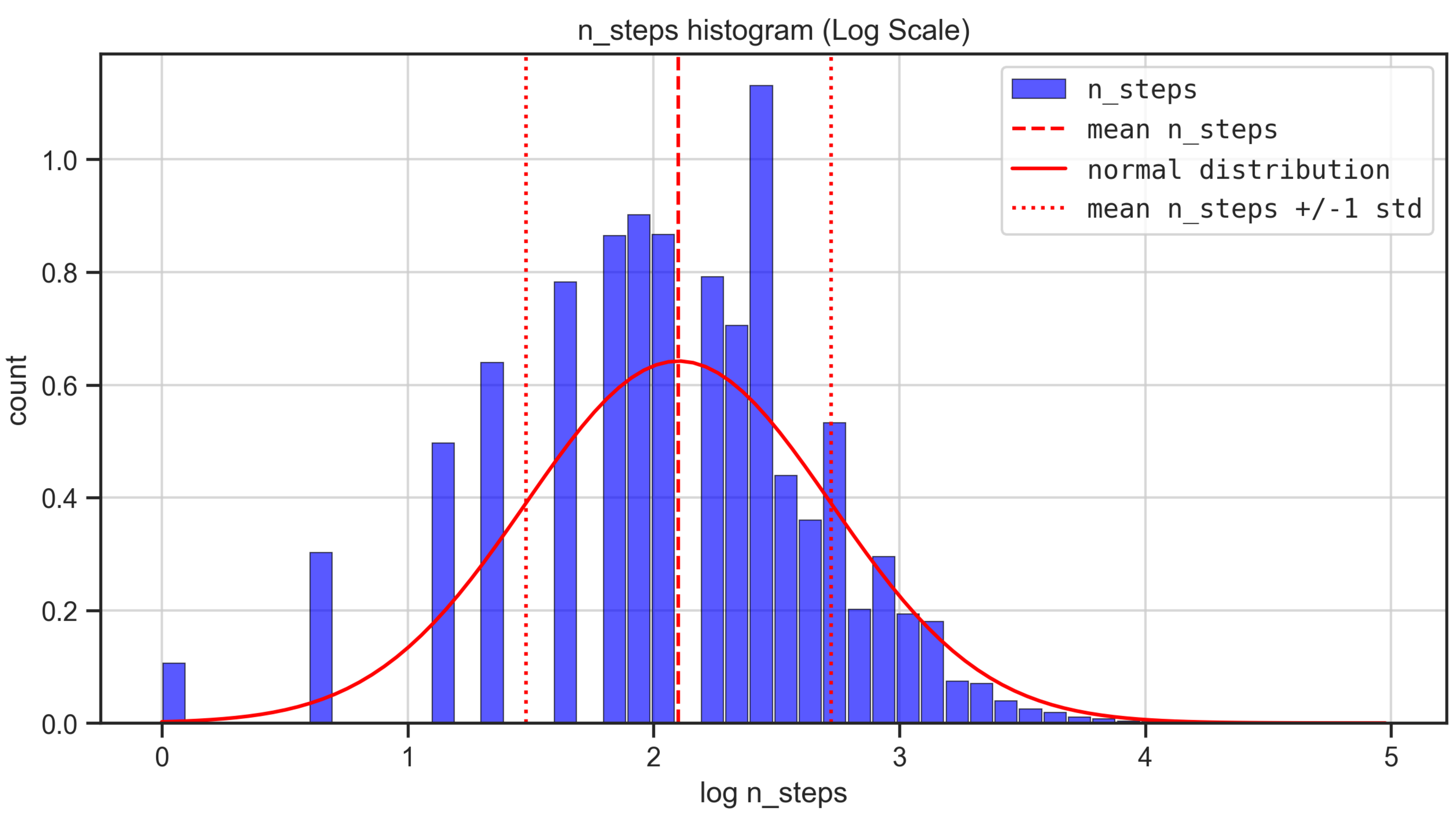
Number of steps

```
fig, ax = plt.subplots(1, 1, figsize=(10, 5), dpi=300)
log_steps_df = df[df['n_steps'] > 0].copy()
log_steps_df['n_steps'] = np.log(log_steps_df['n_steps'])
plt.rcParams['font.family'] = 'monospace'
plt.hist(log_steps_df['n_steps'], bins=50, alpha=0.65, color='blue', edgecolor='black', linewidth=0.5, label='n_steps', rwidth=0.9, density=True)
plt.axvline(log_steps_df['n_steps'].mean(), color='red', linestyle='--', label='mean n_steps')

x = np.linspace(log_steps_df['n_steps'].min(), log_steps_df['n_steps'].max(), 100)
y = stats.norm.pdf(x, log_steps_df['n_steps'].mean(), log_steps_df['n_steps'].std())

plt.plot(x, y, color='red', linestyle='-', label='normal distribution')
plt.axvline(log_steps_df['n_steps'].mean() + log_steps_df['n_steps'].std(), color='red', linestyle='dotted', label='mean n_steps +/-1 std')
plt.axvline(log_steps_df['n_steps'].mean() - log_steps_df['n_steps'].std(), color='red', linestyle='dotted')
plt.title('n_steps histogram (Log Scale)')
plt.xlabel('log n_steps')
plt.ylabel('count')
plt.grid(alpha=0.75)
plt.legend()
```

Out[ ]: <matplotlib.legend.Legend at 0x1b5fa2e1d50>



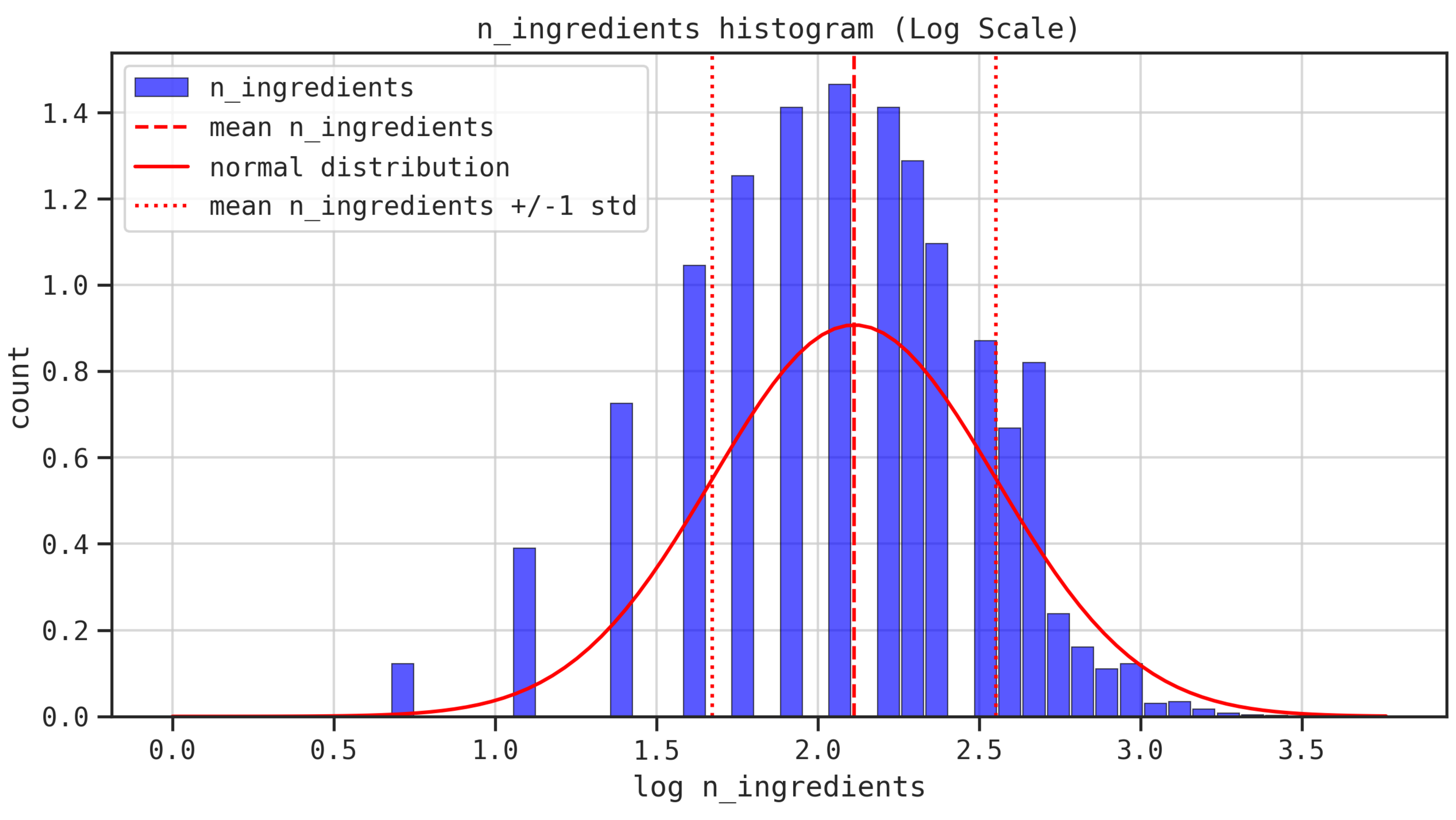
Number of ingredients

```
fig, ax = plt.subplots(1, 1, figsize=(10, 5), dpi=300)
log_steps_df = df[df['n_ingredients'] > 0].copy()
log_steps_df['n_ingredients'] = np.log(log_steps_df['n_ingredients'])
plt.rcParams['font.family'] = 'monospace'
plt.hist(log_steps_df['n_ingredients'], bins=50, alpha=0.65, color='blue', edgecolor='black', linewidth=0.5, label='n_ingredients', rwidth=0.9, density=True)
plt.axvline(log_steps_df['n_ingredients'].mean(), color='red', linestyle='--', label='mean n_ingredients')

x = np.linspace(log_steps_df['n_ingredients'].min(), log_steps_df['n_ingredients'].max(), 100)
y = stats.norm.pdf(x, log_steps_df['n_ingredients'].mean(), log_steps_df['n_ingredients'].std())

plt.plot(x, y, color='red', linestyle='-', label='normal distribution')
plt.axvline(log_steps_df['n_ingredients'].mean() + log_steps_df['n_ingredients'].std(), color='red', linestyle='dotted', label='mean n_ingredients +/-1 std')
plt.axvline(log_steps_df['n_ingredients'].mean() - log_steps_df['n_ingredients'].std(), color='red', linestyle='dotted')
plt.title('n_ingredients histogram (Log Scale)')
plt.xlabel('log n_ingredients')
plt.ylabel('count')
plt.grid(alpha=0.75)
plt.legend()
```

Out[ ]: <matplotlib.legend.Legend at 0x1b5fa400760>

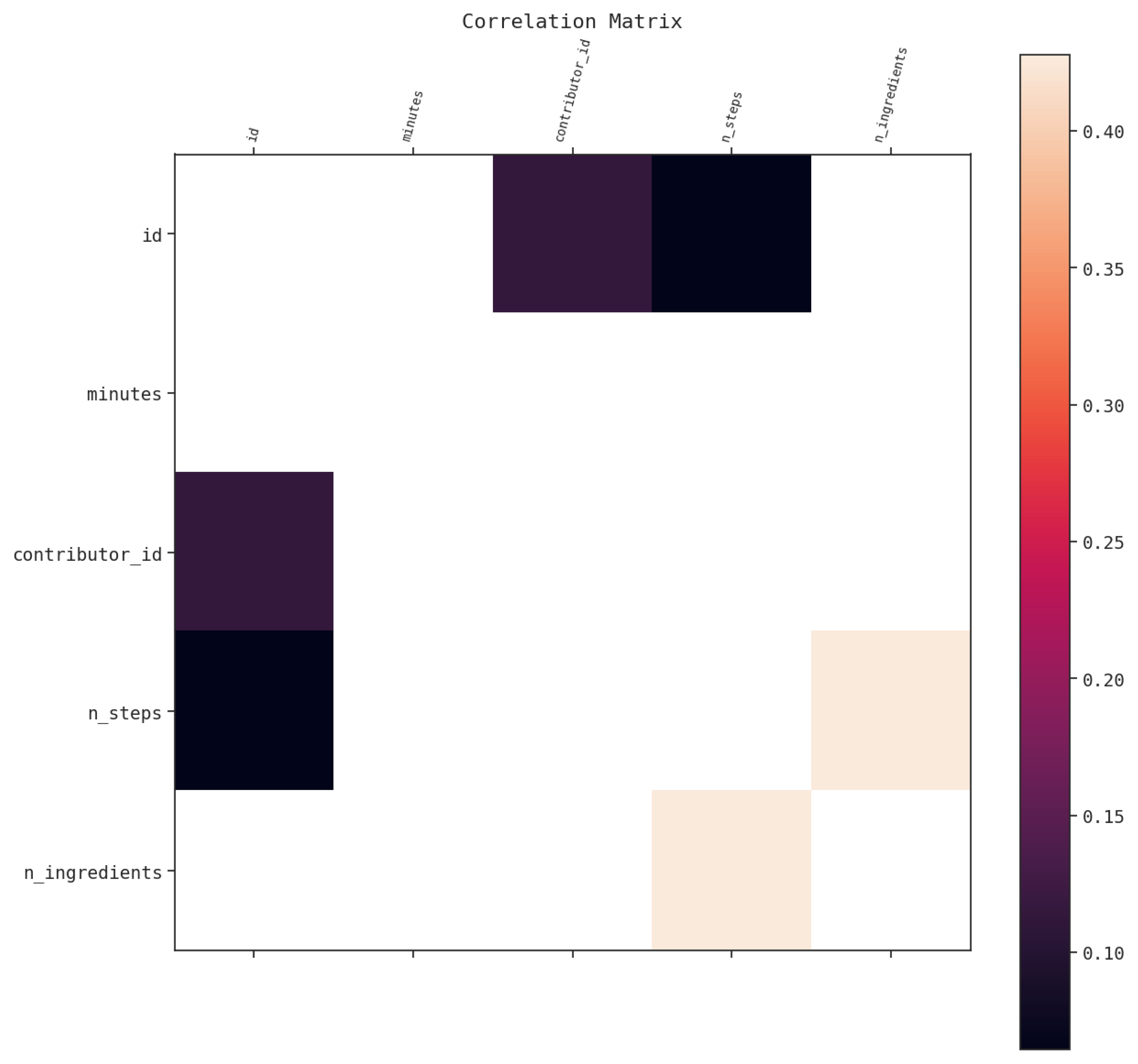


Corelation Matrix

```
f = plt.figure(figsize=(14, 14))
corr_df = df.corr()
corr_df = corr_df - np.diag(np.diag(corr_df))
corr_df = corr_df[corr_df >= 0.05]
plt.matshow(corr_df, fignum=f.number)
plt.xticks(range(df.select_dtypes(['number']).shape[1]), df.select_dtypes(['number']).columns, fontsize=10, rotation=75)
plt.yticks(range(df.select_dtypes(['number']).shape[1]), df.select_dtypes(['number']).columns, fontsize=14)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Correlation Matrix', fontsize=16)
```

```
C:\Users\grecu\AppData\Local\Temp\ipykernel_50752\716418815.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  corr_df = df.corr()
```

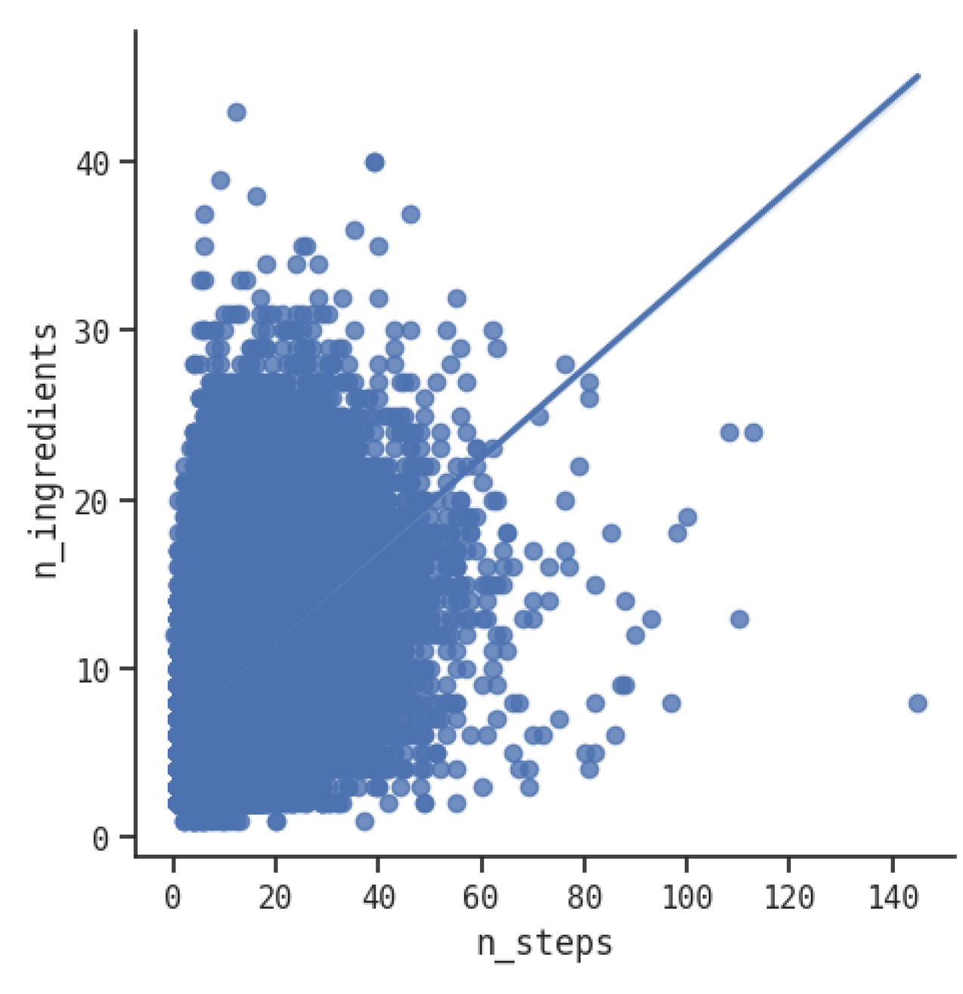
Out[ ]: Text(0.5, 1.0, 'Correlation Matrix')



Scatterplot with best-fit for the highest column correlation (n_steps and n_ingredients)

```
sns.lmplot(x="n_steps", y="n_ingredients", data=df)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x1b5fa557280>



Get rid of nans nad keep only id and description

```
df = df[['id', 'description']].rename(columns={'id': 'recipe_id'})
df['recipe_id'] = df['recipe_id'].astype(str)

df.dropna(subset=['description'], inplace=True)

print(df.head(10))
```

```
   recipe_id                                        description
0     137739  autumn is my favorite time of year to cook! th...
1      31490  this recipe calls for the crust to be prebaked...
2     112140  this modified version of 'mom's' chili was a h...
3      59389  this is a super easy, great tasting, make ahea...
4      44061  my dh's amish mother raised him on this recipe...
6      25274  my italian mil was thoroughly impressed by my ...
7      67888  this recipe is posted by request and was origi...
9      75452                              from ann hodgman's
10    109439  horseradish is one of my favorite condiments a...
11     42198  simple but sexy. this was in my local newspape...
```

Add data to weaviate

```
data = df.to_dict('records')

with client.batch(batch_size=3, dynamic=True) as batch:
    for i in range(0, len(data)):
        batch.add_data_object(data[i], 'Recipe')

    batch.create_objects()
```

```
C:\Users\grecu\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\weaviate\warnings.py:80: Depr
ecationWarning: Dep002: You are batching manually. This means you are NOT using the client's built-in
            multi-threading. Setting `batch_size` in `client.batch.configure()`  to an int value will enabled automatic
            batching. See:
            https://weaviate.io/developers/weaviate/current/restful-api-references/batch.html#example-request-1
  warnings.warn(
```

Create the results for every sentence

```
import json
concepts_moveAway = [{'concept':"Creative cooking ideas inspired by autumn harvest featuring pumpkins, root vegetables, and warm spices",'moveAway': {"concepts": ["summ
            {'concept':"Explore the creation of vibrant, rainbow-colored salads using fresh, organic vegetables sourced from your local farmers market",'moveAway': {"co
            {'concept':"Step-by-step guides to seafood recipes influenced by Greek cuisine, highlighting bright, natural flavors and the health benefits of the Mediterr
            {'concept':"Learn how to prepare heart-healthy holiday treats with zero added sugars and low-fat content, perfect for maintaining a balanced diet during the
            {'concept':"Discover nutrient-rich meal plans designed specifically for bodybuilders, focusing on lean turkey, chickpeas, and other high-protein ingredients
            {'concept':"Comforting and warming soup and stew recipes showcasing autumnal vegetables like squash, sweet potatoes, and fresh herbs",'moveAway': {"concepts
            {'concept':"Dive into the world of colorful, plant-based dishes, showcasing the full spectrum of summer produce from zucchini to bell peppers, and the nutri
            {'concept':"Enjoy the fresh and vibrant flavors of Italian seafood pasta dishes, incorporating fresh tomatoes, basil, and olive oil, key ingredients of a cl
            {'concept':"Learn how to reinvent traditional Christmas recipes with a low-fat, sugar-free twist, without sacrificing taste or festive cheer",'moveAway': {"
            {'concept':"Browse through a collection of high-protein, low-carb dinner ideas incorporating lean chicken, lentils, and a variety of vegetables, designed to

i = 0
for item in concepts_moveAway:
    nearText = {"concepts": [item['concept']],
                "moveAwayFrom": item['moveAway']}

    result = (
        client.query
            .get("Recipe", ["recipe_id", "description"])
            .with_near_text(nearText)
            .with_additional(['certainty'])
            .do()
    )

    result = json.dumps(result, indent=4)

    with open('part1/json_data_concept_'+ str(i) +'.txt', 'w') as outfile:
        outfile.write(result)

    i = i + 1
```

```
import json

for i in range(10):
    with open('part1/json_data_concept_'+ str(i) +'.txt') as f:
        json_data = json.load(f)
    certainty = 0
    for recipe in json_data['data']['Get']['Recipe']:
        certainty += recipe['_additional']['certainty']
    print(i, certainty / float(len(json_data['data']['Get']['Recipe'])))
```

```
0 0.9348696267604828
1 0.9146103939414024
2 0.9531265622377396
3 0.948051675260067
4 0.9533708280324936
5 0.9500026652216911
6 0.9357669696211814
7 0.9458862173557282
8 0.9486092355847359
9 0.9571997037529946
```

Results and Conclusions:

At the first part of analyzing the dataset, after ploting the histograms and the correlation matrix we can see that there is a strong dependecy between the n_steps and the n_ingredients.

For the second part, I opted for an implementation on the cpu, a local one, with a docker compose file, because the one that used the gpu, did not stop after 1 hour and 10 mins of running!, not to mention my concern for the laptop temperature and the speed of the fans.

The results where nearly on point with sentence no. 9 having the most accurate results overall, 0.957 ~ 0.96.

At first the certainty was below 0.9 and because of that i added concepts for the results to stay away from to tweak the model, increasing the overall certainty for all of the sentences proposed.

I can say that i am satisfied with the filtered recipes, i can see a resemblance between the sentences and the result, and also i will probably use this results to compare with the implementation of the recomender system.