Project Title: AI – Powered Data Validation and Standardization in Supply Chain

Data Exploration and Solution Planning

1. Overview of Data Visualization and Analysis

In the phase 1 document we understand about the uncleaned or missing value data present in the supply chain and the tools used to increase the standardization of the dataset. In this phase we are doing data visualization to understand the dataset and data cleaning to remove the inconsistencies.

Objectives:

- This project aims to analyse supply chain data to gain insights and identify improvement opportunities, focusing on operational efficiency and cost management to suggest strategic improvements.
- Analysing and visualizing the Supply Chain by looking at the relationship between the price of the products and the revenue generated. Similarly, various analysis is done for supply chain dataset.
- This analysis makes use of a dataset from Kaggle.

2. Data cleaning and Preparation

2.1 Data Preparation

We are using Supply chain dataset from Kaggle. The dataset is collected from the fashion and beauty startup. The dataset is based on the supply chain of makeup products. The features included in the datasets are Product Type, SKU (Stock Keeping Unit), Price, Availability, Number of Products sold, Revenue generated, Customer demographics, Stock levels, Lead Times, Order Quantities, Shipping time, Shipping Carriers, Shipping cost, Supplier Name, Location, Lead time, Production Volumes, Manufacturing lead time, Manufacturing cost, Inspection Results, Defect rates, Transportation modes, Routes and Costs.

2.2 Handling missing values

Identify and remediate missing data in key fields like Price, Availability, and Number of Products Sold, etc.

- **Numerical Features:** Replace the missing value with the mean or median.
- **Categorical Features:** Assigned a place holder value “Unknown” for missing categories.

Code:

- # Import necessary libraries
- import pandas as pd
- import numpy as np
- # Load the dataset (replace 'Supply_Chain_Dataset.csv' with your actual file path)
- data = pd.read_csv("Supply_Chain_Dataset.csv")
- # Identify numerical features
- numerical_cols = data.select_dtypes(include=['float64', 'int64']).columns
- # Impute numerical features with the median
- data[numerical_cols] = data[numerical_cols].fillna(data[numerical_cols].median())
- # Identify categorical features
- categorical_cols = data.select_dtypes(include=['object']).columns
- # Impute categorical features with "Unknown"
- data[categorical_cols] = data[categorical_cols].fillna("Unknown")

2.3 Managing Outlier

Outlier in supply chain data can significantly impact the accuracy of analyses and the effectiveness of AI models.

- **Detection:** Visualized using boxplots to effectively visualize the distribution and identify potential outliers visually through Z-score analysis.
- **Treatment:**

  Winsorization: Replace outliers with a specified percentile value e.g., 95th percentile

Code:

- **# Calculate Z-scores for numerical features**
- for col in numerical_cols:
- data[f'{col}_Zscore'] = np.abs((data[col] - data[col].mean()) / data[col].std())
- **# Detection: Identify outliers using Z-score (e.g., Z-score > 3)**

- outliers = data[numerical_cols][data[f'{numerical_cols[0]}_Zscore'] > 3]
- outliers
- # **Winsorization** (capping at the 95th percentile)
- #for col in numerical_cols:
- upper_limit = data[col].quantile(0.95)
- data[col] = np.where(data[col] > upper_limit, upper_limit, data[col])
- data[col]
- # **Example**: Boxplot visualization for a specific numerical feature
- plt.figure(figsize=(10, 6))
- plt.boxplot(data[numerical_cols[0]], vert=False)
- plt.title(f'Boxplot of {numerical_cols[0]}')
- plt.show()
- # Analyse the identified outliers to understand their characteristics
- from sklearn.ensemble import IsolationForest
- from sklearn.preprocessing import LabelEncoder
- # Create a copy of the DataFrame to avoid modifying the original
- data_encoded = data.copy()
- # Identify categorical features
- categorical_cols = data_encoded.select_dtypes(include=['object']).columns
- # Encode categorical features using Label Encoding
- for col in categorical_cols:
- le = LabelEncoder()
- data_encoded[col] = le.fit_transform(data_encoded[col])
- # Now, fit the IsolationForest model on the encoded data
- clf = IsolationForest(contamination=0.01) # 1% of data as outliers
- clf.fit(data_encoded)
- outlier_predictions = clf.predict(data_encoded)
- # Identify outliers
- outliers = data_encoded[outlier_predictions == -1]
- outliers

2.4 Resolving Duplicates and Inconsistencies

Handle Inconsistencies in Categorical Features:

- **Standardization:** Converts categorical values to a consistent format (lowercase, remove extra spaces) for better comparisons.
- **Location Handling:** If applicable, implement logic to standardize location data (e.g., address formatting, geocoding).

Code:

- def resolve_duplicates_and_inconsistencies(df):
- **# 1. Handle Duplicate Records**
- # Corrected column names to match the DataFrame
- key_columns = ['Product type', 'Supplier name', 'SKU']
- df = df.drop_duplicates(subset=key_columns, keep='first')
- **# 2. Handle Inconsistencies in Categorical Features**
- # 2.1 Standardize categorical values (e.g., case, spaces)
- # Corrected column names to match the DataFrame
- for col in ['Product type', 'Supplier name', 'Shipping carriers', 'Transportation modes', 'Inspection results']:
- df[col] = df[col].str.strip().str.lower()
- return df
- # Example Usage:
- # Assuming 'data' is your supply chain data DataFrame
- df_clean = resolve_duplicates_and_inconsistencies(data)
- print("Cleaned Data:")
- print(df_clean)

3 Data Visualization and Insights

3.1 Tools for Visualization

The following libraries were employed to facilitate effective data analysis are:

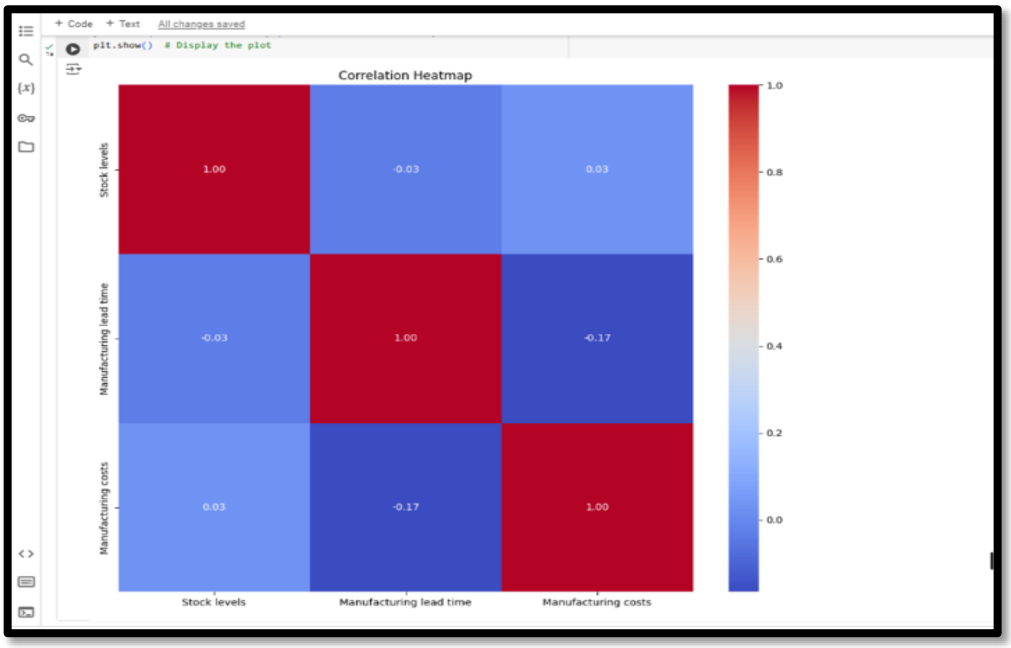
- **Matplotlib:** A foundational plotting library in Python for creating static, animated, and interactive plots.
- **Seaborn:** Seaborn simplifies the process of creating sophisticated plots and enhances the visual appeal of the graphics.
- **Plotly:** A powerful library for creating interactive, web-based visualizations.

3.2 Key Visualizations and Insights

3.2.1 Correlation Heatmap: Shows relationship between the stock levels, manufacturing lead time and manufacturing cost

Code:

- import seaborn as sns
- import matplotlib.pyplot as plt
- # Assuming 'data' is your DataFrame and you want to visualize the correlation between numerical features
- numerical_cols = data.select_dtypes(include=['float64', 'int64']).columns # Select numerical features
- correlation_matrix = data[numerical_cols].corr() # Calculate correlation matrix
- plt.figure(figsize=(12, 10)) # Adjust figure size if needed
- sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f") # Create heatmap
- plt.title('Correlation Heatmap') # Set the title of the plot
- plt.show() # Display the plot

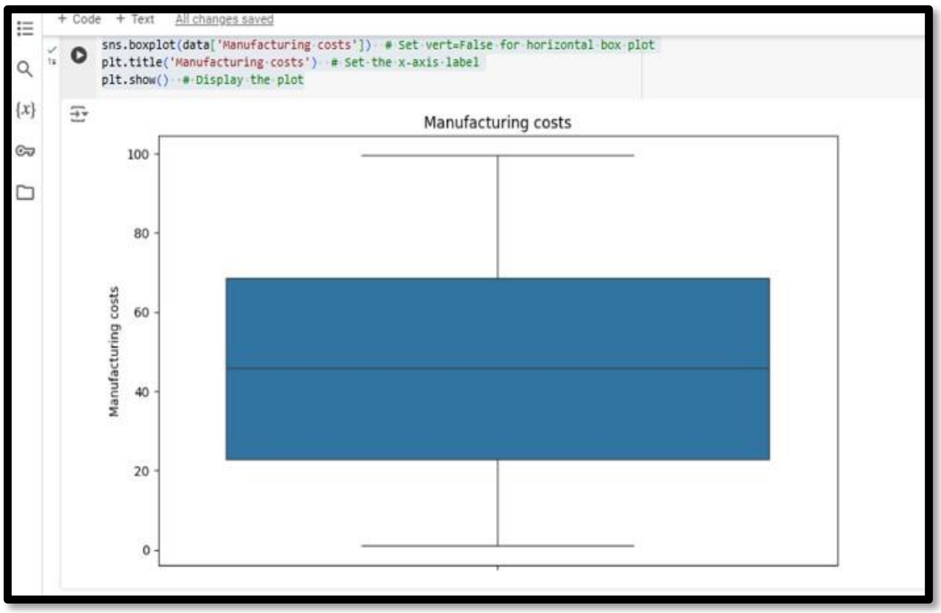


```
plt.show()  # Display the plot
```

3.2.2 Boxplots: Box plots provide a concise view of the central tendency, variability, and skewness of data.

Code:

- import matplotlib.pyplot as plt
- import seaborn as sns
- # Assuming 'data' is your DataFrame and 'column_name' is the column you want to visualize
- plt.figure(figsize=(10, 6)) # Adjust figure size if needed
- sns.boxplot(data['Manufacturing costs']) # Set vert=False for horizontal box plot
- plt.title('Manufacturing costs') # Set the x-axis label
- plt.show() # Display the plot



```
sns.boxplot(data['Manufacturing costs'])  # Set vert=False for horizontal box plot
plt.title('Manufacturing costs')  # Set the x-axis label
plt.show()  # Display the plot
```

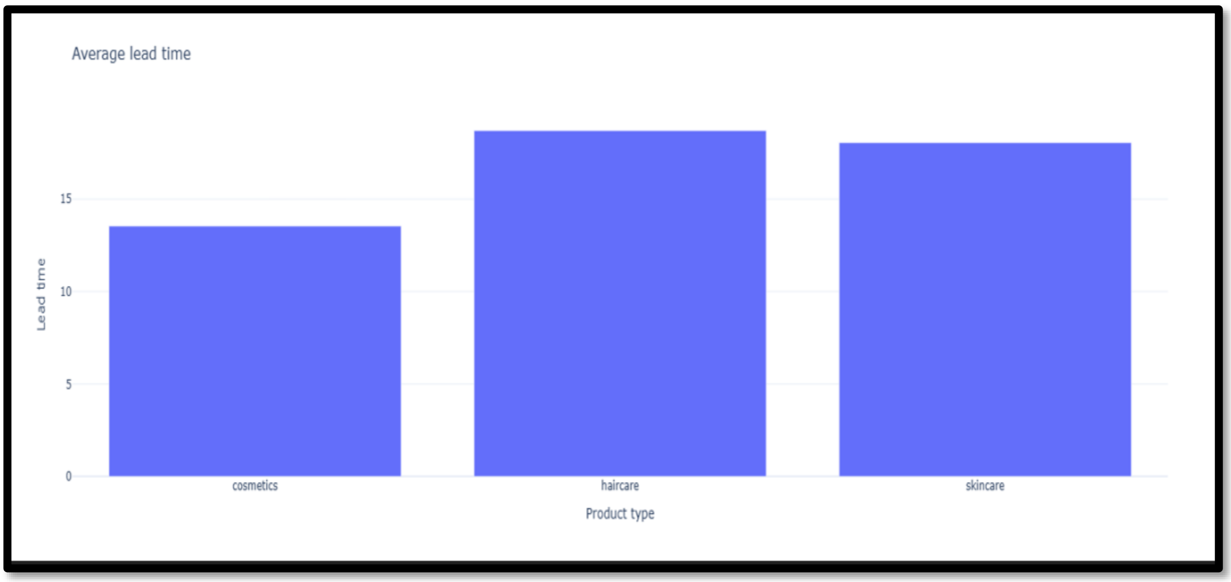
3.2.3 Barplots: Bar plots are ideal for comparing quantities across different categories and are commonly used in both exploratory data analysis (EDA) and for presenting statistical results. Here we are plotting average lead time for cosmetics, haircare and skincare.

Code:

- import pandas as pd
- import numpy as np
- import plotly.express as px

- avg_lead_time = data.groupby("Product type")['Lead time'].apply(lambda x: pd.to_numeric(x, errors='coerce').mean()).reset_index()
- avg_lead_time

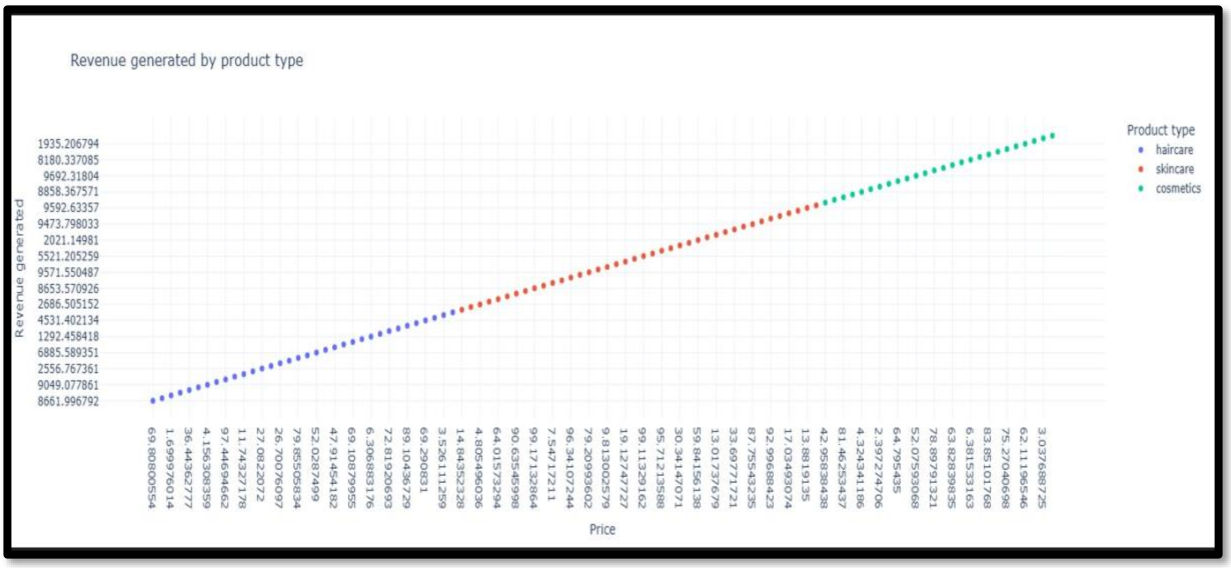
  bar_lead_time = px.bar(avg_lead_time, x='Product type', y='Lead time', title='Average lead time')
- bar_lead_time.show()



3.2.4 Scatterplots: A **scatter plot** is a type of plot used to display the relationship between revenue generated by product types.

Code:

- import pandas as pd
- import numpy as np
- import plotly.express as px
- fig = px.scatter(data, x="Price", y="Revenue generated", color="Product type", title="Revenue generated by product type", \
- hover_data = ["Number of products sold"])
- fig.show()



4. Model Research and Selection Rationale

4.1 Research into Techniques

The techniques used are:

- **Isolation Forests**: Efficient for identifying anomalies in high-dimensional datasets.
- **Autoencoders**: Neural networks used for anomaly detection by learning to compress and reconstruct data.
- **k-Nearest Neighbors (k-NN)**: Detects outliers by evaluating the distance between data points.

Justification for Using Isolation Forests

- **Scalability and Efficiency**: Isolation Forests are highly scalable and can handle large datasets efficiently.
- **Effectiveness in Identifying Anomalies**: One of the core challenges in supply chains is identifying errors such as duplicate entries, data entry mistakes, incorrect inventory levels, or anomalous shipping times. Isolation Forests are particularly good at identifying rare or isolated data points that deviate from normal patterns.

5. Data Transformation and Feature Engineering

5.1 Feature Scaling

- **Standardization:** This is useful for machine learning models that assume data is normally distributed

- **Min-Max Scaling:** The values are scaled to a range between 0 and 1. This can be useful for models that require inputs to be within a specific range or that are sensitive to the magnitude of the data.

Code:

- import pandas as pd
- from sklearn.preprocessing import StandardScaler, MinMaxScaler
- def scale_features(df):
- # Select numerical features that are present in the DataFrame
- numerical_features = df.select_dtypes(include=['float64', 'int64']).columns.tolist()
- # Remove columns that are not supposed to be scaled
- remove_cols = ['SKU', 'Number of products sold', 'Lead time', 'Order quantities']
- numerical_features = [col for col in numerical_features if col not in remove_cols]
- # Create a scaler object (choose Standardization or Min-Max Scaling)
- scaler = StandardScaler()
- # scaler = MinMaxScaler()
- # Fit and transform the scaler on the numerical features
- df[numerical_features] = scaler.fit_transform(df[numerical_features])
- return df
- # Example Usage:
- # Assuming 'data' is your supply chain data DataFrame
- df_scaled = scale_features(data) # Pass the original 'data' DataFrame
- print("Scaled Data:")
- print(df_scaled)

5.2 Encoding Categorical Variables

- **One-Hot Encoding:** One-Hot Encoding is a method used to convert categorical variables (which contain text or labels) into a numerical format that machine learning algorithms can work with.

Code:

- import pandas as pd
- from sklearn.preprocessing import OneHotEncoder
- def encode_categorical_features(df):

- # Select categorical features
- categorical_features = ['Product type', 'Supplier name', 'Shipping carriers',
  'Transportation modes', 'Inspection results']
- # Create a OneHotEncoder object
- # Removed 'sparse=False' as it's no longer needed in newer versions of sklearn
- # Added sparse_output=False to ensure the output is a dense array
- encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
- # Fit and transform the encoder on the categorical features
- encoded_features = encoder.fit_transform(df[categorical_features])
- # Create a new DataFrame with the encoded features
- encoded_df=pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(categorical_features))
- # Concatenate the encoded features with the original DataFrame
- df_encoded = pd.concat([df.drop(categorical_features, axis=1), encoded_df], axis=1)
- return df_encoded
- # Example Usage:
- # Assuming 'df' is your supply chain data DataFrame
- df_encoded = encode_categorical_features(df)
- print("Encoded Data:")
- print(df_encoded)

6. Feasibility Assessment

6.1 Exploratory Data Analysis (EDA):

- **Missing Data Check**: Check for missing values in categorical columns (e.g., Product type, Supplier name, Shipping carriers), as One-Hot Encoding cannot handle missing categories.

6.2 Dimensionality Reduction:

- Apply dimensionality reduction techniques (e.g., **PCA** or **t-SNE**) to reduce the high-dimensional space created by One-Hot Encoding, especially if there are many categories.

7. Conclusion

Phase 2 of the supply chain data project was successfully completed with the following the below mentioned things.

1. **Data Quality** was significantly improved through proper handling of missing values, outliers, duplicates, and inconsistencies.
2. **Data Transformation and Feature Engineering** were conducted with appropriate scaling and encoding techniques to prepare the dataset for machine learning.
3. **Insights** were drawn from the data using effective visualizations, providing a clear understanding of trends and key business factors.
4. **Feasibility** of the project was assessed, confirming that the technical, business, operational, and model-related aspects were viable.
5. Machine learning models were selected and prepared based on the cleaned and transformed data, ensuring the project is on track for the next phase (model training and evaluation).

I will done all program coding part in this phase_2 project.