WEEK 8 PROJECT REPORT

DATA SCIENCE: RETAIL FORECASTING PROJECT

TEAM MEMBER DETAILS:

GROUP NAME: FORECAST NINJA

MEMBER NAME- ASHWINI NAGARAJU

EMAIL: ASHWINIBN3@ICLOUD.COM

COUNTRY: UNITED KINGDOM

UNIVERSITY: UNIVERSITY OF GREENWICH

SPECIALIZATION: DATA SCIENCE

PROBLEM DESCRIPTION:

The project is concerned with assisting a large beverage organization located in Australia to enhance its demand forecasting methodology. This organization sells its commodities through several supermarket chains and constantly operates promotional activities all year round. It is due to these promotions plus other elements such as holidays and seasons that great influence the demand for their product.

The company used an in-house tool for product-level weekly demand. The tool could not return accurate or even understandable forecasts for products with different time series patterns. Some had obvious seasonality, some had trends, and some had neither.

To resolve this, the firm wants to study new AI/ML- bases predicting ways that can give better exactness, higher growth ability, and clearer explainability. The aim is to swap their old setup with a stronger answer that can guess need on a week-by-week basis for each item using past data and related effecting elements.

DATA UNDERSTANDING:

The dataset contains weekly sales record of beverage products from a company operating in Australia. It includes international promotional activities and external factors affecting demand.

Dataset Overview:

Records: 1,218 rows

Features: 12 columns

Time period: Weekly data from Feb 2017 to Dec 2020

Key Features:

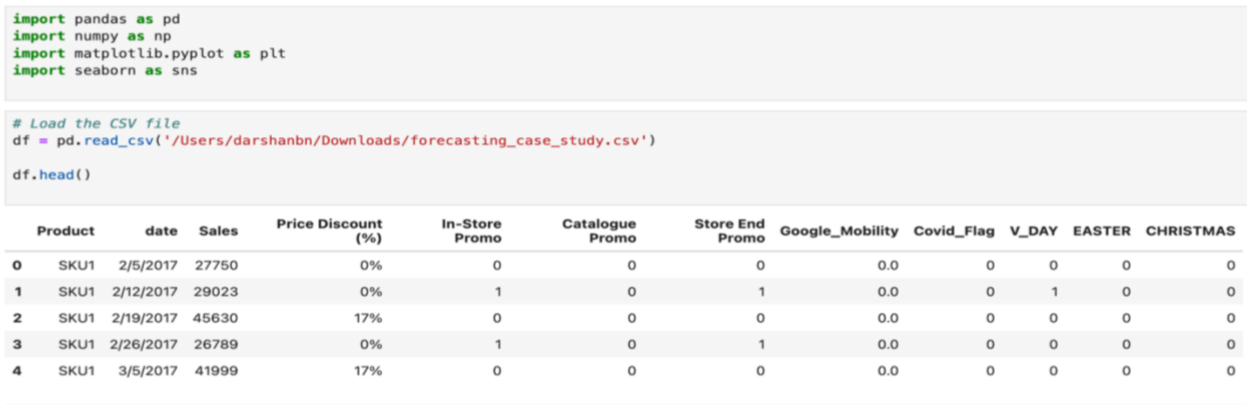
- Product: The unique product code (SKU Identifier) for each beverage item. There are 6 Unique Products in total.
- Date: The week ending date for each sales record.
- Sales: The number of units sold for that product in a given week. This is the target variable we are aiming to forecast.
- Price Discount (%)**:** The percentage discount offered on the product during the week.
- In-Store Promo**:** Indicates whether the product was part of any in-store promotional activity that week
- Catalogue Promo**:** Shows whether the product appeared in a promotional catalogue during the week
- Store End Promo: Flags if the product was placed at the store end (high-visibility area) as part of a promotion
- Google_Mobility**:** A numeric indicator reflecting customer mobility trends (e.g., footfall or movement levels)
- Covid_Flag: A binary indicator representing whether COVID-19-related restrictions were in place during that week
- V_DAY, EASTER, CHRISTMAS: These are binary flags (1 or 0) indicating whether the respective holidays occurred during the given week.

Target Variable:

Sales: This column contains the total number of units sold for a specific product in a given week. It is a numeric, continuous variable and serves as the target variable in all the forecasting models we plan to build.

Data Type:

The dataset is structured, multivariate time series data. Each row represents weekly sales for a specific product. The date column acts as the time index and the sales column is the target variable.



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load the CSV file
df = pd.read_csv('/Users/darshanbn/Downloads/forecasting_case_study.csv')

df.head()
```

	Product	date	Sales	Price Discount (%)	In-Store Promo	Catalogue Promo	Store End Promo	Google_Mobility	Covid_Flag	V_DAY	EASTER	CHRISTMAS
0	SKU1	2/5/2017	27750	0%	0	0	0	0.0	0	0	0	0
1	SKU1	2/12/2017	29023	0%	1	0	1	0.0	0	1	0	0
2	SKU1	2/19/2017	45630	17%	0	0	0	0.0	0	0	0	0
3	SKU1	2/26/2017	26789	0%	1	0	1	0.0	0	0	0	0
4	SKU1	3/5/2017	41999	17%	0	0	0	0.0	0	0	0	0

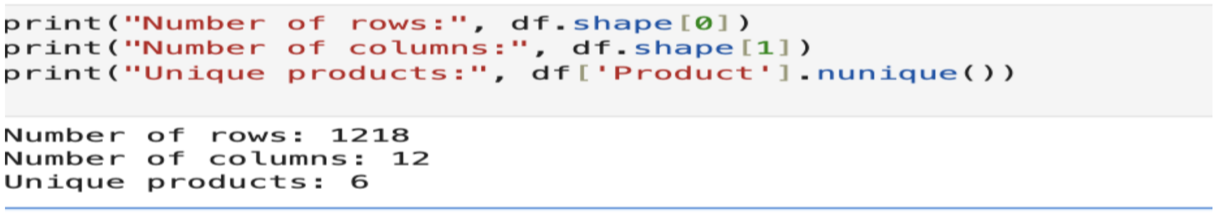
Figure 1: Sample data overview

WHAT TYPE OF DATA I GOT FOR ANALYSIS:

The dataset I am working with is a combination of time-based, numerical and categorical information that's good for forecasting models.

- It is structured as a table, with each row representing weekly sales of a product.
- The date column gives it a time-series structure-meaning trends and seasonality over time can be tracked.
- Numerical values like sales, discounts, and google mobility
- Categorical labels like product names
- Binary flags for promotions, holidays, and Covid

Because it includes multiple features that influence sales over time, I call this multivariate time series data, the right kind of data for building forecasting models using machine learning or deep learning approaches.



```
print("Number of rows:", df.shape[0])
print("Number of columns:", df.shape[1])
print("Unique products:", df['Product'].nunique())
```

```
Number of rows: 1218
Number of columns: 12
Unique products: 6
```

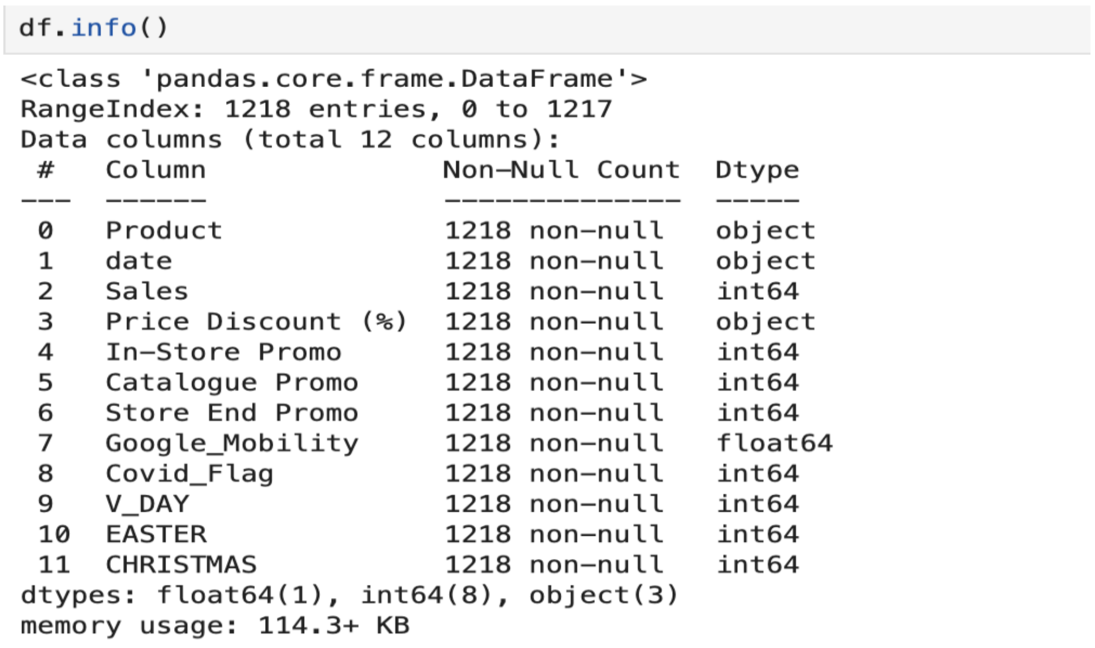
Figure 2: showing total rows and columns of the dataset.

WHAT ARE THE PROBLEMS IN THE DATA:

After exploring the dataset, identified a few issues that could affect the performance of forecasting models.

1. No missing values:

   There are no missing values in the dataset, which is a good sign and do not need to worry about filling the gaps.



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1218 entries, 0 to 1217
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Product               1218 non-null   object 
 1   date                  1218 non-null   object 
 2   Sales                 1218 non-null   int64  
 3   Price Discount (%)    1218 non-null   object 
 4   In-Store Promo        1218 non-null   int64  
 5   Catalogue Promo       1218 non-null   int64  
 6   Store End Promo       1218 non-null   int64  
 7   Google_Mobility       1218 non-null   float64
 8   Covid_Flag            1218 non-null   int64  
 9   V_DAY                 1218 non-null   int64  
 10  EASTER                1218 non-null   int64  
 11  CHRISTMAS             1218 non-null   int64  
dtypes: float64(1), int64(8), object(3)
memory usage: 114.3+ KB
```

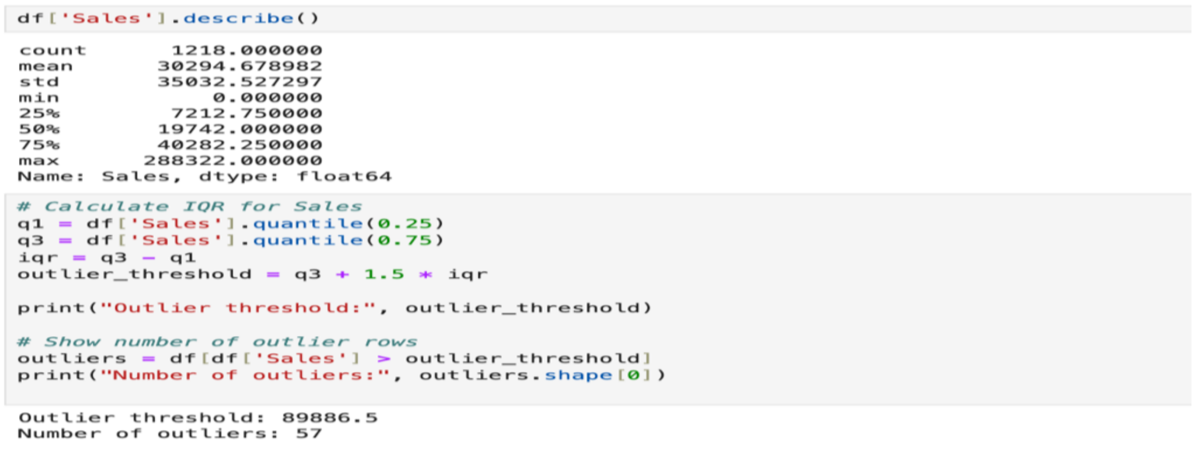
Figure 3: showing no missing values

2. Outliers in sales data

The sales column has extremely high values in some weeks. Within a reasonable range, but there are few extremely high spikes-values going above 89,000 units.

To understand this better, I used the IQR (Interquartile Range) method to calculate the threshold for what counts as an outlier.

Outliers like this can impact model performance if not handled properly. They might lead the model to focus too much on extreme weeks, reducing accuracy for regular weeks.



```
df['Sales'].describe()
```

```
count      1218.000000
mean      30294.678982
std       35032.527297
min           0.000000
25%        7212.750000
50%       19742.000000
75%       40282.250000
max      288322.000000
Name: Sales, dtype: float64
```

```
# Calculate IQR for Sales
q1 = df['Sales'].quantile(0.25)
q3 = df['Sales'].quantile(0.75)
iqr = q3 - q1
outlier_threshold = q3 + 1.5 * iqr

print("Outlier threshold:", outlier_threshold)

# Show number of outlier rows
outliers = df[df['Sales'] > outlier_threshold]
print("Number of outliers:", outliers.shape[0])
```

```
Outlier threshold: 89886.5
Number of outliers: 57
```

Figure 4: Image showing presence of extremely high values indicating potential outliers.

3. Skewed distributions

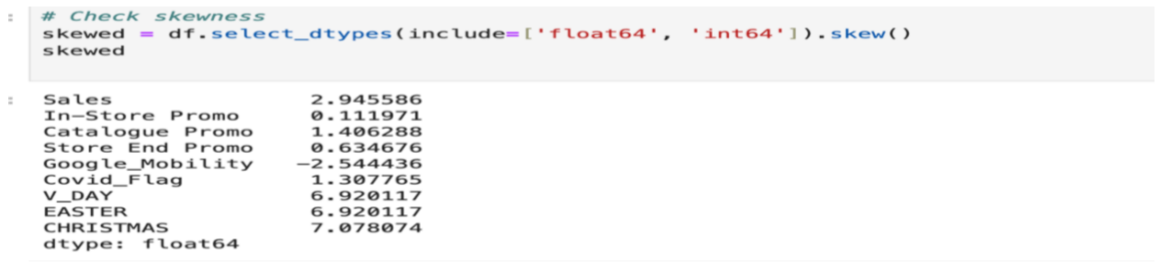
While exploring the dataset, I noticed that some columns have skewed distribution, the data isn't spread out evenly on both sides of the average. This matters because many models in machine learning assume that features are normally distributed.

The sales column is positively skewed, with most sales values on the lower side and a few very high peaks.

Holiday columns like V_day, Easter and Christmas are heavily skewed because they are active for only a few weeks in the year.

On the other side, Google_mobility is negatively skewed, like due to the impact of Covid lockdowns on customer movement.

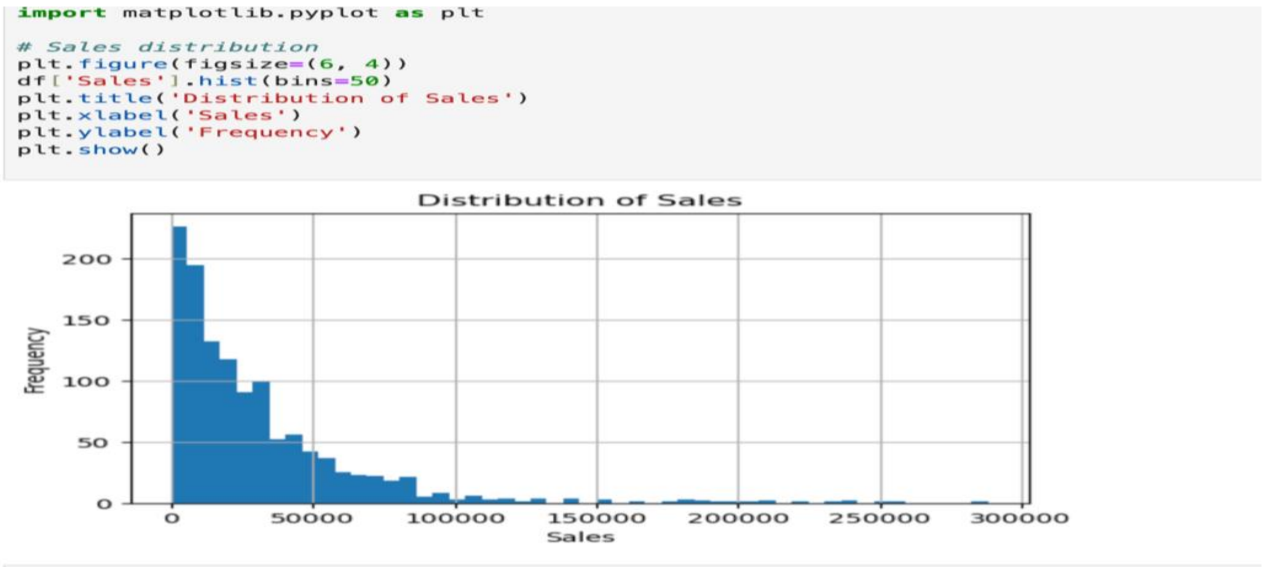
If left untreadted, these skwes can bias the model and reduce overall performance. So I plan to apply log transformations on the sales column and use proper encoding or smoothing strategies.



```
# Check skewness
skewed = df.select_dtypes(include=['float64', 'int64']).skew()
skewed
```

```
Sales                  2.945586
In-Store Promo         0.111971
Catalogue Promo        1.406288
Store End Promo        0.634676
Google_Mobility       -2.544436
Covid_Flag             1.307765
V_DAY                  6.920117
EASTER                 6.920117
CHRISTMAS              7.078074
dtype: float64
```

Figure 5: skewness values showing that sales and holiday flags are highly skewed.



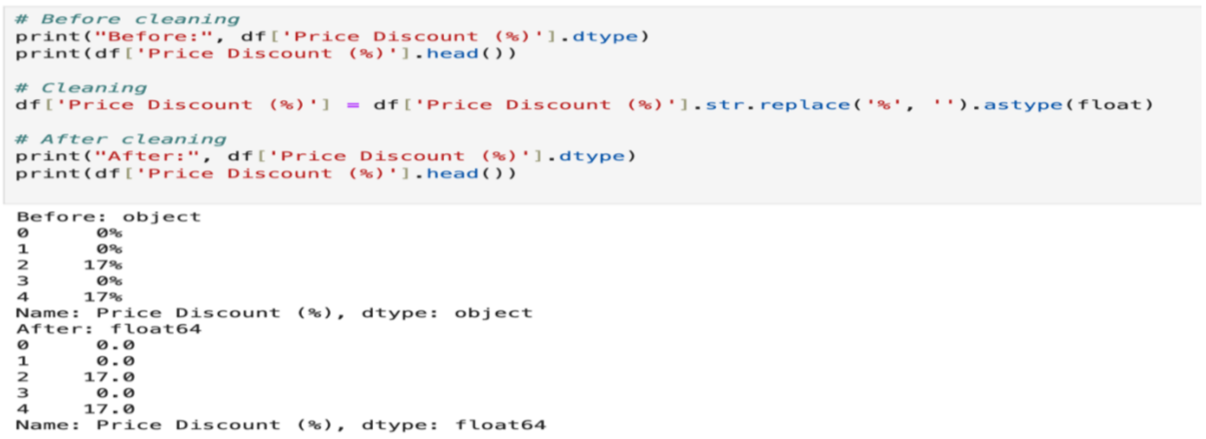
```
import matplotlib.pyplot as plt

# Sales distribution
plt.figure(figsize=(6, 4))
df['Sales'].hist(bins=50)
plt.title('Distribution of Sales')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.show()
```

Figure 6: Histogram of weekly sales showing positive skew.

4. Categorical Vs Numerical format

While checking the dataset, I noticed that the price discount (%) column was stored as text, instead of being a numerical value. For example, values were stored like 17%, 0%, etc.… which cannot be used in calculations or modeling.

To fix this I removed the % sign and converted this column to a numerical format to float.



```
# Before cleaning
print("Before:", df['Price Discount (%)'].dtype)
print(df['Price Discount (%)'].head())

# Cleaning
df['Price Discount (%)'] = df['Price Discount (%)'].str.replace('%', '').astype(float)

# After cleaning
print("After:", df['Price Discount (%)'].dtype)
print(df['Price Discount (%)'].head())
```

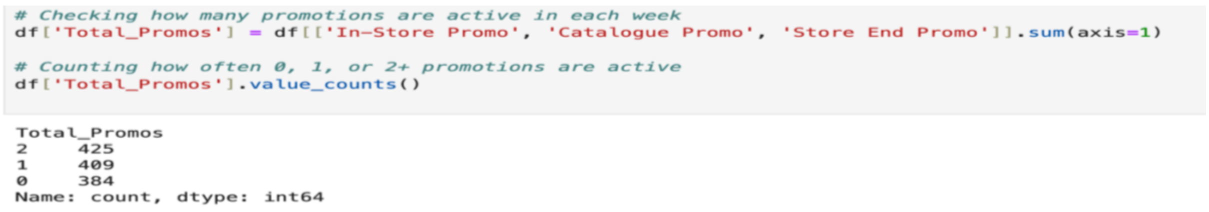
```
Before: object
0      0%
1      0%
2     17%
3      0%
4     17%
Name: Price Discount (%), dtype: object
After: float64
0      0.0
1      0.0
2     17.0
3      0.0
4     17.0
Name: Price Discount (%), dtype: float64
```

Figure 7: Before and after conversion of the price discount column to numerical format.

5. Multiple promotions in a single week

I noticed that multiple promotions were often active in the same week. For example, a product could be a part of both an in-store promo and a catalogue promo at the same time.

This overlap can confuse the model and it may struggle to understand which promotion actually influenced the sales spike. I plan to handle this later during feature engineering, possibly by combining them into a single feature.



```
# Checking how many promotions are active in each week
df['Total_Promos'] = df[['In-Store Promo', 'Catalogue Promo', 'Store End Promo']].sum(axis=1)

# Counting how often 0, 1, or 2+ promotions are active
df['Total_Promos'].value_counts()
```

```
Total_Promos
2    425
1    409
0    384
Name: count, dtype: int64
```

Figure 8: count of overlapping promotions per week.

APPROACHES TO OVERCOME THE PROBLEM:

1. Outliers in sales:

I will apply a log transformation on the sales column. This reduces the effect of extremely high values without removing them, helping the model learn more balanced patterns across both high and low sales weeks.

2. Skewed distribution:

Apply log transformation to sales. For sparse binary features like V_day, Easter and Christmas, I will monitor their impact during feature selection or use target encoding if needed.

3. Incorrect format (categorical vs numerical)

I converted the price discount (%) column from a string format to numeric by removing the % sign and casting it to float. Because model require numerical input for mathematical operations.

4. Multiple promotions in a single week

I will create a new feature called total promos, which counts how many promotion types are active in a given week. Instead of treating overlapping promotions separately, this new feature helps the model understand the intensity of promotional activity more simply and effectively.

5. Missing values

No action is needed because the dataset has no missing values, it is good thing to check for missing data.

GITHUB REPO:

https://github.com/ashwinibn3/Retail-forecasting_project