Your Title Here

Name(s): (your name(s) here)

Website Link: (your website link)

In [ ]:
```
import pandas as pd
import numpy as np
from pathlib import Path

import plotly.express as px
import plotly.io as pio
pio.renderers.default = "notebook"
pd.options.plotting.backend = 'plotly'

# from dsc259r_utils import * # Feel free to uncomment and use this.
```

Step 1: Introduction

What features predict recipe popularity (number of ratings)?

Popularity is defined as the **number of non-zero ratings** a recipe receives in the interactions table. This focuses on engagement/visibility rather than the average star rating.

In [2]:
```
# TODO
```

Step 2: Data Cleaning and Exploratory Data Analysis

In [3]:
```
import pandas as pd
import numpy as np
import ast
import plotly.express as px
import plotly.io as pio

pio.renderers.default = "notebook"
pd.options.plotting.backend = "plotly"

recipes = pd.read_csv("RAW_recipes.csv")
interactions = pd.read_csv("RAW_interactions.csv")
```

```
print("recipes:", recipes.shape)
print("interactions:", interactions.shape)

recipes.head()
```

```
recipes: (83782, 12)
interactions: (731927, 5)
```

Out[3]:

	name	id	minutes	contributor_id	submitted	tags	nutrition
0	1 brownies in the world best ever	333281	40	985201	2008-10-27	['60-minutes-or-less', 'time-to-make', 'course...	[138.4, 10.0, 50.0, 3.0, 3.0, 19.0, 6.0]
1	1 in canada chocolate chip cookies	453467	45	1848091	2011-04-11	['60-minutes-or-less', 'time-to-make', 'cuisin...	[595.1, 46.0, 211.0, 22.0, 13.0, 51.0, 26.0]
2	412 broccoli casserole	306168	40	50969	2008-05-30	['60-minutes-or-less', 'time-to-make', 'course...	[194.8, 20.0, 6.0, 32.0, 22.0, 36.0, 3.0]
3	millionaire pound cake	286009	120	461724	2008-02-12	['time-to-make', 'course', 'cuisine', 'prepara...	[878.3, 63.0, 326.0, 13.0, 20.0, 123.0, 39.0]
4	2000 meatloaf	475785	90	2202916	2012-03-06	['time-to-make', 'course', 'main-ingredient', ...	[267.0, 30.0, 12.0, 12.0, 29.0, 48.0, 2.0]

In [4]:

```
# --- Parse / clean recipes ---
# Parse list-like columns stored as strings
for col in ["tags", "nutrition", "ingredients", "steps"]:
    if col in recipes.columns:
        recipes[col] = recipes[col].apply(ast.literal_eval)
```

```
# Convert dates
recipes["submitted"] = pd.to_datetime(recipes["submitted"], errors="co

# Expand nutrition vector into columns (7 values)
nutrition_cols = ["calories", "total_fat", "sugar", "sodium", "protein
nutrition_df = pd.DataFrame(recipes["nutrition"].to_list(), columns=nu

recipes_clean = pd.concat([recipes.drop(columns=["nutrition"]), nutrit

# Log cook time (very skewed)
recipes_clean["log_minutes"] = np.log1p(recipes_clean["minutes"])

recipes_clean[["id", "name", "minutes", "n_steps", "n_ingredients", "s
```

Out[4]:

	id	name	minutes	n_steps	n_ingredients	submitted	calories	to
0	333281	1 brownies in the world best ever	40	10	9	2008-10-27	138.4	
1	453467	1 in canada chocolate chip cookies	45	12	11	2011-04-11	595.1	
2	306168	412 broccoli casserole	40	6	9	2008-05-30	194.8	
3	286009	millionaire pound cake	120	7	7	2008-02-12	878.3	
4	475785	2000 meatloaf	90	17	13	2012-03-06	267.0	

In [5]:

```
# --- Parse / clean interactions ---
interactions["date"] = pd.to_datetime(interactions["date"], errors="co

# Remove placeholder/non-ratings (rating == 0)
interactions_clean = interactions[interactions["rating"] > 0].copy()

# Popularity = number of (non-zero) ratings per recipe
popularity = (
    interactions_clean
    .groupby("recipe_id")
    .size()
    .reset_index(name="num_ratings")
)
```

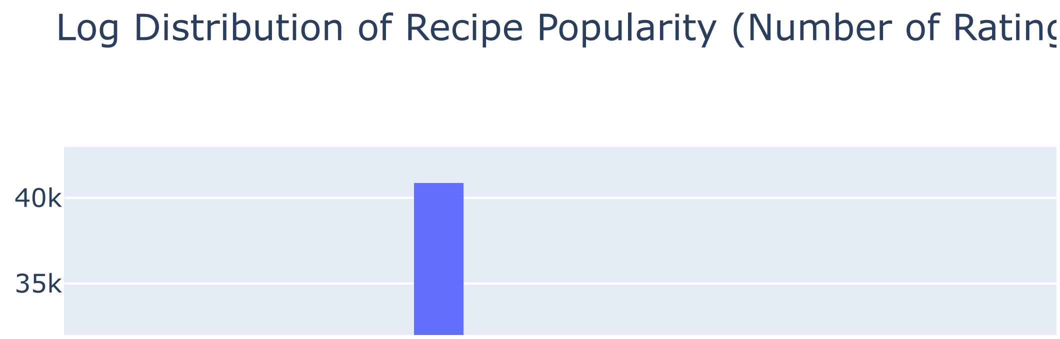
```
# Merge popularity back to recipe-level table
recipe_level = recipes_clean.merge(popularity, left_on="id", right_on=
recipe_level["num_ratings"] = recipe_level["num_ratings"].fillna(0).as

# Log popularity (very skewed)
recipe_level["log_num_ratings"] = np.log1p(recipe_level["num_ratings"]

recipe_level[["id", "name", "num_ratings", "log_num_ratings", "minutes
```

Out[5]:

	id	name	num_ratings	log_num_ratings	minutes	log_minutes	n_
0	333281	1 brownies in the world best ever	1	0.693147	40	3.713572	
1	453467	1 in canada chocolate chip cookies	1	0.693147	45	3.828641	
2	306168	412 broccoli casserole	4	1.609438	40	3.713572	
3	286009	millionaire pound cake	1	0.693147	120	4.795791	
4	475785	2000 meatloaf	2	1.098612	90	4.510860	

In [6]:

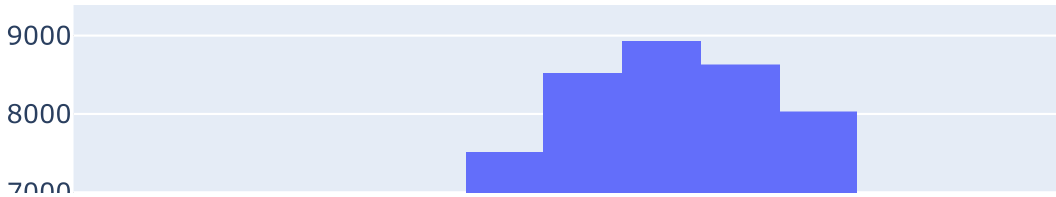
```
# Univariate: popularity distribution (log scale)
fig = px.histogram(
    recipe_level,
    x="log_num_ratings",
    nbins=60,
    title="Log Distribution of Recipe Popularity (Number of Ratings)"
)
fig.show()
```



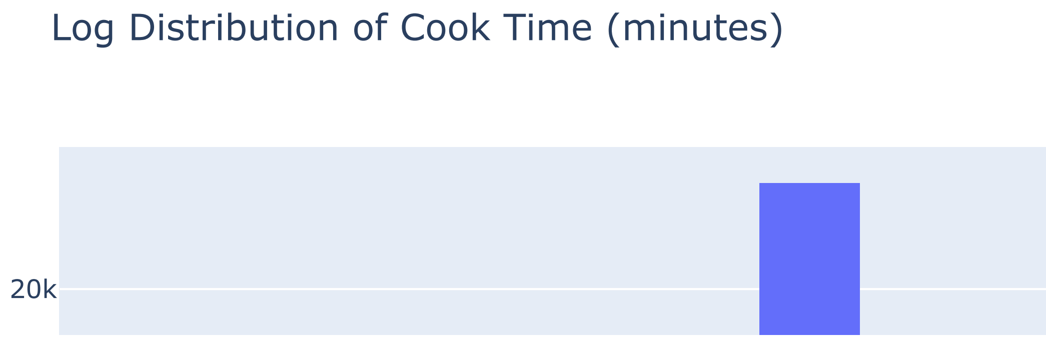
In [7]:

```
# Univariate: ingredient count distribution
fig = px.histogram(
    recipe_level,
    x="n_ingredients",
    nbins=50,
    title="Distribution of Number of Ingredients"
)
fig.show()
```

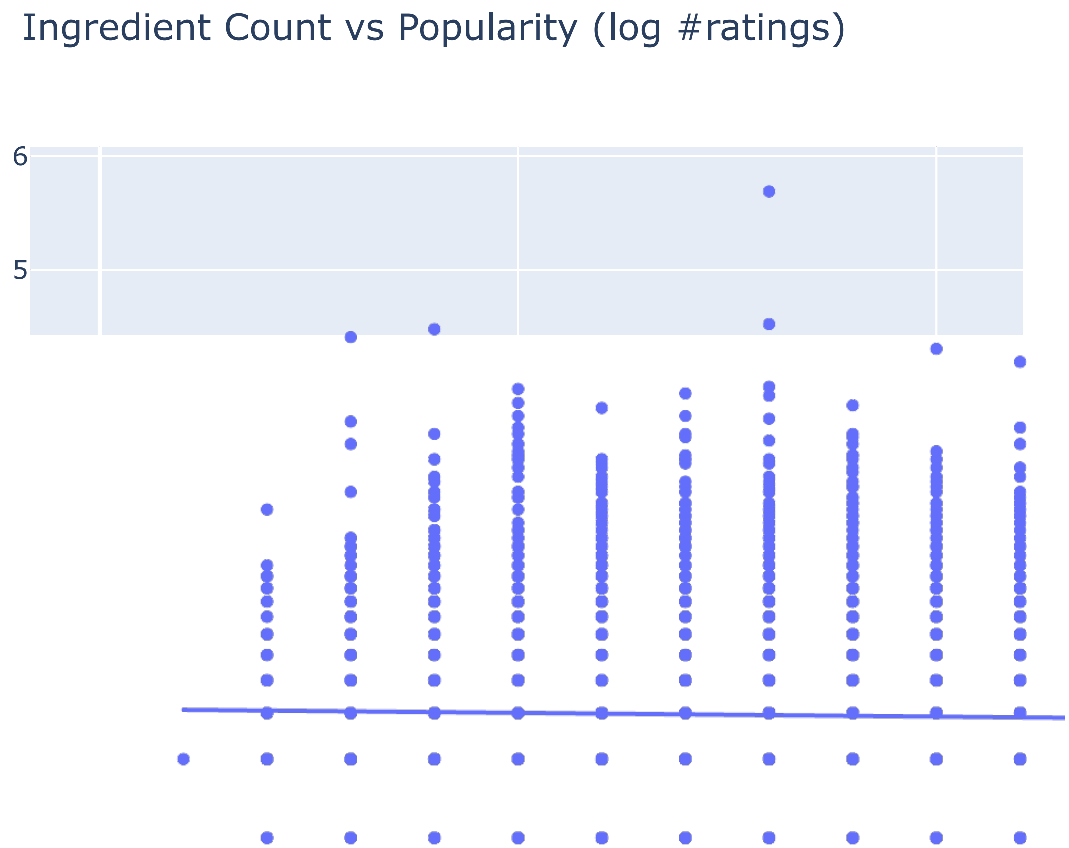
Distribution of Number of Ingredients



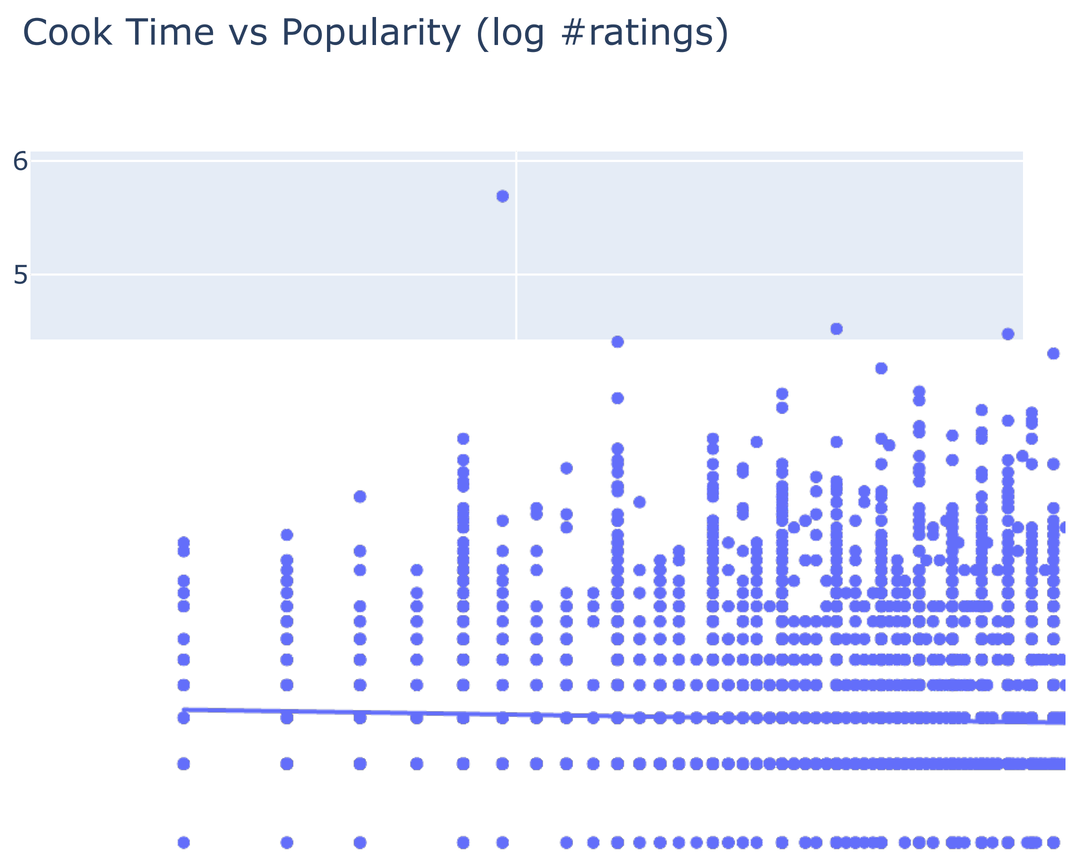
In [8]:
```
# Univariate: cook time distribution (log scale)
fig = px.histogram(
    recipe_level,
    x="log_minutes",
    nbins=60,
    title="Log Distribution of Cook Time (minutes)"
)
fig.show()
```



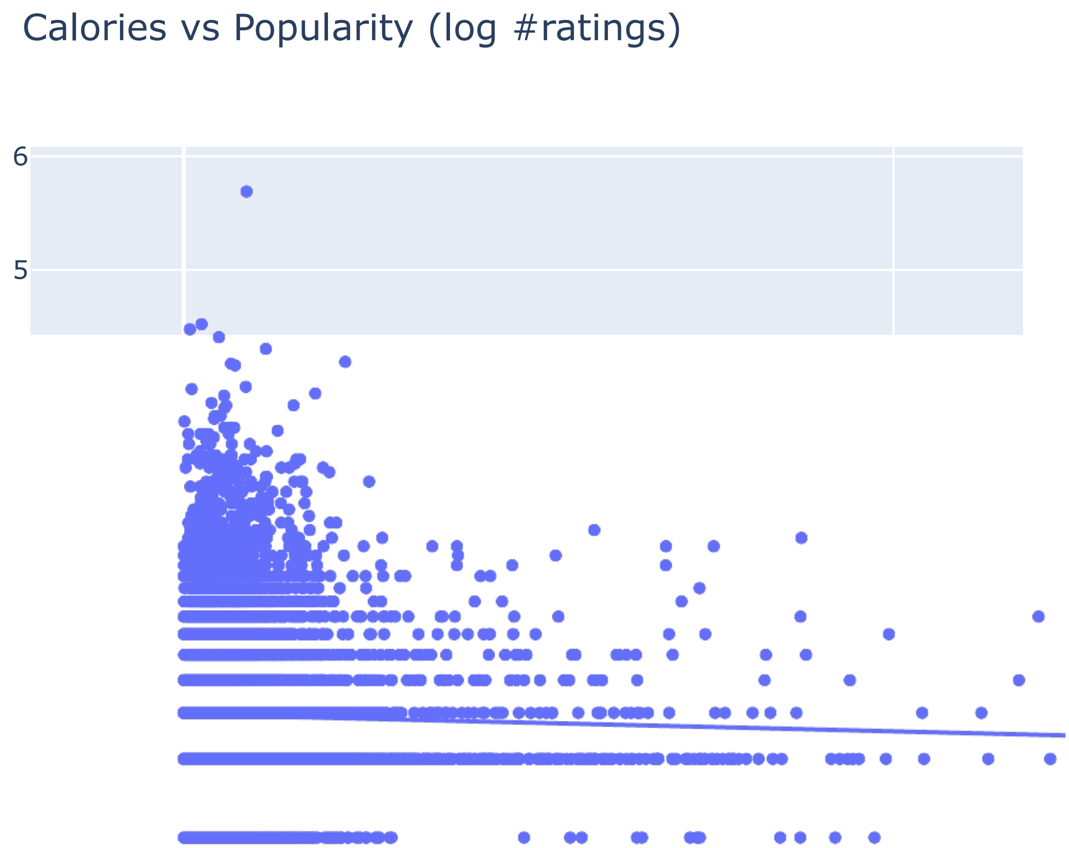
In [9]:
```
# Bivariate: ingredients vs popularity
sample = recipe_level.sample(15000, random_state=42)

fig = px.scatter(
    sample,
    x="n_ingredients",
    y="log_num_ratings",
    trendline="ols",
    title="Ingredient Count vs Popularity (log #ratings)"
)
fig.show()
```




```
In [10]: # Bivariate: cook time vs popularity
fig = px.scatter(
    sample,
    x="log_minutes",
    y="log_num_ratings",
    trendline="ols",
    title="Cook Time vs Popularity (log #ratings)"
)
fig.show()
```

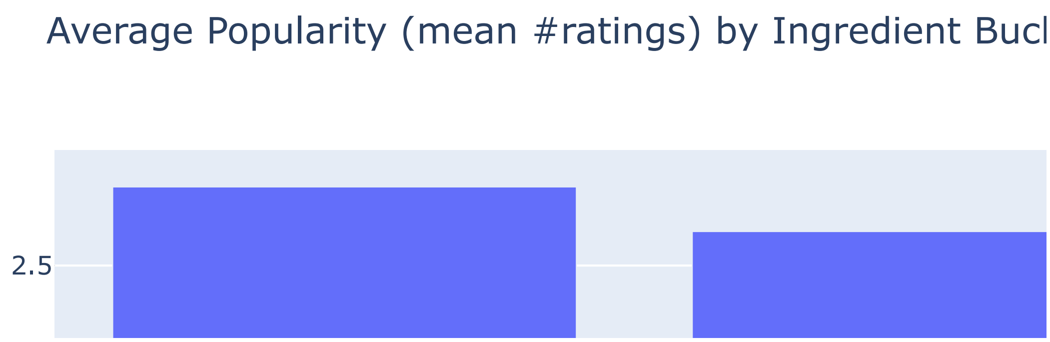


In [11]:
```
# Bivariate: calories vs popularity
fig = px.scatter(
    sample,
    x="calories",
    y="log_num_ratings",
    trendline="ols",
    title="Calories vs Popularity (log #ratings)"
)
fig.show()
```




```
In [12]: # Interesting aggregates: popularity by ingredient bucket
recipe_level["ingredient_bucket"] = pd.cut(
    recipe_level["n_ingredients"],
    bins=[0, 5, 10, 15, 20, 50, np.inf],
    labels=["1-5", "6-10", "11-15", "16-20", "21-50", "50+"],
    include_lowest=True
)

agg_ing = (
    recipe_level
    .groupby("ingredient_bucket", observed=True)["num_ratings"]
    .agg(["mean", "median", "count"])
    .reset_index()
)

fig = px.bar(
    agg_ing,
    x="ingredient_bucket",
    y="mean",
    title="Average Popularity (mean #ratings) by Ingredient Bucket",
```

```
    hover_data=["median", "count"]
)
fig.show()

agg_ing
```

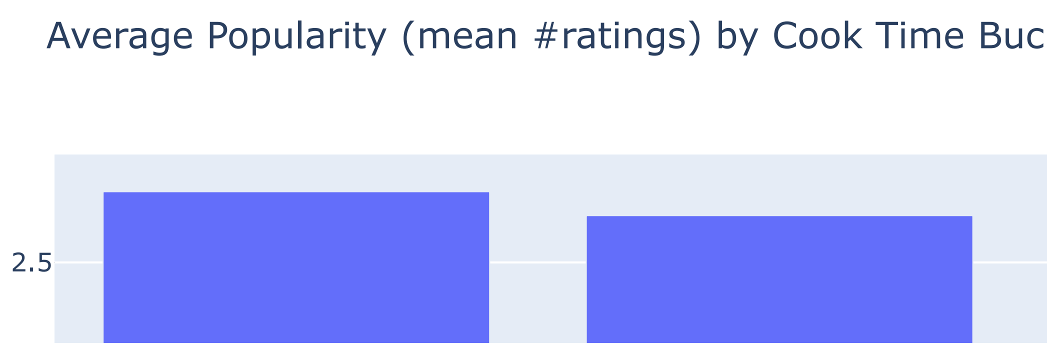


Out[12]:

	ingredient_bucket	mean	median	count
0	1-5	2.816718	2.0	14164
1	6-10	2.637028	1.0	41623
2	11-15	2.496317	1.0	22810
3	16-20	2.431808	1.0	4502
4	21-50	2.704246	1.0	683

In [13]:
```
# Interesting aggregates: popularity by cook time bucket
recipe_level["time_bucket"] = pd.cut(
    recipe_level["minutes"],
    bins=[0, 15, 30, 60, 120, 240, np.inf],
```

```
    labels=["<=15", "15-30", "30-60", "60-120", "120-240", "240+"],
    include_lowest=True
)

agg_time = (
    recipe_level
    .groupby("time_bucket", observed=True)["num_ratings"]
    .agg(["mean", "median", "count"])
    .reset_index()
)

fig = px.bar(
    agg_time,
    x="time_bucket",
    y="mean",
    title="Average Popularity (mean #ratings) by Cook Time Bucket",
    hover_data=["median", "count"]
)
fig.show()

agg_time
```



Out[13]:

	time_bucket	mean	median	count
0	<=15	2.781355	2.0	16680
1	15-30	2.686264	1.0	20632
2	30-60	2.586245	1.0	25416
3	60-120	2.413774	1.0	12328
4	120-240	2.357459	1.0	4297
5	240+	2.699932	1.0	4429

Step 3: Assessment of Missingness

In [14]:
```
# TODO
```

Step 4: Hypothesis Testing

In [15]:
```
# TODO
```

Step 5: Framing a Prediction Problem

In [16]:
```
# TODO
```

Step 6: Baseline Model

In [17]:
```
# TODO
```

Step 7: Final Model

In [18]:
```
# TODO
```

Step 8: Fairness Analysis

In [19]:
```
# TODO
```