All About Calories

Name(s): Sri Supratik Talari

Website Link: https://github.com/SriSupratikTalari/RecipiesandRatings/blob/main/README.md

In [1]:

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

import plotly.express as px
pd.options.plotting.backend = 'plotly'

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

Step 1: Introduction

In [2]:

```
Question_1_4='What type of recipes have the most calories'
```

Step 2: Data Cleaning and Exploratory Data Analysis

In [3]:

```
# Merging the two dataframes after replacing 0 vals in rating with np.nan and
# creating a column that contains the average rating for that recipe
receipes = 'RAW_recipes.csv'
interactions = 'RAW_interactions.csv'
receipes = pd.read_csv(receipes)
interactions = pd.read_csv(interactions)
receipes_interactions = receipes.merge(interactions, left_on='id', right_on='recipe_id',
how='left')
receipes_interactions["rating"] = receipes_interactions["rating"].replace(0, np.nan)
a = receipes_interactions.groupby('id')['rating'].mean()
receipes_interactions['average_rating'] = receipes_interactions['id'].map(a)
receipes_interactions
```

Out[3]:

	Unnamed: 0_x	name	id	minutes	...	date	rating	review	average_rating
0	111	1 brownies in the world best ever	333281	40	...	2008-11-19	4.0	These were pretty good, but took forever to ba...	4.0
1	115	1 in canada chocolate chip cookies	453467	45	...	2012-01-26	5.0	Originally I was gonna cut the recipe in half ...	5.0
2	118	412 broccoli casserole	306168	40	...	2008-12-31	5.0	This was one of the best broccoli casseroles t...	5.0
...	...	...	...	...	...	...	...	...	...
234426	231636	cookies by design sugar shortbread cookies	298509	20	...	2008-06-19	1.0	This recipe tastes nothing like the Cookies by...	3.0
234427	231636	cookies by design sugar shortbread cookies	298509	20	...	2010-02-08	5.0	yummy cookies, i love this recipe me and my sm...	3.0
234428	231636	cookies by design sugar shortbread cookies	298509	20	...	2014-11-01	NaN	I work at a Cookies By Design and can say this...	3.0

234429 rows × 20 columns

In [4]:

```python
# filtering the dataframe to only the columns that I want
# to use and spliting all the strings in nutrition by ',' and getting rid of the brackets
receipes_interactions=receipes_interactions[['id','name','tags','nutrition','n_steps','n_ingredients','rating','average_rating']]
receipes_interactions['nutrition']=list(map(lambda x: list(x[1:-1].split(',')), receipes_interactions['nutrition']))
```

In [5]:

```python
# creating individual columns for all the values of a given index in the nutrition column

# and going through the tags column and filtering it to only return np.nan or the time tag and making it as a new column
# finally I create a column that contains bool representations of the rating column if it has a null value or not
cal=receipes_interactions['nutrition'].transform(lambda x: float(x[0]))
tol_fat=receipes_interactions['nutrition'].transform(lambda x: float(x[1])/100.0)
sugar=receipes_interactions['nutrition'].transform(lambda x: float(x[2])/100.0)
sodium=receipes_interactions['nutrition'].transform(lambda x: float(x[3])/100.0)
protein=receipes_interactions['nutrition'].transform(lambda x: float(x[4])/100.0)
sat_fat=receipes_interactions['nutrition'].transform(lambda x: float(x[5])/100.0)
carbs=receipes_interactions['nutrition'].transform(lambda x: float(x[6])/100.0)
receipes_interactions=receipes_interactions.assign(calories=cal,total_fats=tol_fat,sugars=sugar,sodium=sodium,protein=protein,saturated_fat=sat_fat,carbohydrates=carbs).drop(['nutrition'],axis=1)
import re
import numpy as np

receipes_interactions['filtered_tags'] = receipes_interactions['tags'].apply(
    lambda x: [f"{max(map(int, re.findall(r'(\d+)-minutes-or-less', x, re.IGNORECASE)))}-minutes-or-less"] if re.findall(r'(\d+)-minutes-or-less', x, re.IGNORECASE) else
              [f"{max(map(int, re.findall(r'(\d+)-hours-or-less', x, re.IGNORECASE)))}-hours-or-less"] if re.findall(r'(\d+)-hours-or-less', x, re.IGNORECASE) else np.nan
)
receipes_interactions = receipes_interactions.drop('tags', axis=1)
receipes_interactions['ratings_missing']=receipes_interactions['rating'].isna()
receipes_interactions
receipes_interactions
```

Out[5]:

	id	name	n_steps	n_ingredients	...	saturated_fat	carbohydrates	filtered_tags	ratings_missing
0	333281	1 brownies in the world best ever	10	9	...	0.19	0.06	[60-minutes-or-less]	False
1	453467	1 in canada chocolate chip cookies	12	11	...	0.51	0.26	[60-minutes-or-less]	False
2	306168	412 broccoli casserole	6	9	...	0.36	0.03	[60-minutes-or-less]	False
...	...	...	...	...	...	...	...	...	...
234426	298509	cookies by design sugar shortbread cookies	5	7	...	0.11	0.06	[30-minutes-or-less]	False
234427	298509	cookies by design sugar shortbread cookies	5	7	...	0.11	0.06	[30-minutes-or-less]	False
234428	298509	cookies by design sugar shortbread cookies	5	7	...	0.11	0.06	[30-minutes-or-less]	True

234429 rows × 15 columns

In [6]:

```
receipes_interactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234429 entries, 0 to 234428
Data columns (total 15 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   id               234429 non-null  int64  
 1   name             234428 non-null  object 
 2   n_steps          234429 non-null  int64  
 3   n_ingredients    234429 non-null  int64  
 4   rating           219393 non-null  float64
 5   average_rating   231652 non-null  float64
 6   calories         234429 non-null  float64
 7   total_fats       234429 non-null  float64
 8   sugars           234429 non-null  float64
 9   sodium           234429 non-null  float64
 10  protein          234429 non-null  float64
 11  saturated_fat    234429 non-null  float64
 12  carbohydrates    234429 non-null  float64
 13  filtered_tags    221050 non-null  object 
 14  ratings_missing  234429 non-null  bool   
dtypes: bool(1), float64(9), int64(3), object(2)
memory usage: 25.3+ MB
```

In [7]:

```
# cretaing three functions that returns bins for the calorie, protein, and tota_fat colum
ns
def cal_cat(x):
    if 0 <= x < 25000.0:
        return '[0, 25000.0)'
    else:
        return '[25000.0,50000.0)'
def protein_cat(x):
    if 0 <= x < 25:
        return '[0.0, 25.0)'
    else:
        return '[25.0,50.0)'
def fat_cat(x):
    if 0<=x<17.5:
        return '[0,17.5)'
    else:
        return '[17.5,35.0)'
```

In [8]:

```
# applying those functions to their designated columns and then adding those columns to t
he original dataframe
receipes_interactions['cal_cat']=receipes_interactions['calories'].apply(cal_cat)
receipes_interactions['protein_cat']=receipes_interactions['protein'].apply(protein_cat)
receipes_interactions['fat_cat']=receipes_interactions['total_fats'].apply(fat_cat)
```

In [9]:

```
pip install --upgrade plotly
```

```
Requirement already satisfied: plotly in /Users/srisupratiktalari/miniforge3/envs/dsc80/l
ib/python3.12/site-packages (6.0.0)
Requirement already satisfied: narwhals>=1.15.1 in /Users/srisupratiktalari/miniforge3/en
vs/dsc80/lib/python3.12/site-packages (from plotly) (1.28.0)
Requirement already satisfied: packaging in /Users/srisupratiktalari/miniforge3/envs/dsc8
0/lib/python3.12/site-packages (from plotly) (24.2)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:

```
# creating plots for the Univariate Analysis on sodium column.
```

```
import plotly.express as px

fig = px.box(x=receipes_interactions['sodium'], title="Sodium Distribution")
fig.update_layout(xaxis_title="Sodium (PDV) proportion")
fig.write_html('sodium.html', include_plotlyjs='cdn')
fig.show()
```

In [11]:

```
# creating plots for the Univariate Analysis on protein column.
fig_2 = px.box(x=receipes_interactions['protein'], title="Protein Distribution")
fig_2.update_layout(xaxis_title=" Protein (PDV) proportion")
fig_2.write_html('protein.html', include_plotlyjs='cdn')
fig_2.show()
```

In [12]:

```
# creating plots for the Univariate Analysis on total_fatcolumn.
fig_3 = px.box(x=receipes_interactions['total_fats'], title="Total Fats Distribution")
fig_3.update_layout(xaxis_title="Total Fats (PDV) proportion")
fig_3.write_html('total_fat.html', include_plotlyjs='cdn')
fig_3.show()
```

In [13]:

```
# creating plots for the Univariate Analysis on sugars column.
fig_4 = px.box(x=receipes_interactions['sugars'], title="Sugars Distribution")
fig_4.update_layout(xaxis_title="Sugars (PDV) proportion")
fig_4.write_html('sugars.html', include_plotlyjs='cdn')
fig_4.show()
```

In [14]:

```
# creating bivariate analysis for calories and protein
fig_5 = px.scatter(receipes_interactions, x='protein', y='calories',
                   title="Protein vs Calories",
                   labels={'protein': 'Protein (PDV) proportion', 'calories': 'Calories'
})
fig_5.write_html('proteinxcal.html', include_plotlyjs='cdn')
fig_5.show()
```

In [15]:

```
# creating bivariate analysis for and calories and total_fat
fig_6 = px.scatter(receipes_interactions, x='total_fats', y='calories',
                   title="Total Fats vs Calories",
                   labels={'total_fats': 'Total Fats (PDV) proportion', 'calories': 'Cal
ories'})
fig_6.write_html('totalxcal.html', include_plotlyjs='cdn')
fig_6.show()
```

In [16]:

```
# creating a pivot table between cal_cat and protein_cat witht the count aggregation
a=receipes_interactions.pivot_table(index='cal_cat',columns='protein_cat',values='protein
',aggfunc='count',fill_value=0)

a
```

Out[16]:

protein_cat	[0.0, 25.0)	[25.0,50.0)
cal_cat		
[0, 25000.0)	234414	7
[25000.0,50000.0)	7	1

Step 3: Assessment of Missingness

In [17]:

```
# running the Missingness perumation test on all of the numerical columns to see if ratin
g is MCAR or MAR
base_df = receipes_interactions.copy()
choices = []

for col in ['n_steps', 'n_ingredients', 'average_rating', 'calories', 'total_fats', 'sug
ars',
            'sodium', 'protein', 'saturated_fat', 'carbohydrates']:

    obs_mean_diff = base_df.groupby('ratings_missing')[[col]].mean().diff().iloc[1].valu
```

```
es[0]
    diff = []

    for j in range(500):
        base_df_with_perm=base_df.copy()
        base_df_with_perm = base_df_with_perm.assign(ratings_missing=np.random.permutati
on(base_df['ratings_missing']))
        something = base_df_with_perm.groupby('ratings_missing')[[col]].mean().diff().il
oc[1].values[0]
        diff.append(something)

    p_val = (np.array(diff) >= obs_mean_diff).mean()

    if p_val >= 0.05:
        choices.append(f'{col}:fail to reject')
    else:
        choices.append(f'{col}:reject')
choices
```

Out[17]:

```
['n_steps:reject',
 'n_ingredients:reject',
 'average_rating:fail to reject',
 'calories:reject',
 'total_fats:reject',
 'sugars:reject',
 'sodium:reject',
 'protein:reject',
 'saturated_fat:reject',
 'carbohydrates:reject']
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Step 4: Hypothesis Testing

In [18]:

```
# observed absolute average of means for our permutation test
obs=receipes_interactions.groupby('protein_cat')[['calories']].mean()
obs_stat=abs(obs.iloc[0]- obs.iloc[1])
obs_stat
```

Out[18]:

```
calories    22559.68
dtype: float64
```

In [19]:

```
# conduting our permutation test by shuffling protein_cat
n_repetitions = 500

differences = []
for _ in range(n_repetitions):
    with_shuffled = receipes_interactions.assign(Shuffled_Weights=np.random.permutation(
receipes_interactions['protein_cat']))
    with_shuffled=with_shuffled.groupby('Shuffled_Weights')[['calories']].mean()
    difference = abs(with_shuffled.iloc[0]- with_shuffled.iloc[1])
    differences.append(difference)
differences[:10]
```

Out[19]:

```
[calories    78.61
 dtype: float64,
 calories    5.81
 dtype: float64,
 calories    143.12
 dtype: float64,
 calories    82.59
 dtype: float64,
 calories    94.06
 dtype: float64,
 calories    49.42
 dtype: float64,
 calories    239.15
 dtype: float64,
 calories    28.73
 dtype: float64,
 calories    220.11
 dtype: float64,
 calories    15.19
 dtype: float64]
```

In [20]:

```
# obtaining our p-val
p=(np.array(differences) >= obs_stat.iloc[0]).mean()
p
```

Out[20]:

```
np.float64(0.0)
```

Step 5: Framing a Prediction Problem

In [21]:

```
question='Predict calories of recipes'
```

Step 6: Baseline Model

In [22]:

```
# creating our baseline model(linearRegression) with protein and carbohydrates as our fea
tures and calories as the column to predict
from sklearn.model_selection import train_test_split
X=receipes_interactions[['protein','carbohydrates']]
y=receipes_interactions['calories']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,random_state=42
)
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
pl = Pipeline([('lin_mod', LinearRegression())])
model=pl.fit(X_train,y_train)
model.named_steps['lin_mod'].coef_
print(model.score(X_train, y_train))
from sklearn.metrics import root_mean_squared_error
rmse_dict = {}
rmse_dict['two train feature: protein,carbohydrate']= root_mean_squared_error(y_train, mo
del.predict(X_train))
rmse_dict['two test feature: protein,carbohydrate']= root_mean_squared_error(y_test, mode
l.predict(X_test))
rmse_dict
```

```
0.811680953223733
```

Out[22]:

```
{'two train feature: protein,carbohydrate': np.float64(255.2916748677694),
 'two test feature: protein,carbohydrate': np.float64(248.69192187011018)}
```

Step 7: Final Model

In [23]:

```
# creating our final model with 'protein', 'carbohydrates', 'fat_cat', 'cal_cat','sugars'
as our features
# performing onehotencoding on 'fat_cat' and 'cal_cat' and binaraizer on 'sugars' with a
threshold of 151.3
# and using polynomial degree as our hyperparameter tuning
from sklearn.preprocessing import FunctionTransformer, OneHotEncoder
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn.preprocessing import Binarizer

X = receipes_interactions[['protein', 'carbohydrates', 'fat_cat', 'cal_cat','sugars']]
y = receipes_interactions['calories']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=4
2)


preproc = make_column_transformer(
    (OneHotEncoder(handle_unknown='ignore'), ['fat_cat', 'cal_cat']),(Binarizer(threshold
=151.3),['sugars']),  # OneHotEncoding for categorical variables
    remainder='passthrough'
)


pipeline = make_pipeline(preproc, PolynomialFeatures(), LinearRegression())


param_grid = {
    'polynomialfeatures__degree': [1, 2, 3, 4, 5]
}


grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='neg_mean_squared_error',
n_jobs=-1)
grid_search.fit(X_train, y_train)

# Get best model
best_model = grid_search.best_estimator_

# Evaluate performance
y_pred = best_model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

# Print best polynomial degree and RMSE
print("Best Polynomial Degree:", grid_search.best_params_['polynomialfeatures__degree'])
print("Test Set RMSE:", rmse)

# Store RMSE in dictionary
rmse_dict['Polynomial Regression (Best Degree)'] = rmse
rmse_dict
```

```
Best Polynomial Degree: 1
Test Set RMSE: 219.51823848043935
```

Out[23]:

```
{'two train feature: protein,carbohydrate': np.float64(255.2916748677694),
 'two test feature: protein,carbohydrate': np.float64(248.69192187011018),
 'Polynomial Regression (Best Degree)': np.float64(219.51823848043935)}
```

Step 8: Fairness Analysis

In [24]:

```
# getting the baseline rmse for our fairness test
from sklearn import metrics
import warnings
compute_rmse = lambda x: metrics.mean_squared_error(x['calories'], x['prediction'], squa
red=False)


b = Binarizer(threshold=25.0)


results = X_test.copy()


results['below_25'] = b.transform(results[['protein']])


results['prediction'] = y_pred
results['calories'] = y_test
warnings.filterwarnings("ignore", category=UserWarning)


warnings.filterwarnings("ignore", category=FutureWarning)
obs = results.groupby('below_25')[['calories', 'prediction']].apply(compute_rmse).diff()
.iloc[-1]
obs
```

```
/Users/srisupratiktalari/miniforge3/envs/dsc80/lib/python3.12/site-packages/sklearn/base.
py:486: UserWarning:

X has feature names, but Binarizer was fitted without feature names
```

Out[24]:

```
np.float64(6202.053213109698)
```

In [25]:

```
# conducting our permuation test for fairness
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=FutureWarning)

diff_in_acc = []
for _ in range(500):
    s = (
        results[['below_25', 'prediction', 'calories']]
        .assign(below_25=np.random.permutation(results['below_25']))
        .groupby('below_25')
        [['calories', 'prediction']]
        .apply(compute_rmse)
        .diff()
        .iloc[-1]
    )

    diff_in_acc.append(abs(s))
diff_in_acc
```

Out[25]:

```
[np.float64(186.75756955547592),
    np.float64(171.46377151885548),
    np.float64(165.5512480752634),
    np.float64(12.57573223933528),
    np.float64(206.83260501522741),
    np.float64(55.190936798812686),
    np.float64(44.528296189101326),
    np.float64(121.62911385487368),
    np.float64(53.45028944819936),
    np.float64(134.70733171693158),
    np.float64(212.2669842267058),
    np.float64(73.29115525628691),
    np.float64(71.46300934787942),
    np.float64(99.85705477416556),
    np.float64(143.62926243899025),
    np.float64(211.09021995149416),
    np.float64(198.34509384543696),
    np.float64(150.76680256189772),
    np.float64(163.97936638542333),
    np.float64(168.3137593670125),
    np.float64(59.00348493066721),
    np.float64(202.12322450904642),
    np.float64(214.25448519367873),
    np.float64(110.22902206340325),
    np.float64(152.18556006551484),
    np.float64(153.21369040538823),
    np.float64(168.25750914300613),
    np.float64(197.10134173824758),
    np.float64(142.27300436723553),
    np.float64(188.12944797588958),
    np.float64(108.75400944882142),
    np.float64(90.71321647766217),
    np.float64(213.42635982808685),
    np.float64(126.66977622069123),
    np.float64(157.8105882757055),
    np.float64(145.3480224627701),
    np.float64(188.8763247765284),
    np.float64(191.99195682028764),
    np.float64(114.46030731179096),
    np.float64(216.4638609082356),
    np.float64(73.71303504441329),
    np.float64(197.30134208520494),
    np.float64(1005.241313321944),
    np.float64(209.60448245016786),
    np.float64(107.85400166871636),
    np.float64(125.32914147550531),
    np.float64(119.46034717042156),
    np.float64(84.817863308321),
    np.float64(181.58568034546101),
    np.float64(142.25425425464255),
    np.float64(175.48566091314893),
    np.float64(160.29184993783858),
    np.float64(86.03504379043042),
    np.float64(184.97318988772173),
    np.float64(80.60686132593642),
    np.float64(80.00060476573134),
    np.float64(215.39823561095844),
    np.float64(123.26662621012679),
    np.float64(211.32323366003396),
    np.float64(135.96046491687179),
    np.float64(210.89198337804612),
    np.float64(180.220051246362),
    np.float64(58.234725316329815),
    np.float64(111.31028120311484),
    np.float64(189.32632584050026),
    np.float64(204.52947753780316),
    np.float64(44.1532910829967 3),
    np.float64(177.6544181533228),
    np.float64(153.11681490574517),
    np.float64(52.7627805554311 6),
    np.float64(166.54500220553842),
    np.float64(73.04740248350836),
    np.float64(65.49106423675619),
    np.float64(215.66386069331838),
    np.float64(63.13478581122294),
    np.float64(202.5232250437163),
    np.float64(196.61384087951356),
    np.float64(186.03569269692832),
    np.float64(165.88874948660606),
    np.float64(104.23521975069531),
    np.float64(179.49817402099683),
    np.float64(203.7919766573388),
    np.float64(57.49096597093441),
    np.float64(178.17004482089703),
    np.float64(179.7075496706193),
    np.float64(136.73546992707014),
    np.float64(135.16670973706567),
    np.float64(213.68260994690863),
    np.float64(60.03787279454542),
    np.float64(158.44496630286744),
    np.float64(206.77010495343993),
    np.float64(12.997614857004464),
    np.float64(151.88555849189063),
    np.float64(31.91249340965163),
    np.float64(165.5512480752634),
    np.float64(79.17559579255166),
    np.float64(111.61965880151428),
    np.float64(115.0790623496531),
    np.float64(130.15105087399425),
    np.float64(138.88235856233763),
    np.float64(175.12003465653075),
    np.float64(97.4476571386634),
    np.float64(51.869018939907676),
    np.float64(126.86352761753272),
    np.float64(103.56646374076698),
    np.float64(216.86386099703955),
    np.float64(206.5732297568247),
    np.float64(172.65127589972786),
    np.float64(190.19507785003208),
    np.float64(2.462030253301606),
    np.float64(187.9419475170376),
    np.float64(187.09819541838752),
    np.float64(216.64511095001583),
    np.float64(115.99781977521656),
    np.float64(114.2196803445857),
    np.float64(155.39807647787),
    np.float64(161.91373230218954),
    np.float64(268.5235204285932),
    np.float64(179.4731739432022),
    np.float64(172.4669002267277),
    np.float64(88.31944228909865),
    np.float64(9.27464338625316),
    np.float64(213.43260983104565),
    np.float64(196.98259153077055),
    np.float64(190.76695414083505),
    np.float64(158.8293431218408),
    np.float64(153.97306929667516),
    np.float64(163.8637408857447),
    np.float64(64.09104739934435),
    np.float64(217.25761107230954),
    np.float64(198.43259398915438),
    np.float64(179.07942271152953),
    np.float64(69.76923974449662),
    np.float64(210.37010801746862),
    np.float64(194.70758734430402),
    np.float64(157.1762102172634),
    np.float64(153.1543150992419),
    np.float64(212.35760927747822),
    np.float64(218.0513611874119),
    np.float64(135.55733729230283),
    np.float64(165.1856215363177),
    np.float64(215.30136057955647),
    np.float64(191.14820498724808),
    np.float64(50.679601057439186),
    np.float64(199.28884536412681),
    np.float64(65.95044472833587),
    np.float64(189.80445195372417),
    np.float64(168.25125911810133),
    np.float64(206.5732297568247),
    np.float64(99.79767922177442),
    np.float64(194.27946151148998),
    np.float64(213.24198473943585),
    np.float64(147.5636600455523),
    np.float64(325.95082704646666),
    np.float64(163.45123909464087),
    np.float64(270.13608170052623),
    np.float64(216.32323587406086),
    np.float64(175.02940934345438),
    np.float64(211.44198373514448),
    np.float64(46.35644592483021),
    np.float64(110.13527126666162),
    np.float64(206.09822927004848),
    np.float64(77.64432899756372),
    np.float64(93.0538646987338),
    np.float64(173.08252746354586),
    np.float64(216.95604261661634),
    np.float64(215.56386066295593),
    np.float64(156.82620851645726),
    np.float64(187.1981956700069),
    np.float64(204.96697804013405),
    np.float64(129.19479419592005),
    np.float64(155.40745152459212),
    np.float64(155.97307933085156),
    np.float64(156.363706254359),
    np.float64(0.7588764064001339),
    np.float64(747.0286962889402),
    np.float64(205.56072869807227),
    np.float64(31.89374313621974),
    np.float64(166.23250091498355),
    np.float64(126.83852743745908),
    np.float64(186.91694496034617),
    np.float64(38.34809798550043),
    np.float64(173.35127843078783),
    np.float64(184.37318826257948),
    np.float64(163.23248813944653),
    np.float64(97.2664054176143),
    np.float64(141.52924988008874),
    np.float64(106.66024125213386),
    np.float64(188.12944797588958),
    np.float64(93.91324811761788),
    np.float64(191.75758131666063),
    np.float64(112.19153858503017),
    np.float64(147.969912311296),
    np.float64(178.79192180460103),
    np.float64(247.24773094543798),
    np.float64(120.5884808948463),
    np.float64(213.9857350808763),
    np.float64(123.54475328792734),
    np.float64(64.1129226636053),
    np.float64(210.33573299297484),
    np.float64(174.37628206826656),
    np.float64(163.11998764675548),
    np.float64(118.97596839412242),
    np.float64(1.625539358044108),
    np.float64(69.86299083538438),
    np.float64(180.79817799937223),
    np.float64(102.40395321086689),
    np.float64(185.8513172157406),
    np.float64(171.34502107473986),
    np.float64(160.1199741454342),
    np.float64(112.94154481996355),
    np.float64(175.70128664936055),
    np.float64(189.98570237108592),
    np.float64(126.47602482093197),
    np.float64(135.6073376185169),
    np.float64(137.19484787478737),
    np.float64(136.71046976617959),
    np.float64(124.21975830567582),
    np.float64(198.0888434211268),
    np.float64(171.6418971829739),
    np.float64(199.74197106869985),
    np.float64(163.85124083166315),
    np.float64(178.09191956956266),
    np.float64(92.78198702335209),
    np.float64(117.98846063885682),
    np.float64(72.85365027621043),
    np.float64(190.15132775023537),
    np.float64(214.88886043767903),
    np.float64(195.5700889788281),
    np.float64(180.17630111264438),
    np.float64(156.74495812026544),
    np.float64(193.9450858511352),
    np.float64(17.274771634149516),
    np.float64(210.65448321657573),
    np.float64(87.81318711644411),
    np.float64(885.8773777051206),
    np.float64(121.56348835537668),
    np.float64(105.85710918179092),
    np.float64(178.57629612018846),
    np.float64(122.39161963401276),
    np.float64(203.8732267564113),
    np.float64(200.57009731513185),
    np.float64(183.44506069356726),
    np.float64(44.15329108299673),
    np.float64(172.46690022672766),
    np.float64(52.8690319321683),
    np.float64(144.72926888066118),
    np.float64(185.99819259927366),
    np.float64(207.65760580236682),
    np.float64(218.06073618847876),
    np.float64(134.66045640783238),
    np.float64(80.52248541462953),
    np.float64(86.52879890384662),
    np.float64(173.54502912461925),
    np.float64(39.11998847977526),
    np.float64(115.64156690370444),
    np.float64(117.0197029571911),
    np.float64(171.60439704336446),
    np.float64(151.1418045604442),
    np.float64(157.06683468678435),
    np.float64(142.78863245283574),
    np.float64(184.31068809168463),
    np.float64(78.30371125203399),
    np.float64(164.79186986737108),
    np.float64(195.58571400790572),
    np.float64(160.8074773012746),
    np.float64(172.85127662677428),
    np.float64(190.1857028286596),
    np.float64(175.78566193645966),
    np.float64(164.69499445491007),
    np.float64(122.7853726006285),
    np.float64(93.4132432263415),
    np.float64(152.04805934513877),
    np.float64(177.20129167081893),
    np.float64(117.4999279148461),
    np.float64(168.39188467772442),
    np.float64(104.18834433057151),
    np.float64(149.1355437410304),
    np.float64(201.25759830940973),
    np.float64(131.41668460333383),
    np.float64(5.669359502220743),
    np.float64(154.25744574237063),
    np.float64(197.37634221451236),
    np.float64(98.36016576446733),
    np.float64(199.2450952952542),
    np.float64(95.7538909561938),
    np.float64(212.22948420550983),
    np.float64(127.14165461757561),
    np.float64(9.325674018868085),
    np.float64(198.26071870628863),
    np.float64(25.562399244890457),
    np.float64(216.29823586782445),
    np.float64(199.66384594836032),
    np.float64(126.36977405208324),
    np.float64(169.4168887103218),
    np.float64(208.02635613708662),
    np.float64(141.77612637428717),
    np.float64(164.22624244882667),
    np.float64(35.20004092894689),
    np.float64(180.50755212137653),
    np.float64(58.63914568298941),
    np.float64(181.32630457806496),
    np.float64(191.12945494588536),
    np.float64(160.55435114362464),
    np.float64(128.12603664807975),
    np.float64(142.32925470485074),
    np.float64(205.7482289001493),
    np.float64(188.26694831064432),
    np.float64(121.500987879354),
    np.float64(206.42635460818136),
    np.float64(25.90927946966238),
    np.float64(139.34798647234672),
    np.float64(197.00134156360298),
    np.float64(116.69782538875083),
    np.float64(152.4386863878189),
    np.float64(198.27634373209858),
    np.float64(42.74077175187645),
    np.float64(7.383987350917806),
    np.float64(151.1199294441626),
    np.float64(218.34511121759365),
    np.float64(196.41696552748076),
    np.float64(143.53238686717785),
    np.float64(168.06063335704718),
    np.float64(211.7763589407515),
    np.float64(152.1168097055105),
    np.float64(19.235229100023105),
    np.float64(150.90430329596728),
    np.float64(176.77316525544555),
    np.float64(129.3885455547355),
    np.float64(149.4480454468679),
    np.float64(110.13527126666162),
    np.float64(90.4194635333271),
    np.float64(173.36377847563918),
    np.float64(170.87626931094877),
    np.float64(76.56619206318692),
    np.float64(167.84500749277805),
    np.float64(189.71695175132527),
    np.float64(131.7854371232745),
    np.float64(146.2074023885539),
    np.float64(153.3230659686247),
    np.float64(190.9513295515767),
    np.float64(90.7038413837976),
    np.float64(139.3511114918201),
    np.float64(203.81072668024717),
    np.float64(129.5416716265737),
    np.float64(65.36918777705333),
    np.float64(219.20136126730148),
    np.float64(127.06040403391509),
    np.float64(81.63499738592745),
    np.float64(170.24501690872407),
    np.float64(137.51359991050276),
    np.float64(175.34503543106393),
    np.float64(218.47011122840266),
    np.float64(194.8107125427797),
    np.float64(26.624915219145606),
    np.float64(42.68764602179499),
    np.float64(199.4794706624781),
    np.float64(187.74507203230203),
    np.float64(211.54198379754533),
    np.float64(145.1667714165473),
    np.float64(204.8544779123842),
    np.float64(195.46071377475283),
    np.float64(217.87169871350676),
    np.float64(43.15952749895661),
    np.float64(136.44171803353748),
    np.float64(176.8794156080352),
    np.float64(115.31031422485123),
    np.float64(44.70642361067169),
    np.float64(125.81039498985038),
    np.float64(149.88242280537372),
    np.float64(167.12937959850692),
    np.float64(155.06682482263778),
    np.float64(205.69197883981312),
    np.float64(13.291370169329326),
    np.float64(124.22913337511744),
    np.float64(59.875370779140354),
    np.float64(126.15727251175329),
    np.float64(198.76071952279457),
    np.float64(110.15402142606453),
    np.float64(77.75519817909037),
    np.float64(194.57321208444395),
    np.float64(179.21379813321676),
    np.float64(218.19198620269748),
    np.float64(197.28571705821085),
    np.float64(198.3357188300033),
    np.float64(191.8763315723653),
    np.float64(145.5917738657334),
    np.float64(145.5698987400151),
    np.float64(158.86684329868672),
    np.float64(198.07321839508913),
    np.float64(6.105841212600325),
    np.float64(139.0229844429756),
    np.float64(156.54808215813432),
    np.float64(206.2513544288853),
    np.float64(45.00642768353154),
    np.float64(307.75631850565725),
    np.float64(162.09810812642738),
    np.float64(167.1512546875679),
    np.float64(192.58570807703353),
    np.float64(124.5072604321106),
    np.float64(182.94243364241925),
    np.float64(131.26043353237424),
    np.float64(207.13573031058957),
    np.float64(160.51060094303227),
    np.float64(212.53885937710794),
    np.float64(147.82928652845177),
    np.float64(171.595022008445),
    np.float64(212.50448435840883),
    np.float64(170.560643113708),
    np.float64(217.49511111188394),
    np.float64(150.42305072029458),
    np.float64(203.62010144607143),
    np.float64(194.10758617314076),
    np.float64(150.6511769434745),
    np.float64(178.18566987110702),
    np.float64(28.413527700984247),
    np.float64(159.77622255373703),
    np.float64(212.3451092705131),
    np.float64(130.88230593279926),
    np.float64(180.8700532154984),
    np.float64(145.83552526407857),
    np.float64(198.09196842633202),
    np.float64(132.85106934607217),
    np.float64(156.47308179081568),
    np.float64(164.94499551783787),
    np.float64(136.56671884011584),
    np.float64(171.03876992433047),
    np.float64(109.83214368585473),
    np.float64(161.46060526529388),
    np.float64(83.73189468826311),
    np.float64(100.57893646816117),
    np.float64(155.9762043462867),
    np.float64(207.56385571558346),
    np.float64(24.368631192666896),
    np.float64(154.26369577407354),
    np.float64(38.81060726395825),
    np.float64(108.69463393749041),
    np.float64(200.71384752606406),
    np.float64(218.19198620269748),
    np.float64(123.20632866130038),
    np.float64(79.0068439506112),
    np.float64(57.41909006552959),
    np.float64(176.51066438057978),
    np.float64(139.98861544850797),
    np.float64(155.1793253857485),
    np.float64(32.69375477835703),
    np.float64(183.62318619180198),
    np.float64(96.60702413501431),
    np.float64(159.11059444552117),
    np.float64(213.54823488523616),
    np.float64(200.13884667269812),
    np.float64(214.7044853699871),
    np.float64(148.4824151513194),
    np.float64(104.93210097639167),
    np.float64(204.0169769304365),
    np.float64(110.37277328374697),
    np.float64(141.1979978677724),
    np.float64(167.7012569145909),
    np.float64(103.31646148513826),
    np.float64(213.6982349539888),
    np.float64(136.55734377966456),
    np.float64(156.71058295249142),
    np.float64(170.82314410997503),
    np.float64(20.006064279667868),
    np.float64(197.64821767958674),
    np.float64(12.96532816873102),
    np.float64(183.48881081616923),
    np.float64(197.2232169500448),
    np.float64(1.0692798085998163),
    np.float64(150.716802294599),
    np.float64(216.96073601667607),
    np.float64(141.10112227765285),
    np.float64(205.67010381628262),
    np.float64(67.94421837245665),
    np.float64(114.46030731179096),
    np.float64(119.84785017833053),
    np.float64(99.17446092463538),
    np.float64(7.07563419334474),
    np.float64(136.56984386026477),
    np.float64(122.57287100109465),
    np.float64(172.85127662677422),
    np.float64(174.12940619963223),
    np.float64(24.628010124427732),
    np.float64(203.39822616994925),
    np.float64(81.25061826065382),
    np.float64(205.56385370146262),
    np.float64(155.4293266335837),
    np.float64(210.23260791894256),
    np.float64(110.47277413173441),
    np.float64(204.20760215873187),
    np.float64(216.18261083834895),
    np.float64(20.63419900515254),
    np.float64(120.41347954795532),
    np.float64(48.28772177374043),
    np.float64(131.41980962473366),
    np.float64(215.2669855682385),
    np.float64(98.79766987717255),
    np.float64(174.21378149704012),
    np.float64(109.37276476112197),
    np.float64(206.8888550705766)]
```

In [26]:

```
# getting our p-val for fairness
p_val=(np.array(diff_in_acc) >= obs).mean()
```

```
p_val
```

Out[26]:

```
np.float64(0.0)
```

In [ ]:

```

```

In [ ]:

```

```