CONTENTS

PART I: RESEARCH QUESTION

RESEARCH QUESTION

Can a multiple linear regression model be constructed based on the retail dataset?

JUSTIFICATION

Understanding the factors that influence customer purchasing behavior is crucial for businesses aiming to optimize their marketing strategies and enhance customer satisfaction. By constructing a multiple linear regression model based on the retail dataset, this analysis seeks to quantify the relationships between overall purchase value and key customer attributes. This method is supported by empirical evidence, as a study on impulse buying behavior emphasizes how various factors critically influence customer decisions (Bucko et al., 2018). The findings suggest that customer demographics and transaction value are pivotal in shaping purchasing factors.

HYPOTHESIS

H_0: There are no significant factors influencing customer purchasing behavior in an online retail environment, and any observed patterns are due to random variation.

H_a: There are significant factors, such as customer demographics, purchase frequency, and transaction value, that influence customer purchasing behavior in an online retail environment with an Adjusted R-squared value of >0.65.

This analysis aims to identify critical drivers of customer behavior and assess whether the alternate hypothesis holds. Statistical methods, such as regression and cluster analysis, will examine the relationships between factors and customer purchasing behavior. The findings will offer valuable insights into how these factors influence purchasing decisions in the online retail environment.

CONTEXT

The contribution of this study to the field of data analytics and the MSDA program is in its practical use of statistical techniques and machine learning to solve real business problems. It highlights how data-driven insights can improve decision-making, customer segmentation, and marketing strategies. This study will use regression analysis to quantify the relationship between transaction value and various factors such as time of purchase, purchase frequency, and customer demographics. A study by Yue Shi (2023) demonstrated that an improved linear regression algorithm applied to e-commerce sales data increased prediction accuracy by 23% and processing time by 18% compared to the traditional method. Furthermore, enhanced clustering algorithms can effectively classify retail customers based on purchasing behaviors (Fang & Liu, 2021). Their research found that specific factors, such as purchase frequency and transaction value, significantly influence customer segmentation.

PART II: DATA COLLECTION

DATA COLLECTED

The Online Retail II data set was created by Daqing Chen and is licensed under the Creative Commons Attribution 4.0 International (CC BY 4.0) license (Chen, 2019). The dataset can be accessed through this link: https://archive.ics.uci.edu/dataset/502/online+retail+ii. There are 8 columns and 1,067,371 rows in the data set.

These are the relevant variables for the research question:

Variable	Type	Description
TotalPurchase – Dependent variable	Continuous	Total transaction value
InvoiceNo	Categorical	Unique numbers assigned to each transaction. If this code starts with the letter 'c,' it indicates a cancellation.
StockCode	Categorical	Unique numbers assigned to each distinct product
Description	Categorical	Product name
Quantity	Continuous	Quantities bought per transaction
InvoiceDate	Categorical	Day and time of transaction
UnitPrice	Continuous	Price of product
CustomerID	Categorical	Unique numbers assigned to each customer
Country	Categorical	Country where the customer resides

ADVANTAGES & DISADVANTAGES OF DATA GATHERING METHOD

As the data set is already available, no additional data collection is necessary. The existing data set will be downloaded and prepared for analysis. All canceled orders, as denoted by the InvoiceNo, starting with the letter ‘c,’ will be removed. An initial data exploration of the dataset shows that CustomerID has 243,007 missing values. All missing values will be imputed by inferring based on other available data such as InvoiceNo and StockCode. Imputing missing values is necessary to minimize bias in the analysis, ensuring the dataset remains usable when missing values cannot be ignored (Jafari, 2022). A potential drawback of imputing missing values is the risk of introducing inaccuracies that fail to represent the actual missing data accurately.

Furthermore, the time component in InvoiceDate will be removed. Since the focus is on broader trends, removing the time will simplify the analysis without losing important information. A new column will also be added that directly measures the revenue generated per transaction. This column, TotalPurchase, will be computed by multiplying UnitPrice and Quantity. Python will be utilized for all data exploration and cleaning in this analysis. Overall data sparsity is >5%.

CHALLENGES ENCOUNTERED

The study's limitations are the data's time range and the limited demographic information. The dataset only ranges for 2 years, so long-term trends or seasonal variations outside the period may not be captured. Furthermore, the demographic information only has Country and CustomerID. This limitation restricts the ability to analyze how factors like age, income, or gender influence purchasing behavior. There are no delimitations in this analysis. As Botello (2024) noted, the lack of comprehensive demographic data and limited temporal scope can significantly impact the ability to make nuanced business decisions and accurate forecasts.

Additionally, while imputing missing values was initially considered to minimize bias in the analysis, it was ultimately decided to remove the rows with missing CustomerID values. Given the large size of the dataset, this approach ensured data integrity without significantly impacting the overall analysis.

PART III: DATA EXTRACTION & PREPARATION

DATA EXTRACTION & PREPARATION PROCESS

Here are the steps for preparing the data:

1. Import the necessary libraries for data handling and manipulation (CSV, pandas, numpy) and visualizations (matplotlib, seaborn), and set up to ignore warnings during the execution of your code.

```
# Data Handling and Manipulation
import csv
import pandas as pd
import numpy as np

# Visualizations
import matplotlib.pyplot as plt
import seaborn as sns

# Ignore Warnings
import warnings
warnings.filterwarnings("ignore")
```

2. Specify the file path, read the CSV file using both DictReader and pandas and display the first few rows of the DataFrame.

```
# Specifiy CSV file path
file_path = r'C:\Users\kolgi\OneDrive - Western Governors University\d214\Combined_Online_Retail.csv'

# Open the CSV file and read it using DictReader
with open(file_path, 'r') as csvfile:
    csvreader = csv.DictReader(csvfile)

# Read the CSV file into a pandas DataFrame then open head
df = pd.read_csv(file_path)
```

3. Calculate the total number of elements, missing and zero values, and compute sparsity based on missing and zero values, then print the results.

```
# Calculate the total number of elements in the DataFrame
total_elements = df.size

# Calculate the number of missing values
missing_values = df.isnull().sum().sum()

# Calculate the number of zero values
zero_values = (df == 0).sum().sum()

# Calculate sparsity based on missing values only
sparsity_missing = (missing_values / total_elements) * 100

# Calculate sparsity based on missing values and zero values
sparsity_total = ((missing_values + zero_values) / total_elements) * 100

print(f"Sparsity based on missing values only: {sparsity_missing:.2f}%")
print(f"Sparsity based on missing values and zero values: {sparsity_total:.2f}%")
```

```
Sparsity based on missing values only: 2.90%
Sparsity based on missing values and zero values: 2.97%
```

4. Display the first few rows of the DataFrame, view its basic information including data types and memory usage, and generate descriptive statistics for the numerical columns in the DataFrame.

```
df.head()
```

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	12/1/2009 7:45	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	12/1/2009 7:45	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	12/1/2009 7:45	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	12/1/2009 7:45	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	12/1/2009 7:45	1.25	13085.0	United Kingdom

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1067371 entries, 0 to 1067370
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype  
---  ------       --------------    -----  
 0   Invoice      1067371 non-null  object 
 1   StockCode    1067371 non-null  object 
 2   Description  1062989 non-null  object 
 3   Quantity     1067371 non-null  int64  
 4   InvoiceDate  1067371 non-null  object 
 5   Price        1067371 non-null  float64
 6   Customer ID  824364 non-null   float64
 7   Country      1067371 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 65.1+ MB
```

```
df.describe()
```

	Quantity	Price	Customer ID
count	1.067371e+06	1.067371e+06	824364.000000
mean	9.938898e+00	4.649388e+00	15324.638504
std	1.727058e+02	1.235531e+02	1697.464450
min	-8.099500e+04	-5.359436e+04	12346.000000
25%	1.000000e+00	1.250000e+00	13975.000000
50%	3.000000e+00	2.100000e+00	15255.000000
75%	1.000000e+01	4.150000e+00	16797.000000
max	8.099500e+04	3.897000e+04	18287.000000

5. Check for missing values in each column, display the columns that have missing values, and then remove rows with missing Customer ID directly in the DataFrame.

```
# Check for missing values in each column
missing_values = df.isnull().sum()

# Display columns with missing values
print("Columns with missing values:")
print(missing_values[missing_values > 0])
```

```
Columns with missing values:
Description      4382
Customer ID    243007
dtype: int64
```

```
# Remove rows with missing Customer ID in place
df.dropna(subset=['Customer ID'], inplace=True)
```

6. Filter out rows where the 'Invoice' column starts with 'C,' as this indicates canceled orders that are unnecessary for the analysis.

```
# Filter out rows where the 'Invoice' column starts with 'C'
df = df[~df['Invoice'].str.startswith('C')]
```

7. Find duplicated rows in the DataFrame (excluding the first occurrence), count the number of duplicate rows, and then remove these duplicates from the DataFrame.

```
# Find duplicated rows
duplicates = df[df.duplicated()]

# Count the number of duplicated rows
num_duplicates = duplicates.shape[0]

print(f'Number of duplicate rows: {num_duplicates}')

# Remove duplicates
df.drop_duplicates(inplace=True)
```

```
Number of duplicate rows: 26124
```

8. Count the number of NaN values in each column of the DataFrame and print the results.

```
# Count NaN values in each column
nan_values_per_column = df.isna().sum()

print('Number of NaN values per column:')
print(nan_values_per_column)
```

```
Number of NaN values per column:
Invoice        0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
Price          0
Customer ID    0
Country        0
dtype: int64
```

9. Calculate the length of each StockCode and count the occurrences of each length, then print the results.

```
# Calculate the length of each StockCode and count occurrences
stock_code_length_counts = df['StockCode'].apply(len).value_counts()

print(stock_code_length_counts)
```

```
StockCode
5     690646
6      84992
4       1821
7       1049
1        693
2        248
12        31
3         16
Name: count, dtype: int64
```

10. Filter for StockCode values with lengths less than 5, count the occurrences of these short StockCode values, and retrieve the descriptions for them, then print the unique short StockCode values, their counts, and their descriptions.

```
# Filter for StockCodes with lengths less than 5
short_stock_codes = df[df['StockCode'].apply(len) < 5]['StockCode'].unique()

# Count occurrences of these short StockCodes
short_stock_code_counts = df[df['StockCode'].isin(short_stock_codes)]['StockCode'].value_counts()

# Get descriptions for these short StockCodes
short_stock_code_descriptions = df[df['StockCode'].isin(short_stock_codes)][['StockCode', 'Description']].drop_duplicates()

print("Unique StockCodes with lengths less than 5:")
print(short_stock_codes)

print("\nOccurrences of these StockCodes:")
print(short_stock_code_counts)

print("\nDescriptions of these StockCodes:")
print(short_stock_code_descriptions)
```

```
Unique StockCodes with lengths less than 5:
['POST' 'C2' 'M' 'PADS' 'D' 'DOT']

Occurrences of these StockCodes:
StockCode
POST    1803
M        688
C2       248
PADS      18
DOT       16
D          5
Name: count, dtype: int64

Descriptions of these StockCodes:
       StockCode                 Description
89          POST                     POSTAGE
9292          C2                   CARRIAGE
11310          M                      Manual
62299       PADS  PADS TO MATCH ALL CUSHIONS
160443         D                    Discount
842968       DOT              DOTCOM POSTAGE
```

11. Define a list of StockCode values to remove, filter out the rows where StockCode is in that list, and then display the number of rows remaining after the removal.

```python
# Define the list of StockCode values to remove
stock_codes_to_remove = ['POST', 'C2', 'M', 'PADS', 'D', 'DOT']

# Remove rows where StockCode is in the list
df = df[~df['StockCode'].isin(stock_codes_to_remove)]

# Show the number of rows after removal
print(f"Number of rows after removing specified StockCodes: {df.shape[0]}")
```

```
Number of rows after removing specified StockCodes: 776718
```

12. Identify non-capitalized descriptions in the DataFrame, count the number of unique non-capitalized descriptions, and then display the count along with the unique non-capitalized descriptions.

```python
# Identify non-capitalized descriptions
non_capitalized_descriptions = df[df['Description'].apply(lambda x: not x.isupper())]

# Count the number of non-capitalized descriptions
num_non_capitalized = non_capitalized_descriptions['Description'].nunique()

# Display non-capitalized descriptions and count
print(f"Number of non-capitalized descriptions: {num_non_capitalized}")
print(non_capitalized_descriptions['Description'].unique())
```

```
Number of non-capitalized descriptions: 29
['BAG 500g SWIRLY MARBLES' 'POLYESTER FILLER PAD 40x40cm'
 'POLYESTER FILLER PAD 60x40cm' 'POLYESTER FILLER PAD 65CMx65CM'
 'POLYESTER FILLER PAD 45x45cm' 'BAG 125g SWIRLY MARBLES'
 'ESSENTIAL BALM 3.5g TIN IN ENVELOPE' 'FOLK ART GREETING CARD,pack/12'
 'BAG 250g SWIRLY MARBLES' 'POLYESTER FILLER PAD 30CMx30CM'
 ' Bank Charges' 'This is a test product.'
 'Adjustment by john on 26/01/2010 16'
 'Adjustment by john on 26/01/2010 17' 'POLYESTER FILLER PAD 45x30cm'
 'Bank Charges' ' SET Of 6 SOLDIER SKITTLES'
 'THE KING GIFT BAG 25x24x12cm' 'Adjustment by Peter on Jun 25 2010 '
 'FRENCH BLUE METAL DOOR SIGN, No' 'NUMBER TILE COTTAGE GARDEN, No'
 'NUMBER TILE VINTAGE FONT, No ' 'NUMBER TILE VINTAGE FONT No '
 'FRENCH BLUE METAL DOOR SIGN No' 'NUMBER TILE COTTAGE GARDEN No'
 '3 TRADITIONAl BISCUIT CUTTERS  SET' 'FLOWERS HANDBAG blue and orange'
 'Next Day Carriage' 'High Resolution Image']
```

13. Define a list of unusual descriptions, count the occurrences of each description in the DataFrame, filter the counts for the specified weird descriptions, and then display the results showing how many times each weird description appears. Then, remove the rows containing these descriptions.

```
# List of weird descriptions
weird_descriptions = [
    ' Bank Charges',
    'This is a test product.',
    'Adjustment by john on 26/01/2010 16',
    'Adjustment by john on 26/01/2010 17',
    'Bank Charges',
    'Adjustment by Peter on Jun 25 2010 ',
    'Next Day Carriage',
    'High Resolution Image'
]

# Count occurrences of each weird description
description_counts = df['Description'].value_counts()

# Filter counts for the weird descriptions
weird_description_counts = {desc: description_counts.get(desc, 0) for desc in weird_descriptions}

# Display the results
for desc, count in weird_description_counts.items():
    print(f"'{desc}': {count} times")
```

```
' Bank Charges': 2 times
'This is a test product.': 12 times
'Adjustment by john on 26/01/2010 16': 18 times
'Adjustment by john on 26/01/2010 17': 14 times
'Bank Charges': 29 times
'Adjustment by Peter on Jun 25 2010 ': 3 times
'Next Day Carriage': 79 times
'High Resolution Image': 3 times
```

```
# Drop rows with the weird descriptions
df = df[~df['Description'].isin(weird_descriptions)]
```

14. Generate a new column, Total_purchase, by multiplying the Quantity and Price columns.

```
# Generate new column TotalPurchase
df['Total_purchase'] = df['Quantity'] * df['Price']
```

15. Ensure the InvoiceDate column is in datetime format, extract the day of the week and month, then apply one-hot encoding to these extracted features. Finally, display the first few rows of the DataFrame to confirm the changes.

```
# Ensure InvoiceDate is in datetime format
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'], errors='coerce')

# Extract Invoice Day of Week
df['invoice_day_of_week'] = df['InvoiceDate'].dt.strftime('%A')  # Extract weekday name

# Extract Invoice Month
df['invoice_month'] = df['InvoiceDate'].dt.month_name()  # Extract full month name

# One-hot encode the day of the week and month
df = pd.get_dummies(df, columns=['invoice_day_of_week', 'invoice_month'], prefix=['day_of_week', 'month'])

# Display the first few rows to confirm changes
print(df.head())
```

16. Create a new column is_UK, assigning 1 for rows where the Country is 'United Kingdom' and 0 for other countries, then display the first few rows to verify the changes.

```
# Create a new column 'is_UK', assigning 1 for 'United Kingdom' and 0 for others
df['is_UK'] = df['Country'].apply(lambda x: 1 if x == 'United Kingdom' else 0)

# Display the first few rows to verify
df.head()
```

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country	Total_purchase	day_of_week_Friday	...	month_February	month_
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom	83.4	0	...	0	
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom	81.0	0	...	0	

17. Filter the DataFrame to retrieve rows where the Price is 0, display the result, and remove these rows from the DataFrame. Finally, the descriptive statistics will be displayed to verify the changes.

```
# Filter the DataFrame to get rows where Price is 0
price_zero_df = df[df['Price'] == 0]

# Display the result
price_zero_df.head()
```

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country	Total_purchase	day_of_week_Friday	...	month_February	mo
4674	489825	22076	6 RIBBONS EMPIRE	12	2009-12-02 13:34:00	0.0	16126.0	United Kingdom	0.0	0	...	0	
6781	489998	48185	DOOR MAT FAIRY CAKE	2	2009-12-03 11:19:00	0.0	15658.0	United Kingdom	0.0	0	...	0	
18738	490961	22065	CHRISTMAS PUDDING TRINKET POT	1	2009-12-08 15:25:00	0.0	14108.0	United Kingdom	0.0	0	...	0	
18739	490961	22142	CHRISTMAS CRAFT WHITE FAIRY	12	2009-12-08 15:25:00	0.0	14108.0	United Kingdom	0.0	0	...	0	

```
# Remove rows where Price is 0
df = df[df['Price'] != 0]

# Display the first few rows to verify
df.describe()
```

18. Select relevant columns for outlier detection, calculate the 15th percentile (Q1) and 85th percentile (Q3), compute the interquartile range (IQR), and detect outliers based on the IQR rule. Then, count the number of outliers per column and display the rows with any outliers, along with the count of these rows.

```
# Select relevant columns for outlier detection
columns_for_outliers = ['Price', 'Quantity', 'Total_purchase']

# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = df[columns_for_outliers].quantile(0.15)
Q3 = df[columns_for_outliers].quantile(0.85)

# Compute IQR
IQR = Q3 - Q1

# Detect outliers: Values below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR
outliers = (df[columns_for_outliers] < (Q1 - 1.5 * IQR)) | (df[columns_for_outliers] > (Q3 + 1.5 * IQR))

# Count the number of outliers per column
outlier_counts = outliers.sum()

# Filter out the rows that have any outliers
outlier_rows = df[outliers.any(axis=1)]

print("Outlier counts per column:")
print(outlier_counts)

print(f"\nNumber of rows with any outliers: {outlier_rows.shape[0]}")
print(outlier_rows.head())
```

```
Outlier counts per column:
Price             16766
Quantity          36728
Total_purchase    41447
dtype: int64

Number of rows with any outliers: 70558
```

19. Use the IsolationForest model to detect outliers in the specified numerical columns (Price, Quantity, Total_purchase). Fit the model and create a new column, Is_Outlier, to identify outliers (-1 for outliers, 1 for inliers). Then, display the first few rows of the DataFrame to verify the results, along with the total counts of outliers and inliers.

```
from sklearn.ensemble import IsolationForest

# Initialize IsolationForest model with contamination level of 10%
model = IsolationForest(contamination=0.20, random_state=0)

# Select the relevant numerical columns to check for outliers
columns_for_outliers = ['Price', 'Quantity', 'Total_purchase']

# Fit the model and calculate outlier scores (fit on the selected columns)
df['Outlier_Scores'] = model.fit_predict(df[columns_for_outliers])

# Create a new column to identify outliers (-1 for outliers, 1 for inliers)
df['Is_Outlier'] = df['Outlier_Scores'].apply(lambda x: 1 if x == -1 else 0)

# Display the first few rows of the DataFrame to check results
print(df.head())

# Count the number of outliers and inliers
outlier_count = df['Is_Outlier'].sum()
inlier_count = len(df) - outlier_count

print(f"Number of outliers detected: {outlier_count}")
print(f"Number of inliers: {inlier_count}")
```

20. Separate the outliers for further analysis, remove them from the main dataset, drop the outlier-related columns (Outlier_Scores and Is_Outlier), reset the index, and then display the first few rows of the cleaned dataset.

```
# Separate the outliers for analysis
outliers_data = df[df['Is_Outlier'] == 1]

# Remove the outliers from the main dataset
df = df[df['Is_Outlier'] == 0]

# Drop the 'Outlier_Scores' and 'Is_Outlier' columns from the cleaned dataset
df = df.drop(columns=['Outlier_Scores', 'Is_Outlier'])

# Reset the index of the cleaned data
df.reset_index(drop=True, inplace=True)

# Display the first few rows of the cleaned dataset
df.head()
```

TOOLS AND TECHNIQUES USED

Python's libraries like pandas and numpy are employed for data handling, while matplotlib and seaborn are used for visualization. The sklearn library is utilized for modeling and outlier detection. After loading the dataset and necessary libraries, the data cleaning involves removing duplicates and irrelevant entries such as canceled orders and unnecessary descriptions, handling missing values, and filtering outliers using the IQR rule and IsolationForest.

JUSTIFICATION FOR TOOLS AND TECHNIQUES

Python was used for this analysis. According to Canales (2022), Python is better suited for research than R because of its powerful libraries and ease of handling large datasets more efficiently. These libraries make managing and building models like multiple linear regression easy. Python is preferred over SAS and R due to its superior flexibility and extensive open-source libraries, which offer more comprehensive solutions for data manipulation and visualization compared to SAS's proprietary system (Ochoa, 2024).

A disadvantage to Python is that it can be CPU-intensive, especially for computational-heavy tasks. This drawback may result in longer processing times for data cleaning, model training, and evaluation. Additionally, Python's memory consumption can be relatively high. This is especially noticeable when handling massive datasets like in this analysis.

PART IV: ANALYSIS

SHAPIRO-WILK TEST FOR NORMALITY

The Shapiro-Wilk test will determine whether a dataset follows a normal distribution, with a p-value greater than 0.05 indicating normality (Chopra et al., 2019). This test is crucial for accurately assessing distribution.

```
Shapiro-Wilk test for log_Quantity:
  Statistic=0.9240524172782898, p-value=0.0
  The data in log_Quantity does not appear to be normally distributed.

Shapiro-Wilk test for log_Price:
  Statistic=0.9717838168144226, p-value=0.0
  The data in log_Price does not appear to be normally distributed.

Shapiro-Wilk test for log_Total_purchase:
  Statistic=0.9538493752479553, p-value=0.0
  The data in log_Total_purchase does not appear to be normally distributed.
```

A significant disadvantage to this analysis is the failure to achieve normality, as indicated by the results of the Shapiro-Wilk test. Despite applying a natural logarithm transformation to the variables, the

dataset remained unnormalized. The persistent deviation may have impacted the validity of statistical tests and models that assume a normal distribution.

MULTIPLE LINEAR REGRESSION

Multiple Linear Regression will be used in this research because of the continuous nature of the dependent variable. This method will utilize stepwise regression to identify significant independent variables in the model (Sahay, 2016).

OLS Regression Results

Dep. Variable:	total_purchase	**R-squared:**	0.167
Model:	OLS	**Adj. R-squared:**	0.167
Method:	Least Squares	**F-statistic:**	6579.
Date:	Mon, 14 Oct 2024	**Prob (F-statistic):**	0.00
Time:	21:15:20	**Log-Likelihood:**	-6.7792e+05
No. Observations:	622817	**AIC:**	1.356e+06
Df Residuals:	622797	**BIC:**	1.356e+06
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.0021	0.007	300.678	0.000	1.989	2.015
price	0.5577	0.002	272.949	0.000	0.554	0.562
is_UK	-0.5255	0.003	-163.485	0.000	-0.532	-0.519
day_of_week_Monday	0.2040	0.006	35.893	0.000	0.193	0.215
day_of_week_Tuesday	0.2390	0.006	42.160	0.000	0.228	0.250
day_of_week_Wednesday	0.2279	0.006	40.158	0.000	0.217	0.239
day_of_week_Thursday	0.2493	0.006	44.411	0.000	0.238	0.260
day_of_week_Friday	0.2662	0.006	46.256	0.000	0.255	0.277
day_of_week_Saturday	0.8543	0.037	23.194	0.000	0.782	0.927
day_of_week_Sunday	-0.0386	0.006	-6.809	0.000	-0.050	-0.027
month_January	-0.0610	0.005	-12.412	0.000	-0.071	-0.051
month_February	-0.0700	0.005	-14.335	0.000	-0.080	-0.060
month_March	-0.0808	0.004	-18.203	0.000	-0.090	-0.072
month_April	-0.0464	0.005	-9.885	0.000	-0.056	-0.037
month_May	-0.0286	0.005	-6.310	0.000	-0.037	-0.020
month_June	-0.0866	0.004	-19.303	0.000	-0.095	-0.078
month_July	-0.0726	0.005	-15.867	0.000	-0.082	-0.064
month_August	-0.0231	0.005	-5.035	0.000	-0.032	-0.014
month_October	-0.1063	0.004	-27.149	0.000	-0.114	-0.099
month_November	-0.2065	0.004	-55.256	0.000	-0.214	-0.199
month_December	-0.1708	0.004	-40.507	0.000	-0.179	-0.163

Omnibus:	101054.785	**Durbin-Watson:**	0.638
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	26049.809
Skew:	-0.192	**Prob(JB):**	0.00
Kurtosis:	2.075	**Cond. No.**	1.08e+15

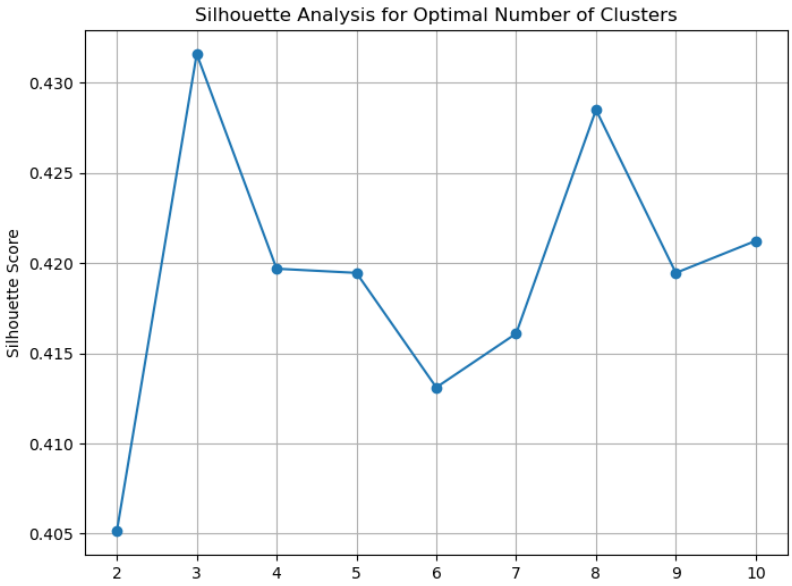
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.78e-24. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

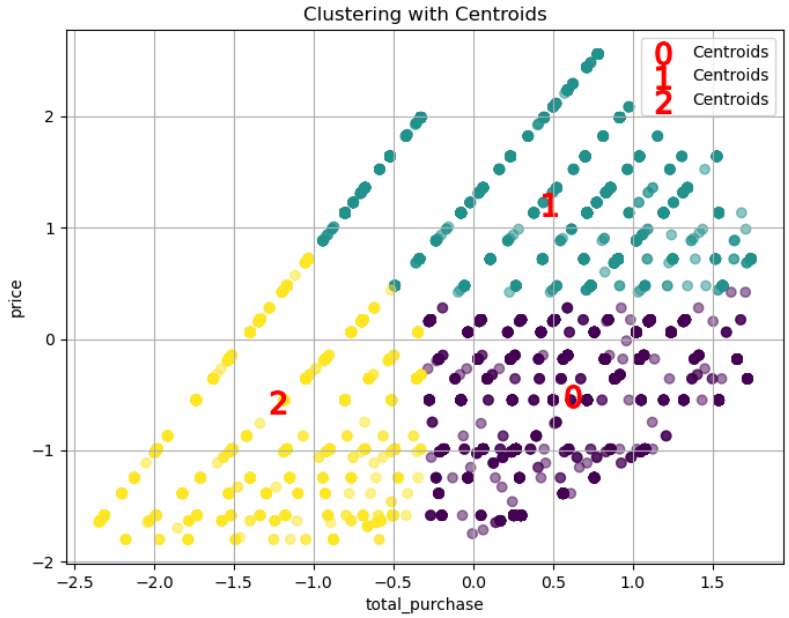
A disadvantage of multiple linear is the linearity assumption of regression. This can be restrictive, especially when the relationship between variables is non-linear. As previously discussed, achieving perfect normality was not possible for this dataset.

The F-statistic tests whether at least one of the predictors is statistically significant. The p-value of 0.00 indicates that the overall model is statistically significant. This means that the independent variables collectively explain a significant portion of the variance in total_purchase. Meanwhile, the AIC and BIC values help assess model complexity. Lower values indicate a better model fit with fewer parameters. However, the large values here reflect the considerable sample size and the complexity of the model.

CLUSTER ANALYSIS

Cluster analysis was performed to uncover patterns within the data. Hassas (n.d.) explains that "cluster analysis helps retailers segment customers, stores, and products based on key attributes, leading to more informed decisions and improved financial performance."





This graph helps determine the optimal number of clusters based on the silhouette score.

The score peaks when the number of clusters is 3. This suggests that 3 clusters represent the optimal number for the data.

It is important to note that cluster analysis can struggle with large datasets. As the computational complexity increases, clusters may be complex to interpret without clear distinctions. Moreover, cluster analysis is sensitive to the selection of variables and the clustering algorithm, which can lead to inconsistent results if not chosen carefully.

PART V: DATA SUMMARY & IMPLICATIONS

RESULTS

The analysis indicates that a multiple linear regression model can be constructed, but its predictive capabilities are limited. The current evidence aligns with the null hypothesis, which states that no significant factors influence customer purchasing behavior and that any patterns observed are due to random variation.

Key Metric	Result	
R-squared	0.167	16.7%
Adjusted R-squared	0.167	16.7%
Mean Squared Error	0.514	

The analysis revealed an adjusted r-squared of 0.17 or 17%. This indicates that the current model explains only a tiny portion of the variance in purchasing behavior. This suggests that the independent variables included in the model do not account for a significant portion of the factors influencing customer purchases. A low R-squared does not necessarily mean the model is not ideal since many complex and unobserved factors often drive customer behavior. However, this low value does signal that there is room for improvement.

Another metric is the mean squared error (MSE), which measures how well the regression model's predictions match the actual data. An MSE of 0.514 means the squared error between the predicted behavior and the actual behavior. This suggests that the model is moderately accurate, but there is still a noticeable discrepancy between the predicted and actual values.

	Cluster	total_purchase	price	is_UK
0	0	0.713623	-0.514771	0.865511
1	1	0.712062	1.215699	0.912476
2	2	-1.183804	-0.562995	0.979559

There are 3 clusters determined in the cluster analysis. Cluster 0 has customers with the highest total purchases but purchase lower-priced items. Despite purchasing lower-priced items, customers in this group still make large overall purchases. This could indicate a segment that buys frequently. These customers can be seen as potentially loyal and value cost-effective options. Cluster 1 has customers with a high total purchase and tends to purchase higher-priced items. These customers spend more on higher-priced items, thus making them a high-value segment. Cluster 2 has customers with low total purchases and buys lower-priced items. Customers with low overall spend and lower-priced items represent infrequent buyers. All three clusters have a majority of UK-based customers.

LIMITATIONS

Despite applying transformations, the low R-squared suggests that other unaccounted influential factors might be driving customer behavior. This indicates that important factors may be missing from the dataset. It can also mean that the relationships between the included variable and purchasing behavior are more complex than a linear model can capture.

Moreover, this analysis does not account for time-based factors like seasonality or trends. Ignoring these factors could lead to an incomplete understanding of customer behavior. This is especially important as purchasing behavior fluctuates significantly throughout the year.

RECOMMENDATION

Given the analysis results, it is evident that the current set of variables does not fully capture the complexity of customer purchasing behavior. While the multiple linear regression model could be

constructed, it did not provide a comprehensive explanation of the data due to the complexity of customer behavior. However, the insights from the cluster analysis provide valuable customer segmentations based on their purchasing behavior.

Use the result from the cluster analysis to develop marketing strategies for each cluster. For cluster 0 (high total purchase, low-priced items), focus on loyalty programs and bulk purchase promotions to encourage more frequent purchases. For Cluster 1 (high total purchase, high-priced items), offer exclusive deals to retain these high-value customers. For Cluster 2 (low total purchase, low-price items), engage these customers with promotions and tailored offers to boost their engagement and spending.

Additionally, consider expanding the data collection process to capture additional variables that may better explain purchases. Variables like seasonal trends, product returns, and customer demographics should be included in future analyses to predict better purchasing behavior.

FUTURE STUDY PROPOSALS

Perform time series analysis to investigate the role of seasonality and time-based factors on purchasing behavior. Examining trends, like holidays or sales periods, over time could help uncover patterns that were not captured in the current analysis. Additionally, incorporating purchase recency and frequency would help better segment customers and improve the targeting of marketing initiatives.

The limited performance of the linear regression model suggests that non-linear relationships might exist between the variables and purchasing behavior. A promising future direction would be to apply non-linear models. Consider using Random Forest, a machine learning algorithm that can model complex interactions between variables and is more flexible in capturing non-linear patterns. Another option is neural networks. These can handle complex data and may provide deeper insights into customer behavior. This approach could improve the predictive power and provide a more accurate understanding of the factors that drive purchasing behavior.

SOURCES

Botello, C. (2021, February 24). 3 reasons why retailers struggle to maximize data and how to overcome these challenges. Sada. Retrieved September 26, 2024, from https://sada.com/insights/blog/3-reasons-why-retailers-struggle-to-maximize-data-and-how-to-overcome-these-challenges/

Bucko, J., Kakalejčík, L., & Ferencová, M. (2018). Online shopping: Factors that affect consumer purchasing behaviour. *Cogent Business & Management, 5*(1). https://doi.org/10.1080/23311975.2018.1535751

Canales Luna, J. (2022, December). Python vs R for data science: Which should you learn? DataCamp. Retrieved September 20, 2024, from https://www.datacamp.com/blog/python-vs-r-for-data-science-whats-the-difference

Chen, D. (2019). Online Retail II. UCI Machine Learning Repository. https://doi.org/10.24432/C5CG6D.

Chopra, R., England, A., & Alaudeen, M. (2019). Data Science with Python : Combine Python with Machine Learning Principles to Discover Hidden Patterns in Raw Data. Packt Publishing.

Fang, C., & Liu, H. (2021). Research and Application of Improved Clustering Algorithm in Retail Customer Classification. Symmetry (20738994), 13(10), 1789. https://doi.org/10.3390/sym13101789.

Hassas, A. (n.d.). Cluster analysis applications in retail industry. Solvoyo. Retrieved October 15, 2024, from https://www.solvoyo.com/blogs/retail/cluster-analysis-applications-in-retail-industry/

Jafari, R. (2022). Hands-On Data Preprocessing in Python : Learn How to Effectively Prepare Data for Successful Data Analytics. Packt Publishing.

Ochoa, D. (2024, May 18). SAS vs Python: A comparison for data science in 2024. HalfNine. Retrieved September 26, 2024, from https://www.halfnine.com/blog/post/sas-vs-python

Sahay, A. (2016). Applied regression and modeling : a computer integrated approach (First edition.). Business Expert Press.

Shi, Y. (2023). Application of Improved Linear Regression Algorithm in Business Behavior Analysis. Procedia Computer Science, 228, 1101–1109. https://doi.org/10.1016/j.procs.2023.11.144

Srivastava, P. R., Sharma, D., & Kaur, I. (2022). Differential effects of online signals on sales performance of local brand clothing products. Journal of Enterprise Information Management, 35(6), 1522-1547. https://doi.org/10.1108/JEIM-01-2020-0039

Walker, M. (2020). Python Data Cleaning Cookbook : Modern Techniques and Python Tools to Detect and Remove Dirty Data and Extract Key Insights. Packt Publishing.