Veydant Sharma D15C 50 EXP1

EXPERIMENT 1

Aim: Introduction to Data science and Data preparation using Pandas steps.

Theory:

For machine learning algorithms to perform effectively, it is essential to preprocess raw data and convert it into a clean, usable dataset. This often involves encoding categorical data into numerical representations since most algorithms require numeric input. One common approach is to transform categorical labels into column vectors with binary values, a process known as one-hot encoding. This ensures that the categorical data is represented in a way that the algorithm can interpret without assigning implicit ordinal relationships between categories.

Another critical preprocessing step is handling missing values, which are often represented as NaNs (Not a Number). Missing values can arise from various issues, such as incomplete data collection, sensor malfunctions, or user input errors. These missing entries can lead to errors or biases in the training process if not addressed properly.

Problem Statement

The given dataset provides comprehensive details about retail product sales, focusing on the relationship between product attributes, outlet characteristics, and sales performance. This analysis aims to address the following key objectives:

- **Product Performance**: Identifying products or product types that drive the highest sales and those underperforming.
- **Outlet Insights**: Understanding the impact of outlet size, location, and type on overall sales performance.
- **Pricing Analysis**: Investigating how pricing strategies, such as maximum retail price (MRP), influence customer purchasing behavior.
- **Possible sales Prediction**: Developing models to predict item-level sales and provide actionable insights for inventory and pricing strategies.

By preprocessing the dataset and applying statistical analysis, the goal is to extract meaningful patterns that can guide data-driven decisions in retail operations.

Dataset Overview

Veydant Sharma D15C 50 EXP1

The dataset comprises **12 columns**, each detailing specific aspects of retail products, their pricing, visibility, and sales performance across outlets. Below is a breakdown of the dataset's columns and their relevance:

1. **Item_Identifier**: A unique code for each product, essential for distinguishing between items in the inventory.
2. **Item_Weight**: Represents the weight of each product, which may impact logistics and consumer preference.
3. **Item_Fat_Content**: Categorizes items as Low Fat or Regular Fat, reflecting their nutritional value and target audience.
4. **Item_Visibility**: A measure of the shelf visibility of an item, influencing its likelihood of being purchased.
5. **Item_Type**: Broad categories like Dairy, Beverages, or Snacks, helping analyze trends across product types.
6. **Item_MRP**: The maximum retail price, a key factor in determining product affordability and customer demand.
7. **Outlet_Identifier**: A unique code assigned to each retail outlet, linking products to specific store locations.
8. **Outlet_Establishment_Year**: Indicates the year the outlet began operations, useful for analyzing store maturity's effect on sales.
9. **Outlet_Size**: Categorizes stores as Small, Medium, or Large, impacting foot traffic and product demand.
10. **Outlet_Location_Type**: Describes whether the store is Urban, Suburban, or in a Tier 3 area, capturing demographic influences.
11. **Outlet_Type**: Differentiates between store types, such as Grocery Stores or Supermarkets, reflecting varying business models.
12. **Item_Outlet_Sales**: The target variable, representing the total sales of a product at a specific outlet.

Steps:

1. **Loading the Data in Pandas**

   The initial step is to load the dataset into a Pandas DataFrame. This can be done using the `read_csv` method to load the data from a CSV file, which forms the foundation for any data analysis or preprocessing tasks.

Importing of the dataset

```
[2] import pandas as pd
    df = pd.read_csv("/content/market_data.csv")
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Out
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	

2. Getting a Quick Overview of the Dataset
   To gain a better understanding of the dataset's structure, we can use the `df.info()` method, which gives details about the columns (or features) in the dataset, along with their data types.

Description of Dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Item_Identifier            8523 non-null   object 
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object 
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object 
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object 
 7   Outlet_Establishment_Year  8523 non-null   int64  
 8   Outlet_Size                6113 non-null   object 
 9   Outlet_Location_Type       8523 non-null   object 
 10  Outlet_Type                8523 non-null   object 
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

3. **Removing Irrelevant Columns**
   In most datasets, not every column contributes to the analysis. Some columns may be redundant or unnecessary. These columns only add to the size of the dataset without providing meaningful insights. For instance, in this case, the column "Outlet_Establishment_Year" is irrelevant for our analysis and can be removed to streamline the data.

Drop columns that aren't useful

```
[5] cols = ["Outlet_Establishment_Year"]
    df = df.drop(cols, axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Item_Identifier       8523 non-null   object 
 1   Item_Weight           7060 non-null   float64
 2   Item_Fat_Content      8523 non-null   object 
 3   Item_Visibility       8523 non-null   float64
 4   Item_Type             8523 non-null   object 
 5   Item_MRP              8523 non-null   float64
 6   Outlet_Identifier     8523 non-null   object 
 7   Outlet_Size           6113 non-null   object 
 8   Outlet_Location_Type  8523 non-null   object 
 9   Outlet_Type           8523 non-null   object 
 10  Item_Outlet_Sales     8523 non-null   float64
dtypes: float64(4), object(7)
memory usage: 732.6+ KB
```

Veydant Sharma   D15C   50   EXP1

4. **Eliminating Rows with Excessive Missing Values**
   Datasets often contain rows with missing values that can distort analysis results. In this step, we remove rows that have too many missing values, as they may not be useful for the analysis. The `dropna()` method helps us achieve this, ensuring that only rows with complete data are retained.
5. **Handling Missing Values**
   Upon inspecting the dataset, we find that columns have a significant number of missing entries. This can introduce bias or errors in our analysis. To tackle this, we categorize the items based on their type, then calculate the mean weight for each category. These means are then used to fill the missing values, ensuring the dataset remains consistent and accurate.
6. **Detecting Outliers (Manually)**
   Outliers can significantly affect the performance of models, so identifying and handling them is crucial. For the "Item_Outlet_Sale" column, we use the Interquartile Range (IQR) method to identify outliers. The first quartile (Q1) is 834.2474, the second quartile (Q2) is 3101.2964, and the IQR is 2267.049. Using this, we calculate the lower and upper bounds to flag outliers as values that fall outside this range. Any data points lower than the lower bound or higher than the upper bound can be considered outliers.

Item_Iden	Item_Wei	Item_Fat_	Item_Visib	Item_Type	Item_MRP	Outlet_Ide	Outlet_Est	Outlet_Siz	Outlet_Lo	Outlet_Ty	Item_Outlet_Sales
FDA46	13.6	Low Fat	0.117818	Snack Foo	192.9136	OUT049	1999	Medium	Tier 1	Supermark	2527.377
FDC02	21.35	Low Fat	0.069103	Canned	259.9278	OUT018	2009	Medium	Tier 3	Supermark	6768.523
NCP30	20.5	Low Fat	0.032835	Household	40.2822	OUT045	2002		Tier 2	Supermark	707.0796
FDY25		Low Fat	0.03381	Canned	180.5976	OUT027	1985	Medium	Tier 3	Supermark	7968.294
NCR53		Low Fat	0.144338	Health and	224.4404	OUT027	1985	Medium	Tier 3	Supermark	6976.252
FDY56	16.35	Regular	0.062764	Fruits and	227.6062	OUT017	2007		Tier 2	Supermark	7222.598
FDH19		Low Fat	0.032928	Meat	173.1738	OUT027	1985	Medium	Tier 3	Supermark	7298.5
FDY55	16.75	Low Fat	0.081253	Fruits and	256.4988	OUT013	1987	High	Tier 3	Supermark	7452.965
FDU23	17.85	Low Fat	0.147024	Breads	93.7436	OUT018	2009	Medium	Tier 3	Supermark	1134.523
DRE60	9.395	Low Fat	0.159658	Soft Drinks	224.972	OUT045	2002		Tier 2	Supermark	7696.648
DRP47	15.75	Low Fat	0.141399	Hard Drink	250.5382	OUT017	2007		Tier 2	Supermark	2775.72
FDU55	16.2	Low Fat	0.035984	Fruits and	260.6278	OUT045	2002		Tier 2	Supermark	4425.573
FDN58		Regular	0.056597	Snack Foo	230.9984	OUT027	1985	Medium	Tier 3	Supermark	9267.936
FDI44	16.1	Low Fat	0.100389	Fruits and	76.0328	OUT049	1999	Medium	Tier 1	Supermark	1853.587
FDW56		Low Fat	0.070557	Fruits and	191.2162	OUT027	1985	Medium	Tier 3	Supermark	7504.232
FDE45	12.1	Low Fat	0.040522	Fruits and	178.5002	OUT018	2009	Medium	Tier 3	Supermark	5552.106
FDR35		Low Fat	0.020597	Breads	200.0742	OUT027	1985	Medium	Tier 3	Supermark	8958.339

7. Data Scaling: Standardization and Normalization
Standardization and normalization are key techniques in preprocessing that ensure the features in

the dataset are on a similar scale. Standardization (also known as Z-score scaling) transforms the data so that it has a mean of 0 and a standard deviation of 1. Normalization, on the other hand, rescales the data to a specific range, typically between 0 and 1. Both methods are essential for improving the performance and accuracy of machine learning models, as they help avoid any one feature dominating the others due to differences in scale.

Standardization and Normalization of the data

```python
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import pandas as pd


numerical_columns = ['Item_Weight', 'Item_Visibility', 'Item_MRP', 'Item_Outlet_Sales']

# Initialize scalers
standard_scaler = StandardScaler()
minmax_scaler = MinMaxScaler()

# Standardization of the data
standardized_data = standard_scaler.fit_transform(df[numerical_columns])
df_standardized = pd.DataFrame(standardized_data, columns=numerical_columns)

# Normalization of the data
normalized_data = minmax_scaler.fit_transform(df[numerical_columns])
df_normalized = pd.DataFrame(normalized_data, columns=numerical_columns)


print("Standardized Data:")
print(df_standardized.head())

print("\nNormalized Data:")
print(df_normalized.head())
```

Conclusion:

- This experiment focused on the essential steps of data preprocessing using Pandas for data science applications.
- We worked with a retail dataset, performing initial data exploration and analysis.
- Identified and addressed missing values to maintain data integrity.
- Used the **Interquartile Range (IQR) method** to detect and eliminate outliers, ensuring dataset consistency.
- Explored **feature selection techniques** to retain the most relevant variables for analysis.
- Applied **data transformation methods** to enhance data structure and usability.
- Highlighted the significance of **standardization and normalization** in preparing data for machine learning models.