CS146 - LBA

Data Processing

Load data and change column name to make it more readable

In [1]:
```
import pandas as pd

df = pd.read_csv("CS146-LBA-data.csv")

df = df.drop(['Timestamp', 'Your name', 'Email Address'], axis=1)

df.rename(columns={'Grocery store brand':'store_brand',
                   'Grocery store street address': 'address',

                   'Product 1 brand': 'apple_brand_1',
                   'Product 1 price (€)': 'apple_price_1',
                   'Product 2 brand': 'apple_brand_2',
                   'Product 2 price (€)': 'apple_price_2',
                   'Product 3 brand': 'apple_brand_3',
                   'Product 3 price (€)': 'apple_price_3',

                   'Product 1 brand.1': 'banana_brand_1',
                   'Product 1 price (€).1': 'banana_price_1',
                   'Product 2 brand.1': 'banana_brand_2',
                   'Product 2 price (€).1': 'banana_price_2',
                   'Product 3 brand.1': 'banana_brand_3',
                   'Product 3 price (€).1': 'banana_price_3',

                   'Product 1 brand.2': 'tomato_brand_1',
                   'Product 1 price (€).2': 'tomato_price_1',
                   'Product 2 brand.2': 'tomato_brand_2',
                   'Product 2 price (€).2': 'tomato_price_2',
                   'Product 3 brand.2': 'tomato_brand_3',
                   'Product 3 price (€).2': 'tomato_price_3',

                   'Product 1 brand.3': 'potato_brand_1',
                   'Product 1 price (€).3': 'potato_price_1',
                   'Product 2 brand.3': 'potato_brand_2',
                   'Product 2 price (€).3': 'potato_price_2',
                   'Product 3 brand.3': 'potato_brand_3',
                   'Product 3 price (€).3': 'potato_price_3',

                   'Product 1 brand.4': 'flour_brand_1',
                   'Product 1 price (€).4': 'flour_price_1',
                   'Product 2 brand.4': 'flour_brand_2',
                   'Product 2 price (€).4': 'flour_price_2',
                   'Product 3 brand.4': 'flour_brand_3',
                   'Product 3 price (€).4': 'flour_price_3',

                   'Product 1 brand.5': 'rice_brand_1',
                   'Product 1 price (€).5': 'rice_price_1',
                   'Product 2 brand.5': 'rice_brand_2',
                   'Product 2 price (€).5': 'rice_price_2',
                   'Product 3 brand.5': 'rice_brand_3',
                   'Product 3 price (€).5': 'rice_price_3',

                   'Product 1 brand.6': 'milk_brand_1',
                   'Product 1 price (€).6': 'milk_price_1',
                   'Product 2 brand.6': 'milk_brand_2',
                   'Product 2 price (€).6': 'milk_price_2',
                   'Product 3 brand.6': 'milk_brand_3',
```

```
                      'Product 3 price (€).6': 'milk_price_3',

                      'Product 1 brand.7': 'butter_brand_1',
                      'Product 1 price (€).7': 'butter_price_1',
                      'Product 2 brand.7': 'butter_brand_2',
                      'Product 2 price (€).7': 'butter_price_2',
                      'Product 3 brand.7': 'butter_brand_3',
                      'Product 3 price (€).7': 'butter_price_3',

                      'Product 1 brand.8': 'egg_brand_1',
                      'Product 1 price (€).8': 'egg_price_1',
                      'Product 2 brand.8': 'egg_brand_2',
                      'Product 2 price (€).8': 'egg_price_2',
                      'Product 3 brand.8': 'egg_brand_3',
                      'Product 3 price (€).8': 'egg_price_3',

                      'Product 1 brand.9': 'chicken_breast_brand_1',
                      'Product 1 price (€).9': 'chicken_breast_price_1',
                      'Product 2 brand.9': 'chicken_breast_brand_2',
                      'Product 2 price (€).9': 'chicken_breast_price_2',
                      'Product 3 brand.9': 'chicken_breast_brand_3',
                      'Product 3 price (€).9': 'chicken_breast_price_3',
                     }, inplace=True)

item_list = ['apple', 'banana', 'tomato', 'potato', 'flour',
             'rice', 'milk', 'butter', 'egg', 'chicken_breast']
```

Map store address to neighborhood

In [2]:
```
address_book = pd.concat([pd.read_csv("CS146-Berlin-supermarkets.csv"),
                          pd.read_csv("CS146-London-supermarkets.csv")])

df['neighborhood'] = None

for _, entry in address_book.iterrows():
    df.loc[df['address'].str.contains(entry['Supermarket'].split(",")[-1
].lstrip()), 'neighborhood'] \
    = entry['Neighborhood']

print ("Number of data entry missing neighborhood assignment:", df['neig
hborhood'].isnull().sum())
print ("")
print (df[df['neighborhood'].isnull()][['store_brand', 'address', 'neigh
borhood']])
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:7: UserWar
ning: This pattern has match groups. To actually get the groups, use st
r.extract.
  import sys

Number of data entry missing neighborhood assignment: 7

   store_brand                               address neighborhood
8         ALDI  Karl-Marx Straße 231, 12055 Berlin           None
13       EDEKA                      EDEKA Annenstraße         None
21        Lidl                      Glasgower str. 42         None
25        ALDI            ALDI, Frankfurter Allee 117         None
46        ALDI                          Kiefholzstraße         None
62        ALDI  Waterfront, Cape Town, South Africa         None
63        ALDI                Cape Town, South Africa         None
```

Handel unmapped enty manually

In [3]:
```
df.loc[df['address'] == "Karl-Marx Straße 231, 12055 Berlin ", 'neighbor
hood'] = "Neukölln"
df.loc[df['address'] == "EDEKA Annenstraße", 'neighborhood'] = "Mitte"
df.loc[df['address'] == "Glasgower str. 42", 'neighborhood'] = "Neuköll
n"
df.loc[df['address'] == "ALDI, Frankfurter Allee 117", 'neighborhood'] =
 "Friedrichshain"
df.loc[(df['store_brand'] == "ALDI") & (df['address'] == "Kiefholzstraß
e"), 'neighborhood'] = "Alt-Treptow"
df.loc[df['address'] == "Waterfront, Cape Town, South Africa", 'neighbor
hood'] = "Cape Town"
df.loc[df['address'] == "Cape Town, South Africa", 'neighborhood'] = "Ca
pe Town"

print ("Number of data entry missing neighborhood assignment:", df['neig
hborhood'].isnull().sum())
```

```
Number of data entry missing neighborhood assignment: 0
```

Build dataset into dictionary

In [4]:
```
import math
import re
import numpy as np

data = dict()

for item in item_list:
    item_data = []
    for _, entry in df.iterrows():
        for i in range(1, 4):
            brand_string = "{:s}_brand_{:d}".format(item, i)
            price_string = "{:s}_price_{:d}".format(item, i)
            if isinstance(entry[brand_string], str) \
            and not re.match(r".*[O,o]nly.*", entry[brand_string]) and n
ot math.isnan(entry[price_string]):
                item_data.append([entry[brand_string], entry[price_strin
g],
                                          entry['store_brand'], entry['neighborh
ood']])

    data[item] = item_data
```

Handle different representation for 'no brand' and convert array data into pandas df

In [5]:
```
import re
import pandas as pd

for key, value_set in data.items():
    no_brand_count = 0
    for value in value_set:
        if re.match(r".*[N,n]o[ ,-][B,b]rand.*", value[0]) or re.match(r
".*[B,b]rand [U,u]nknown.*", value[0])\
        or re.match(r".*[N,n]ormal.*", value[0]) or re.match(r"-", value
[0]):
            value[0] = "no_brand"
            no_brand_count += 1
    data[key] = pd.DataFrame(value_set, columns=['brand', 'price', 'stor
e', 'location'])
    print (len(value_set), "data entry for", key, "with", no_brand_count
, "sold with no brand")
```

```
225 data entry for apple with 37 sold with no brand
180 data entry for banana with 46 sold with no brand
208 data entry for tomato with 49 sold with no brand
202 data entry for potato with 44 sold with no brand
176 data entry for flour with 11 sold with no brand
190 data entry for rice with 8 sold with no brand
209 data entry for milk with 7 sold with no brand
220 data entry for butter with 5 sold with no brand
214 data entry for egg with 27 sold with no brand
144 data entry for chicken_breast with 16 sold with no brand
```

Use KNN to cluster brand name

In [33]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score
import numpy as np

np.random.seed(5)
brand_cluster_dict = dict()

for key, df in data.items():
    brand_name = np.array(df['brand'])

    # build knn cluster model using all text in the rejected set
    vectorizer = TfidfVectorizer(stop_words='english')
    X = vectorizer.fit_transform(brand_name)
    true_k = 10
    knn_cluster_model = KMeans(n_clusters=true_k, init='k-means++', max_
iter=100, n_init=1)
    knn_cluster_model.fit(X)
    brand_mapping = knn_cluster_model.predict(X)

    # use the most common word as the keyword of each brand cluster
    order_centroids = knn_cluster_model.cluster_centers_.argsort()[:, ::
-1]
    terms = vectorizer.get_feature_names()
    brand_cluster_name = [terms[index] for index in order_centroids[:, 0
]]
    brand_cluster_dict[key] = brand_cluster_name

    df['brand'] = [brand_cluster_name[i] for i in brand_mapping]
```

Calculate simple average price for each product

In [7]:
```
for key, df in data.items():
    print ("simple average price for {:s} is €{:.3f}".format(key, df['pr
ice'].mean()))
```

```
simple average price for apple is €2.283
simple average price for banana is €1.451
simple average price for tomato is €3.459
simple average price for potato is €1.365
simple average price for flour is €1.059
simple average price for rice is €2.361
simple average price for milk is €1.047
simple average price for butter is €4.079
simple average price for egg is €2.581
simple average price for chicken_breast is €10.178
```

Build stan code

In [8]:
```
import pystan

stan_code = """
    data {
        // Y
        int<lower=0> n;
        real<lower=0> price[n];

        // X
        int<lower=0> b;
        vector[b] brand[n];
        int<lower=0> s;
        vector[s] store[n];
        int<lower=0> l;
        vector[l] location[n];
    }
    parameters {
        // base
        real<lower=0> baseSigma;
        real<lower=0> mu;
        real<lower=0> base;

        // multiplier
        row_vector<lower=0>[b] brandMultiplier;
        row_vector<lower=0>[s] storeMultiplier;
        row_vector<lower=0>[l] locationMultiplier;

        // model
        real<lower=0> modelSigma;
    }
    model {
        baseSigma ~ inv_gamma(1, 1);
        mu ~ cauchy(0,1);
        base ~ normal(mu, sqrt(baseSigma));

        brandMultiplier ~ lognormal(0, 1);
        storeMultiplier ~ lognormal(0, 1);
        locationMultiplier ~ lognormal(0, 1);

        modelSigma ~ gamma(1, 1);

        for(i in 1:n) {
            price[i] ~ normal(base * (brandMultiplier * brand[i]) * (sto
reMultiplier * store[i])
            * (locationMultiplier * location[i]), sqrt(modelSigma));
        }
    }
"""

stan_model = pystan.StanModel(model_code=stan_code)
```

```
INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_bca1181cf8c1049
1650d358698362862 NOW.
/usr/local/lib/python3.7/site-packages/Cython/Compiler/Main.py:367: Fut
ureWarning: Cython directive 'language_level' not set, using 2 for now
(Py2). This will change in a later release! File: /var/folders/k8/dkych
j2n5c98xy85t13b1md40000gn/T/tmpmmqt6n0q/stanfit4anon_model_bca1181cf8c1
0491650d358698362862_5634475610633302309.pyx
  tree = Parsing.p_module(s, pxd, full_module_name)
```

Run stan model

In [10]:

```
stan_model_result = dict()

for key, df in .items():
    price = np.array(df['price']).tolist()
    n = len(price)
    brand = np.array(pd.get_dummies(df['brand']), dtype=int)
    b = 10
    store = np.array(pd.get_dummies(df['store']), dtype=int)
    s = np.unique(np.array((df['store']))).size
    location = np.array(pd.get_dummies(df['location']), dtype=int)
    l = np.unique(np.array((df['location']))).size

    stan_data = {
        'n': n,
        'price': price,
        'b': b,
        'brand': brand,
        's': s,
        'store': store,
        'l': l,
        'location': location,
    }

    stan_model_result[key] = stan_model.sampling(data=stan_data)
    print ("Completed stan model for", key)
```

```
Completed stan model for apple
Completed stan model for banana
Completed stan model for tomato
Completed stan model for potato
Completed stan model for flour
Completed stan model for rice
Completed stan model for milk
Completed stan model for butter
Completed stan model for egg

WARNING:pystan:20 of 4000 iterations ended with a divergence (0.5%).
WARNING:pystan:Try running with adapt_delta larger than 0.8 to remove t
he divergences.

Completed stan model for chicken_breast
```

Display result

In [62]:
```
for key, result in stan_model_result.items():
    r = result.extract()
    print ("Posterior result for", key)
    print ("Base price mean at {:.3f} with 95% interval ({:.2f}, {:.2f}
)"
            .format(np.mean(r['base']), np.percentile(r['base'], 2.5), np
.percentile(r['base'], 97.5)))
    print ("")
    for i in range(10):
        print ("Mean multiplier for brand {:s} is {:.3f}"
               .format(brand_cluster_dict[key][i], np.mean(r['brandMulti
plier'][:,i])))
    print ("")
    s = np.unique(np.array((data[key]['store']))).size
    s_name = pd.get_dummies(data[key]['store']).columns.values
    for i in range(s):
        print ("Mean multiplier for store {:s} is {:.3f}"
               .format(s_name[i], np.mean(r['storeMultiplier'][:,i])))
    print ("")
    l = np.unique(np.array((data[key]['location']))).size
    l_name = pd.get_dummies(data[key]['location']).columns.values
    for i in range(l):
        print ("Mean multiplier for location {:s} is {:.3f}"
               .format(l_name[i], np.mean(r['locationMultiplier'][:,i
])))
    print ("\n")
```

```
Posterior result for apple
Base price mean at 2.245 with 95% interval (0.65, 5.49)

Mean multiplier for brand braeburn is 1.730
Mean multiplier for brand pink is 1.104
Mean multiplier for brand no_brand is 1.009
Mean multiplier for brand rewe is 0.886
Mean multiplier for brand gala is 0.892
Mean multiplier for brand bio is 1.054
Mean multiplier for brand granny is 0.778
Mean multiplier for brand kanzi is 1.496
Mean multiplier for brand wahl is 1.108
Mean multiplier for brand oaklands is 0.838

Mean multiplier for store ALDI is 1.130
Mean multiplier for store EDEKA is 1.095
Mean multiplier for store Lidl is 1.005
Mean multiplier for store REWE is 1.143

Mean multiplier for location Alt-Treptow is 1.101
Mean multiplier for location Cape Town is 0.715
Mean multiplier for location Friedrichshain is 1.080
Mean multiplier for location Kreuzberg is 1.195
Mean multiplier for location Lichtenberg is 1.009
Mean multiplier for location London is 1.326
Mean multiplier for location Mitte is 1.114
Mean multiplier for location Neukölln is 1.232
Mean multiplier for location Prenzlauer Berg is 1.100
Mean multiplier for location Schöneberg is 1.157
Mean multiplier for location Tempelhof is 0.864


Posterior result for banana
Base price mean at 1.835 with 95% interval (0.49, 4.62)

Mean multiplier for brand rewe is 1.106
Mean multiplier for brand no_brand is 1.167
Mean multiplier for brand bananen is 1.255
Mean multiplier for brand edeka is 1.082
Mean multiplier for brand gut is 0.773
Mean multiplier for brand bio is 1.222
Mean multiplier for brand chiquita is 1.007
Mean multiplier for brand oaklands is 0.757
Mean multiplier for brand lidl is 1.010
Mean multiplier for brand gutbio is 1.049

Mean multiplier for store ALDI is 1.027
Mean multiplier for store EDEKA is 1.226
Mean multiplier for store Lidl is 0.994
Mean multiplier for store REWE is 1.056

Mean multiplier for location Alt-Treptow is 0.922
Mean multiplier for location Cape Town is 1.172
Mean multiplier for location Friedrichshain is 1.103
Mean multiplier for location Kreuzberg is 1.148
Mean multiplier for location Lichtenberg is 1.210
Mean multiplier for location London is 0.845
```

```
Mean multiplier for location Mitte is 1.035
Mean multiplier for location Neukölln is 1.009
Mean multiplier for location Prenzlauer Berg is 1.031
Mean multiplier for location Schöneberg is 1.150
Mean multiplier for location Tempelhof is 0.892


Posterior result for tomato
Base price mean at 2.997 with 95% interval (0.90, 7.90)

Mean multiplier for brand rispentomaten is 1.168
Mean multiplier for brand tomatoes is 1.393
Mean multiplier for brand no_brand is 1.412
Mean multiplier for brand bio is 1.129
Mean multiplier for brand rewe is 1.164
Mean multiplier for brand strauchtomaten is 0.676
Mean multiplier for brand edeka is 0.975
Mean multiplier for brand cherry is 0.958
Mean multiplier for brand gutbio is 1.288
Mean multiplier for brand mini is 1.078

Mean multiplier for store ALDI is 1.296
Mean multiplier for store EDEKA is 1.193
Mean multiplier for store Lidl is 1.165
Mean multiplier for store REWE is 1.258

Mean multiplier for location Alt-Treptow is 1.078
Mean multiplier for location Cape Town is 0.893
Mean multiplier for location Friedrichshain is 1.412
Mean multiplier for location Kreuzberg is 1.525
Mean multiplier for location Lichtenberg is 0.789
Mean multiplier for location London is 1.306
Mean multiplier for location Mitte is 1.346
Mean multiplier for location Neukölln is 1.102
Mean multiplier for location Prenzlauer Berg is 1.242
Mean multiplier for location Schöneberg is 1.168
Mean multiplier for location Tempelhof is 0.650


Posterior result for potato
Base price mean at 1.637 with 95% interval (0.47, 3.97)

Mean multiplier for brand festkochend is 1.115
Mean multiplier for brand speisekartoffeln is 1.334
Mean multiplier for brand no_brand is 1.295
Mean multiplier for brand rewe is 0.992
Mean multiplier for brand kartoffeln is 0.822
Mean multiplier for brand potatoes is 1.099
Mean multiplier for brand bio is 0.669
Mean multiplier for brand beste is 1.418
Mean multiplier for brand oaklands is 1.074
Mean multiplier for brand gutbio is 0.970

Mean multiplier for store ALDI is 0.967
Mean multiplier for store EDEKA is 1.110
Mean multiplier for store Lidl is 1.052
Mean multiplier for store REWE is 1.092
```

```
Mean multiplier for location Alt-Treptow is 1.193
Mean multiplier for location Cape Town is 0.835
Mean multiplier for location Friedrichshain is 0.831
Mean multiplier for location Kreuzberg is 0.778
Mean multiplier for location Lichtenberg is 1.106
Mean multiplier for location London is 0.706
Mean multiplier for location Mitte is 0.904
Mean multiplier for location Neukölln is 1.344
Mean multiplier for location Prenzlauer Berg is 0.921
Mean multiplier for location Schöneberg is 1.089
Mean multiplier for location Tempelhof is 2.035


Posterior result for flour
Base price mean at 1.064 with 95% interval (0.27, 2.65)

Mean multiplier for brand no_brand is 1.556
Mean multiplier for brand belbake is 0.945
Mean multiplier for brand aurora is 1.141
Mean multiplier for brand bio is 0.644
Mean multiplier for brand gut is 0.538
Mean multiplier for brand weizenmehl is 1.134
Mean multiplier for brand kathi is 1.318
Mean multiplier for brand ja is 1.580
Mean multiplier for brand wurzener is 1.354
Mean multiplier for brand roggenmehl is 0.729

Mean multiplier for store ALDI is 0.760
Mean multiplier for store EDEKA is 1.047
Mean multiplier for store Lidl is 0.810
Mean multiplier for store REWE is 1.359

Mean multiplier for location Alt-Treptow is 1.210
Mean multiplier for location Cape Town is 1.090
Mean multiplier for location Friedrichshain is 0.965
Mean multiplier for location Kreuzberg is 1.598
Mean multiplier for location Lichtenberg is 0.921
Mean multiplier for location London is 0.794
Mean multiplier for location Mitte is 1.295
Mean multiplier for location Neukölln is 1.482
Mean multiplier for location Prenzlauer Berg is 1.016
Mean multiplier for location Schöneberg is 0.785
Mean multiplier for location Tempelhof is 0.781


Posterior result for rice
Base price mean at 2.007 with 95% interval (0.57, 5.16)

Mean multiplier for brand oryza is 1.378
Mean multiplier for brand bon is 1.067
Mean multiplier for brand golden is 0.961
Mean multiplier for brand gut is 1.404
Mean multiplier for brand edeka is 0.601
Mean multiplier for brand uncle is 0.652
Mean multiplier for brand basmati is 1.041
Mean multiplier for brand wurzener is 1.686
```

```
Mean multiplier for brand no_brand is 1.709
Mean multiplier for brand ja is 0.786

Mean multiplier for store ALDI is 0.793
Mean multiplier for store EDEKA is 1.660
Mean multiplier for store Lidl is 0.692
Mean multiplier for store REWE is 1.521

Mean multiplier for location Alt-Treptow is 0.872
Mean multiplier for location Cape Town is 0.822
Mean multiplier for location Friedrichshain is 1.123
Mean multiplier for location Kreuzberg is 0.946
Mean multiplier for location Lichtenberg is 1.092
Mean multiplier for location London is 1.135
Mean multiplier for location Mitte is 1.328
Mean multiplier for location Neukölln is 0.944
Mean multiplier for location Prenzlauer Berg is 1.267
Mean multiplier for location Schöneberg is 1.292
Mean multiplier for location Tempelhof is 1.189


Posterior result for milk
Base price mean at 1.371 with 95% interval (0.38, 3.51)

Mean multiplier for brand ja is 1.011
Mean multiplier for brand gut is 0.946
Mean multiplier for brand meierkamp is 0.653
Mean multiplier for brand milbona is 1.240
Mean multiplier for brand milch is 1.285
Mean multiplier for brand edeka is 0.900
Mean multiplier for brand milsani is 1.111
Mean multiplier for brand marke is 0.827
Mean multiplier for brand no_brand is 0.930
Mean multiplier for brand weihenstephan is 1.303

Mean multiplier for store ALDI is 0.858
Mean multiplier for store EDEKA is 1.031
Mean multiplier for store Lidl is 0.994
Mean multiplier for store REWE is 1.052

Mean multiplier for location Alt-Treptow is 0.899
Mean multiplier for location Cape Town is 1.985
Mean multiplier for location Friedrichshain is 0.946
Mean multiplier for location Kreuzberg is 0.969
Mean multiplier for location Lichtenberg is 0.895
Mean multiplier for location London is 0.760
Mean multiplier for location Mitte is 0.989
Mean multiplier for location Neukölln is 0.966
Mean multiplier for location Prenzlauer Berg is 1.099
Mean multiplier for location Schöneberg is 0.993
Mean multiplier for location Tempelhof is 0.838


Posterior result for butter
Base price mean at 3.556 with 95% interval (0.99, 9.65)

Mean multiplier for brand gold is 1.308
```

```
Mean multiplier for brand butter is 1.125
Mean multiplier for brand kerrygold is 1.137
Mean multiplier for brand gut is 1.359
Mean multiplier for brand edeka is 0.952
Mean multiplier for brand milsani is 0.895
Mean multiplier for brand manor is 1.255
Mean multiplier for brand arla is 1.116
Mean multiplier for brand ja is 0.844
Mean multiplier for brand ungesalzen is 1.389

Mean multiplier for store ALDI is 1.498
Mean multiplier for store EDEKA is 1.199
Mean multiplier for store Lidl is 1.181
Mean multiplier for store REWE is 1.053

Mean multiplier for location Alt-Treptow is 1.130
Mean multiplier for location Cape Town is 1.346
Mean multiplier for location Friedrichshain is 1.285
Mean multiplier for location Kreuzberg is 1.504
Mean multiplier for location Lichtenberg is 1.146
Mean multiplier for location London is 0.776
Mean multiplier for location Mitte is 1.234
Mean multiplier for location Neukölln is 1.053
Mean multiplier for location Prenzlauer Berg is 1.027
Mean multiplier for location Schöneberg is 1.046
Mean multiplier for location Tempelhof is 0.831


Posterior result for egg
Base price mean at 2.435 with 95% interval (0.74, 5.94)

Mean multiplier for brand gut is 1.387
Mean multiplier for brand bio is 0.879
Mean multiplier for brand no_brand is 2.468
Mean multiplier for brand luisenhof is 1.558
Mean multiplier for brand bodenhaltung is 0.698
Mean multiplier for brand edeka is 0.847
Mean multiplier for brand simply is 0.689
Mean multiplier for brand hofland is 0.999
Mean multiplier for brand ja is 1.118
Mean multiplier for brand demeter is 0.812

Mean multiplier for store ALDI is 1.014
Mean multiplier for store EDEKA is 1.317
Mean multiplier for store Lidl is 0.928
Mean multiplier for store REWE is 1.215

Mean multiplier for location Alt-Treptow is 1.052
Mean multiplier for location Cape Town is 1.089
Mean multiplier for location Friedrichshain is 1.225
Mean multiplier for location Kreuzberg is 1.241
Mean multiplier for location Lichtenberg is 1.139
Mean multiplier for location London is 0.834
Mean multiplier for location Mitte is 1.193
Mean multiplier for location Neukölln is 1.041
Mean multiplier for location Prenzlauer Berg is 1.204
Mean multiplier for location Schöneberg is 1.184
```

```
Mean multiplier for location Tempelhof is 0.803


Posterior result for chicken_breast
Base price mean at 7.460 with 95% interval (1.97, 20.95)

Mean multiplier for brand stolle is 2.093
Mean multiplier for brand gut is 1.253
Mean multiplier for brand metzgerei is 1.066
Mean multiplier for brand no_brand is 0.872
Mean multiplier for brand birchwood is 1.106
Mean multiplier for brand rewe is 1.463
Mean multiplier for brand landjunker is 0.474
Mean multiplier for brand friki is 1.160
Mean multiplier for brand gutbio is 2.435
Mean multiplier for brand biofino is 0.821

Mean multiplier for store ALDI is 1.822
Mean multiplier for store EDEKA is 2.058
Mean multiplier for store Lidl is 0.747
Mean multiplier for store REWE is 1.454

Mean multiplier for location Alt-Treptow is 1.669
Mean multiplier for location Cape Town is 0.530
Mean multiplier for location Friedrichshain is 1.123
Mean multiplier for location Kreuzberg is 1.219
Mean multiplier for location Lichtenberg is 1.704
Mean multiplier for location London is 1.557
Mean multiplier for location Mitte is 1.233
Mean multiplier for location Neukölln is 0.922
Mean multiplier for location Prenzlauer Berg is 0.780
Mean multiplier for location Schöneberg is 1.209
Mean multiplier for location Tempelhof is 1.531
```