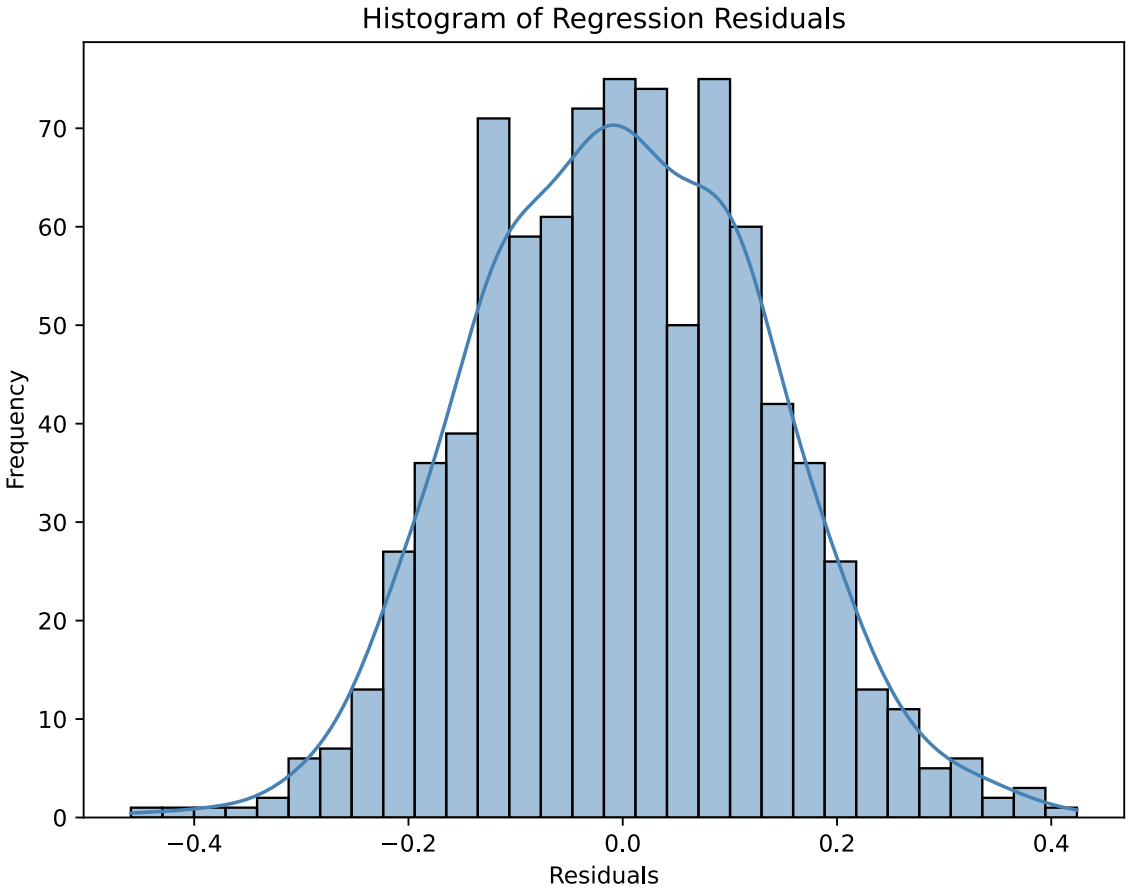
Exercise 10

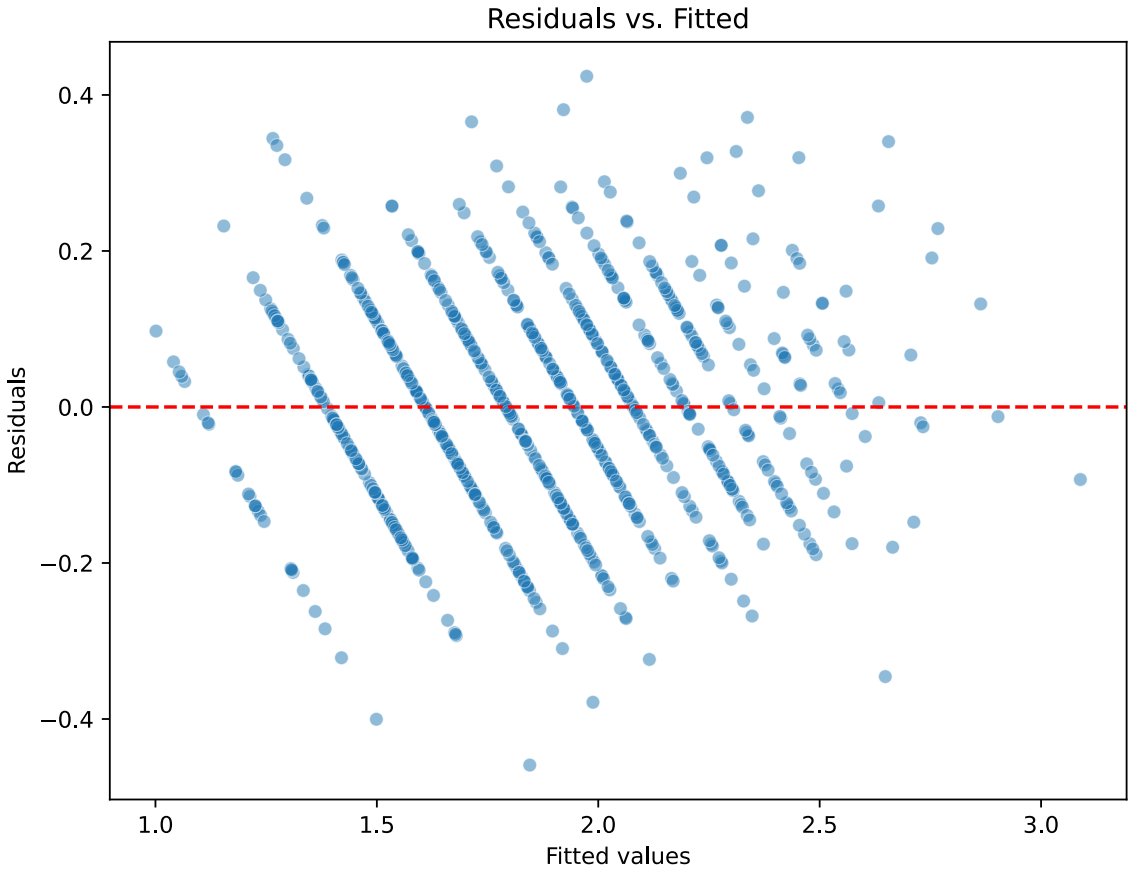
```
import os
import polars as pl
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
import bambi as bmb
import arviz as az
import numpy as np
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Load data
sl_data = pl.read_parquet(os.path.join('data', 'soft_launch.parquet'))
loyal_data = pl.read_parquet(os.path.join('data', 'loyalty.parquet'))

# Standardize brand names
brand_names = {
    'JIF': 'Jif', 'Jiff': 'Jif',
    'SKIPPY': 'Skippy', 'Skipy': 'Skippy', 'Skipp': 'Skippy',
    'Peter Pan': 'PeterPan', 'Peter Pa': 'PeterPan',
    "Harmon's": 'Harmons',
}

pb_data = (
    sl_data
    .with_columns([
        pl.col('brand').cast(pl.Utf8).replace(brand_names).alias('brand')
    ])
    .drop_nans(['units', 'sales'])
    .remove(
        (pl.col('brand') == "None") | (pl.col('loyal') == "None") |
        (pl.col('texture') == "None") | (pl.col('size') == "None")
    )
    .with_columns([
        pl.col('price').cast(pl.Float64).alias('price'),
        pl.col('loyal').cast(pl.Int64).alias('loyal'),
        pl.col('promo').cast(pl.Int64).alias('promo')
    ])
    .unique()
    .filter(pl.col('units') > 0)
    .select('customer_id', 'units', 'brand', 'coupon', 'ad', 'texture',
'size', 'price')
```

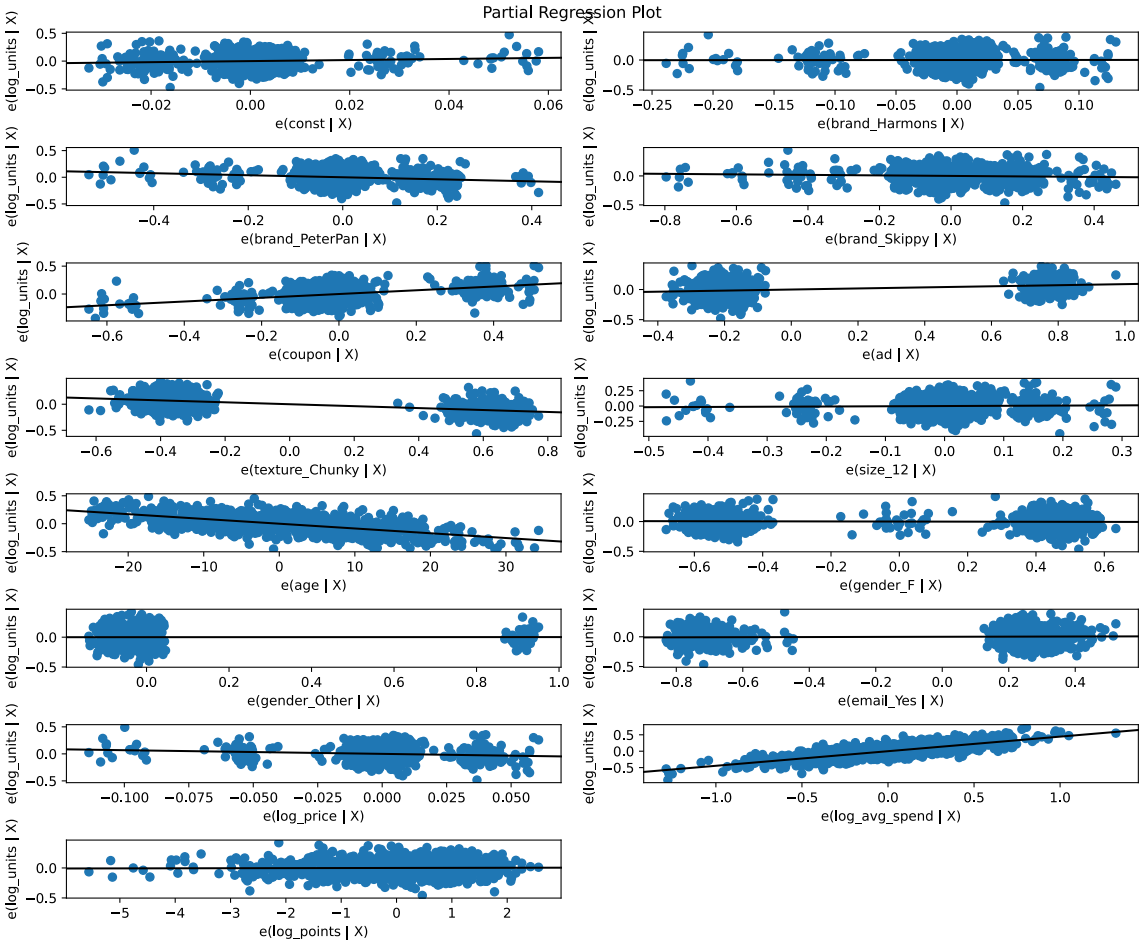
```
)

loyal_data = (
    loyal_data
    .drop(['loyal', 'units'])
    .unique() #One duplicate customer_id
)

#Combine pb data and loyal data
pbloyal_data = pb_data.join(loyal_data, on="customer_id", how="inner")
pbloyal_data = pbloyal_data.drop("customer_id")

# Design matrix and outcome
X = pbloyal_data.select(pl.exclude('units'))
y = pbloyal_data.select('units')

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Transform y
y_train = (
    y_train
    .with_columns((pl.col('units') + 1).log().alias('log_units'))
    .select('log_units')
    .to_pandas()
)
y_train = y_train['log_units']   # convert to Series

# Transform X
X_train = (
    X_train
    .with_columns((pl.col('price') + 1).log().alias('log_price'))
    .with_columns((pl.col('avg_spend') + 1).log().alias('log_avg_spend'))
    .with_columns((pl.col('points') + 1).log().alias('log_points'))
    .to_dummies(columns=['brand', 'texture', 'size', 'gender', 'email'],
drop_first=False)
    .select(pl.exclude('price', 'avg_spend', 'points', 'brand_Jif',
'texture_Smooth', 'size_16', 'gender_M', 'email_No'))
    .to_pandas()
)

# Add constant
X_train = sm.add_constant(X_train)

# Fit model
fit_01 = sm.OLS(y_train, X_train).fit()
```

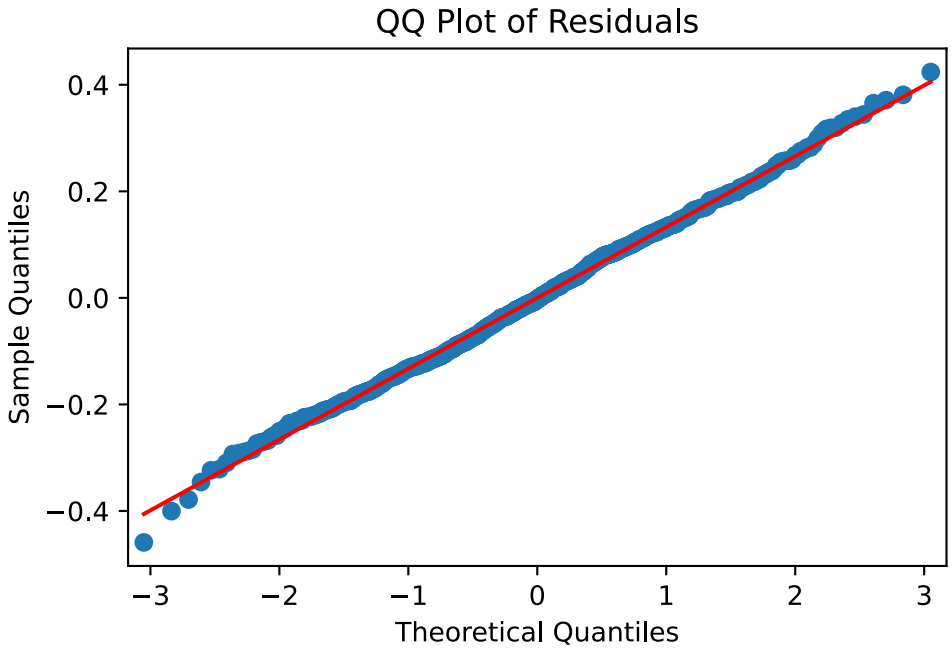
```
# Get fitted values and residuals
fitted = fit_01.fittedvalues
residuals = fit_01.resid

# Histogram of Residuals
plt.figure(figsize=(8,6))
sns.histplot(residuals, kde=True, bins=30, color="steelblue")

plt.title("Histogram of Regression Residuals")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.show()

plt.figure(figsize=(8,6))
sns.scatterplot(x=fitted, y=residuals, alpha=0.5)

# Add horizontal line at 0
plt.axhline(0, color="red", linestyle="--")

# Labels and title
plt.xlabel("Fitted values")
plt.ylabel("Residuals")
plt.title("Residuals vs. Fitted")
plt.show()

"""# Save the dffits values
X_train['dffits'] = fit_01.get_influence().dffits[0]

fig = plt.figure(figsize = (4, 4))
plt.ylabel("DFFITS (Absolute Values)")
plt.xlabel("Observation Number")
plt.scatter(
  X_train.index,
  np.abs(X_train['dffits']),
  s = 3
)
plt.axhline(
  y = 2 * np.sqrt(len(fit_01.params) / len(X_train)),
  color = 'r',
  linestyle = 'dashed'
)
plt.show()""" #This plots correctly but throws an error that prevents the
other plots from loading

#Partial regression plot
fig = sm.graphics.plot_partregress_grid(fit_01, fig=plt.figure(figsize=(12,
10)))
```

```
fig.tight_layout(pad=1.0)
plt.show()

#QQ Plot
sm.qqplot(fit_01.resid, line='s')
plt.title('QQ Plot of Residuals')
plt.show()

# Calculate VIF for each predictor
vif_data = pd.DataFrame({
    "feature": X_train.columns,
    "VIF": [variance_inflation_factor(X_train.values, i)
            for i in range(len(X_train.columns))]
})

# Sort by VIF
vif_data = vif_data.sort_values(by="VIF", ascending=False)
print(vif_data)
```




Residuals vs. Fitted
Residuals
Fitted values
0.4
0.2
0.0
−0.2
−0.4
1.0
1.5
2.0
2.5
3.0


Partial Regression Plot
e(log_units | X)
e(const | X)
e(brand_Harmons | X)
e(brand_PeterPan | X)
e(brand_Skippy | X)
e(coupon | X)
e(ad | X)
e(texture_Chunky | X)
e(size_12 | X)
e(age | X)
e(gender_F | X)
e(gender_Other | X)
e(email_Yes | X)
e(log_price | X)
e(log_avg_spend | X)
e(log_points | X)



	feature	VIF
0	const	6517.788366
12	log_price	104.991965
1	brand_Harmons	90.464005
7	size_12	19.510049
4	coupon	6.348390
3	brand_Skippy	4.849947
2	brand_PeterPan	4.707939
9	gender_F	1.059636
10	gender_Other	1.054915
11	email_Yes	1.022872
5	ad	1.016537
6	texture_Chunky	1.016422
14	log_points	1.015901
13	log_avg_spend	1.011728
8	age	1.005256

We did all the same transforms that were in the previous pb dataset (log transform price; removing loyalty, promo, and sales; normalizing brand names; etc.). With the loyalty dataset, we removed the loyal column since it was a constant value, and we removed the units and customer_id columns since they were duplicates from the other dataset. We did a log transform on avg_spend and points, and we created dummy variables for gender and email. The histogram of residuals for the combined dataset appears more normal than the standalone pb dataset. Additionally, the QQ plot is fitted more to the line and the plot of residuals and fitted values still has a clouded shape. The DFFITS and partial regression plot revealed outliers, but these were naturally occurring values (i.e. some customers spend a lot and have a lot of points), so they were not removed.

```
#Repeating the transforms from X_train and y_train onto X_test and y_test
# Transform y
y_test = (
    y_test
    .with_columns((pl.col('units') + 1).log().alias('log_units'))
    .select('log_units')
    .to_pandas()
)
y_test = y_test['log_units']   # convert to Series

# Transform X
X_test = (
    X_test
    .with_columns((pl.col('price') + 1).log().alias('log_price'))
    .with_columns((pl.col('avg_spend') + 1).log().alias('log_avg_spend'))
    .with_columns((pl.col('points') + 1).log().alias('log_points'))
    .to_dummies(columns=['brand', 'texture', 'size', 'gender', 'email'],
drop_first=False)
    .select(pl.exclude('price', 'avg_spend', 'points', 'brand_Jif',
'texture_Smooth', 'size_16', 'gender_M', 'email_No'))
    .to_pandas()
)

# Add constant
X_test = sm.add_constant(X_test)

# Fit model
fit_02 = sm.OLS(y_test, X_test).fit()
print(fit_02.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              log_units   R-squared:                       0.869
Model:                            OLS   Adj. R-squared:                  0.860
Method:                 Least Squares   F-statistic:                     96.79
Date:                Tue, 30 Sep 2025   Prob (F-statistic):           6.80e-82
Time:                        18:15:53   Log-Likelihood:                140.72
No. Observations:                 219   AIC:                            -251.4
Df Residuals:                     204   BIC:                            -200.6
Df Model:                          14
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
const              1.8019      0.864      2.086      0.038       0.099
3.505
```

```
brand_Harmons     -0.1933      0.214     -0.904      0.367      -0.615
0.228
brand_PeterPan    -0.2258      0.088     -2.569      0.011      -0.399
-0.052
brand_Skippy      -0.0882      0.057     -1.536      0.126      -0.201
0.025
coupon             0.2769      0.057      4.828      0.000       0.164
0.390
ad                 0.0731      0.022      3.295      0.001       0.029
0.117
texture_Chunky    -0.1611      0.019     -8.422      0.000      -0.199
-0.123
size_12           -0.0715      0.101     -0.707      0.480      -0.271
0.128
age               -0.0076      0.001    -10.529      0.000      -0.009
-0.006
gender_F           0.0230      0.019      1.227      0.221      -0.014
0.060
gender_Other       0.0850      0.050      1.707      0.089      -0.013
0.183
email_Yes          0.0120      0.020      0.592      0.554      -0.028
0.052
log_price         -1.1219      0.440     -2.549      0.012      -1.990
-0.254
log_avg_spend      0.4329      0.023     18.604      0.000       0.387
0.479
log_points         0.0026      0.008      0.326      0.745      -0.013
0.018
==============================================================================
Omnibus:                        0.999   Durbin-Watson:                   2.073
Prob(Omnibus):                  0.607   Jarque-Bera (JB):                0.950
Skew:                          -0.160   Prob(JB):                        0.622
Kurtosis:                       2.963   Cond. No.                     5.01e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 5.01e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

This is what we get for the testing dataset

```
#Full dataset
y = (
    y
```

```
    .with_columns((pl.col('units') + 1).log().alias('log_units'))
    .select('log_units')
    .to_pandas()
)
y = y['log_units']   # convert to Series

# Transform X
X = (
    X
    .with_columns((pl.col('price') + 1).log().alias('log_price'))
    .with_columns((pl.col('avg_spend') + 1).log().alias('log_avg_spend'))
    .with_columns((pl.col('points') + 1).log().alias('log_points'))
    .to_dummies(columns=['brand', 'texture', 'size', 'gender', 'email'],
drop_first=False)
    .select(pl.exclude('price', 'avg_spend', 'points', 'brand_Jif',
'texture_Smooth', 'size_16', 'gender_M', 'email_No'))
    .to_pandas()
)

# Add constant
X = sm.add_constant(X)

# Fit model
fit_03 = sm.OLS(y, X).fit()
print(fit_03.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              log_units   R-squared:                       0.867
Model:                            OLS   Adj. R-squared:                  0.866
Method:                 Least Squares   F-statistic:                     504.9
Date:                Tue, 30 Sep 2025   Prob (F-statistic):               0.00
Time:                        18:15:53   Log-Likelihood:                660.71
No. Observations:                1095   AIC:                            -1291.
Df Residuals:                    1080   BIC:                            -1216.
Df Model:                          14
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
const              1.0627      0.334      3.182      0.002       0.407
1.718
brand_Harmons     -0.0116      0.084     -0.139      0.889      -0.176
0.152
brand_PeterPan    -0.1933      0.034     -5.612      0.000      -0.261
-0.126
```

```
brand_Skippy      -0.0485      0.023     -2.074      0.038      -0.094
-0.003
coupon             0.3288      0.023     14.275      0.000       0.284
0.374
ad                 0.0852      0.010      8.650      0.000       0.066
0.104
texture_Chunky    -0.1806      0.008    -21.534      0.000      -0.197
-0.164
size_12            0.0244      0.040      0.612      0.541      -0.054
0.102
age               -0.0084      0.000    -24.954      0.000      -0.009
-0.008
gender_F          -0.0034      0.008     -0.415      0.678      -0.020
0.013
gender_Other       0.0169      0.022      0.774      0.439      -0.026
0.060
email_Yes          0.0129      0.009      1.424      0.155      -0.005
0.031
log_price         -0.7284      0.171     -4.249      0.000      -1.065
-0.392
log_avg_spend      0.4413      0.010     42.790      0.000       0.421
0.462
log_points         0.0014      0.003      0.419      0.675      -0.005
0.008
==============================================================================
Omnibus:                        0.546   Durbin-Watson:                   1.991
Prob(Omnibus):                  0.761   Jarque-Bera (JB):                0.628
Skew:                           0.014   Prob(JB):                        0.730
Kurtosis:                       2.886   Cond. No.                     4.28e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 4.28e+03. This might indicate that there
are
strong multicollinearity or other numerical problems.
```

This is what we get for the full dataset.

Here are the frequentist interpretations of the parameters:

brand_Harmons – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that choosing Harmons brand peanut butter changes the number of units sold between −16.1% and +16.4%, relative to Jif peanut butter, holding all other variables fixed. This is not statistically significant.

brand_PeterPan – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that choosing PeterPan brand peanut

butter changes the number of units sold between −23.0% and −11.8%, relative to Jif peanut butter, holding all other variables fixed. This is statistically significant.

brand_Skippy – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that choosing Skippy brand peanut butter changes the number of units sold between −9.0% and -.03%, relative to Jif peanut butter, holding all other variables fixed. This is statistically significant.

coupon – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that having a coupon increases the number of units sold between +32.8% and +45.4%, holding all other variables fixed. This is statistically significant.

ad – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that having an ad increases the number of units sold between +6.8% and +11.0%, holding all other variables fixed. This is statistically significant.

texture_Chunky – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that having a chunky texture decreases the number of units sold between −17.9% and −15.1%, relative to smooth peanut butter, holding all other variables fixed.

size_12 – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that having a 12-oz jar increases the number of units sold between −5.3% and +10.7%, relative to 16-oz peanut butter, holding all other variables fixed. This is not statistically significant.

age – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that a one unit increase in age will change the number of units sold between −1.0% and −0.7%, holding all other variables fixed. This is statistically significant.

gender_F – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that being a female will change the number of units sold between −1.8% and 6.2%, relative to male customers, holding all other variables fixed. This is not statistically significant.

gender_Other – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that being a gender other than male or female will change the number of units sold between −12.5% and 4.7%, relative to male customers, holding all other variables fixed. This is not statistically significant.

email_Yes – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that receiving Harmons emails will change the number of units sold between −3.5% and 5.0%, relative to male customers, holding all other variables fixed. This is not statistically significant.

price - Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that a 1% change in price changes the number of units sold between −225.6% and −45.9%, holding all other variables fixed. This is statistically significant.

avg_spend - Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that a 1% change in average spend changes the number of units sold between 40.5% and 50.5%, holding all other variables fixed. This is statistically significant.

avg_spend - Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that a 1% change in points changes the number of units sold between −1.6% and 1.4%, holding all other variables fixed. This is not statistically significant.

```
#profit function used in expected loss
def profit(log_price, log_units, cost, scale):
    log_cost = np.log(cost)
    log_profit = log_units * (log_price - log_cost)
    profit = np.exp(log_profit) * scale
    return profit

X = X.drop(columns=['age', 'log_avg_spend', 'gender_F', 'gender_Other',
'log_points', 'email_Yes'])
fit_04 = sm.OLS(y, X).fit()

# Construct a new design matrix for prediction
X_new = pl.DataFrame({
    'brand_Harmons': [1, 1, 1, 1, 1, 1],
    'brand_PeterPan': [0, 0, 0, 0, 0, 0],
    'brand_Skippy': [0, 0, 0, 0, 0, 0],
    'coupon': [0, 1, 0, 0, 1, 0],
    'ad': [0, 0, 0, 1, 0, 1],
    'texture_Chunky': [0, 0, 0, 0, 0, 0],
    'size_12': [0, 0, 0, 0, 0, 0],
    'log_price': [3.50, 3.50, 3.50, 4.0, 4.0, 4.0],
    'constant': [1, 1, 1, 1, 1, 1]
}).with_columns(
    pl.col('log_price').log().alias('log_price')
).to_pandas()

# Confidence and prediction intervals
fit_04.get_prediction(X_new).summary_frame(alpha=0.05)

# Get preditions for X_new
fr_pred = fit_04.predict(X_new)

# Calculate profit for each action
fr_pred['profit'] = profit(
```

```
    log_price = X_new['log_price'],
    log_units = fr_pred,
    cost = 1.50,
    scale = 10_000
)

# Average across actions
print(np.mean(fr_pred['profit']))
```

```
82303.06166047683
```

For our predictions, we created a table with six rows:

-Harmons Peanut Butter at $3.50

-Harmons Peanut Butter at $3.50 with a coupon

-Harmons Peanut Butter at $3.50 with an ad

-Harmons Peanut Butter at $4.00

-Harmons Peanut Butter at $4.00 with a coupon

-Harmons Peanut Butter at $4.00 with an ad

Predictions:

Row 1 – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that pricing Harmons peanut butter at $3.50 will lead to a total sale between $3.53 and $14.33.

Row 2 – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that pricing Harmons peanut butter at $3.50 will lead to a total sale between $3.66 and $13.59 units.

Row 3 – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that pricing Harmons peanut butter at $3.50 will lead to a total sale between $3.53 and $14.33.

Row 4 – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that pricing Harmons peanut butter at $4.00 will lead to a total sale between $6.02 and $20.33.

Row 5 – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that pricing Harmons peanut butter at $4.00 will lead to a total sale between $3.81 and $13.73.

Row 6 – Based on OLS interval estimates, 95% of such intervals will contain the true value of the parameter, such that we are 95% confident that pricing Harmons peanut butter at $4.00 will lead to a total sale between $6.02 and $20.33.

From our analysis, it looks like providing a coupon leads to a smaller sale, and increasing the price from $3.50 to $4.00 grants more profit, even with the decreased correlation between price and units. The mean predicted profit from our output is $82,303, which is very promising. ```