Research Question

Perishable goods are a unique segment of products consisting of items with a limited shelf life before spoiling renders them unsaleable, like flowers, meat, vegetables, or fruits. Proper inventory and stock management are one strategy many businesses use to reduce waste, but accurate demand forecasting is another necessary tool in reducing inventory waste (Sid, 2021).

Businesses that can forecast both the timing and amount of sales for goods can anticipate production, warehousing, and shipping needs and know what future constraints will be placed on labor and cash flow. (Milano, 2018)

Many companies are looking for new predictors for better supply chain models beyond their historical sales data. New data sources to predict demand allows companies to evaluate potential predictor variables through more than just their transactions. Mobile phones and social media have provided additional data that businesses have used successfully to predict demand (Brea, 2020).

Understanding consumer behavior may depend on various factors, including demographic data, lifestyle trends, and shopping patterns. In one case, Nounós Creamery reduced overproduction by 40% and decreased product

loss by implementing forecasting tools that incorporated data from multiple sources (Traasdahl, 2021).

This study aims to determine if the variables in the available data set can be used to build a successful predictive model for forecasting the demand for perishable goods. The data includes expenditures on fruit and vegetables from 22,448 households to be used as target variables. The data also contains variables for each household, including demographic data and many local area data, including the availability of other store types, employment availability, and street density. (Peng, 2022)

The question for this study is whether or not a regression model can be built to predict a household's expenditures on perishable goods (fruits and vegetables) based on the available research data?

The hypothesis for this research is:

It is possible to build a regression model to statistically significantly predict household expenditures on perishable goods based on the available variables in the research data.

Data Collection

The data set for this study was initially created for the research by Ke Peng and Nikhil Kaza in their study Availability of neighbourhood supermarkets

and convenience stores, broader built environment context, and the purchase of fruits and vegetables in US households. The complete data set is publicly available on the University of North Carolina's Dataverse site with a "Public Domain Dedication" waiver. (Peng & Kaza, 2019).

The data was acquired by downloading the data set and available data dictionary as a .zip file and extracting the files. The data set is formatted as a .tab file.

One advantage of this data acquisition is that it is sourced from an academic institution as part of a peer-reviewed study. The integrity of the data set is much higher as both authors, and their academic colleagues have reviewed it. The data contains thousands of observations with no null or missing values.

However, acquiring data from a published academic study presents the disadvantage of limited opportunities to gather additional observations to develop a model further.

Since the purpose of this study is to evaluate the data and determine whether the available variables are sufficient to predict a household's expenditures, the availability of additional data is not a concern, as there is an adequate number of observations to determine if a regression model can be built for predictions.

Data Extraction and Preparation

All preparation and analysis will be done using the python programming language in a Jupyter Notebook. Python is a general programming language with a robust library of packages well suited for data analysis. Its use within data science is rapidly growing and is suitable for predictive modeling (Kan, 2018).

An advantage to Python is that its use is widespread among not just data scientists but professionals in many other fields. Its widespread use means there is a greater chance that colleagues and professionals reviewing and collaborating on the project will have an understanding of the code developed to perform the analysis.

A disadvantage to using Python is that while it is widespread, it relies heavily on community-maintained packages for managing data and performing analysis.

Extracting and preparing this data for analysis relies primarily on many functions within the pandas library for Python.

Pandas is an optimal tool for working with data frames in Python and provides several functions for examining and manipulating data.

The following steps were taken to extract and prepare the data:

1. Read the downloaded data set and write as a pandas data frame using the pandas function “read_csv()”

```
df = pd.read_csv('data_table1-3.csv')
```

```
df.head()
```

	household_code	fruit_paid	vege_paid	ln_fruit_paid	ln_vege_paid	super5_cat	con5_10	regional_10000	dest5_10	mix5_
0	1	166.260000	148.790000	5.113553	5.002536	1	0.2	1.548161	27.9	2.9661
1	2	14.030000	39.890000	2.641198	3.686126	1	0.2	1.548161	27.9	2.9661
2	3	245.490000	425.100000	5.503256	6.052324	1	0.2	1.548161	27.9	2.9661
3	4	9.720000	142.570000	2.274186	4.959833	1	0.1	0.876350	4.4	3.9882
4	5	330.579999	480.969999	5.800848	6.175805	0	0.1	1.972699	1.3	5.6183

5 rows × 24 columns

The advantage of extracting and writing the data with this function is that it can quickly and simply format the provided data into a data frame ready for further exploration and manipulation with only a single line of code.

A disadvantage to extracting data with this method is that it is best suited for static datasets that are available as a single large file. It is not well suited for live data sets that are updated regularly and accessed through API calls or queries.

2. Examine the data with the pandas .info() function and check for null or missing values, datatypes, number of observations, and number of

columns.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22448 entries, 0 to 22447
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   household_code  22448 non-null  int64
 1   fruit_paid      22448 non-null  float64
 2   vege_paid       22448 non-null  float64
 3   ln_fruit_paid   22310 non-null  float64
 4   ln_vege_paid    22334 non-null  float64
 5   super5_cat      22448 non-null  int64
 6   con5_10         22448 non-null  float64
 7   regional_10000  22448 non-null  float64
 8   dest5_10        22448 non-null  float64
 9   mix5_10         22448 non-null  float64
 10  design5_10      22448 non-null  float64
 11  auto_bg100      22448 non-null  float64
 12  poverty_tr100   22448 non-null  float64
 13  edu             22448 non-null  int64
 14  income          22448 non-null  int64
 15  race            22448 non-null  int64
 16  hh_size         22448 non-null  int64
 17  age             22448 non-null  int64
 18  marital_status  22448 non-null  int64
 19  children        22448 non-null  int64
 20  employed_new    22448 non-null  int64
 21  urban           22448 non-null  int64
 22  msa             22448 non-null  int64
 23  zip             22448 non-null  int64
dtypes: float64(11), int64(13)
memory usage: 4.1 MB
```

The information provided from this step shows that all columns except for ln_fruit_paid1 and ln_vege_paid contain 22,448 non-null values, meaning there are no missing or null values for those variables. In the case of the two columns with null values, both correspond to variables the original researchers of this dataset created for their research purposes. These will be dropped from the dataset as they are unnecessary for this study.

An advantage to examining the data with the .info() function is that it

provides quite a bit of valuable insights into the data frame in a straightforward printout. The shape of the data, number of observations, null or missing values, datatypes, variable names, and even memory usage are all included and make an initial examination quick and easy.

A disadvantage to this tool is that if only certain insights are needed, it provides too much information and can make finding those particular insights more time-consuming.

For example, if finding variables with null values is needed, then a function like .isnull() would be better suited for just that purpose

3. Examine the data with .describe()

```
df.describe()
```

	household_code	fruit_paid	vege_paid	ln_fruit_paid	ln_vege_paid	super5_cat	con5_10	regional_10000
count	22448.000000	22448.000000	22448.000000	22310.000000	22334.000000	22448.000000	22448.000000	22448.000000
mean	11224.500000	133.700733	135.791781	4.452355	4.525334	0.635157	1.212745	9.637722
std	6480.323757	124.415704	118.907040	1.077241	0.991359	0.825646	1.762062	11.980695
min	1.000000	0.000000	0.000000	-2.302585	-2.302585	0.000000	0.000000	0.000000
25%	5612.750000	46.957499	53.017500	3.868802	3.985831	0.000000	0.200000	2.287601
50%	11224.500000	99.035000	104.165000	4.602768	4.651481	0.000000	0.600000	5.864709
75%	16836.250000	180.362498	182.600000	5.199146	5.209882	1.000000	1.600000	12.270072
max	22448.000000	1631.569999	1660.030002	7.397298	7.414591	2.000000	17.900000	136.262620

Another tool for examining the structure and makeup of the data

frame is the .describe() function, which displays essential information for all numerical columns that summarize their central tendencies, dispersions, and distribution.

Many of the categorical values are represented in this data set as numerical values. This is because the researchers that initially created this dataset formatted these categorical variables from their string values to numerical values for their research. The corresponding string values for these variables are defined in the provided data dictionary (Peng, 2022) and their published study (Peng & Kaza, 2019).

The exception to the defined categorical values is “age” which is missing from the data dictionary and published research article. Due to the lack of clarification on how the numerical values map to categorical values, it will be dropped from the data along with the 2 columns identified in step 2.

An advantage to the .describe() function is its thorough breakdown of descriptive statistics within a data frame. It provides a wide range of helpful insight into numerical values that can provide deeper insight into a data frame.

A disadvantage to this function is that it can only provide insight into quantitive numerical data where means, quartiles, minimums, and maximums are helpful. For nonquantitative data or categorical data, like zip codes or marital status, it is not well suited for statistical understanding.

4. Create a “total expenditure” column representing the sum of household expenditures for both fruits and vegetables.

Create total expenditure variable

```
df.insert(loc = 3, column='total expenditure', value= df['fruit_paid'] + df['vege_paid'])
```

Since this study is intended to look at households' total expenditure, a sum of both expenditures on fruits and vegetables is needed. Those two expenditure columns can then be dropped from the data frame.

To build that variable .insert() is used to create a new column in the data frame with the sum of fruit_pad and vege_paid.

An advantage to using the .insert() function is that it is not only able to create and add the needed values to the data frame, but it's possible to specify the column name and location within the data

frame.

A disadvantage of this tool is its ability to insert a column within the data set requires a specific index number for location. This requires a user's understanding of the organizational structure of the data frame, which can be difficult for large datasets. Placing a new column with this tool between or adjacent to similar columns requires understanding what those columns are and their location index values along their axis.

5. Drop unneeded columns using pandas .drop() function

drop identified columns

```
df.drop(['ln_fruit_paid', 'ln_vege_paid', 'age', 'fruit_paid','vege_paid'], axis=1, inplace=True)
```

As stated in the steps above; ln_fruit_paid1, ln_vege_paid, age, 'fruit_paid' and 'vege_paid' columns are not needed for this study and can be dropped using the .drop() function.

This function is an advantage because it is a quick and simple tool to remove unneeded columns or rows from a data frame.

A disadvantage to this tool is that it is destructive in that the changes to the data frame can't be undone, and returning these values to the data frame would require extracting the data set again and redoing any preparations before dropping the columns.

6. Format categorical variables for regression modeling.

   All categorical data in this data frame are represented as numerical values rather than strings. To prepare this data set for regression modeling, the categorical variables need to be converted to dummy variables, either by identifying binary categorical data as 1 or 0 for "yes" and "no" or one hot encoding all categorical variables containing three or more distinct values.

   Based on both the data dictionary (Peng, 2022) and the original published research (Peng & Kaza, 2019) the following variables all have multiple numerical values that map to string values: edu, income, race, marital_status & urban
   These will need to be one hot encoded.
   The variable "children" is a binary variable of "yes" or "no" and is already represented in the data frame as the proper binary values, 1 and 0, that would be needed for representation with a dummy

variable.

To one hot encode the identified columns, there are two necessary steps:

a. Convert all numerical values to their mapped string values

```
# Education
df['edu'].replace({1:'high school or below', 2:'college or higher', 3:'no female head'}, inplace=True)

# Income
df['income'].replace({1:'below 20000', 2:'20000-59999', 3:'60000+'}, inplace=True)

# Race
df['race'].replace({1:'white', 2:'black', 3:'asian', 4:'other'}, inplace=True)

# Marital Status
df['marital_status'].replace({1:'married', 2:'widowed', 3:'divorced', 4:'seprated/single'},
                             inplace=True)

# Urban
df['urban'].replace({1:'urbanized area', 2:'urban cluster', 3:'non-urban'}, inplace=True)

df[['edu','income','race','marital_status','urban']].sample(10)
```

	edu	income	race	marital_status	urban
9530	high school or below	60000+	white	married	non-urban
22427	high school or below	60000+	white	married	urbanized area
11657	high school or below	20000-59999	white	married	non-urban
20211	high school or below	20000-59999	white	married	non-urban
21591	high school or below	20000-59999	white	married	urbanized area
19767	college or higher	60000+	white	married	urbanized area
16891	college or higher	20000-59999	black	seprated/single	urbanized area
12101	college or higher	60000+	white	seprated/single	urbanized area
18477	high school or below	below 20000	other	married	urbanized area
11991	college or higher	60000+	white	married	urbanized area

b. One hot encode identified columns using pandas' get_dummies() function

```
variables = ['edu','income','race','marital_status','urban']

for var in variables:
    df['one hot'] = df[var]
    df = pd.get_dummies(df, columns=['one hot'], prefix= '', prefix_sep='')
```

```
# Group one hot encoded variables into their original category heading
education = ['high school or below', 'college or higher', 'no female head']
income = ['below 20000', '20000-59999', '60000+']
race = ['white', 'black', 'asian', 'other']
marital_status = ['married', 'widowed', 'divorced', 'seprated/single']
urban = ['urbanized area', 'urban cluster', 'non-urban']
one_hot = education + income + race + marital_status + urban
```

```
df[one_hot].head()
```

	high school or below	college or higher	no female head	below 20000	20000-59999	60000+	white	black	asian	other	married	widowed	divorced	seprated/single	ur
0	0	1	0	0	0	1	0	1	0	0	1	0	0	0	
1	0	1	0	0	1	0	0	0	0	1	0	0	0	1	
2	0	1	0	0	0	1	0	0	1	0	1	0	0	0	
3	1	0	0	0	1	0	1	0	0	0	1	0	0	0	
4	0	1	0	1	0	0	1	0	0	0	1	0	0	0	

One hot encoding categorical data breaks each category into multiple columns with a heading representing each distinct value within that category. A value of 1 is assigned to the column representing that categories original value, and a 0 for all other columns within that category.

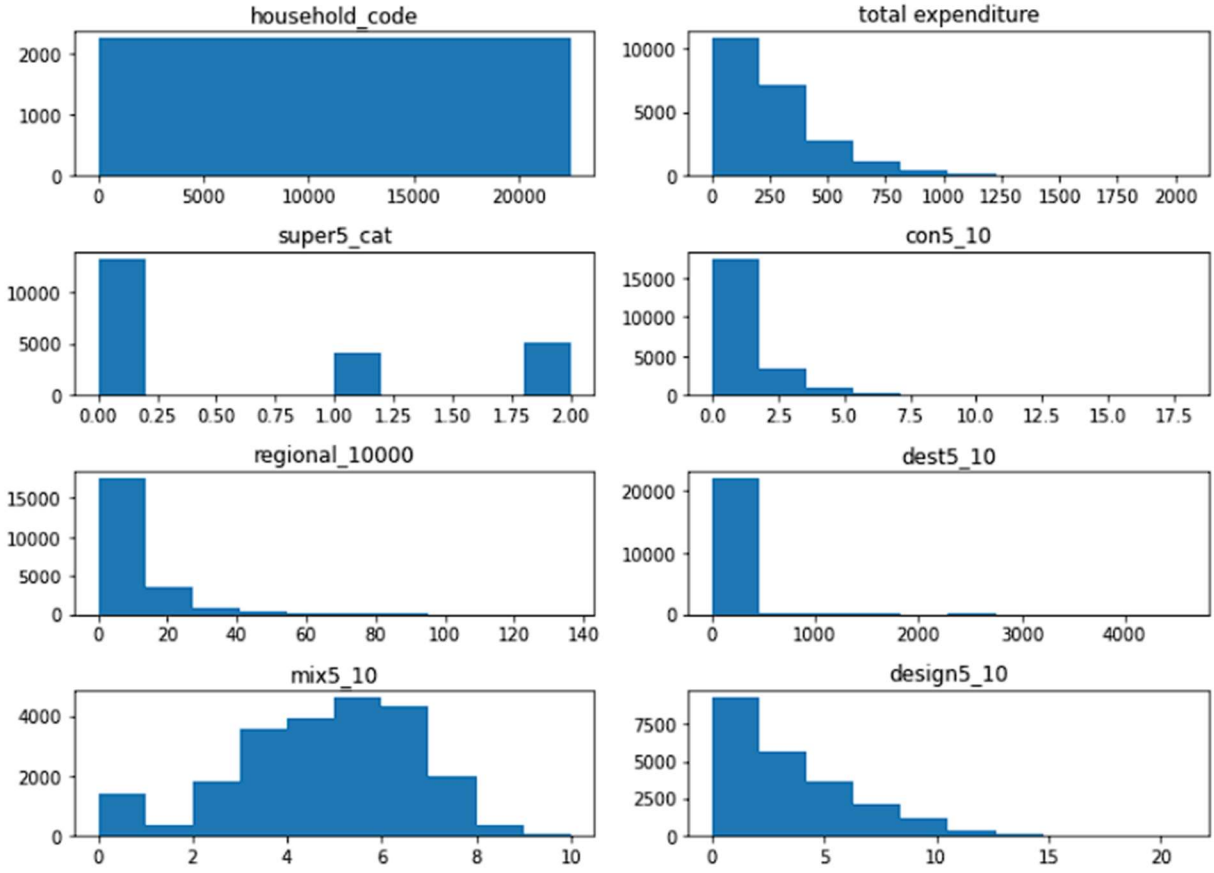
For example, a household with “asian” as its original value for “race” would be assigned a new variable “asian” with a value of 1, and three variables of “black”, “white” & “other”, with values of 0.

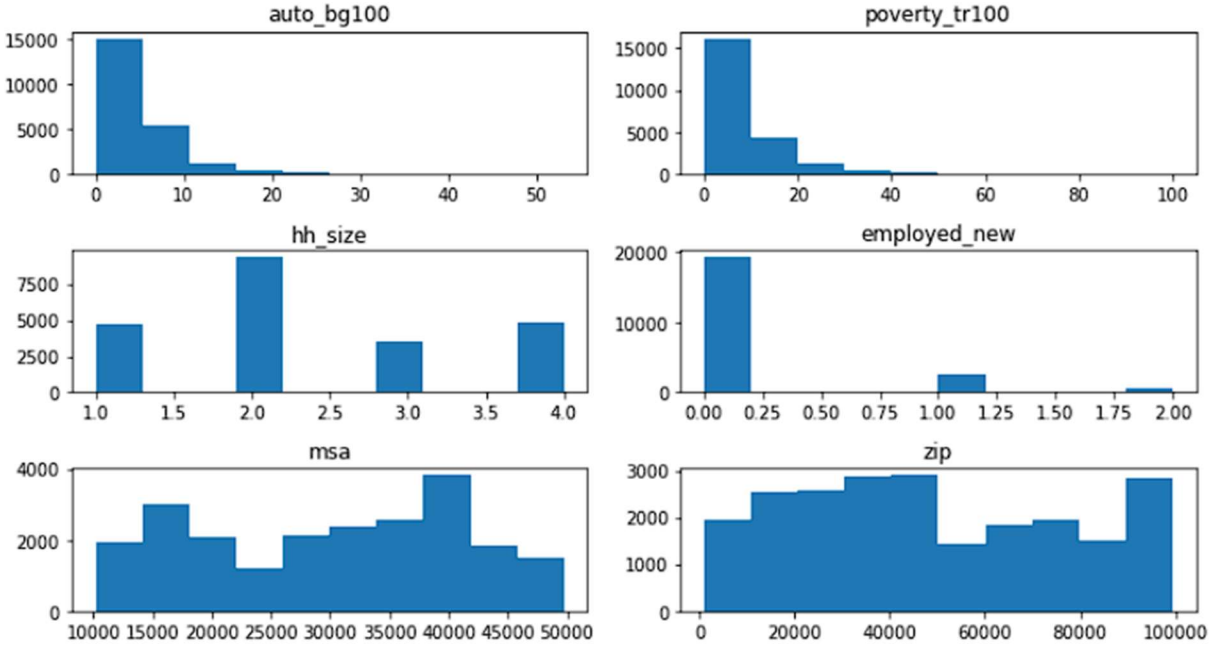
An advantage of one-hot encoding the categorical data is that it accurately represents the categorical data in a way that is useful for many predictive modeling strategies, including regression models.

A disadvantage to one-hot encoding is that while it is well suited for categories with a low number of distinct values, categories with a large number of distinct values would require an excessive number of new columns. For large datasets, multiple categorical variables with a large number of distinct values would create an impractical number of new variables through one-hot encoding.

7. Examine distributions of non-categorical data

```
non_binary = df.drop(columns=categories).nunique()[lambda x: x > 2].index.tolist()

plt.figure(figsize=(10,35))

i = 1

for col in non_binary:
    plt.subplot(20, 2, i)
    plt.hist(df[col])
    plt.title(col)
    i += 1

plt.tight_layout();
```


household_code
total expenditure
super5_cat
con5_10
regional_10000
dest5_10
mix5_10
design5_10


auto_bg100
poverty_tr100
hh_size
employed_new
msa
zip

Based on the output of the histogram plots several variables show a right skewed distribution including: total expenditure, cons_5_10, regional_10000, dest5_10, design5_10, auto_bg100, poverty_tr100 and employed_new

8. Transform skewed data to prep for regression model using logarithmic transformation.

   Although linear regression does not require normal distribution of the predictor variables, it helps improve accuracy. (Knee, 2021)

   To transform the skewed data, the np.log() function is used for each variable.

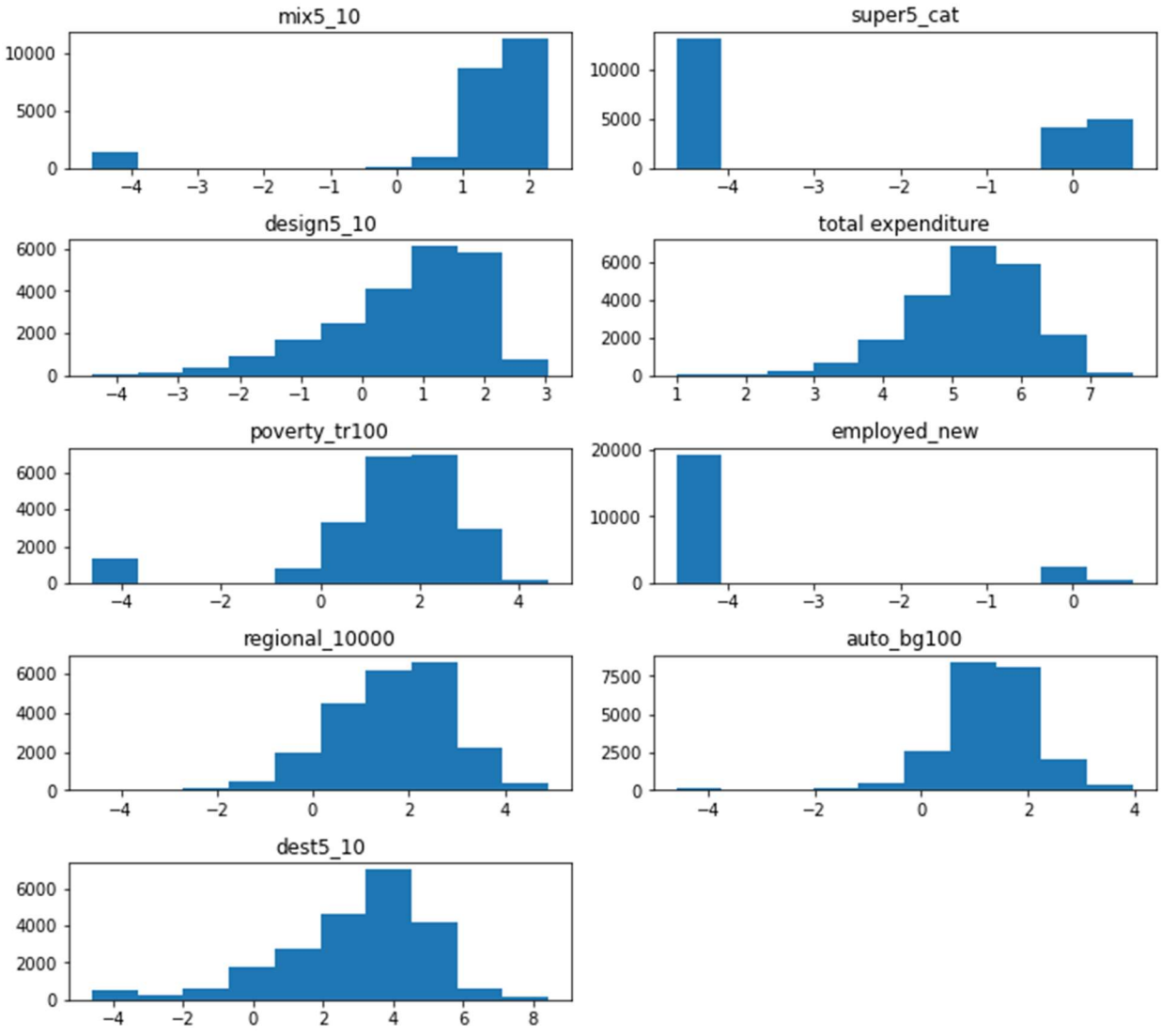
   An advantage to using logarithmic transformation for normalizing these variables is that it can easily be inverted using an exponential function. A successful model that predicts using these variables will return predictions of the logarithmically transformed target variable. An inverse of that transformed value can be found by applying an exponential function, and a predicted value that matches the raw value

for the actual value is much easier to find.

A disadvantage to using the logarithmic transformation is that the model is now training on the transformed data. The output for the predicted value will represent the logarithm of the true predicted value and will need to be inverse transformed first.

```
skewed = ['mix5_10', 'super5_cat', 'design5_10', 'total expenditure', 'poverty_tr100', 'employed_new',
          'regional_10000', 'auto_bg100', 'dest5_10']

df_log = df

for var in skewed:
    df_log[var] = np.log(df_log[var]+.01)
```

```
plt.figure(figsize=(10,35))

i = 1

for col in skewed:
    plt.subplot(20, 2, i)
    plt.hist(df_log[col])
    plt.title(col)
    i += 1

plt.tight_layout();
```



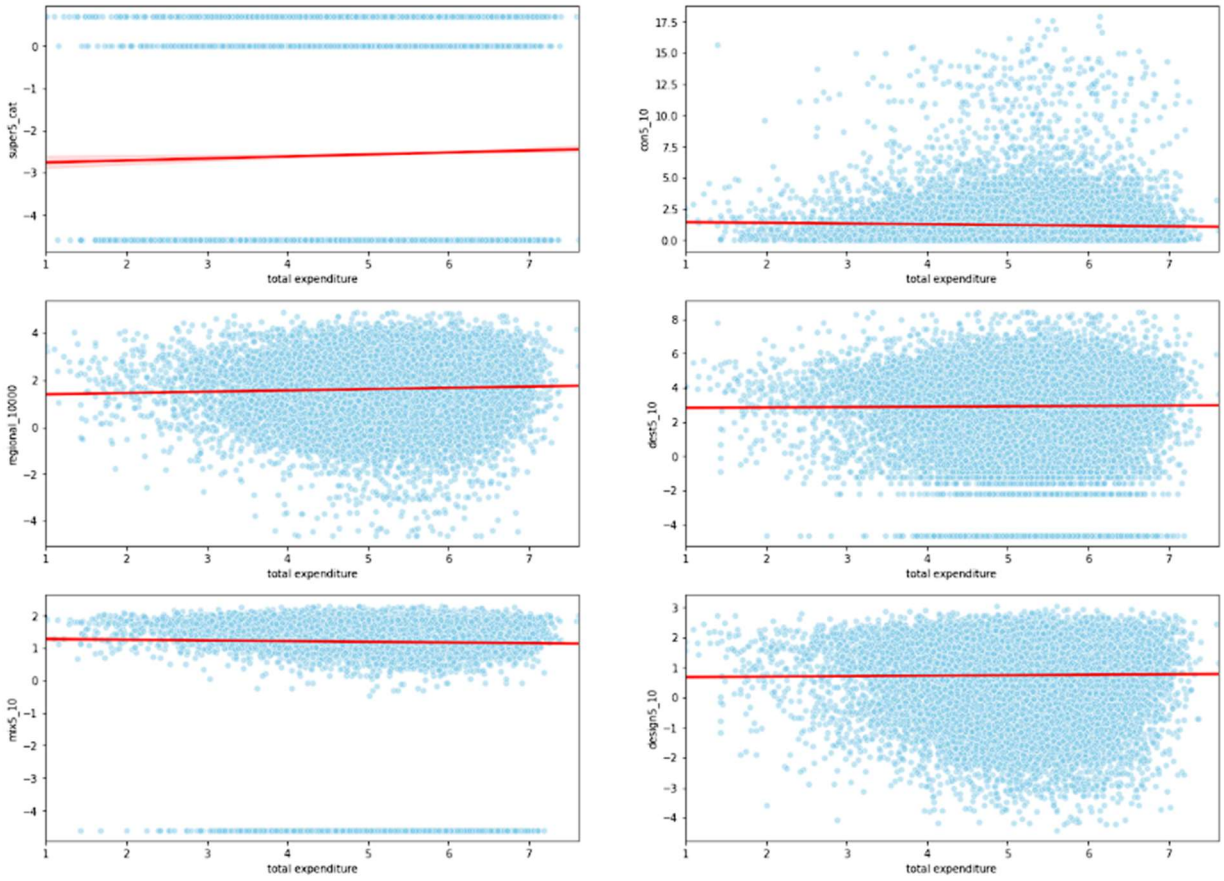
https://medium.com/swlh/log-transformations-in-linear-regression-the-basics-95bc79c1ad35

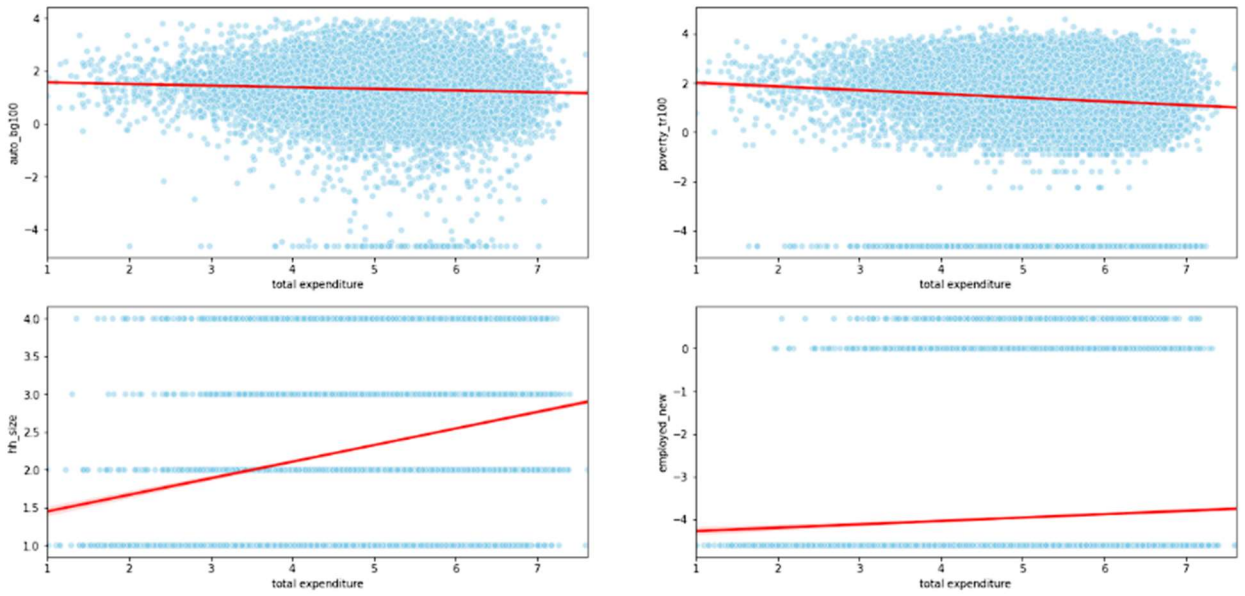
9. Examine relationship between predictor variables and target variables with seaborn regplot.

   Before modeling, it is helpful to understand any linear relationships between predictor variables and the target variable. Seaborn's regplot

function calculates a linear regression between the two variables and plots a regression line.

```
cont = ['super5_cat', 'con5_10', 'regional_10000', 'dest5_10', 'mix5_10',
        'design5_10', 'auto_bg100', 'poverty_tr100', 'hh_size', 'employed_new']

plt.figure(figsize=(20,25))

i = 1

for var in cont:
    plt.subplot(5, 2, i)
    sns.scatterplot(data=df_log, x="total expenditure", y=var, color='skyblue', alpha=.5)
    sns.regplot(data=df_log, x="total expenditure", y=var, scatter=False, color='r')
    i += 1
```





An advantage of the regplot function is that it is a simple way to look at the relationship between each continuous predictor variable and the target variable and expose any existing relationships before modeling.

A disadvantage is that it is best suited for continuous numerical data and is not well suited to explore individual regressions between categorical variables and the target variable.

Analysis

A linear regression based on ordinary least squares (OLS) will be used to build the multiple regression model for this study.

Linear regression is suited for us when both continuous and categorical data are needed to predict a continuous target variable(Mitchel, 2019). A

multiple regression model is simply a linear regression applied to multiple predictor variables.

With ordinary least squares, the regression line is an estimate that minimizes the sum of squared residuals. An advantage to OLS is that it is very efficient computationally and, therefore, quite fast. A disadvantage is that it is sensitive to outliers (Bruce, Bruce, Gedeck, 2020).

To assess the model's performance and evaluate the strength of the relationship of the predictor variables to the target variables, the following metrics will be examined:

R-squared: This is the proportion of variance in the target variable that is predictable by the predictor variables. This is measured between 0 and 1, with a score of 1 representing a perfect fit and a score of 0, meaning the model performs no better than random chance. (Bruce, Bruce, Gedeck, 2020). This is the value to be used to determine the statistical significance of the entire model.

t-statistic – A measurement of the significance of the coefficient. A high t-statistic indicates a high likelihood the variable's coefficient is outside the range of random chance. (Bruce, Bruce, Gedeck, 2020).

p-value – An inverse to the t-statistic, it also measures the significance of a variable's coefficient. It measures the probability of obtaining results as a random chance. A lower p-value indicates a higher likelihood the variable's coefficient is not the result of randomness. (Bruce, Bruce, Gedeck, 2020). For this model, the statistical significance will be determined by an alpha of .05, or in other words, a 95% chance the variable's coefficient is not related to random chance.

Condition number – A measure of the model's sensitivity and a good indicator for multicollinearity. A high condition number indicates multiple variables strongly related to each other. A low condition number is preferable for a predictive model. (McAleer, 2020)

OLS

```
data = df_log

data['intercept'] = 1

predictors = data.drop(columns = ['household_code', 'total expenditure'])
target = data['total expenditure']

x = predictors
y = target
x = sm.add_constant(x)

model = sm.OLS(y, x).fit()
model.summary()
```

OLS Regression Results

Dep. Variable:	total expenditure	**R-squared:**	0.121
Model:	OLS	**Adj. R-squared:**	0.120
Method:	Least Squares	**F-statistic:**	123.3
Date:	Mon, 05 Sep 2022	**Prob (F-statistic):**	0.00
Time:	11:48:39	**Log-Likelihood:**	-27900.
No. Observations:	22448	**AIC:**	5.585e+04
Df Residuals:	22422	**BIC:**	5.606e+04
Df Model:	25		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
super5_cat	0.0133	0.003	4.621	0.000	0.008	0.019
con5_10	-0.0059	0.005	-1.311	0.190	-0.015	0.003
regional_10000	0.0277	0.008	3.588	0.000	0.013	0.043
dest5_10	0.0157	0.007	2.124	0.034	0.001	0.030
mix5_10	-0.0239	0.005	-5.197	0.000	-0.033	-0.015
design5_10	0.0034	0.013	0.275	0.783	-0.021	0.028
auto_bg100	-0.0307	0.008	-3.784	0.000	-0.047	-0.015
poverty_tr100	-0.0152	0.003	-4.603	0.000	-0.022	-0.009
hh_size	0.0918	0.011	8.123	0.000	0.070	0.114
children	0.1790	0.021	8.592	0.000	0.138	0.220
employed_new	-0.0009	0.004	-0.222	0.824	-0.009	0.007
msa	2.846e-07	5.12e-07	0.556	0.578	-7.19e-07	1.29e-06
zip	-3.414e-07	2.32e-07	-1.472	0.141	-7.96e-07	1.13e-07
college or higher	0.7701	0.012	64.096	0.000	0.747	0.794
high school or below	0.6531	0.014	46.658	0.000	0.626	0.681
no female head	0.4612	0.017	27.340	0.000	0.428	0.494
20000-59999	0.6054	0.012	49.927	0.000	0.582	0.629
60000+	0.8225	0.012	66.963	0.000	0.798	0.847
below 20000	0.4565	0.017	27.260	0.000	0.424	0.489
asian	0.6382	0.027	23.257	0.000	0.584	0.692
black	0.3106	0.018	16.895	0.000	0.275	0.347
other	0.4804	0.023	20.781	0.000	0.435	0.526
white	0.4552	0.013	35.722	0.000	0.430	0.480
divorced	0.3933	0.014	28.728	0.000	0.366	0.420
married	0.6507	0.016	41.217	0.000	0.620	0.682
seprated/single	0.3185	0.015	21.954	0.000	0.290	0.347
widowed	0.5219	0.020	25.544	0.000	0.482	0.562
non-urban	0.6366	0.013	48.540	0.000	0.611	0.662
urban cluster	0.6157	0.022	28.148	0.000	0.573	0.659
urbanized area	0.6322	0.014	46.511	0.000	0.606	0.659
intercept	1.8845	0.024	79.750	0.000	1.838	1.931

Omnibus:	1901.420	Durbin-Watson:	1.951
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2730.123
Skew:	-0.686	Prob(JB):	0.00
Kurtosis:	4.019	Cond. No.	1.33e+16

An initial OLS model returns an r-squared value of .121 and a condition number of 1.33e+16. With an r-squared number of .121 the predictor values account for only 12.1% of the variance in this model and are therefore not statistically significant.

Furthermore, a condition number of 1.33e+16 is high and indicates the likelihood of strong multicollinearity within the model.

This model will benefit from reducing the number of predictor variables to reduce the likelihood of multicollinearity. Selecting variables in the initial model with a tvalue above 0, and the top variables with a pvalue greater than .05 should reduce the number of needed predictor variables for modeling with the most negligible impact on the r-squared value of the model.

```
pvalues = pd.DataFrame(model.pvalues, columns =['pvalue']).reset_index().rename(columns={'index':'variable'})
tvalues = pd.DataFrame(model.tvalues, columns =['tvalue']).reset_index().rename(columns={'index':'variable'})
key_vals = pvalues.merge(tvalues, how='inner', on='variable')
key_vals = key_vals[key_vals['pvalue'] < 0.05]
key_vals = key_vals[key_vals['tvalue'] > 0]
```

```
key_vals.sort_values(by='tvalue', ascending=False, inplace=True)
key_vals.reset_index(drop=True, inplace=True)
reduced = key_vals.sort_values(by='tvalue',ascending=False).head(10)['variable'].tolist()
reduced
```

```
['intercept',
 '60000+',
 'college or higher',
 '20000-59999',
 'non-urban',
 'high school or below',
 'urbanized area',
 'married',
 'white',
 'divorced']
```

```
data = df_log

data['intercept'] = 1

predictors = df_log[reduced]
target = df_log['total expenditure']

x = predictors
y = target
x = sm.add_constant(x)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

model = sm.OLS(y_train, x_train).fit()
model.summary()
```

OLS Regression Results

Dep. Variable:	total expenditure	**R-squared:**	0.106
Model:	OLS	**Adj. R-squared:**	0.105
Method:	Least Squares	**F-statistic:**	235.8
Date:	Mon, 05 Sep 2022	**Prob (F-statistic):**	0.00
Time:	12:17:08	**Log-Likelihood:**	-22499.
No. Observations:	17958	**AIC:**	4.502e+04
Df Residuals:	17948	**BIC:**	4.510e+04
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
intercept	4.3484	0.046	94.906	0.000	4.259	4.438
60000+	0.4024	0.025	16.012	0.000	0.353	0.452
college or higher	0.3561	0.025	14.372	0.000	0.308	0.405
20000-59999	0.1560	0.025	6.207	0.000	0.107	0.205
non-urban	0.0697	0.033	2.133	0.033	0.006	0.134
high school or below	0.2414	0.028	8.769	0.000	0.187	0.295
urbanized area	0.1074	0.032	3.338	0.001	0.044	0.170
married	0.3219	0.018	17.469	0.000	0.286	0.358
white	0.0401	0.017	2.338	0.019	0.006	0.074
divorced	0.0082	0.022	0.381	0.703	-0.034	0.050

Omnibus:	1430.606	**Durbin-Watson:**	1.963
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2013.331
Skew:	-0.662	**Prob(JB):**	0.00
Kurtosis:	3.967	**Cond. No.**	18.4

The new model with a reduced number of predictor variables has a much lower condition number, 18.4, compared to the first model built with all available predictor variables, 1.33e+16. The likelihood of multicollinearity within this new model is very low.

Additionally, the r-square value is now .106, a drop of only .015. Unfortunately, this value is still very low, and the predictor variables for this model account for only 10.6% of the model's variance and are not statistically significant.

Data Summary and Implications

The multiple regression model for this study was built using least ordinary squares (OLS) and returned an r-squared value of .106. The variables for this model do not account for a high enough variance in the model and are not statistically significant. Based on the available data set, it is not possible to build a multiple regression model that statistically significantly predicts household expenditures on perishable goods.

Therefore, this study does not support the alternative hypothesis and can not reject the null hypothesis.

One limitation of this study is the access to more variables and observations. As a starting point for developing a model, this data set was ideal in that it provided thousands of households as observations and a considerable number of variables to examine. However, additional variables like actual income, expenditures on other household needs, dietary restrictions, and

more comprehensive demographic data might aid in building a more accurate model. Unfortunately, additional data for these households is unavailable and limits the capabilities of this study.

Seeing as demand forecasting is already seen as a highly valuable tool for many businesses, with many already implementing their strategies to developing demand forecasting based on a wide array of consumer data (Traasdahl, 2021) it is my recommendation to acquire additional data on consumers beyond what is provided in this data set. Developing a successful predictive model relies on the right predictor variables, many of which may come from unconventional sources (Brea, 2020).

Peng and Kaza cite in their published research they originally sourced their data from the Nielsen Homescan Consumer Panel Dataset for 2010(Peng & Kaza, 2019). Seeing as this data set lacks sufficient variables to develop a model that statistically significantly predicts house expenditures on perishable goods, I would recommend accessing the Nielsen data to discover additional variables that might be available for these households.

Another option with this data set is to examine expenditures based on zip codes rather than households. Retailers focused on demand forecasting might find forecasting for all households in their area more worthwhile than individual households. It may be that aggregating all data by grouping data at the household level into zip code-wide data may provide stronger predictor variables for forecasting demand for an area more accurately.

Finally, a different predictive model might be better suited to predict consumer behavior based on this data set accurately. While a regression model was not a good option, other predictive models may provide business partners in the supply chain of perishable goods insights into future demand. A final recommendation for this data set is to explore options in predictive modeling, such as classification models that might predict which households are more likely to spend more on perishable goods.

Sources

Bruce, P., Bruce, A. G., & Gedeck, P. (2020). Practical statistics for data scientists: 50+ essential concepts using r and Python. O'Reilly.

Van den Broeck, M. (n.d.). *Quantifying model fit: Python*. Datacamp. Retrieved September 2022, from https://campus.datacamp.com/courses/introduction-to-regression-with-statsmodels-in-python/assessing-model-fit-e78fd9fe-6303-4048-8748-33b19c4222fe

Mitchell, M. (2019, November 8). *Selecting the correct predictive modeling technique*. Medium. Retrieved September 2022, from https://towardsdatascience.com/selecting-the-correct-predictive-modeling-technique-ba459c370d59

Singh, N. (2020, July 27). *Advantages and disadvantages of linear regression*. OpenGenus IQ: Computing Expertise & Legacy. Retrieved September 2022, from https://iq.opengenus.org/advantages-and-disadvantages-of-linear-regression/

Milano, S. (2018, December 12). *The advantages of demand forecasting*. Small Business - Chron.com. Retrieved September 2022, from

https://smallbusiness.chron.com/advantages-demand-forecasting-60405.html

Chen, C., Wang, Y., Huang, G., & Xiong, H. (2019). Hierarchical demand forecasting for factory production of Perishable Goods. 2019 IEEE International Conference on Big Data (Big Data). https://doi.org/10.1109/bigdata47090.2019.9006161

Huber, J., Gossmann, A., & Stuckenschmidt, H. (2017). Cluster-based hierarchical demand forecasting for perishable goods. Expert Systems with Applications, 76, 140–151. https://doi.org/10.1016/j.eswa.2017.01.022

Peng, Ke, 2022, "Availability of neighbourhood supermarkets and convenience stores, broader built environment context, and the purchase of fruits and vegetables in US households", https://doi.org/10.15139/S3/U9NMA9, UNC Dataverse, V1, UNF:6:21gunr4KSHmfLrBDyeu9ig== [fileUNF]

Peng, K., & Kaza, N. (2019). Availability of neighbourhood supermarkets and convenience stores, broader built environment context, and the purchase of fruits and vegetables in US households. Public Health Nutrition, 22(13), 2436–2447. https://doi.org/10.1017/s1368980019000910

Date, S. (2022, June 17). What are dummy variables and how to use them in a regression model. Medium. Retrieved August 25, 2022, from https://towardsdatascience.com/what-are-dummy-variables-and-how-to-use-them-in-a-regression-model-ee43640d573e

Kan, E. (2018, December 10). Data science 101: Is Python better than R? Medium. Retrieved August 25, 2022, from https://towardsdatascience.com/data-science-101-is-python-better-than-r-b8f258f57b0f

Traasdahl , A. (2020, June 11). How AI is Changing Perishable Food Forecasting. SupplyChainBrain RSS. Retrieved August 25, 2022, from https://www.supplychainbrain.com/blogs/1-think-tank/post/31411-how-ai-and-analytics-are-changing-fresh-and-perishable-food-forecasting

Lu, C.-J., & Chang, C.-C. (2014). A hybrid sales forecasting scheme by combining independent component analysis with K-means clustering

and support vector regression. *The Scientific World Journal*, *2014*, 1–8. https://doi.org/10.1155/2014/624017

Brea, C. (2020, November 26). *Predicting consumer demand in an unpredictable world*. Harvard Business Review. Retrieved September 2022, from https://hbr.org/2020/11/predicting-consumer-demand-in-an-unpredictable-world