# Capstone Project:
# Data Analysis and Visualization Dashboard

## OMIS 114: Professor Wu

Alex S, Charlie G, Lucas I

# 1) Exploration - Dataset Info

**Sales Column Data Types:**

- Numeric columns
  - age
  - income
  - price
  - quantity
  - revenue
  - purchase frequency
  - avg order value
  - customer lifespan
- Categorical
  - gender
  - location
  - product category
  - brand
- Time
  - registration date
  - purchase date

**Shape:** (10000, 15)

**Added Columns:**

- customer lifetime value
- order recency

```
Sales Data Column Types:
customer_id            object
age                   float64
gender                 object
location               object
income                float64
registration_date      object
purchase_date          object
product_category       object
brand                  object
price                 float64
quantity                int64
revenue               float64
purchase_frequency      int64
avg_order_value       float64
customer_lifespan     float64
```

# 1) Exploration - Finding Missing Values

```python
#missing value checks
missing_counts = sales_data.isna().sum()
total_missing = missing_counts.sum()
missing_score = max(0, 100 - (total_missing / sales_data.size) * 100)

print(f'Missing value table: \n{missing_counts}\n')
print(f'Total missing values: \n{total_missing:.2f}\n')
print(f'Missing value score: \n{missing_score:.2f}%')
```

->

```
Missing value table:
customer_id            0
age                  792
gender               524
location             408
income               557
registration_date      0
purchase_date          0
product_category       0
brand                  0
price                  0
quantity               0
revenue                0
purchase_frequency     0
avg_order_value        0
customer_lifespan      0
dtype: int64

Total missing values:
2281.00

Missing value score:
98.48%
```

# 1) Exploration - Handling Missing Values

**Handling Outlier Values**

The method behind filling in NaN in the dataset

```
# Handling age missing values (take the median)
age_imputer = SimpleImputer(strategy='median')
sales_data['age'] = age_imputer.fit_transform(sales_data[['age']])

# Handling gender missing values (fill unkown)
sales_data['gender'] = sales_data['gender'].fillna('Unknown')

#Handling missing location values (fill with unknown)
sales_data['location'] = sales_data['location'].fillna('Unknown')

#Handling missing income values (take the median)
age_imputer = SimpleImputer(strategy='median')
sales_data['income'] = age_imputer.fit_transform(sales_data[['income']])
```

# 1) Exploration - Finding Outliers

**We performed the same method we learned in class**

- Define the bounds for each column / series
- count the number of data entries that exceed those bounds
- Display them in a new Data Frame

```python
#counting function using IQR
def count_outliers(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    return ((series < lower) | (series > upper)).sum()

#Another outlier table
outlier_table = pd.DataFrame({
    'Column': numeric_cols,
    'Outlier_Count': [count_outliers(sales_data[col]) for col in numeric_cols]
})
```

->

```
Outlier Counts (IQR):
age                   25
income               333
price                290
quantity              49
revenue              591
avg_order_value      606
customer_lifespan      0
dtype: int64

Outlier Sum:
1894
```

# 1) Exploration - Handling Outliers

**Takes the median:**

- Income (if negative)

**Cap method (IQR):**

- Income
- Avg order value
- Customer lifespan

**Unchanged:**

- Price
- Revenue
- Quantity
- Age

```
#negative check
median_income = sales_data['income'].median()
sales_data.loc[sales_data['income'] < 0, 'income'] = median_income

#IQR capping (sets outliers to the positive or negative IQR * 1.5 as defined in iqr_bounds)
low, up = iqr_bounds(sales_data['income'])
sales_data['income'] = sales_data['income'].clip(lower=low, upper=up)


# Avg_order_value : Cap
low, up = iqr_bounds(sales_data['avg_order_value'])
sales_data['avg_order_value'] = sales_data['avg_order_value'].clip(lower=low, upper=up)

# Customer_lifespan : Cap
low, up = iqr_bounds(sales_data['customer_lifespan'])
sales_data['customer_lifespan'] = sales_data['customer_lifespan'].clip(lower=low, upper=up)

# Price, Revenue, Quantity, Age : No need to change
#     Price: Statistical outliers may occur because prices might just be higher.
#     Quantity: Same with quantity. Some people might just order a lot
#     Revenue: Since it's a function of price * quantity, don't change
#     Age: Min and max are already 18 and 80, so no need to change these
```

# 2) Cleaning & Preprocessing - New Columns

**Customer Lifetime Value**

- Multiplies average order value with customer lifespan to create a “customer lifetime value” metric

**Purchase Recency:**

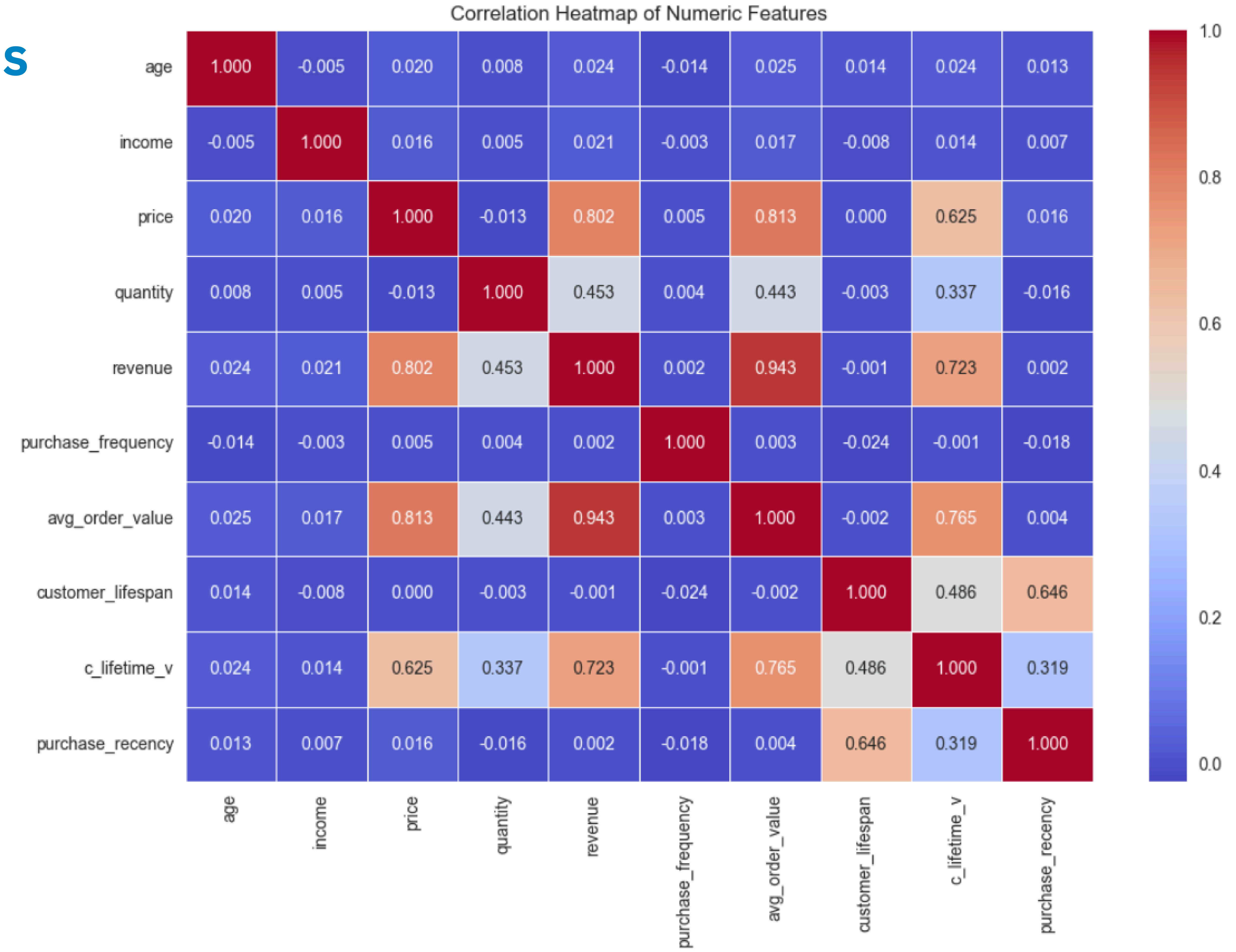
- Uses datetime features to get how long it’s been since a given customer’s last purchase

```
# Create derived features (customer lifetime value, purchase recency)
sales_data.head()

# customer lifetime value = avg order value * customer lifespan
sales_data['c_lifetime_v'] = sales_data['avg_order_value'] * sales_data['customer_lifespan']

# purchase recency = most recent purchase date (days ago)

# convert purchase date into a date and time parameter
sales_data['purchase_date'] = pd.to_datetime(
    sales_data['purchase_date'])

# calculate the number of days it has been since the last purchase with datetime operations
today = datetime.now()

sales_data['purchase_recency'] = (today - sales_data['purchase_date']).dt.days

sales_data.head()
```

# 3) Statistical Analysis

**Findings:**

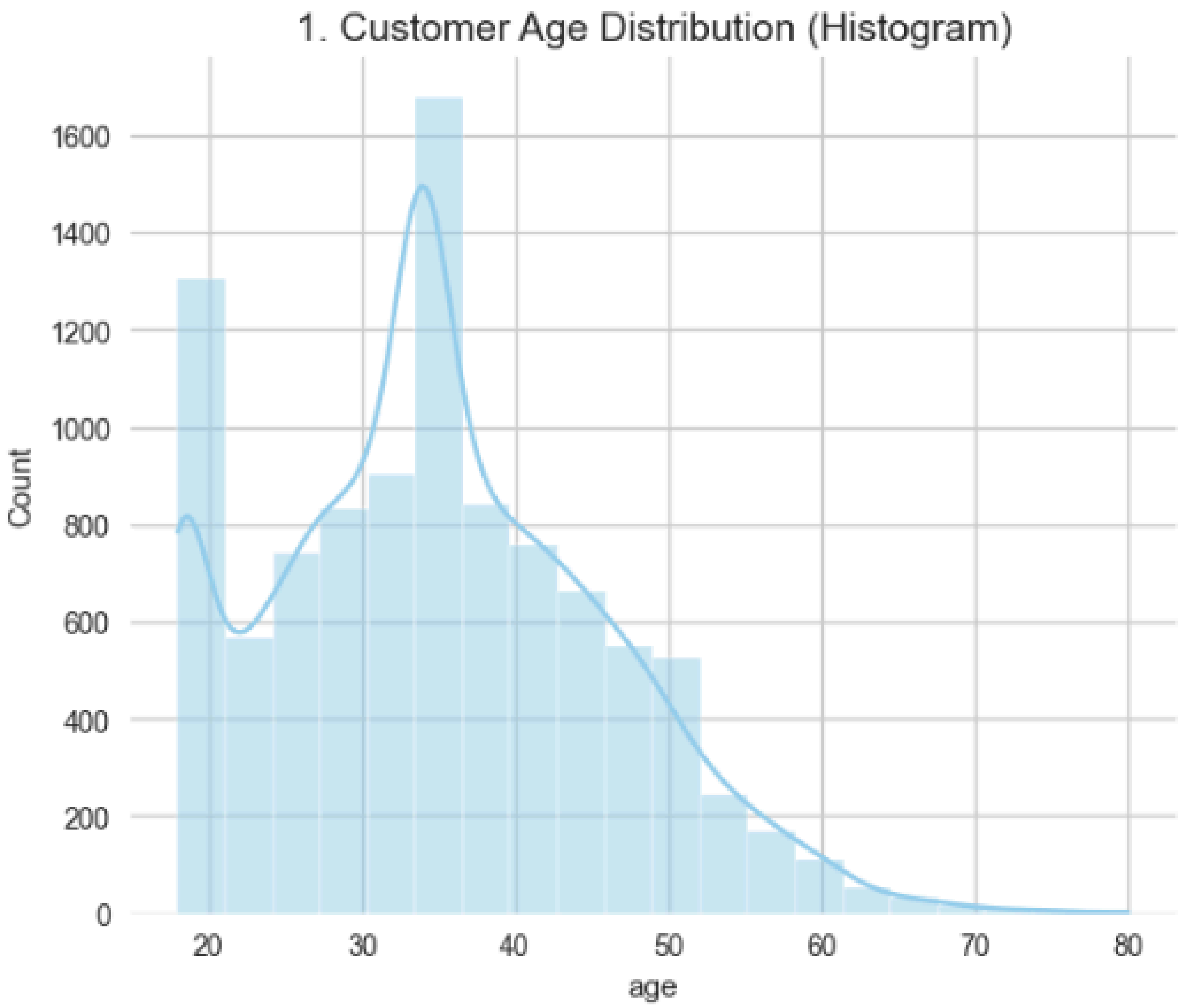
- Results that show correlation are ones that **are derived from each other**
- For example, revenue and price are highly correlated because *revenue = price * quantity*
- The further the abstraction, the lower the correlation (like customer lifetime value & quantity).
  - They aren't directly related but, there's a **few layers of calculations in between**



# 4) Data Visualization

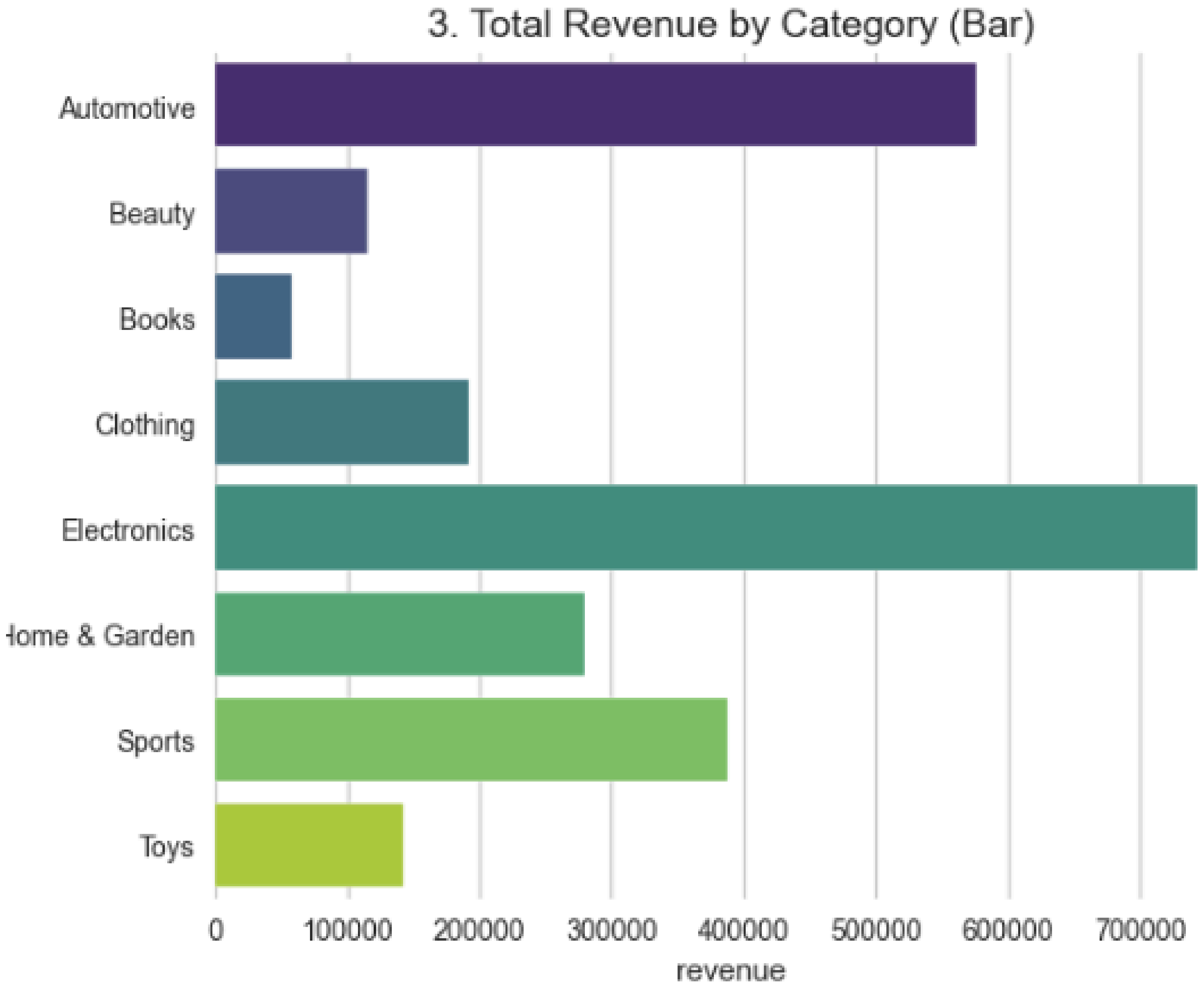
**Insight:**

Lots of 18 year old customers / younger than 18




**Insight:**

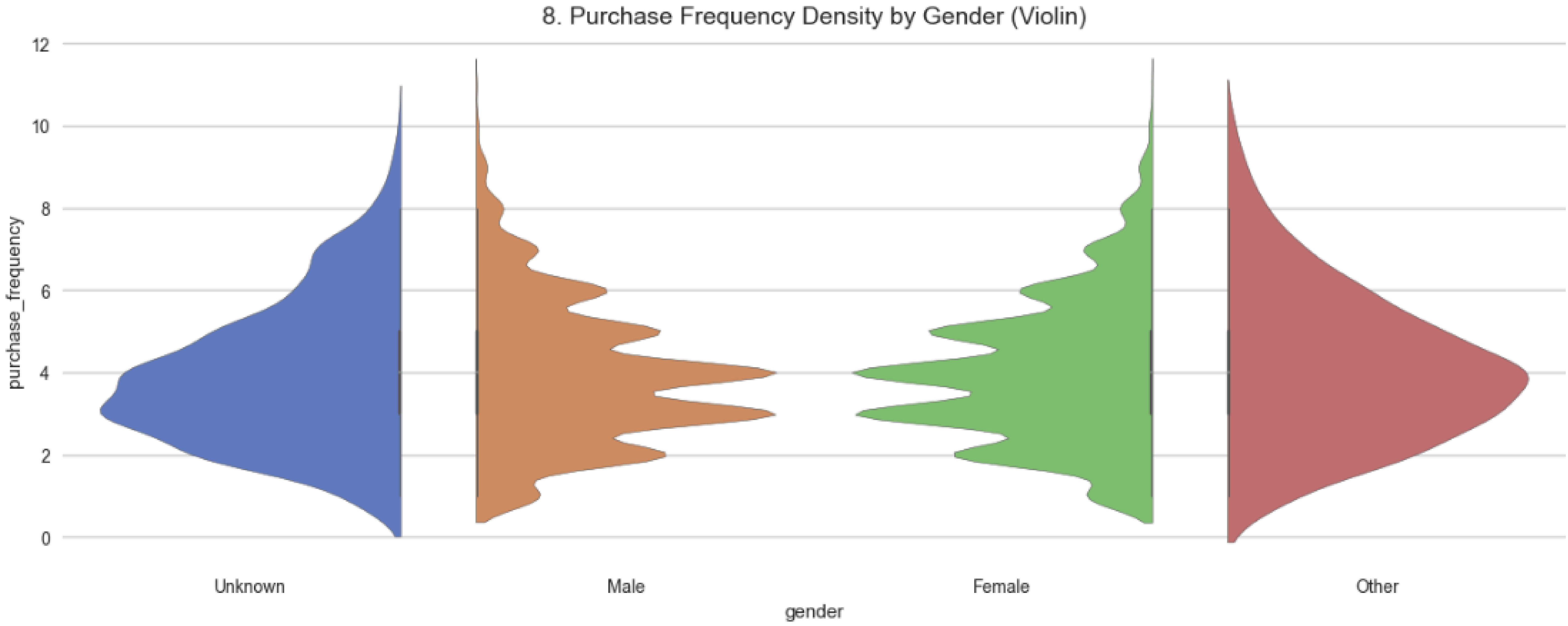
Focus on top performers (auto, electronic)
Drop low performing product categories (books, beauty)



# 4) Data Visualization

**Insight:**

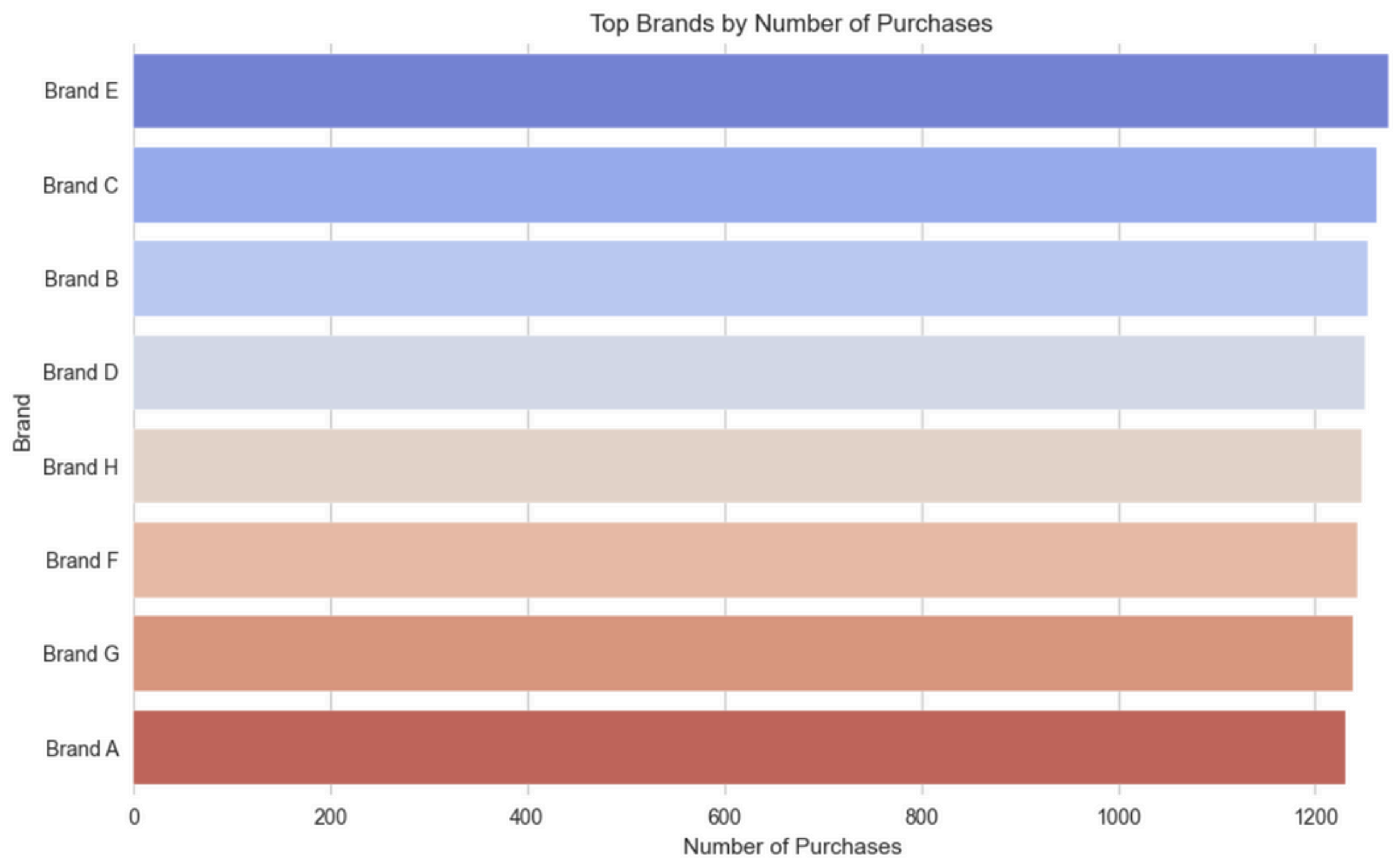
No difference in gender customer data. Here, purchase frequency is the same + gender counts are equal

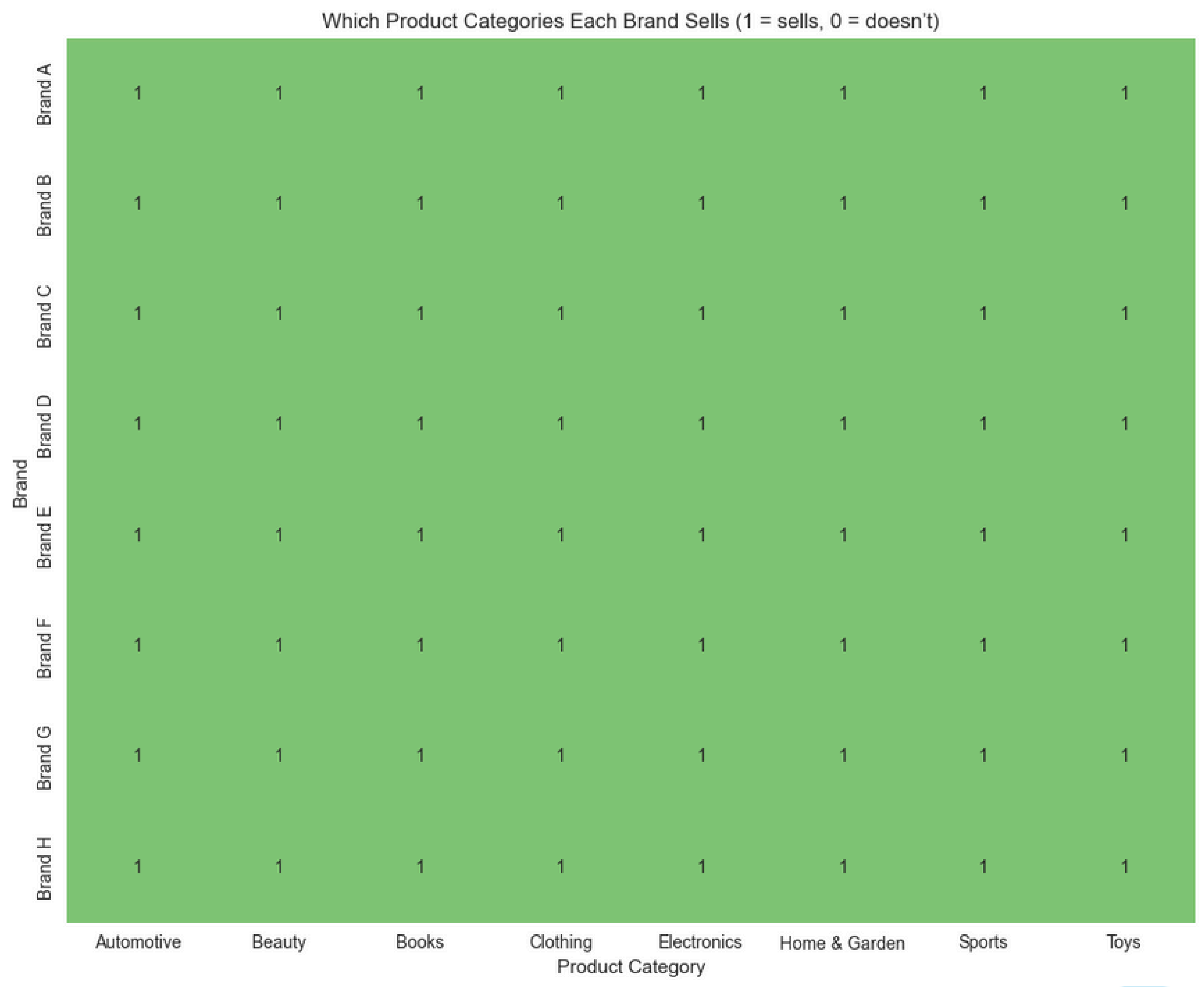


# 4) Data Visualization - Brand Analysis

**Insights:**

- Very little differentiates each brand from one another. They are each purchased the **same amount**, and they all **offer every single product category**. So, how do we determine **which ones perform the best**? → next slide



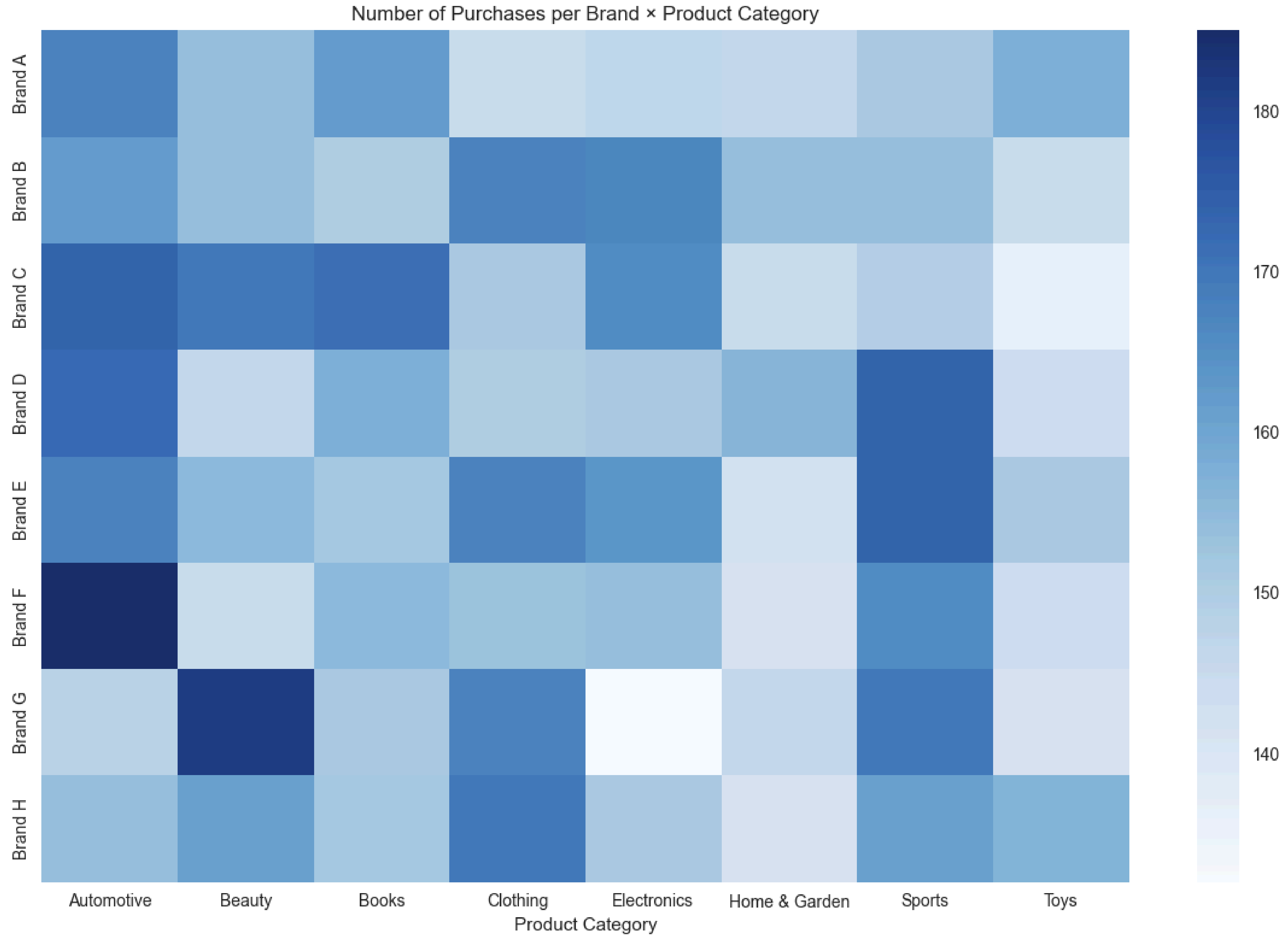
# 4) Data Visualization - Brand Analysis

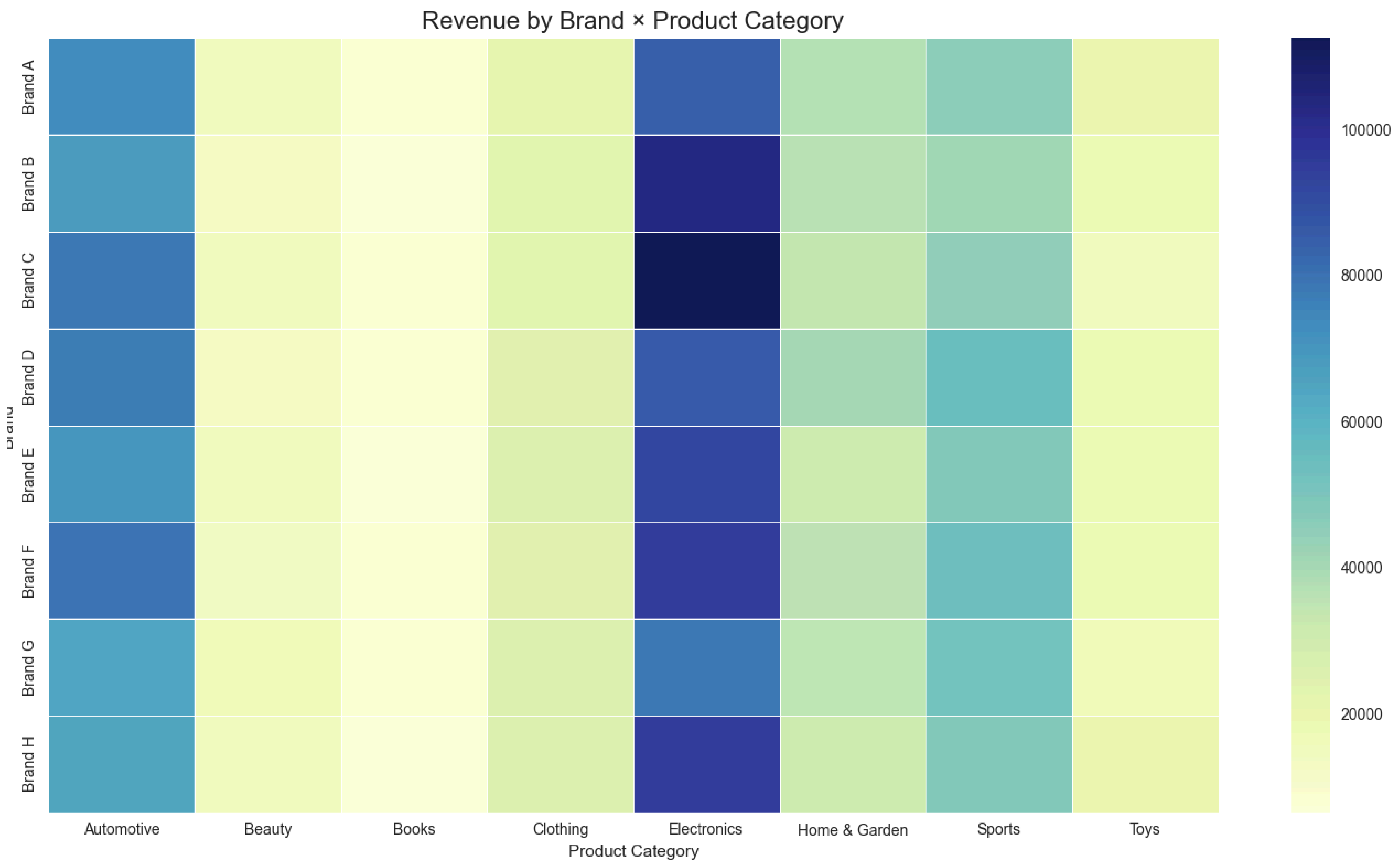
**Insights:**

- Every brand sells **every product category**
- Standouts include
  - Brand F = Automotive
  - Brand G = Beauty
  - Brand D/E = Sports

**Recommendation:**

- Focus on suppliers with **clear product offerings** and cut purchases from the **less effective suppliers**



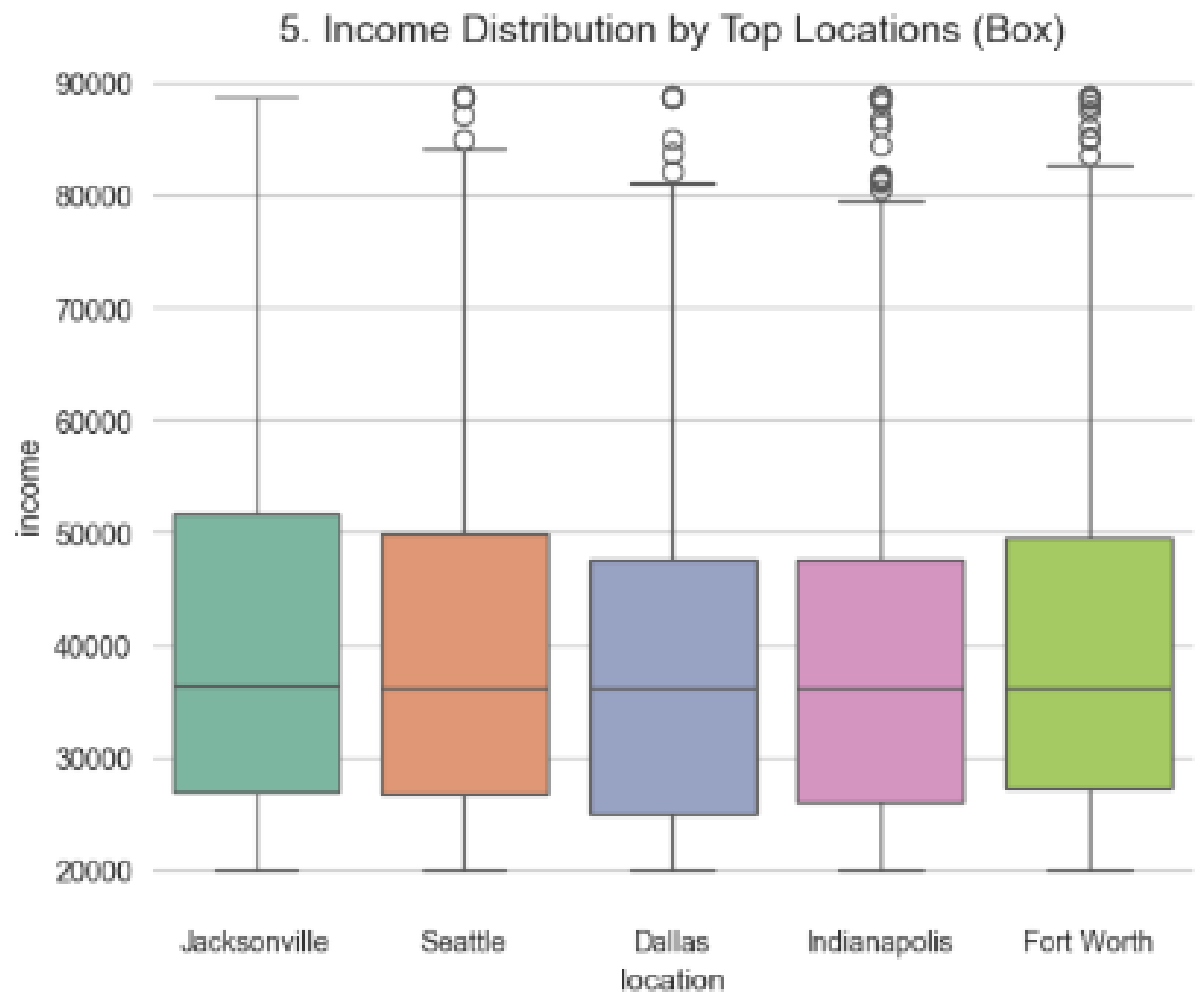


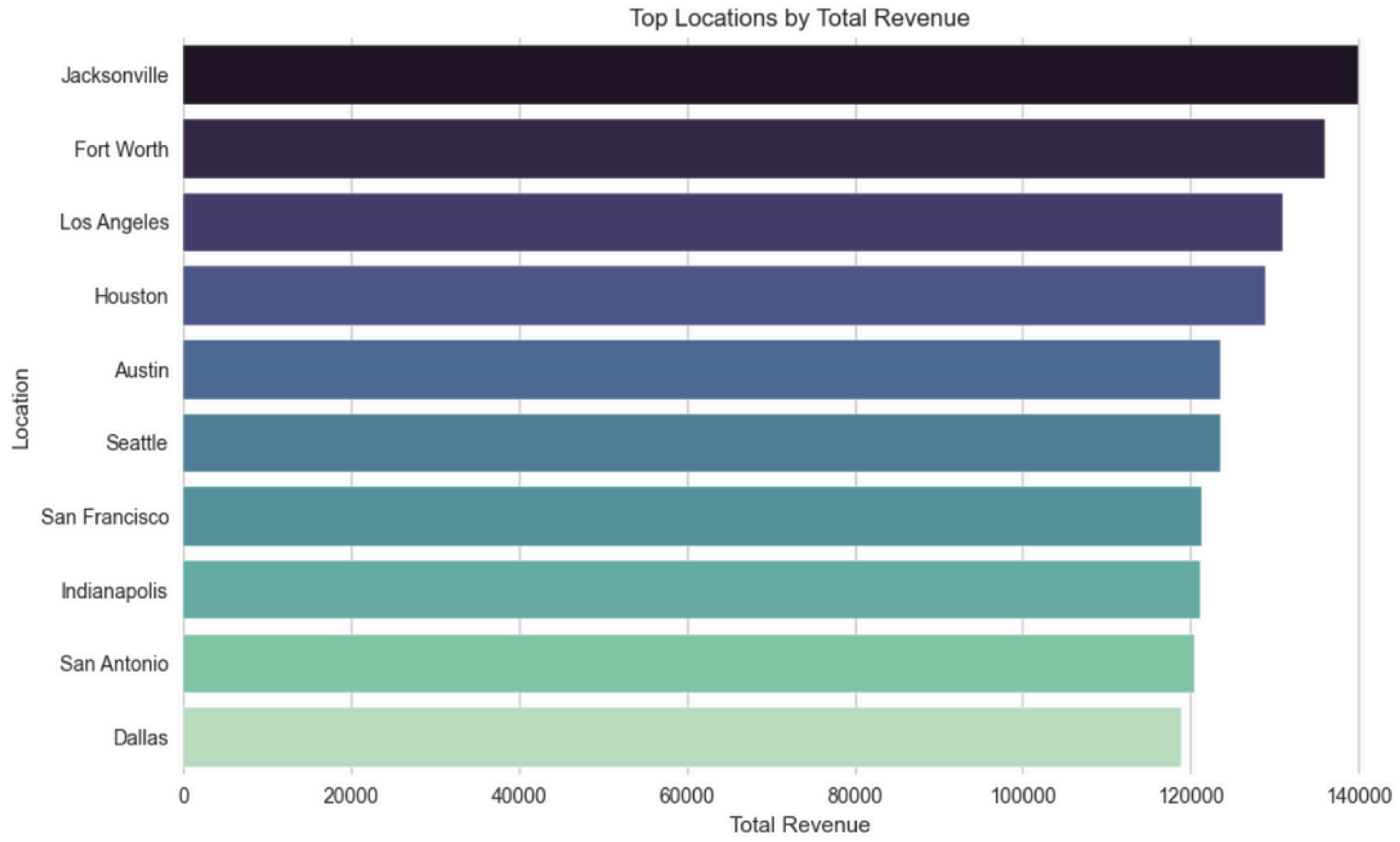
Revenue by Brand × Product Category
Brand
Brand A
Brand B
Brand C
Brand D
Brand E
Brand F
Brand G
Brand H
Automotive
Beauty
Books
Clothing
Electronics
Home & Garden
Sports
Toys
Product Category
100000
80000
60000
40000
20000

# 4) Data Visualization - Location Analysis

**Insights:**

- Sales revenue doesn't change too dramatically across locations ($140k top performer | $105k bottom performer)
- Customer makeup doesn't change much across locations (customers generally have the same income)



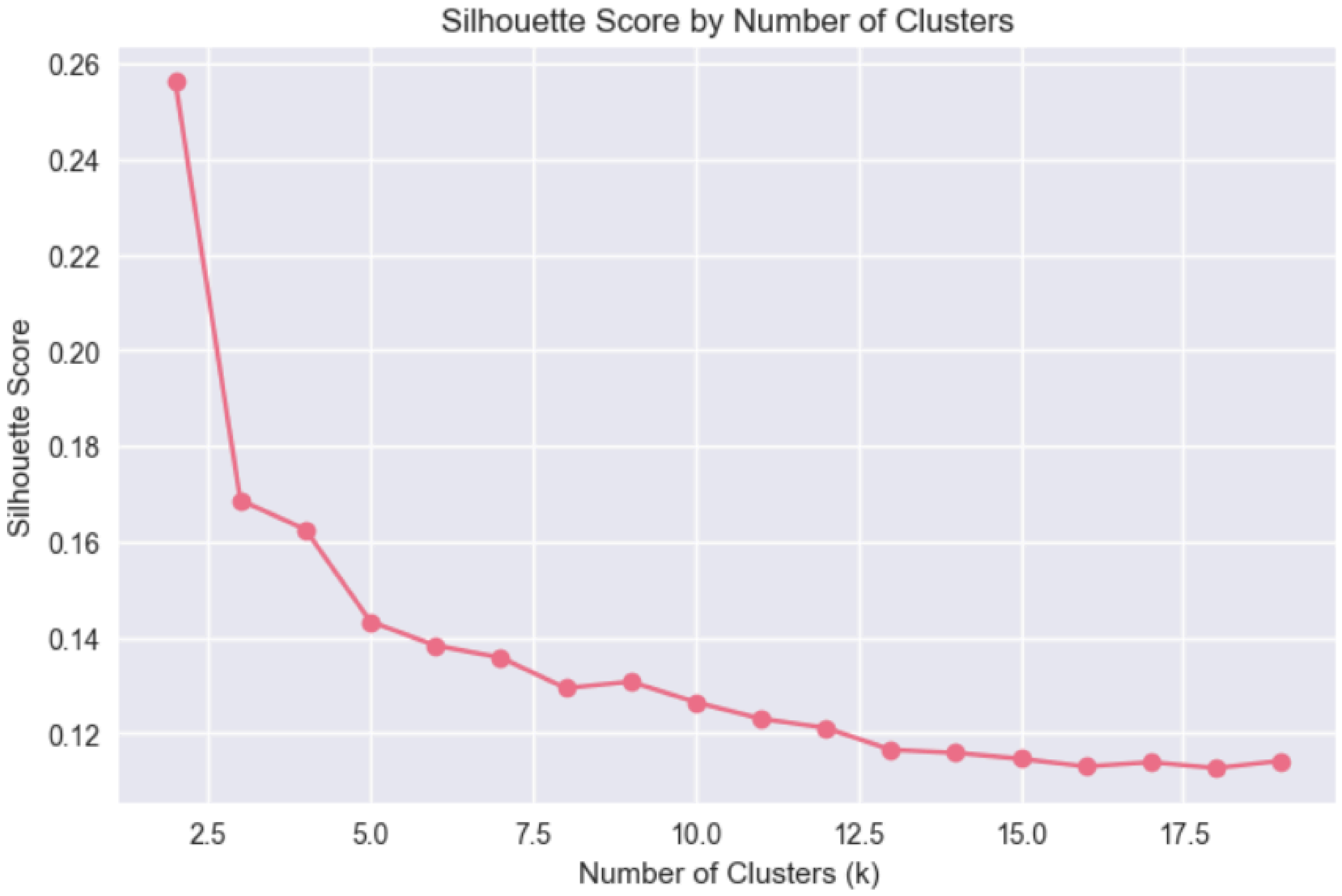
# 5) Machine Learning Implementation - Classification

**Testing the K-Means approach**

Our K Means approach used

- purchase_frequency
- avg_order_value
- revenue
- customer_lifespan
- purchase_recency
- price
- quantity
- age
- income

K Means doesn’t really make much sense here



# 5) Machine Learning Implementation - Classification

**The Goal:**

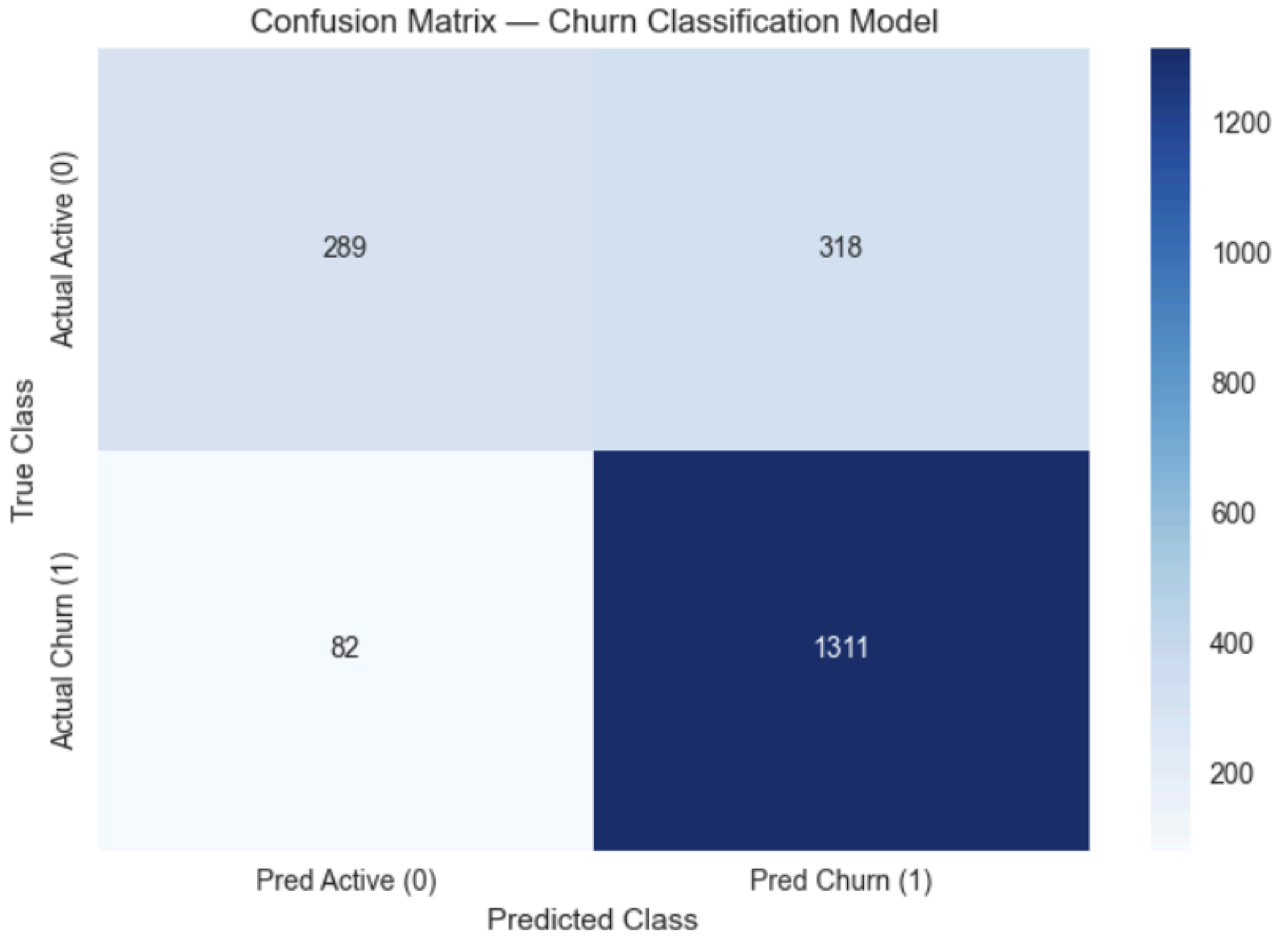
- Train a random forest classifier model that identifies churned customers (180 days no purchases) without seeing purchase history

**How:**

- Pass through <u>numeric & categorical values</u> as X (features)
- Pass through “<u>Churn</u>” as a binary Y label
  - 1 = no longer active
  - 0 = active customer

**Switching to a Customer Churn Approach**

(180 days no purchases)

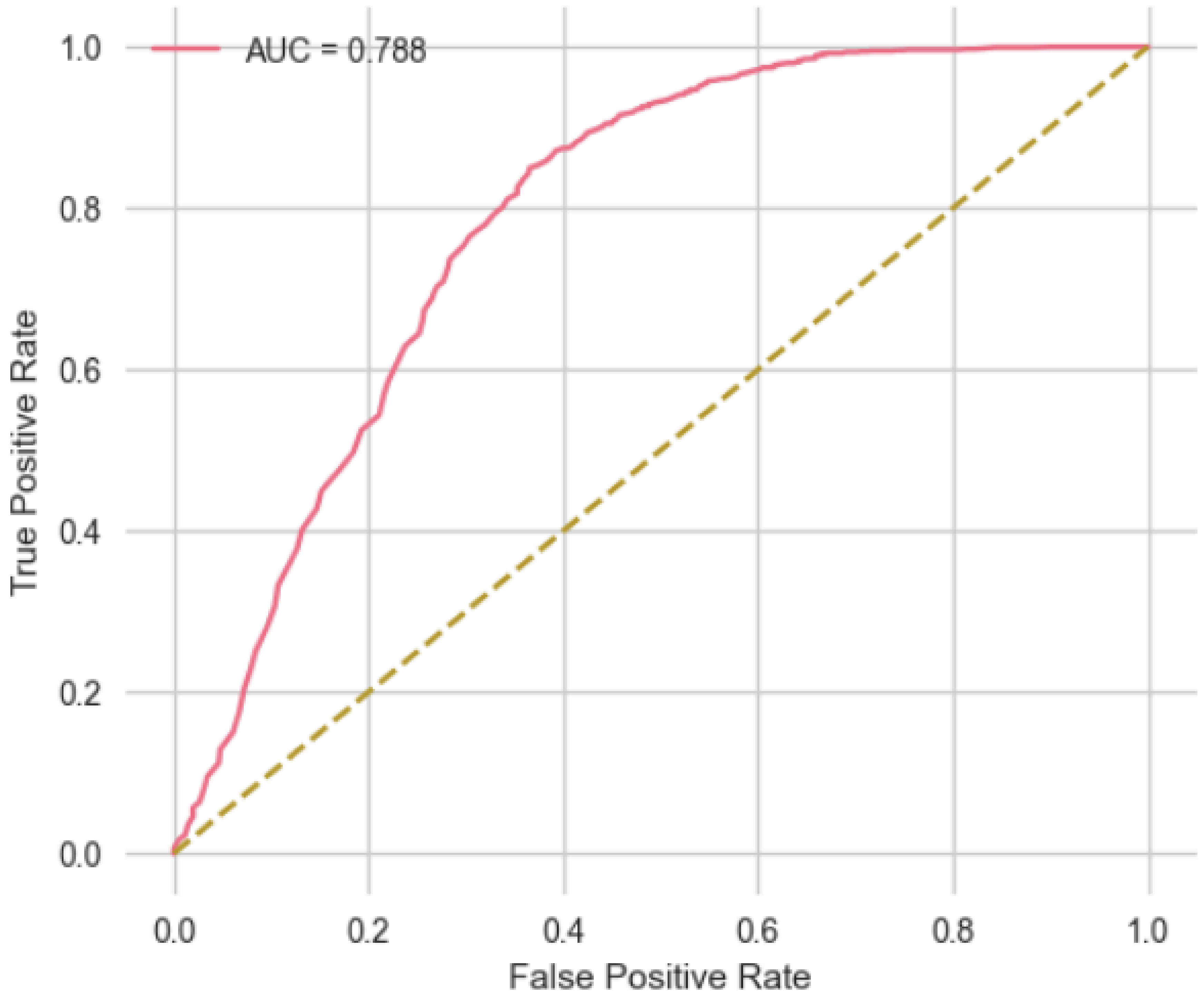


# 5) Machine Learning Implementation - Classification

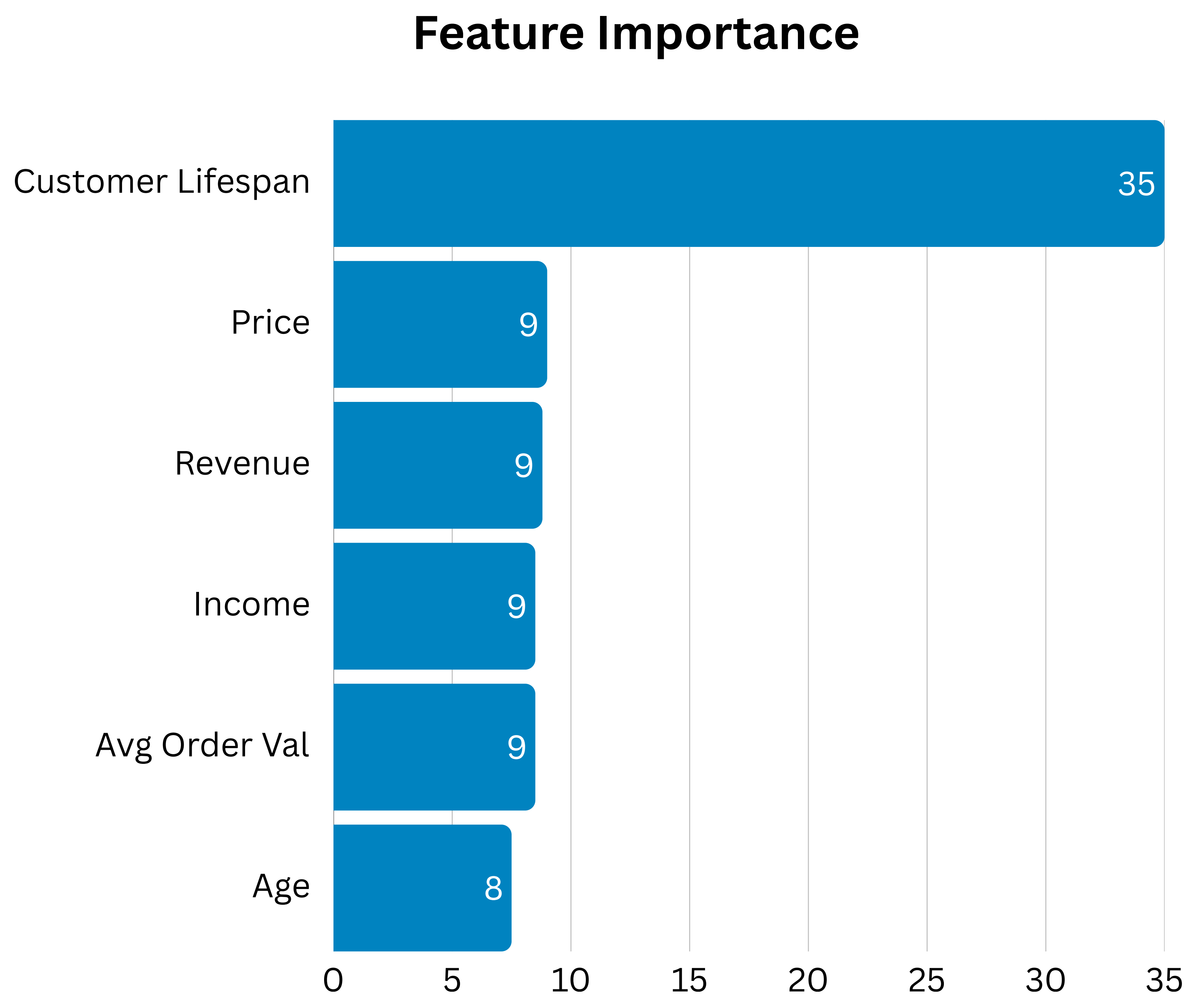
**Interpretation:**

- Significantly better than random guessing **(0.79 > 0.50)**
- Sharp increase towards the left of the chart shows:
  - Better at detecting churners
  - Frequently identifies active customers as churned customers

ROC Curve



# 5) Machine Learning Implementation - Classification



# 5) Machine Learning Implementation - Regression

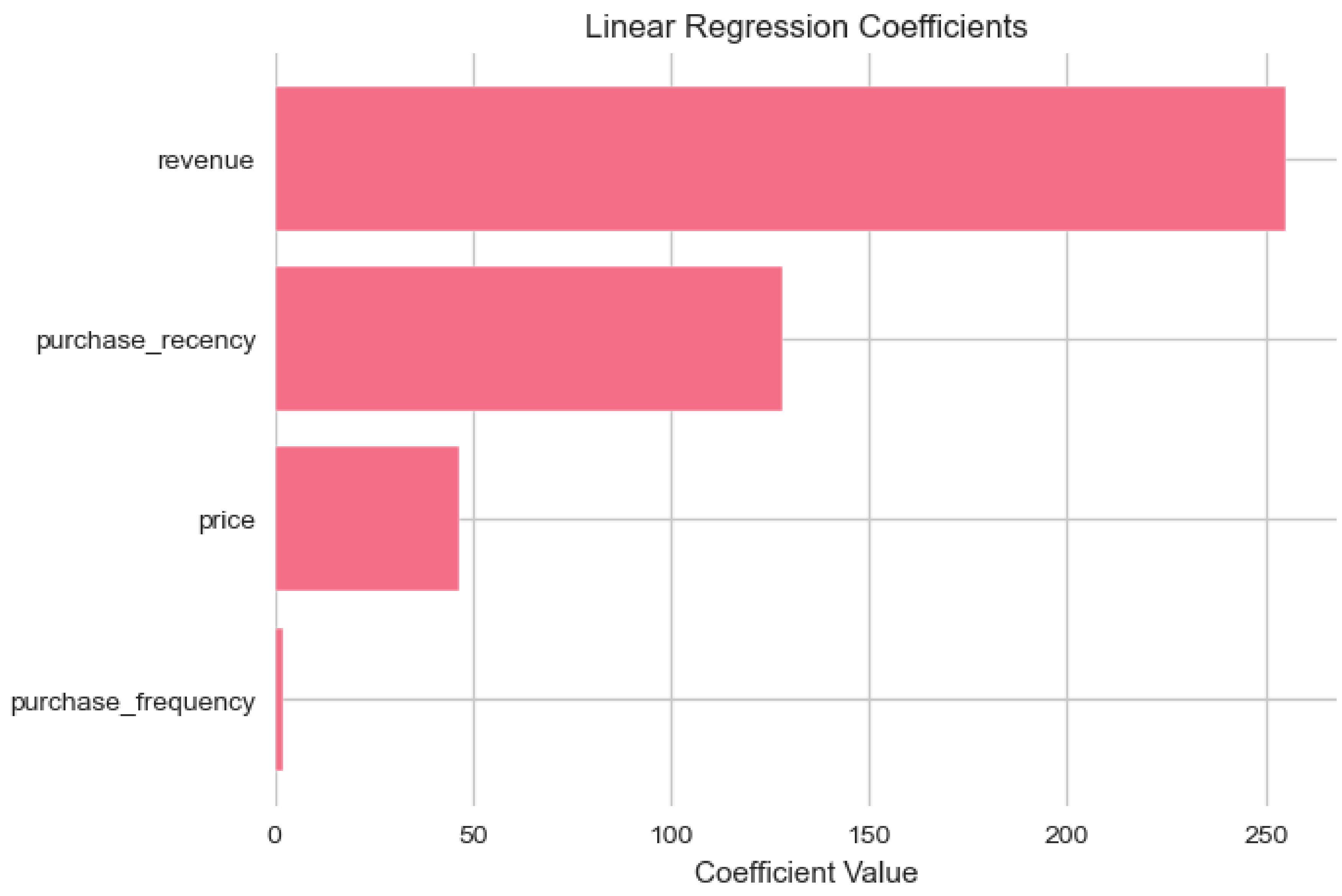
**Method**

- Linear Regression
- Target: Customer value

**Results:**

- Coefficients: [ 46.81073959, 255.2853121 2.04515846, 127.96401042]
- Intercept: 359.07
- MAE: 165.16
- MSE: 61924.18
- $R^2$: 0.61

## Feature Importance



# 6) Insights & Recommendations - Customer Churn

Our classification prediction model showed that it's **78% effective** at identifying customers who have churned vs. customers who are still active.

We can use this to our advantage by **offering deals to customers who haven't purchased for 3-6 months** to try and get their business back.

It's better at identifying churned customers than active customers (given **94% churn recall** vs 48% active recall). This is okay because the **cost of losing a customer to churn is much higher** than sending a promotional deal to an active customer.

With this method, we can **keep active customers for longer** without giving unnecessary promotions to our most active customers.

# 6) Insights & Recommendations - Products and Suppliers

Perhaps the biggest takeaway from this analysis was that **we're ordering every single product category from every single supplier**, which doesn't make much business sense.

Automotive and electronic product categories make up just about **52%** of our total revenue across all categories.

This business should **eliminate the products** and **suppliers** that **don't pull their weight**.

Even though we can't see the operational cost data, we can assume that cutting some of the fat on this business would **increase net profit in the long run.**

> “ We recommend they emphasize automotive and electronic sales. Establish a few <u>key suppliers</u> to provide these goods. ”

# Questions?