final-notebook

June 11, 2024

1 Predicting Recipe Rating based on Recipe Length

Name(s): Jahnavi Naik

Website Link: https://jahnavi-naik.github.io/recipe-analysis/

[ ]:

```
import pandas as pd
import numpy as np
from pathlib import Path

import plotly.express as px
pd.options.plotting.backend = 'plotly'
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score


# from dsc80_utils import * # Feel free to uncomment and use this.
```

1.1 Step 1: Introduction

[ ]:

```
#merging the datasets
recipes_fp = 'food_data/RAW_recipes.csv'
review_fp = 'food_data/RAW_interactions.csv'
recipes = pd.read_csv(recipes_fp)
reviews =  pd.read_csv(review_fp)

temp_df = recipes.merge(reviews, how='left', left_on='id', right_on='recipe_id')
temp_df['rating'].replace(0.0, np.nan, inplace=True)
#SAY WHY YOU DID IN WEBSITE

#average rating per recipe
```

```
avg_rating = temp_df.groupby('id').mean()[['rating']].reset_index()
avg_rating

#final dataframe
df = temp_df.merge(avg_rating, how = 'left', on = 'id')
df = df.rename(columns = {'rating_y': 'avg_rating', 'rating_x': 'rating'})
```

[ ]:
```
#COLUMN DESCRIPTIONS
recipes_desc = {
        'id' : 'The recipe ID, which is unique per recipe',
        'minutes' : 'The number of minutes it takes to complete a recipe',
        'nutrition' : 'a string (that looks like a list) of various nutrition␣
 ↪facts including calories (#), total fat (PDV), sugar (PDV), sodium (PDV),␣
 ↪protein (PDV), saturated fat (PDV), carbohydrates (PDV)',
        'n_steps' : 'the number of steps in the recipe',
        'ingredients' : 'a string (that looks like a list) of ingredients used␣
 ↪in the recipe',
        'n_ingredients' : 'the number of ingredients used in the recipe'
}
reviews_desc = {
    'recioe_id': 'the recipe id, matching the id column in the recipes␣
 ↪dataframe',
    'rating' : 'the rating given by the reviewer on a 1 - 5 scale'
}

recipes_desc_df = pd.DataFrame(list(recipes_desc.items()), columns=['Column',␣
 ↪'Description'])
reviews_desc_df = pd.DataFrame(list(reviews_desc.items()), columns=['Column',␣
 ↪'Description'])
reviews_desc_df
```

[ ]:
```
      Column                                        Description
0  recioe_id  the recipe id, matching the id column in the r…
1     rating  the rating given by the reviewer on a 1 - 5 scale
```

[ ]:
```
recipes_html_table = recipes_desc_df.to_html(index=False)
reviews_html_table = reviews_desc_df.to_html(index=False)
with open('recipes_descriptions.html', 'w') as f:
    f.write(recipes_html_table)
with open('reviews_descriptions.html', 'w') as f:
    f.write(reviews_html_table)
```

[ ]:
```
# Question: how does the length of the recipe/ ingredients affect the ratings␣
 ↪of the recipe?
```

1.2 Step 2: Data Cleaning and Exploratory Data Analysis

[ ]:

```
#split up the nutrition column to have each value in the list have its own␣
 ↪column and make then floats so easier to work with and pull numbers for␣
 ↪analysis
new_df = df.copy()

new_df['nutrition'] = new_df['nutrition'].astype(str).str.strip('[]')
new_df['nutrition'] = new_df['nutrition'].str.split(', ').apply(lambda x:␣
 ↪[float(i) for i in x])

def split_list(row):
    return pd.Series(row['nutrition'])

nutri = new_df[['nutrition']]
nutri = new_df.apply(split_list, axis=1)
nutri.columns = ['calories (#)', 'total fat (PDV)', 'sugar (PDV)', 'sodium␣
 ↪(PDV)',' protein (PDV)', 'saturated fat (PDV)', 'carbohydrates (PDV)']
new_df = pd.concat([new_df, nutri], axis=1).drop(columns = ['nutrition',␣
 ↪'recipe_id', 'review'])
new_df['has_sugar'] = new_df['ingredients'].apply(lambda ingredients: 'sugar'␣
 ↪in ingredients)
```

[ ]:

```
new_df
```

[ ]:

```
                                                        name      id  minutes  \
0                1 brownies in the world    best ever  333281       40
1                  1 in canada chocolate chip cookies  453467       45
2                               412 broccoli casserole  306168       40
3                               412 broccoli casserole  306168       40
4                               412 broccoli casserole  306168       40
…                                                      …       …        …
234424                          zydeco ya ya deviled eggs  308080       40
234425            cookies by design   cookies on a stick  298512       29
234426  cookies by design   sugar shortbread cookies  298509       20
234427  cookies by design   sugar shortbread cookies  298509       20
234428  cookies by design   sugar shortbread cookies  298509       20

        contributor_id   submitted  \
0               985201  2008-10-27
1              1848091  2011-04-11
2                50969  2008-05-30
3                50969  2008-05-30
4                50969  2008-05-30
…                    …           …
234424           37779  2008-06-07
234425          506822  2008-04-15
```

```
234426          506822  2008-04-15
234427          506822  2008-04-15
234428          506822  2008-04-15

                                                     tags  n_steps  \
0       ['60-minutes-or-less', 'time-to-make', 'course…       10
1       ['60-minutes-or-less', 'time-to-make', 'cuisin…       12
2       ['60-minutes-or-less', 'time-to-make', 'course…        6
3       ['60-minutes-or-less', 'time-to-make', 'course…        6
4       ['60-minutes-or-less', 'time-to-make', 'course…        6
…                                                   …      …
234424  ['60-minutes-or-less', 'time-to-make', 'course…        7
234425  ['30-minutes-or-less', 'time-to-make', 'course…        9
234426  ['30-minutes-or-less', 'time-to-make', 'course…        5
234427  ['30-minutes-or-less', 'time-to-make', 'course…        5
234428  ['30-minutes-or-less', 'time-to-make', 'course…        5

                                                    steps  \
0       ['heat the oven to 350f and arrange the rack i…
1       ['pre-heat oven the 350 degrees f', 'in a mixi…
2       ['preheat oven to 350 degrees', 'spray a 2 qua…
3       ['preheat oven to 350 degrees', 'spray a 2 qua…
4       ['preheat oven to 350 degrees', 'spray a 2 qua…
…                                                   …
234424  ['in a bowl , combine the mashed yolks and may…
234425  ['place melted butter in a large mixing bowl a…
234426  ['whip sugar and shortening in a large bowl , …
234427  ['whip sugar and shortening in a large bowl , …
234428  ['whip sugar and shortening in a large bowl , …

                                              description  \
0       these are the most; chocolatey, moist, rich, d…
1       this is the recipe that we use at my school ca…
2       since there are already 411 recipes for brocco…
3       since there are already 411 recipes for brocco…
4       since there are already 411 recipes for brocco…
…                                                   …
234424                          deviled eggs, cajun-style
234425  i've heard of the 'cookies by design' company,…
234426  i've heard of the 'cookies by design' company,…
234427  i've heard of the 'cookies by design' company,…
234428  i've heard of the 'cookies by design' company,…

                                              ingredients  …  rating  \
0       ['bittersweet chocolate', 'unsalted butter', '…  …     4.0
1       ['white sugar', 'brown sugar', 'salt', 'margar…  …     5.0
2       ['frozen broccoli cuts', 'cream of chicken sou…  …     5.0
```

```
3       ['frozen broccoli cuts', 'cream of chicken sou…  …     5.0
4       ['frozen broccoli cuts', 'cream of chicken sou…  …     5.0
…                                                   …  …     …
234424  ['hard-cooked eggs', 'mayonnaise', 'dijon must…  …     5.0
234425  ['butter', 'eagle brand condensed milk', 'ligh…  …     1.0
234426  ['granulated sugar', 'shortening', 'eggs', 'fl…  …     1.0
234427  ['granulated sugar', 'shortening', 'eggs', 'fl…  …     5.0
234428  ['granulated sugar', 'shortening', 'eggs', 'fl…  …     NaN

        avg_rating calories (#)  total fat (PDV)  sugar (PDV)  sodium (PDV)  \
0              4.0        138.4             10.0         50.0           3.0
1              5.0        595.1             46.0        211.0          22.0
2              5.0        194.8             20.0          6.0          32.0
3              5.0        194.8             20.0          6.0          32.0
4              5.0        194.8             20.0          6.0          32.0
…            …          …              …          …           …
234424         5.0         59.2              6.0          2.0           3.0
234425         1.0        188.0             11.0         57.0          11.0
234426         3.0        174.9             14.0         33.0           4.0
234427         3.0        174.9             14.0         33.0           4.0
234428         3.0        174.9             14.0         33.0           4.0

         protein (PDV)  saturated fat (PDV)  carbohydrates (PDV)  has_sugar
0                  3.0                 19.0                  6.0       True
1                 13.0                 51.0                 26.0       True
2                 22.0                 36.0                  3.0      False
3                 22.0                 36.0                  3.0      False
4                 22.0                 36.0                  3.0      False
…                …                  …                  …        …
234424             6.0                  5.0                  0.0      False
234425             7.0                 21.0                  9.0       True
234426             4.0                 11.0                  6.0       True
234427             4.0                 11.0                  6.0       True
234428             4.0                 11.0                  6.0       True

[234429 rows x 23 columns]
```

[ ]:

```python
df_html = new_df[['name', 'id', 'minutes', 'n_steps', 'n_ingredients',␣
 ↪'calories (#)', 'has_sugar', 'avg_rating']].head().to_html(index=False)
with open('df_head.html', 'w') as f:
    f.write(df_html)
```

[ ]:

```python
# DISTRIBUTION OF NUMBER OF STEPS
num_steps = new_df.copy()
num_steps = new_df.drop_duplicates(subset='id')
fig = px.histogram(num_steps, x='n_steps', nbins=30,
                   title='Distribution of Number of Steps in Recipe',
                   labels={'n_steps': 'Number of Steps', 'count': 'Frequency'})
# Show the plot
fig.show()
```

[ ]:

```
fig.write_html('univariate-n_steps.html', include_plotlyjs='cdn')
```

[ ]:

```
num_steps['n_steps'].describe()
Q1 = new_df['n_steps'].quantile(0.25)
Q3 = new_df['n_steps'].quantile(0.75)

# Step 2: Calculate IQR
IQR = Q3 - Q1

# Step 3: Determine outlier boundaries
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
upper_bound
```

[ ]:

```
23.5
```

[ ]:

```
#MINUTES DISTRIBUTION

minutes = new_df.copy()
minutes = minutes[minutes['minutes'] < 120]



#The distribution of minutes is skewed to the right with a very long tail.␣
 ↪Because of this, I decided to only show values where the time taken was less␣
 ↪than 120 minutes, or 2 hours,
#as I ruled out the other values that are considered outliers using the IQR␣
 ↪test. This allows us to take a better look at the distribution of the␣
 ↪minutes for most of the data
Q1 = new_df['minutes'].quantile(0.25)
Q3 = new_df['minutes'].quantile(0.75)

# Step 2: Calculate IQR
IQR = Q3 - Q1

# Step 3: Determine outlier boundaries
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

#Step 4: Filter out the outliers
filtered_minutes = new_df[(new_df['minutes'] >= lower_bound) &␣
 ↪(new_df['minutes'] <= upper_bound)]
```

```
fig = px.histogram(filtered_minutes, x='minutes', nbins=30,
                   title='Distribution of Time Taken for Recipe (minutes)',
                   labels={'log_calories': 'Minutes', 'count': 'Frequency'})
# Show the plot
fig.show()
```

[ ]:

```
fig.write_html('univariate-minutes.html', include_plotlyjs='cdn')
```

[ ]:

```
#NUMBER OF STEPS VS NUMBER OF INGREDIENTS
step_ingredients = new_df.groupby('id').mean()#[['n_steps', 'n_ingredients']]
fig = px.scatter(step_ingredients, x='avg_rating', y='n_steps',
                 title='Average Rating vs. Number of Steps',
                 labels={'avg_rating': 'Average Rating', 'n_steps': 'Number of␣
 ↪Steps'})

# Show the plot
fig.show()
```

[ ]:

```
fig.write_html('n_steps-rating.html', include_plotlyjs='cdn')
```

[ ]:

```
calories_ingredients = new_df.groupby('id').mean()#[['n_steps',␣
 ↪'n_ingredients']]
fig = px.scatter(step_ingredients, x='n_ingredients', y='n_steps',
                 title='Number of Calories vs. Number of Steps',
                 labels={'avg_rating': 'Average Rating', 'n_steps': 'Number of␣
 ↪Steps'})

# Show the plot
fig.show()
```

[ ]:

```
Q1 = new_df['calories (#)'].quantile(0.25)
Q3 = new_df['calories (#)'].quantile(0.75)

# Step 2: Calculate IQR
IQR = Q3 - Q1

# Step 3: Determine outlier boundaries
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

#Step 4: Filter out the outliers
filtered_calories = new_df[(new_df['calories (#)'] >= lower_bound) &␣
 ↪(new_df['calories (#)'] <= upper_bound)]

pivot_table = filtered_calories[['calories (#)', 'n_ingredients']].
 ↪pivot_table(index = 'n_ingredients', aggfunc = ['mean', 'median', 'std'])
pivot_table
```

```
[ ]:                    mean          median             std
              calories (#) calories (#) calories (#)
     n_ingredients
     1                157.229630        144.20   113.179463
     2                212.604051        144.70   205.470241
     3                215.911121        163.60   191.614204
     4                237.279823        183.60   193.219294
     5                262.716944        219.00   191.958962
     6                285.390661        239.90   207.755204
     7                302.350313        252.20   205.388476
     8                318.771192        275.20   204.539190
     9                336.953357        294.90   207.894473
     10               341.251078        304.10   201.831227
     11               363.584400        324.80   207.621016
     12               364.438348        326.40   202.957711
     13               384.200156        359.50   205.592255
     14               402.556815        374.30   211.273798
     15               422.722760        392.50   215.983254
     16               433.215455        416.70   210.721340
     17               458.555769        419.70   216.732349
     18               469.239942        440.20   215.431545
     19               469.590974        439.10   224.048137
     20               498.537207        452.70   237.817307
     21               452.178998        416.80   218.114314
     22               572.306952        620.50   242.775594
     23               476.350549        471.10   211.118148
     24               505.260976        454.50   214.630693
     25               540.551163        606.40   271.928627
     26               544.987273        572.70   178.086653
     27               583.072222        589.85   214.616784
     28               559.124324        491.70   257.784868
     29               442.145000        336.20   169.228984
     30               580.293103        594.30   165.961585
     31               348.242857        219.60   172.555632
     32               363.100000        363.10          NaN
     33               338.200000        338.20          NaN
```

```
[ ]: html_pivot = pivot_table.to_html()
     with open('pivot_table.html', 'w') as f:
         f.write(html_pivot)
```

1.3 Step 3: Assessment of Missingness

```
[ ]: # TODO
     nan_counts = df.isna().sum()
     nan_counts
```

```
[ ]: name                 1
     id                   0
     minutes              0
     contributor_id       0
     submitted            0
     tags                 0
     nutrition            0
     n_steps              0
     steps                0
     description        114
     ingredients          0
     n_ingredients        0
     user_id              1
     recipe_id            1
     date                 1
     rating           15036
     review              58
     avg_rating        2777
     dtype: int64
```

```
[ ]: #PERMUTATION TEST TO ASSESS MISSINGNESS OF REVIEW
     #RATING VS. MINUTES

     #null hypothesis: missingness of rating does not depend on the number of minutes
     #alternative hypothesis: the missingness of the rating depends on the number of␣
      ↪minutes

     def stat(avg_rating, minutes):
         is_missing = avg_rating.isna()
         mean_missing = minutes[is_missing].mean()
         mean_notmissing = minutes[~is_missing].mean()

         return np.abs(mean_notmissing - mean_missing)

     #make a copy of the og dataframe, drop the duplicates of the recipe because the␣
      ↪description and number of steps doesnt change within a recipe
     shuffled = new_df.copy().drop_duplicates(subset=['id'])

     observed = stat(shuffled['avg_rating'], shuffled['minutes'])

     stats = []

     for i in range(1000):
         shuffled['shuffled rating'] = np.random.permutation(shuffled['avg_rating'])
         curr = stat(shuffled['shuffled rating'], shuffled['minutes'])
         stats = np.append(stats, curr)
```

```
p_value = np.mean(np.array(stats) >= observed)
p_value

# we get a p-value of 0.035 and there fore reject the null hypothesis at the 0.
 ↪01 significance level
```

[ ]: 0.039

[ ]:
```
fig = px.histogram(
    pd.DataFrame(stats), x=0, nbins=10, histnorm='probability',
    title='Empirical Distribution of Mean Differences in Minutes (Missing vs.␣
 ↪Not Missing Ratings) ')
fig.add_vline(x=observed, line_color='red')
fig.update_xaxes(title_text='Mean Differences in Minutes')
```

[ ]:
```
fig.write_html('missing_test.html', include_plotlyjs='cdn')
```

[ ]:
```
#null hypothesis: missingness of rating does not depend on the number of␣
 ↪calories in the recipe
#alternative hypothesis: the missingness of the rating depends on the number of␣
 ↪calories in the recipe

def stat(avg_rating, calories):
    is_missing = avg_rating.isna()
    mean_missing = calories[is_missing].mean()
    mean_notmissing = calories[~is_missing].mean()

    return np.abs(mean_notmissing - mean_missing)

#make a copy of the og dataframe, drop the duplicates of the recipe because the␣
 ↪description and number of steps doesnt change within a recipe
shuffled = new_df.copy().drop_duplicates(subset=['id'])

observed = stat(shuffled['avg_rating'], shuffled['calories (#)'])

stats = []

for i in range(1000):
    shuffled['shuffled rating'] = np.random.permutation(shuffled['avg_rating'])
    curr = stat(shuffled['shuffled rating'], shuffled['calories (#)'])
    stats = np.append(stats, curr)

p_value = np.mean(np.array(stats) >= observed)
p_value

#we get a p-value of 0.0, and therefore reject the null hypothesis at a 0.01␣
 ↪significant level
```

[ ]:

```
0.0
```

1.4 Step 4: Hypothesis Testing

[ ]:

```
# TODO
#null hyp: there is no relationship between time it takes for a recipe and the␣
 ↪average rating of a recipe
#alt hype: recipes that take over 37 minutes have a lower average rating than␣
 ↪ones that take 37 or less minutes

# #FIX UP DATAFRAME
# #remove outliers
Q1 = new_df['minutes'].quantile(0.25)
Q3 = new_df['minutes'].quantile(0.75)

# Step 2: Calculate IQR
IQR = Q3 - Q1

# Step 3: Determine outlier boundaries
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

#Step 4: Filter out the outliers
hyp_test_df = new_df[(new_df['minutes'] >= lower_bound) & (new_df['minutes'] <=␣
 ↪upper_bound)]

# #keep only necessary columns and remove duplicates because we only need one␣
 ↪for each recipe, as the minutes and average rating wont change
# hyp_test_df = new_df[['id','minutes', 'avg_rating']].drop_duplicates()
hyp_test_df = new_df.copy()
```

[ ]:

```
hyp_test_df['under 37'] = hyp_test_df['minutes'] <=37
hyp_test_df
```

[ ]:

```
                                               name      id  minutes  \
0                1 brownies in the world    best ever  333281       40
1                  1 in canada chocolate chip cookies  453467       45
2                              412 broccoli casserole  306168       40
3                              412 broccoli casserole  306168       40
4                              412 broccoli casserole  306168       40
…                                                 …       …        …
234424                         zydeco ya ya deviled eggs  308080       40
234425          cookies by design   cookies on a stick  298512       29
234426  cookies by design   sugar shortbread cookies  298509       20
234427  cookies by design   sugar shortbread cookies  298509       20
234428  cookies by design   sugar shortbread cookies  298509       20
```

```
        contributor_id  submitted  \
0               985201 2008-10-27
1              1848091 2011-04-11
2                50969 2008-05-30
3                50969 2008-05-30
4                50969 2008-05-30
…                    …          …
234424           37779 2008-06-07
234425          506822 2008-04-15
234426          506822 2008-04-15
234427          506822 2008-04-15
234428          506822 2008-04-15

                                                     tags  n_steps  \
0       ['60-minutes-or-less', 'time-to-make', 'course…       10
1       ['60-minutes-or-less', 'time-to-make', 'cuisin…       12
2       ['60-minutes-or-less', 'time-to-make', 'course…        6
3       ['60-minutes-or-less', 'time-to-make', 'course…        6
4       ['60-minutes-or-less', 'time-to-make', 'course…        6
…                                                   …      …
234424  ['60-minutes-or-less', 'time-to-make', 'course…        7
234425  ['30-minutes-or-less', 'time-to-make', 'course…        9
234426  ['30-minutes-or-less', 'time-to-make', 'course…        5
234427  ['30-minutes-or-less', 'time-to-make', 'course…        5
234428  ['30-minutes-or-less', 'time-to-make', 'course…        5

                                                    steps  \
0       ['heat the oven to 350f and arrange the rack i…
1       ['pre-heat oven the 350 degrees f', 'in a mixi…
2       ['preheat oven to 350 degrees', 'spray a 2 qua…
3       ['preheat oven to 350 degrees', 'spray a 2 qua…
4       ['preheat oven to 350 degrees', 'spray a 2 qua…
…                                                   …
234424  ['in a bowl , combine the mashed yolks and may…
234425  ['place melted butter in a large mixing bowl a…
234426  ['whip sugar and shortening in a large bowl , …
234427  ['whip sugar and shortening in a large bowl , …
234428  ['whip sugar and shortening in a large bowl , …

                                              description  \
0       these are the most; chocolatey, moist, rich, d…
1       this is the recipe that we use at my school ca…
2       since there are already 411 recipes for brocco…
3       since there are already 411 recipes for brocco…
4       since there are already 411 recipes for brocco…
…                                                   …
234424                          deviled eggs, cajun-style
```

```
234425  i've heard of the 'cookies by design' company,…
234426  i've heard of the 'cookies by design' company,…
234427  i've heard of the 'cookies by design' company,…
234428  i've heard of the 'cookies by design' company,…

                                              ingredients  …  avg_rating  \
0       ['bittersweet chocolate', 'unsalted butter', '…  …         4.0
1       ['white sugar', 'brown sugar', 'salt', 'margar…  …         5.0
2       ['frozen broccoli cuts', 'cream of chicken sou…  …         5.0
3       ['frozen broccoli cuts', 'cream of chicken sou…  …         5.0
4       ['frozen broccoli cuts', 'cream of chicken sou…  …         5.0
…                                                    …  …         …
234424  ['hard-cooked eggs', 'mayonnaise', 'dijon must…  …         5.0
234425  ['butter', 'eagle brand condensed milk', 'ligh…  …         1.0
234426  ['granulated sugar', 'shortening', 'eggs', 'fl…  …         3.0
234427  ['granulated sugar', 'shortening', 'eggs', 'fl…  …         3.0
234428  ['granulated sugar', 'shortening', 'eggs', 'fl…  …         3.0

        calories (#) total fat (PDV)  sugar (PDV)  sodium (PDV)  \
0              138.4            10.0         50.0           3.0
1              595.1            46.0        211.0          22.0
2              194.8            20.0          6.0          32.0
3              194.8            20.0          6.0          32.0
4              194.8            20.0          6.0          32.0
…                …             …          …            …
234424          59.2             6.0          2.0           3.0
234425         188.0            11.0         57.0          11.0
234426         174.9            14.0         33.0           4.0
234427         174.9            14.0         33.0           4.0
234428         174.9            14.0         33.0           4.0

         protein (PDV)  saturated fat (PDV)  carbohydrates (PDV)  has_sugar  \
0                  3.0                 19.0                  6.0       True
1                 13.0                 51.0                 26.0       True
2                 22.0                 36.0                  3.0      False
3                 22.0                 36.0                  3.0      False
4                 22.0                 36.0                  3.0      False
…                  …                  …                  …        …
234424             6.0                  5.0                  0.0      False
234425             7.0                 21.0                  9.0       True
234426             4.0                 11.0                  6.0       True
234427             4.0                 11.0                  6.0       True
234428             4.0                 11.0                  6.0       True

        under 37
0          False
1          False
```

```
2          False
3          False
4          False
…          …
234424     False
234425      True
234426      True
234427      True
234428      True

[234429 rows x 24 columns]
```

[ ]:

```
observed = hyp_test_df.groupby('under 37')['avg_rating'].mean()
observed_diff = observed[True] - observed[False]

reps = 1000
diffs = []

for i in range(reps):
    with_shuffled = hyp_test_df.assign(shuffled_rating=np.random.
 ↪permutation(hyp_test_df['avg_rating']))
    group_means = (with_shuffled.groupby('under 37').mean().loc[:
 ↪,'shuffled_rating'])
    diff = group_means.loc[True] - group_means.loc[False]
    diffs.append(diff)
```

[ ]:

```
fig = px.histogram(
    pd.DataFrame(diffs), x=0, nbins=10, histnorm='probability',
    title='Empirical Distribution of the Mean Differences <br> in Average␣
 ↪Rating (Recipes Under 37 Minutes - Over 37 Minutes)')
fig.add_vline(x=observed_diff, line_color='red')
fig.update_layout(xaxis_range=[-0.03, 0.04], margin=dict(t=60))
fig.update_xaxes(title_text='Mean Differences in Rating')
```

[ ]:

```
fig.write_html('hyp_test.html', include_plotlyjs='cdn')
```

[ ]:

```
p_value = np.mean(np.array(diffs) >= observed_diff)
p_value
```

[ ]:

```
0.0
```

1.5 Step 5: Framing a Prediction Problem

[ ]:

```
# TODO

#The prediction problem I will focus on will be predicting the rating of a␣
 ↪recipe. For this, I will use a Random Forest Classifier,
# as the rating can be considered as categorical when I round the average␣
 ↪rating column to be an integer
```

1.6 Step 6: Baseline Model

[ ]:
```
# TODO
new_df.columns
#remove columns that have na in the average col. menas that there was no rating␣
 ↪on that recipe
new_rating = new_df[~new_df['avg_rating'].isna()]

new_rating = new_rating[['n_steps','n_ingredients', 'avg_rating']]
new_rating['rounded_rating'] = new_rating['avg_rating'].round().astype(int)
new_rating = new_rating.drop(columns=['avg_rating'])
new_rating
```

[ ]:
```
        n_steps  n_ingredients  rounded_rating
0            10              9               4
1            12             11               5
2             6              9               5
3             6              9               5
4             6              9               5
…          …             …             …
234424        7              8               5
234425        9             10               1
234426        5              7               3
234427        5              7               3
234428        5              7               3

[231652 rows x 3 columns]
```

[ ]:
```
X_train, X_test, y_train, y_test = (
    train_test_split(new_rating.drop(columns=['rounded_rating']),␣
 ↪new_rating['rounded_rating'], random_state=1)
)
```

[ ]:
```
preprocessor = ColumnTransformer(
    transformers=[
        ('std-scalar', StandardScaler(), ['n_steps', 'n_ingredients'])
    ]
)
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(random_state=42))
])
pipeline.fit(X_train, y_train)
```

```
[ ]: Pipeline(steps=[('preprocessor',
                      ColumnTransformer(transformers=[('std-scalar',
                                                       StandardScaler(),
                                                       ['n_steps',
                                                        'n_ingredients'])])),
                     ('classifier', RandomForestClassifier(random_state=42))])
```

```
[ ]: y_pred = pipeline.predict(X_test)
```

```
[ ]: f1_score(y_test, y_pred, average='weighted')
```

```
[ ]: 0.6386647045521429
```

1.7 Step 7: Final Model

```
[ ]: fin_mod = new_df.copy()
     fin_mod = fin_mod[~fin_mod['avg_rating'].isna()]
     fin_mod = fin_mod[['avg_rating', 'minutes', 'n_steps', 'n_ingredients',␣
      ↪'calories (#)', 'has_sugar']]
     fin_mod['rounded_rating'] = fin_mod['avg_rating'].round().astype(int)
     fin_mod = fin_mod.drop(columns=['avg_rating'])
     fin_mod
```

```
[ ]:         minutes  n_steps  n_ingredients  calories (#)  has_sugar  \
     0            40       10              9         138.4       True
     1            45       12             11         595.1       True
     2            40        6              9         194.8      False
     3            40        6              9         194.8      False
     4            40        6              9         194.8      False
     …          …       …             …            …         …
     234424       40        7              8          59.2      False
     234425       29        9             10         188.0       True
     234426       20        5              7         174.9       True
     234427       20        5              7         174.9       True
     234428       20        5              7         174.9       True

             rounded_rating
     0                    4
     1                    5
     2                    5
     3                    5
     4                    5
     …                  …
     234424               5
     234425               1
     234426               3
     234427               3
```

```
234428               3

[231652 rows x 6 columns]
```

[ ]:

```
# TODO
#transform calories (#) with standard scalar
#one hot encode ingredients
#one hot encode tags
#transform minutes with standard scalar

X = fin_mod.drop(columns = ['rounded_rating'])
y = fin_mod['rounded_rating']
X_train, X_test, y_train, y_test = (
    train_test_split(X, y, random_state=1)
)

preproc = ColumnTransformer(
    transformers = [
        ('std-scale', StandardScaler(), ['calories (#)', 'minutes']),
        ('one-hot', OneHotEncoder(drop='first', handle_unknown='ignore'),␣
 ↪['has_sugar']),
    ],
    remainder= 'passthrough'
)
pl = Pipeline([
    ('preprocessor', preproc),
    ('forest', RandomForestClassifier(max_depth = 3, n_estimators = 50))
])

#FIND THE CORRECT HYPERPARAMETERS
param_grid = {
    'forest__max_depth': [3, 5, 7, 10, None],
    'forest__n_estimators': [50, 100, 200]
}

grid_search = GridSearchCV(pl, param_grid, cv=5, scoring='accuracy', n_jobs=1)

grid_search.fit(X_train, y_train)
```

[ ]:

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('preprocessor',
ColumnTransformer(remainder='passthrough',
                                                          transformers=[('std-
scale',
StandardScaler(),
['calories '
                                                                          '(#)',
'minutes']),
                                                                        ('one-
hot',
OneHotEncoder(drop='first',
       handle_unknown='ignore'),
['has_sugar'])])),
                                       ('forest',
                                        RandomForestClassifier(max_depth=3,
n_estimators=50))]),
             n_jobs=1,
             param_grid={'forest__max_depth': [3, 5, 7, 10, None],
                         'forest__n_estimators': [50, 100, 200]},
             scoring='accuracy')
```

[ ]:

```
best_params = grid_search.best_params_
best_score = grid_search.best_score_

best_params
```

[ ]:

```
{'forest__max_depth': None, 'forest__n_estimators': 200}
```

[ ]:

```
y_pred = grid_search.predict(X_test)
```

[ ]:

```
f1_score(y_test, y_pred, average='weighted')
```

[ ]:

```
0.9133475035483493
```

1.8 Step 8: Fairness Analysis

[ ]:

```
# TODO
#split between recipes over 35 minutes and under 35 minutes
#run permutation test to check difference in precision between two groups
#null hyp: the model is fair. The precision for shorter recipes is roughly the␣
 ↪same as longer recipes
#alt hyp: the model is unfair. The precision for shorter recipes is lower than␣
 ↪longer recipes
```

[ ]:

```
new_df['minutes'].median()
```

[ ]:

```
35.0
```

[ ]:

```
shorter = fin_mod[fin_mod['minutes'] <= 35]
shorter_y = shorter['rounded_rating']
shorter_x = shorter.drop(columns = ['rounded_rating'])
shorter_pred = grid_search.predict(shorter_x)
shorter_prediction = precision_score(shorter_y, shorter_pred,␣
 ↪average='weighted')
```

```
shorter_prediction
```

[ ]:
```
0.9792886999818651
```

[ ]:
```
longer = fin_mod[fin_mod['minutes'] > 35]
longer_y = longer['rounded_rating']
longer_x = longer.drop(columns = ['rounded_rating'])
longer_pred = grid_search.predict(longer_x)
longer_prediction = precision_score(longer_y, longer_pred, average='weighted')
longer_prediction
```

[ ]:
```
0.9788856657967512
```

[ ]:
```
observed = shorter_prediction - longer_prediction
observed
```

[ ]:
```
0.000403034185113893
```

[ ]:
```
#permutation test
diffs = []
precision_df = fin_mod.copy()
```

[ ]:
```
for i in range(100):
    shuffled = np.random.permutation(precision_df['rounded_rating'])
    precision_df = precision_df.assign(shuffled_rating = shuffled)

    shorter = precision_df[precision_df['minutes'] <= 35]
    shorter_y = shorter['shuffled_rating']
    shorter_x = shorter.drop(columns = ['shuffled_rating', 'rounded_rating'])
    shorter_pred = grid_search.predict(shorter_x)
    shorter_prediction = precision_score(shorter_y, shorter_pred,␣
 ↪average='weighted')

    longer = precision_df[precision_df['minutes'] > 35]
    longer_y = longer['shuffled_rating']
    longer_x = longer.drop(columns = ['shuffled_rating', 'rounded_rating'])
    longer_pred = grid_search.predict(longer_x)
    longer_prediction = precision_score(longer_y, longer_pred,␣
 ↪average='weighted')

    diffs.append(shorter_prediction - longer_prediction)
```

[ ]:
```
diffs
```

[ ]:
```
[-0.0016998339853376843,
 -0.0034367070983210013,
 -0.0005883149926669828,
 -0.0022284249237168874,
 -0.0027897479966420002,
 -0.001078699308566633,
 0.0004479573064095632,
 -0.0016253886811437024,
 -0.003105552859107963,
 -0.005947199475976106,
 -0.0002718036995270623,
 0.002274193855278938 7,
 -0.0028210665538923596,
 0.0004650937305172853,
 0.0024180220464607993,
 0.0037865806552470627,
 0.005655988940146184,
 -0.00042169587053553226,
 -0.000714131958881814,
 0.0035712826804881193,
 0.0009404385836205842,
 -0.0010053325669157065,
 -3.408168565377512e-05,
 0.0006741225340823886,
 -0.003669073336166506,
 0.0013772367314695,
 -0.0017880815787645332,
 -0.004516893834347835,
 -0.0033615726863786,
 0.0025014192168028027,
 0.0004240023547243954,
 0.002648199668868312,
 0.0007866152003118687,
 0.004613735020790877,
 -0.0013676838275579195,
 0.0028710705372679834,
 0.0008992945629394677,
 0.0014695961740602836,
 0.0009339743577975179,
 -9.718639657463335e-05,
 -0.001766939802653078,
 -0.001333015934697368,
 0.0027650655015563075,
 -0.004749897251573376,
 -0.0026602914799384036,
 0.002859615652082814,
 -0.002145991081213494,
 -0.0014358815111293888,
 0.0008729149426810467,
 0.003563366465701878,
 0.0024910337518007086,
 -0.0024335211613922825,
 0.0008458757452383114,
 0.004326250539573473,
 0.0014667774167907988,
 0.002206024450906341,
 0.0010228784865179419,
 -0.00021576026878245003,
 -0.0029616605824426268,
 -0.001559995383600854,
 -0.0023563412786240523,
 -0.0015365031994247769,
 0.0009453699304010632,
 -0.0034655224087873915,
 0.004441390572462622,
 -0.003698968554837O9,
 -0.001864270810740476,
 0.0007865923628006533,
 -0.0008794365861319875,
 0.004138231193082542,
 0.0018301860594152064,
 0.0017263235794487963,
 -0.004624106255835869,
 -0.00046171471217171245,
 -0.0016752134984703293,
 0.001845591446487438,
 0.002409355676392555,
 0.001719731601813268,
 0.001484121061421817,
 -0.0014363030173651925,
 0.0005326241544622023,
 0.0018881591354553695,
 -0.0013406083118409073,
 -0.001491404182760947,
 -0.0008065737031349718,
 -0.0014488471832587724,
 -0.00028051794522354623,
 -0.0004510262879324767,
 0.003579459252160966,
 0.0018446227783069924,
 -0.0032880719350415477,
 0.0013483530758955364,
 0.0047582224497441095,
 -0.001695151678702711,
 -0.001604868132056514,
 0.003462059346159618З,
 -0.0012418937713889466,
 -0.001738371263766858,
 -0.00463602531163454,
 -2.206453360742433e-05]
```

[ ]:

```
p_value = np.mean(np.abs(diffs) >= np.abs(observed))
p_value
```

[ ]:

```
0.94
```

[ ]:

```
fig = px.histogram(
    pd.DataFrame(diffs), x=0, nbins=10, histnorm='probability',
    title='Empirical Distribution of the Precision Differences <br> in Shorter␣
 ↪vs Longer Recipes (shorter - longer)')
fig.add_vline(x=observed, line_color='red')
fig.update_layout(xaxis_range=[-0.01, 0.01], margin=dict(t=60))
fig.update_xaxes(title_text= 'Difference in Precision')
```

[ ]:

```
fig.write_html('fairness.html', include_plotlyjs='cdn')
```

[ ]:

```

```