**Technical Report: Team Project AAI 500**

**Introduction**

This report presents an analysis of the wholesale data for a range of products. The objective is to understand the expenditure patterns of clients across different regions and product categories.

**Data Overview**

The dataset contains information on the expenditure of clients on various products, including Fresh, Milk, Grocery, Frozen, Detergents\_Paper, and Delicassen. The data is categorized by region.

**Descriptive Statistics**

The descriptive statistics of the dataset provide insights into the distribution and central tendencies of the expenditure data.

**1-1**df = pd.read\_csv('wholesale.csv')

print (df.head(5) , "\n")

print ((df.sum() / df.sum().sum()) \* 100)

# This code runs the first 5 rows of our dataset and also runs the total proportions of each of the products, Regions and Channels

**1-2**

df.info()

# This line of code runs the information about each of the columns, describing counts, null values, data types and dimension of our dataset

**1-3**

plt.figure(figsize=(6,7))

df.groupby('Region')[['Fresh', 'Milk', 'Grocery', 'Frozen',

       'Detergents\_Paper', 'Delicassen']].sum().plot(kind='bar')

plt.xticks(rotation = 45)

plt.xlabel('Region')

plt.ylabel('Total Sales')

plt.show();

# This block of code summarizes a bar plot of the total consumption of each product per region. We wanted to visualize the products that were being bought the most per region while also analyzing the proportions of each of those products.

**1-4**

df.groupby('Region')[['Fresh', 'Milk', 'Grocery', 'Frozen',

       'Detergents\_Paper', 'Delicassen']].sum().sum(axis=1) / df.sum().sum() \* 100

# This block runs the “groupby” function to analyze the proportion of the products being sold in each region. In this case we see that Region 3 composes 73% of all total sales.

**1-5**

df.groupby('Channel')[['Fresh', 'Milk', 'Grocery', 'Frozen',

       'Detergents\_Paper', 'Delicassen']].sum().sum(axis=1) / df.sum().sum() \* 100

# This block is very similar to the previous one but we wanted to analyze how were the total sales per channel.

**1-6**

**1-7**

**1-8**

**1-9 – 1-10**

df\_products =  df.drop(columns=['Channel', 'Region'], axis= 1)

df\_products.columns

# These two lines of code represent a subset of our data, in order to utilize the

the columns in a more efficient way for the next blocks of codes we decided to drop channel and region columns

**1-11 – 1-12**

def confidence\_evaluation(df):

    num\_columns = len(df.columns)

    fig, axs = plt.subplots(num\_columns, 2, figsize=(10, num\_columns\*5)) # Adjust figsize as needed

    for idx, column in enumerate(df.columns):

        mean = np.mean(df[column])

        standard\_deviation = np.std(df[column])

        shape\_df = df[column].shape[0]

        # Calculating the Gamma distribution parameter

        variance\_gamma = standard\_deviation\*\*2

        k\_gamma = mean\*\*2 / variance\_gamma # Shape

        theta\_gamma = variance\_gamma / mean # Scale

        # Preparing plot data

        x = np.linspace(0, 150, 10000) # Generating linear space

        y = stats.gamma.pdf(x, a=k\_gamma, scale=theta\_gamma) # Gamma PDF

        # Plotting Gamma PDF

        axs[idx, 0].plot(x, y)

        axs[idx, 0].set\_yscale('log')

        axs[idx, 0].set\_title(f'{column}, Gamma PDF')

        # Simulating samples to estimate the means distribution

        number\_samples\_per\_mean = 50

        num\_means = 1000

        means = [np.random.gamma(k\_gamma, theta\_gamma, number\_samples\_per\_mean).mean() for \_ in range(num\_means)]

        # Plotting the means distribution

        sns.histplot(means, kde=True, ax=axs[idx, 1]) # KDE for smoothed curve

        axs[idx, 1].set\_title(f'{column}, Means Distribution')

        # Calculating the Standard Error and Confidence Intervals

        standard\_error = standard\_deviation / np.sqrt(shape\_df)

        lower = mean - 2\*standard\_error

        upper = mean + 2\*standard\_error

        # Print the confidence interval and the mean

        print(f'For {column}: 95% confidence interval = [{round(lower,2)}, {round(upper,2)}], mean = {round(mean,2)}')

    plt.tight\_layout() # Adjust layout to prevent overlap

    plt.show();

# On this block of code, we created a function to plot and calculate the confidence intervals for the categories of products we wanted to analyze. On block 1-7 we saw the distributions of each of the products and resembled a gamma distribution, therefore in order to construct a sampling distribution of these categories we decided to use the Gamma function with their corresponding parameters to plot the sampling distribution a long with the confidence intervals to have an idea of where the mean of each of the products being sold was going to land. The block also plots the approximation of the probability density function of the product category to confirm that we are representing the original population distribution. The libraries we used were Seaborn, matplot.lib, Numpy, and Stats.Gamma

**1-13**

rowlabel = ['Channel 1', 'Channel 2']

collabel = ['Region 1', 'Region 2', 'Region 3']

table = pd.crosstab(df['Channel'], df['Region'], margins=False, normalize=False)

table.index = rowlabel

table.columns = collabel

table

# To check for independence of the two categorical variables we were working with we needed to create a contingency table to perform the Chi-squared test for independecnce. The code above shows how to create a contingency table.

rowlabel = ['Channel 1','Channel 2']

collabel = ['Region 1','Region 2', 'Region 3']

prop = pd.crosstab(df['Channel'], df['Region'], margins = False, normalize= True)

prop.index=rowlabel

prop.columns=collabel

prop

# On the code block on top is very similar to the previous contingency table except we wanted to see the proportion of each of the elements of the channel and region

# Convert the previously created 'table' into a

# 'Table' object from statsmodels.

table = sm.stats.Table(table)

# Print the fitted values of the table. Fitted values in the context of a contingency table

# are the expected frequencies of observations for each cell of the table under the null

# hypothesis of independence between the variables. In other words, it shows what the distribution

# of 'Channel' across 'Region' would look like if there were no association between the two.

print(table.fittedvalues)

# Perform a test for nominal association between 'Channel' and 'Region'. This test evaluates

# whether there is a statistically significant association between the two categorical variables.

# The test used here is likely a Chi-squared test of independence, which is common for this type

# of analysis. The result of this test includes the Chi-squared statistic and the p-value, among

# other details.

X2 = table.test\_nominal\_association()

print(X2)

**1-14**

sns.pairplot(data= np.sqrt(df\_products))#Plotting several scatter plots to see the correlation between each variable

plt.show();

# This block of code we used the seaborn library to plot a correlation matrix to get a better understanding of the relationship of our features and to explore how strong their relations might be before we start building a regression model for prediction.

**1-15**

fitd2 = smf.ols(formula='Grocery~  Detergents\_Paper', data = df).fit()#Fitting a Linear Model  To  Build an equation to predict how much groceries will be bought

print(fitd2.summary())

print(f'AIC: {fitd2.aic}')

#This is our first attempt to create a predicting equation for our target variable groceries. We chose groceries because the amount of sales that the Fresh product brought, most likely the trend was going to continue, but we if we bring in more of the other correlated products to predict the second highest bought product then we can strategize in maximize the related products. This Ordinary Least Square performed quite well using Detergents\_paper and Milk as the predictors but then we remove Milk to find out that the R-squared value did not change significantly when we removed Milk. The block of code also uses another test, the AIC to see how well our model is performing.

**1-16**

fitd = smf.glm(formula='Grocery~ Detergents\_Paper + Milk',family = sm.families.Gamma(link = sm.families.links.identity()) ,data = df\_products).fit()#Fitting a Linear Model  To  Build an equation to predict how much groceries will be bought

print(fitd.summary())

print(f'AIC: {fitd.aic}')

#In comparison to the previous model, this time we used a Generalized Linear Model with the Gamma family and the identity link function to see how it performed in comparison to the OLS. The result was in favor of the GLM and the same formula was used as the OLS when we compared the AIC of both models. Another detail we noticed was that by adding milk in the GLM the AIC dropped a bit more so we decided to include it in our regression model as well.

**1-17**

#Checking the predictted values agaisnt the actual points

sns.scatterplot(x = fitd.predict(), y = df\_products['Grocery'], hue = df\_products['Grocery'])

# line for perfect predictions

plt.plot(df['Grocery'], df['Grocery'], color='blue', linewidth=2)

plt.show()

# Once we chose our regression model, we wanted to check how good our model was at predicting the values so we used seaborn and the ”predict” function to create a scatter plot against the actual Grocery values. We also added a “best fit line” which represented the perfect fit of the Grocery sales.

**1-18**

sns.scatterplot(x =fitd.resid\_deviance, y = np.log(fitd.fittedvalues))

plt.title('Residual Plot')

plt.show();

# This last block of code is another scatter plot to see how the residuals of our model that we chose was performing against the fitted values or predicted values. The scatter plot shows variability when the fitted plots are converted to a log format. Generally we want to see that the points in the scatter plot to be spread and without a necessary pattern. In this care we see som dispersion among the points and seem to be clustering near the center of the x-axis or at 0. This suggest that our model still shows weakness in predicting values due to the nature of the distribution of the Groceries feature. A solution could have been to consider removing outliers from our data but, since we are working with sales, it is important to consider all points because there a vendors and regions that might do consume that amount of products.

**Conclusion**

*In conclusion, our biggest obstacle was to work with a dataset that was highly skewed but at the end we were successful in visualizing the data and understanding how sales behave in a given region and channel. The skewness of the products allowed us to explore and apply the gamma function and taught us that even with highly skewed data we are able to extract useful information for business purposes. In our linear models, we were able to predict future sales of groceries using 2 other products which can help understand future sales and trends a given region or channels is experiencing. They can also now understand what are their confidence intervals and understand were their mean sales might fall on the long-run with 95% confidence. Finding out that channel and region were independent categorical variables will help marketing create a one-size fits all concept to all region and channels when creating a marketing campaign.*

**References**

*List any references or sources used in the analysis.*