Project Title	**Supply Chain Management**
Tools	ML, Python, SQL, Excel
Domain	Data Analyst
Project Difficulties level	intermediate

Dataset : Dataset is available in the given link. You can download it at your convenience.

Click here to download data set

About Dataset

Supply chain analytics is a valuable part of data-driven decision-making in various industries such as manufacturing, retail, healthcare, and logistics. It is the process of collecting, analyzing and interpreting data related to the movement of products and services from suppliers to customers.

Here is a dataset we collected from a Fashion and Beauty startup. The dataset is based on the supply chain of Makeup products. Below are all the features in the dataset:

- Product Type
- SKU
- Price
- Availability
- Number of products sold
- Revenue generated

- Customer demographics
- Stock levels
- Lead times
- Order quantities
- Shipping times
- Shipping carriers
- Shipping costs
- Supplier name
- Location
- Lead time
- Production volumes
- Manufacturing lead time
- Manufacturing costs
- Inspection results
- Defect rates
- Transportation modes
- Routes
- Costs

Supply Chain Management Machine Learning Project

Project Overview

The goal of this project is to implement a machine learning model to optimize supply chain management. This can involve various tasks such as demand forecasting, inventory optimization, supplier selection, and transportation planning. We'll focus on a demand forecasting model, which is a critical component in supply chain management.

Step 1: Data Preparation

Ensure your dataset includes the following columns:

- Date
- ProductID

- HistoricalSales
- Promotion
- Price
- Weather (if applicable)
- EconomicIndicators (if applicable)

Step 2: Data Preprocessing

First, import the necessary libraries and load the dataset.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Load the dataset
data = pd.read_csv('supply_chain_data.csv')

# Display the first few rows of the dataset
print(data.head())
```

Data Cleaning and Feature Engineering

```
# Handle missing values
data = data.dropna()

# Convert date to datetime
```

```
data['Date'] = pd.to_datetime(data['Date'])

# Create additional time-based features
data['Month'] = data['Date'].dt.month
data['DayOfWeek'] = data['Date'].dt.dayofweek
data['Quarter'] = data['Date'].dt.quarter

# Drop the original date column
data = data.drop(columns=['Date'])

# One-hot encode categorical variables if any
data = pd.get_dummies(data, columns=['Promotion', 'Weather', 'EconomicIndicators'])

# Define the target variable and features
target = 'HistoricalSales'
features = data.drop(columns=[target])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, data[target], test_size=0.2,
random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Step 3: Model Building

We'll use a neural network for demand forecasting.

```
# Define the model
model = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=[X_train_scaled.shape[1]]),
    layers.Dense(64, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(1)
])

# Compile the model
```

```
model.compile(optimizer='adam', loss='mse')

# Train the model
history = model.fit(X_train_scaled, y_train, epochs=50, validation_split=0.2)

# Plot training and validation loss
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='val')
plt.xlabel('Epoch')
plt.ylabel('Mean Squared Error')
plt.legend()
plt.show()
```

Step 4: Model Evaluation

Evaluate the model on the test set.

```
# Evaluate the model
test_predictions = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, test_predictions)
print(f'Mean Squared Error on Test Set: {mse}')

# Plot true vs predicted values
plt.scatter(y_test, test_predictions)
plt.xlabel('True Values')
plt.ylabel('Predictions')
plt.title('True vs Predicted Sales')
plt.show()
```

Step 5: Model Deployment

For simplicity, we'll use a saved model to make future predictions.

```
# Save the model
model.save('demand_forecasting_model.h5')

# Load the model for future predictions
loaded_model = keras.models.load_model('demand_forecasting_model.h5')

# Example prediction with new data
new_data = np.array([[0.5, 1.2, 0.3, 0, 1, 0, 0, 0, 1]])  # Example new data
new_data_scaled = scaler.transform(new_data)
predicted_sales = loaded_model.predict(new_data_scaled)
print(f'Predicted Sales: {predicted_sales[0][0]}')
```

Final Report

Title: Supply Chain Management Demand Forecasting Report

1. Overview

- **Objective:** To forecast product demand using a neural network model.
- **Model Performance:** Mean Squared Error on Test Set: `MSE_value`

2. Data Preprocessing

- Missing values were handled.
- Date features were engineered (Month, Day of Week, Quarter).
- Categorical variables were one-hot encoded.
- Data was split into training and testing sets and standardized.

3. Model Building

- A neural network model with three hidden layers was used.
- The model was trained for 50 epochs with a validation split of 20%.

4. Model Evaluation

- The model's Mean Squared Error on the test set is `MSE_value`.
- A scatter plot of true vs predicted sales values shows good predictive performance.

5. Model Deployment

- The trained model was saved and can be loaded for future predictions.
- An example prediction was made using new data, demonstrating the model's capability.

Conclusion: The neural network model provides accurate demand forecasts, which can be used to optimize inventory levels, reduce stockouts, and improve supply chain efficiency. Regular updates and retraining of the model with new data will help maintain its accuracy.

This project plan outlines the steps to implement a demand forecasting model for supply chain management using machine learning, complete with data preparation, model building, evaluation, and deployment.

Sample report

Supply Chain Analysis: Data Exploration and Visualization¶

You are going to see different types of plots I have used . Feel Free to explore them

Streamlit Dashboard:

To complement the exploratory data analysis, I have created an interactive Streamlit dashboard. The dashboard provides real-time visualization and analysis of the supply chain data, making it easier to identify trends, patterns, and areas for improvement.

Dashboard Link: Click Here To Explore The Dashboard📊

Summary

Welcome to this notebook where I dive deep into the analysis of supply chain data. This notebook focuses on key metrics essential for understanding the performance and efficiency of supply chain operations.

Objectives:

- **Data Exploration**: Gain insights into the supply chain data by exploring production volumes, stock levels, order quantities, revenue, costs, lead times, shipping costs, transportation routes, risks, and sustainability factors.
- **Visualization**: Create informative visualizations to better understand the relationships and distributions within the data.
- **Dashboard Development**: Build an interactive Streamlit dashboard to visualize the key metrics and facilitate real-time analysis.

Key Sections:

1. **Data Preprocessing**
    - Handling missing values.
    - Data cleaning and preparation.
2. **Exploratory Data Analysis (EDA)**
    - Descriptive statistics.
    - Visualization of key metrics.
3. **Visualizations**
    - Production volumes, stock levels, and lead times.
    - Revenue distribution by location.
    - Manufacturing costs by supplier.
    - Comparison of price and manufacturing costs by product type.
    - Relationship between production volume, stock levels, and order quantities.
    - Distribution of shipping costs by shipping carriers.
    - Average lead time by product type.
    - Transportation routes and their frequency.

- Supply chain risk distribution by risk factors.
- Sustainability factors in the supply chain.

4. **Streamlit Dashboard**
    - Introduction to the interactive dashboard.
    - Instructions on how to access and use the dashboard.
    - Link to the deployed Streamlite dashboard.

In [1]:
```
!pip install duckdb
```

Show hidden output

In [2]:
```
import pandas as pd
import duckdb
import plotly.graph_objects as go
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df = pd.read_csv('/kaggle/input/supply-chain-analysis/supply_chain_data.csv')

df.head()
```

Out[3]:

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Stock levels	Lead times	Order quantities	...	Location	Lea time

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Product type', 'SKU', 'Price', 'Availability',
       'Number of products sold', 'Revenue generated', 'Customer demographics',
       'Stock levels', 'Lead times', 'Order quantities', 'Shipping times',
       'Shipping carriers', 'Shipping costs', 'Supplier name', 'Location',
       'Lead time', 'Production volumes', 'Manufacturing lead time',
       'Manufacturing costs', 'Inspection results', 'Defect rates',
       'Transportation modes', 'Routes', 'Costs'],
      dtype='object')
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Product type             100 non-null    object
 1   SKU                      100 non-null    object
```

```
 3   Availability              100 non-null    int64
 4   Number of products sold   100 non-null    int64
 5   Revenue generated         100 non-null    float64
 6   Customer demographics     100 non-null    object
 7   Stock levels              100 non-null    int64
 8   Lead times                100 non-null    int64
 9   Order quantities          100 non-null    int64
 10  Shipping times            100 non-null    int64
 11  Shipping carriers         100 non-null    object
 12  Shipping costs            100 non-null    float64
 13  Supplier name             100 non-null    object
 14  Location                  100 non-null    object
 15  Lead time                 100 non-null    int64
 16  Production volumes        100 non-null    int64
 17  Manufacturing lead time   100 non-null    int64
 18  Manufacturing costs       100 non-null    float64
 19  Inspection results        100 non-null    object
 20  Defect rates              100 non-null    float64
 21  Transportation modes      100 non-null    object
 22  Routes                    100 non-null    object
 23  Costs                     100 non-null    float64
dtypes: float64(6), int64(9), object(9)
```

In [6]:
```
df.isnull().sum()
```

Out[6]:

Product type	0
SKU	0
Price	0
Availability	0
Number of products sold	0
Revenue generated	0
Customer demographics	0
Stock levels	0
Lead times	0
Order quantities	0
Shipping times	0
Shipping carriers	0
Shipping costs	0
Supplier name	0
Location	0
Lead time	0

In [7]:
```
df.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```
query = """
    SELECT SUM("Revenue generated")::DECIMAL(8, 2) AS total_revenue
    FROM df
"""
result = duckdb.query(query).df()

print(result)
```

```
   total_revenue
0      577604.82
```

In [9]:
```
total_revenue = result['total_revenue'][0]

fig = go.Figure()
```

[Reference link]