**Phase-5**

| **Date** | **31-10-2023** |
| --- | --- |
| **Team ID** | **3909** |
| **Project Name** | **Product Demand using Machine learnings** |

**Table of Contents**

| **S.No** | **Title** |
| --- | --- |
| 1 | Introduction |
| 2 | Problem Statement |
| 3 | Literature survey |
| 4 | Design Think Approach |
| 5 | Development Phases |
| 6 | Data collection and Feature engineering |
| 7 | Data visualization, Modeling & Evaluation |
| 8 | Prediction Insights |
| 9 | Conclusion |
| 10 | References |

**1. Introduction:**

The objective of product demand forecasting using machine learning is to develop a model that can accurately predict the future demand for a product, based on historical data and other relevant factors. This information can then be used to make informed business decisions, such as how much inventory to stock, how to allocate marketing resources, and when to launch new products.

**2. Problem Statement**

Traditional product demand forecasting methods are often inaccurate and time-consuming. These methods often rely on subjective judgment and historical data alone, which can lead to errors. Additionally, traditional methods may not be able to account for the dynamic nature of the market and the impact of external factors.

The key challenges for product demand forecasting include:

* **Data quality and quantity:** Demand forecasting requires accurate and complete data, including historical sales data, market trends, and economic conditions. However, many businesses struggle to collect, clean, and integrate this data.
* **Volatility and uncertainty in demand:** Demand can be unpredictable and volatile, due to factors such as economic conditions, competitor actions, and consumer behavior. This can make it difficult to forecast demand accurately.
* **Complex supply chains:** Modern supply chains are often complex and global, with multiple tiers of suppliers and customers. This can make it difficult to track inventory levels and forecast demand across the entire supply chain.
* **Globalization and geopolitical factors:** Globalization has introduced new challenges for demand forecasting, such as trade policies, currency fluctuations, and international relations. These factors can create uncertainty and volatility in demand.
* **Multidimensional supply networks:** Many businesses now operate in multidimensional supply networks, with sales channels that include online and offline stores, as well as third-party marketplaces. This can make it difficult to forecast demand across all channels.
* **Fragmentation:** Forecasting is often siloed within different departments, such as sales, marketing, and supply chain. This can lead to miscommunication and inconsistent forecasts.
* **Lack of visibility:** Many businesses lack visibility into their supply chain and customer demand. This can make it difficult to identify and respond to changes in demand.
* **Lack of collaboration:** Demand forecasting often requires collaboration between different departments and stakeholders. However, this collaboration can be difficult to achieve, especially in large organizations.
* **Lack of expertise:** Demand forecasting is a complex task that requires expertise in statistics, forecasting methods, and supply chain management. Many businesses lack this expertise in-house.

**3. Literature Survey:**

**Demand forecasting model for time-series pharmaceutical data using shallow and deep neural network model:**

Demand forecasting is important for pharmaceutical companies to optimize their production and supply chain operations, make better pricing decisions, and develop new marketing strategies. Statistical methods, such as ARIMA and SARIMA, are the most common methods for demand forecasting. However, these methods are not always accurate for forecasting pharmaceutical demand. Machine learning methods, such as artificial neural networks (ANN), are more accurate for forecasting pharmaceutical demand.

**Machine learning in prediction demand for Fast-moving consumer goods: An Exploratory research:**

Gather historical sales data for FMCG products. Preprocess the data to remove outliers and trends, ensuring data quality for training the Machine Learning models and Evaluate the performance of the trained models using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). Compare the predictions made by Machine Learning models with traditional statistical methods like Autoregressive Integrated Moving Average (ARIMA) to assess their accuracy.

**Fashion Retail: Forecasting Demand for New items**

The authors of the paper propose a new method for forecasting demand for new items in fashion retail. Their method combines deep learning and tree-based machine learning algorithms. The deep learning algorithm learns the complex relationships between item attributes and demand, while the tree-based algorithm provides interpretability and helps to identify the most important attributes for forecasting demand.

**Demand production using Machine learning methods and stacked generalization**

The authors of the paper propose a new method for forecasting demand for new items in fashion retail. Their method combines deep learning and tree-based machine learning algorithms. Deep learning is a type of machine learning that can learn complex relationships between data points. This makes it well-suited for forecasting demand for new items, which often have limited historical sales data. Tree-based machine learning is a type of machine learning that can identify the most important features of a dataset. This makes it well-suited for identifying the most important attributes for forecasting demand.

**Impact of Product Platform and Market Demand on Manufacturing System Performance and Production Cost**

The paper proposes a solution to the challenges of implementing product platform strategy by using a product platform scalability approach. Product platform scalability is a way to improve the scalability of manufacturing systems by designing product platforms that can be adapted to meet the needs of different customers. The paper argues that the multiple product platform scalability approach is more effective than the single product platform scalability approach for high levels of customer customization. This is because the multiple product platform approach allows manufacturers to offer a wider range of customized products that meet the specific needs of each customer.

**4. Design Thinking Approach**

**Empathize**

Before diving into solving the problem, it's crucial to empathize with the users and understand their needs. In this case, our primary users are businesses that need to forecast product demand accurately. We need to gather insights into what factors are most important to them when considering demand forecasting and how accurate predictions can benefit them.

**Actions:**

* Conduct surveys or interviews with businesses to gather their perspectives on demand forecasting.
* Analyze historical demand data to identify critical demand drivers.
* Seek feedback from domain experts in the supply chain and operations management industry.

**Define**

Based on our understanding of the problem and the users' needs, we will define clear objectives and success criteria for our project.

**Objectives**:

* Develop a machine learning model that achieves a Mean Absolute Error (MAE) of less than $X on the test data.
* Create a user-friendly web application for businesses to input product details and receive demand predictions.

**Ideate:**

Brainstorm potential solutions and approaches to address the problem. This phase involves thinking creatively and considering various algorithms and techniques for product demand prediction.

**Actions:**

* Explore different machine learning algorithms such as linear regression, decision trees, random forests, and neural networks.
* Experiment with feature engineering techniques to enhance model performance.
* Consider incorporating external data sources (e.g., economic indicators, social media data, weather data) to improve predictions.

**Prototype**

Create a prototype of the machine learning model and the user interface for demand prediction.

**Actions:**

* Develop a Jupyter Notebook or Python script for data pre-processing, model training, and evaluation.
* Create a simple web interface using tools like Flask or Django to allow businesses to input product details and receive demand predictions.
* Test the prototype with a subset of the dataset to ensure it meets performance objectives.

**Test:**

Evaluate the model's performance using appropriate metrics and gather feedback from users.

**Actions:**

* Split the dataset into training and testing sets.
* Train the model on the training set and evaluate it on the testing set.
* Use metrics such as MAE, Root Mean Square Error (RMSE), and R-squared to assess model performance.
* Collect user feedback on the web interface for usability and accuracy.

**Implement:**

Once the prototype meets the defined objectives and receives positive feedback, proceed with full implementation.

**Actions:**

* Train the final machine learning model on the entire dataset.
* Deploy the model as part of a production-ready web application.
* Conduct thorough testing to ensure the application is robust and user-friendly.

**Iterate:**

Continuous improvement is essential. Gather user feedback and iterate on the model and interface to enhance accuracy and usability.

**Actions:**

* Monitor the model's performance and retrain it periodically with updated data.
* Address user feedback and make necessary improvements to the web interface.
* Stay informed about advancements in machine learning and product demand forecasting models for potential enhancements.

**Flow Diagram:**

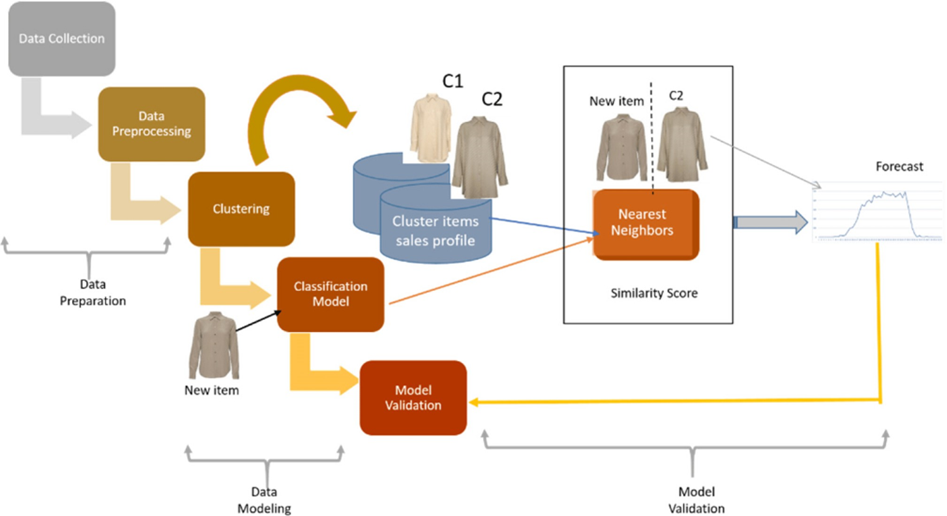
****

Figure 1: Flow chart

**5. Development Phases:**

The development of a machine learning model for product demand forecasting can be divided into the following phases:

1. **Data collection and preparation:** This phase involves gathering historical sales data and other relevant factors, such as product price, seasonality, and economic conditions. The data is then cleaned and preprocessed to ensure that it is in a format that can be used by the machine learning model.
2. **Feature engineering:** This phase involves creating new features from the existing data that can be used to improve the performance of the machine learning model. For example, a new feature could be created to represent the average demand for a product in a particular month.
3. **Model selection and training:** This phase involves selecting a machine learning algorithm and training it on the prepared data. The algorithm will learn to identify patterns in the data and make predictions about future demand.
4. **Model evaluation and deployment:** Once the model is trained, it is important to evaluate its performance on a held-out test set. If the model performs well on the test set, it can be deployed to production.

**6. Data Collection and Feature Engineering:**

The data collection process for product demand forecasting will vary depending on the specific industry and business model. However, some common data sources include:

* Historical sales data
* Product information (e.g., price, features, categories)
* Marketing data (e.g., advertising spend, social media engagement)
* Economic data (e.g., GDP growth rate, unemployment rate)

Once the data has been collected, it is important to clean and prepare it for machine learning. This may involve removing outliers, imputing missing values, and normalizing the data.

Feature engineering is the process of creating new features from the existing data that can be used to improve the performance of the machine learning model. Some common feature engineering techniques for product demand forecasting include:

* Creating lagged features (e.g., demand for a product last month)
* Creating rolling features (e.g., average demand for a product in the past 3 months)
* Creating seasonal features (e.g., demand for a product in the summer)
* Creating indicator features (e.g., whether a product is on sale)

**7. Data Visualization, Modeling & Evaluation:**

**Importing Libraries**

*# Data Manipulation*  
**import** numpy **as** np  
**import** pandas **as** pd  
  
*# Data Visualization*  
**import** matplotlib.pyplot **as** plt  
%matplotlib inline  
**import** seaborn **as** sns  
**import** plotly.express **as** px  
**import** bokeh  
  
**import** seaborn **as** sns  
color\_pal = sns.color\_palette()  
plt.style.use('fivethirtyeight')  
  
*# For Analysis and Forecasting*  
**from** scipy **import** stats  
  
*# Others*  
**import** datetime  
**import** os  
**import** pickle  
**import** requests

**Importing Data**

df = pd.read\_csv(r"/content/Historical Product Demand.csv")

df.head()

Product\_Code Warehouse Product\_Category Date Order\_Demand  
0 Product\_0993 Whse\_J Category\_028 2012/7/27 100.0  
1 Product\_0979 Whse\_J Category\_028 2012/1/19 500.0  
2 Product\_0979 Whse\_J Category\_028 2012/2/3 500.0  
3 Product\_0979 Whse\_J Category\_028 2012/2/9 500.0  
4 Product\_0979 Whse\_J Category\_028 2012/3/2 500.0

df.tail()

Product\_Code Warehouse Product\_Category Date Order\_Demand  
84885 Product\_1241 Whse\_J Category\_019 2012/8/17 1000.0  
84886 Product\_1239 Whse\_J Category\_019 2012/8/21 200.0  
84887 Product\_1241 Whse\_J Category\_019 2012/9/21 20000.0  
84888 Product\_1239 Whse\_J Category\_019 2012/9/13 1000.0  
84889 Product\_1241 Whse\_J NaN NaN NaN

**DATA CLEANING AND PREPROCESSING**

df.describe()

Order\_Demand  
count 8.488900e+04  
mean 8.810712e+03  
std 4.283116e+04  
min 0.000000e+00  
25% 1.000000e+02  
50% 1.000000e+03  
75% 5.000000e+03  
max 4.000000e+06

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 84890 entries, 0 to 84889  
Data columns (total 5 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Product\_Code 84890 non-null object   
 1 Warehouse 84890 non-null object   
 2 Product\_Category 84889 non-null object   
 3 Date 84888 non-null object   
 4 Order\_Demand 84889 non-null float64  
dtypes: float64(1), object(4)  
memory usage: 3.2+ MB

print("Number of attributes with null vaules: ", df.isnull().any().sum())  
print("Percentage of missing values: ",df.isnull().any(axis=1).sum()/len(df)\*100)

Number of attributes with null vaules: 3  
Percentage of missing values: 0.0023559901048415596

df.dropna(axis=0, how="any", inplace=True)

df["Date"] = pd.to\_datetime(df['Date'])  
*# df["Order\_Demand"] = df["Order\_Demand"].str.replace("(", "")*  
*# df["Order\_Demand"] = df["Order\_Demand"].str.replace(")", "")*  
*#Changing the datatype to float*  
df["Order\_Demand"] = df["Order\_Demand"].astype(float)

df = df.sort\_values(by=['Date', 'Product\_Code'])  
df = df.set\_index('Date')  
df.head()

Product\_Code Warehouse Product\_Category Order\_Demand  
Date   
2011-01-08 Product\_0965 Whse\_A Category\_006 2.0  
2011-05-31 Product\_1724 Whse\_A Category\_003 108.0  
2011-06-24 Product\_1521 Whse\_S Category\_019 85000.0  
2011-06-24 Product\_1521 Whse\_S Category\_019 7000.0  
2011-09-02 Product\_1507 Whse\_C Category\_019 1250.0

category\_yearly\_demand = df.groupby([df.index.year, 'Product\_Category'])['Order\_Demand'].mean()

padded\_category\_data = {}  
**for** category, category\_data **in** category\_yearly\_demand.groupby(level = 'Product\_Category'):  
*# print(f"Category: {category}")*  
 padded\_category\_data[category] = [0 **for** \_ **in** range(7)]  
 **for** year, total\_demand **in** category\_data.items():  
 index = ((year[0] - 2010) % 7) - 1  
 padded\_category\_data[category][index] = total\_demand  
  
fig = plt.figure(figsize=(12, 25))  
rows, cols = 11, 3  
x = [2011, 2012, 2013, 2014, 2015, 2016, 2017]  
  
**for** title, data **in** padded\_category\_data.items():  
 *# Create subplots in the grid*  
 ax = fig.add\_subplot(rows, cols, int(title[-2:]))  
 *# Plotting data on the current subplot*  
 ax.plot(x, data)  
 ax.set\_title(title)  
plt.tight\_layout()

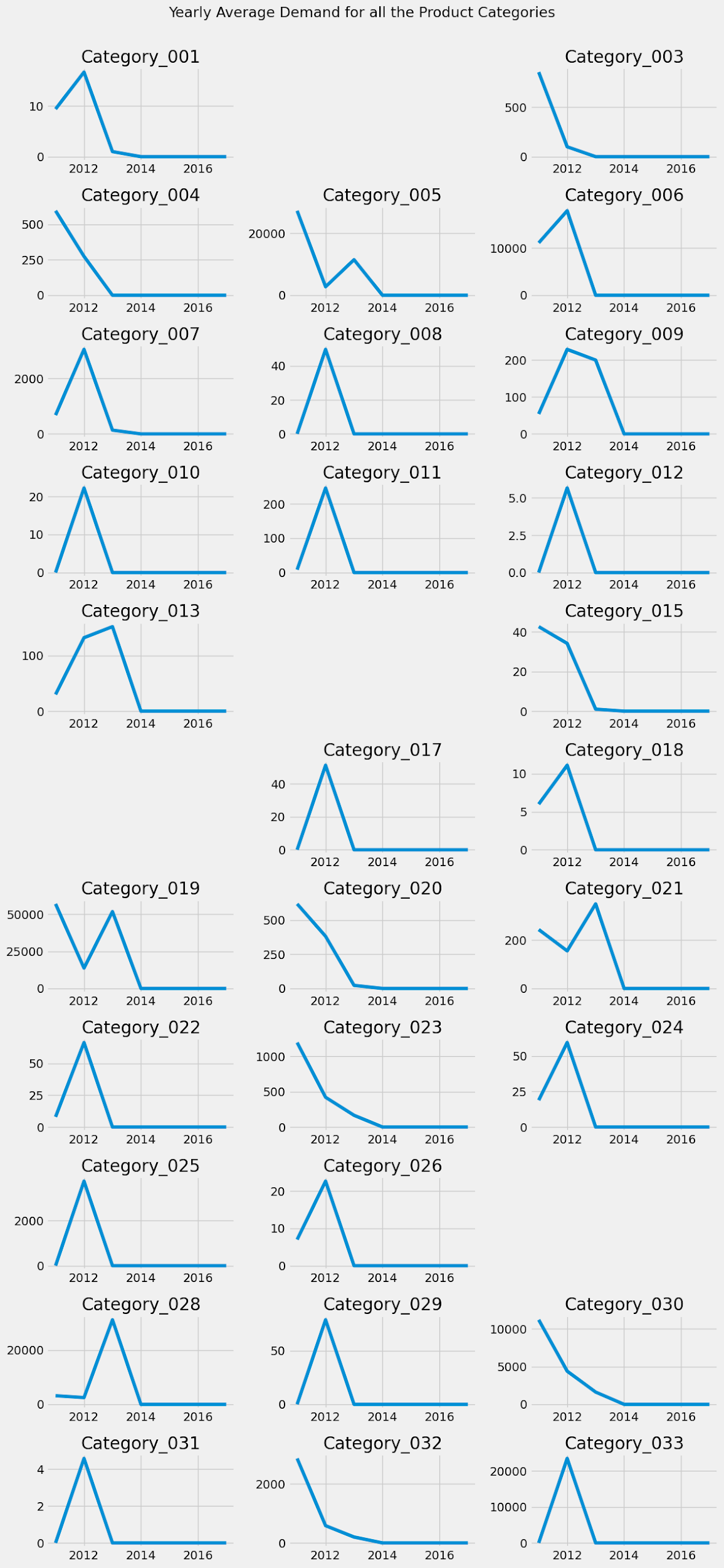


Figure 2: Category wise product demand

padded\_yearly\_categories = {}  
**for** year, year\_data **in** category\_yearly\_demand.groupby(level = 'Date'):  
*# print(f"Category: {category}")*  
 padded\_yearly\_categories[year] = [0 **for** \_ **in** range(33)]  
 **for** category, total\_demand **in** year\_data.items():  
*# print(category)*  
 index = (int(category[1][-2:]) % 33) - 1  
 padded\_yearly\_categories[year][index] = total\_demand  
  
x = [i+1 **for** i **in** range(33)]  
rows = len(padded\_yearly\_categories)  
cols = 1  
  
fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(10, 20))  
  
colors = plt.cm.viridis(np.linspace(0, 1, len(x)))  
  
**for** i, (year, data) **in** enumerate(padded\_yearly\_categories.items()):  
 *# Calculate the row and column indices for the subplot*  
  
 *# Create a bar plot in the current subplot*  
 bars = axes[i].bar(x, data, color=colors)  
 axes[i].bar\_label(bars, labels=x, fontsize = 8)  
  
 *# Set the category title as the subplot title*  
 axes[i].set\_title(year)  
  
 *# Hide only the y-axis scales (ticks)*  
 *#axes[row\_idx, col\_idx].get\_yaxis().set\_visible(False)*  
  
plt.tight\_layout()  
fig.suptitle("Yearwise Average Demand of all Product Categories", y=1.01)  
plt.show()

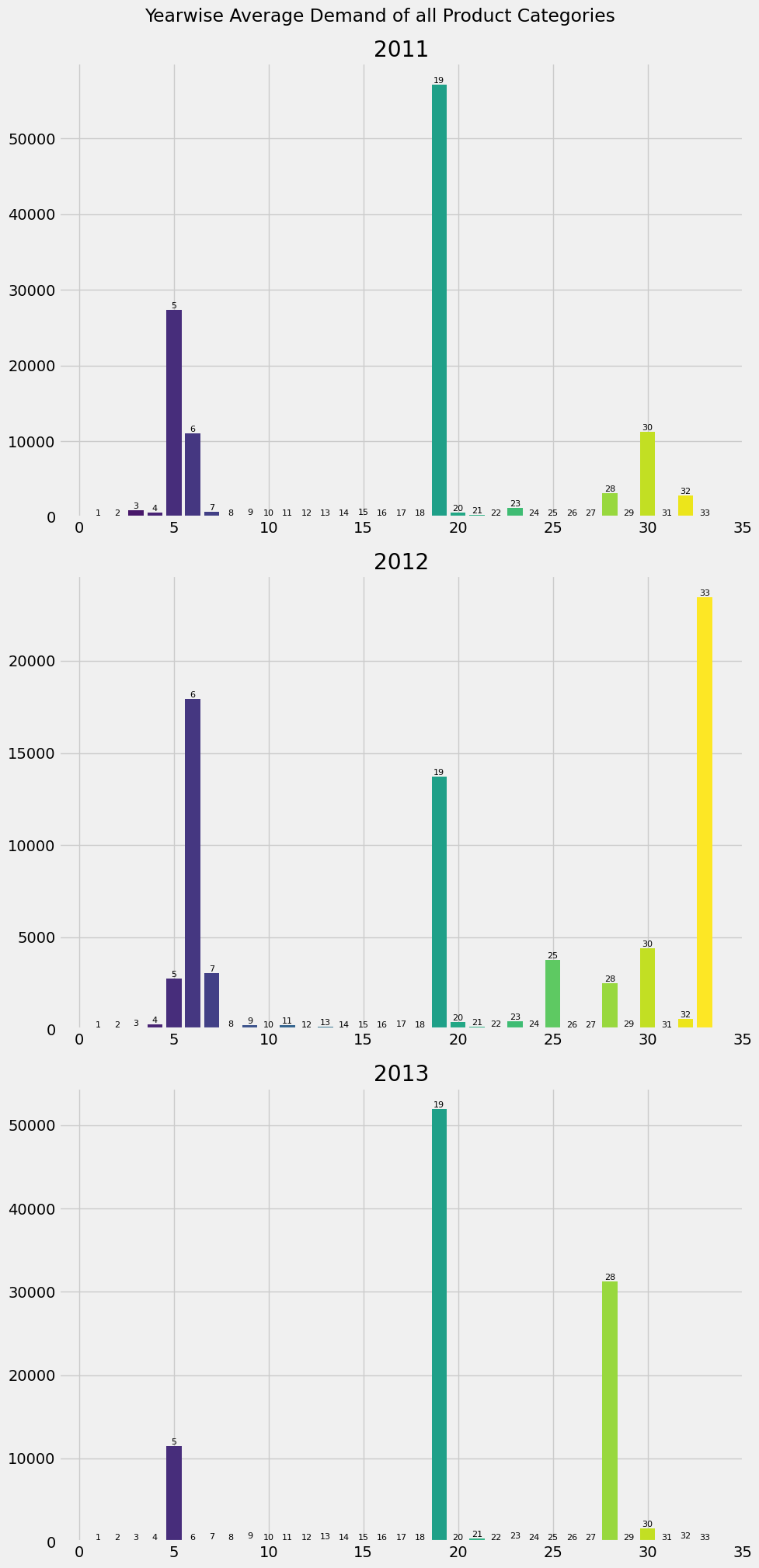


Figure 3 Year Wise average demand of all product categories

warehouse\_yearly\_demand = df.groupby([df.index.year, 'Warehouse'])['Order\_Demand'].mean()

demand\_data = {}  
warehouses = []  
years = [2011 + i **for** i **in** range(7)]  
**for** warehouse, warehouse\_data **in** warehouse\_yearly\_demand.groupby(level='Warehouse'):  
 warehouses.append(warehouse)  
 demand\_data[warehouse] = [0 **for** i **in** range(7)]  
 **for** year, year\_data **in** warehouse\_data.items():  
 index = ((year[0] - 2010) % 7) - 1  
 demand\_data[warehouse][index] = year\_data  
*# print(demand\_data)*  
  
*# Determine the number of warehouses and the number of years*  
num\_warehouses = len(warehouses)  
num\_years = len(years)  
  
*# Set the width of the bars*  
bar\_width = 0.15  
  
*# Create a figure*  
fig, ax = plt.subplots(figsize=(12, 8))  
  
*# Define the index for the x-axis*  
x = np.arange(num\_years)  
  
*# Create a grouped bar chart*  
**for** i, warehouse **in** enumerate(warehouses):  
 *# Offset the x-position for each warehouse*  
 x\_pos = x + i \* bar\_width  
  
 *# Plot the demand values for the current warehouse*  
 ax.bar(x\_pos, demand\_data[warehouse], width=bar\_width, label=warehouse)  
  
*# Set x-axis labels and tick positions*  
ax.set\_xticks(x + (num\_warehouses - 1) \* bar\_width / 2)  
ax.set\_xticklabels(years)  
  
*# Set labels and title*  
ax.set\_xlabel('Year')  
ax.set\_ylabel('Demand')  
ax.set\_title('Yearly Average Demand by Warehouse')  
  
*# Add a legend to distinguish the warehouses*  
ax.legend()  
  
*# Show the chart*  
plt.show()

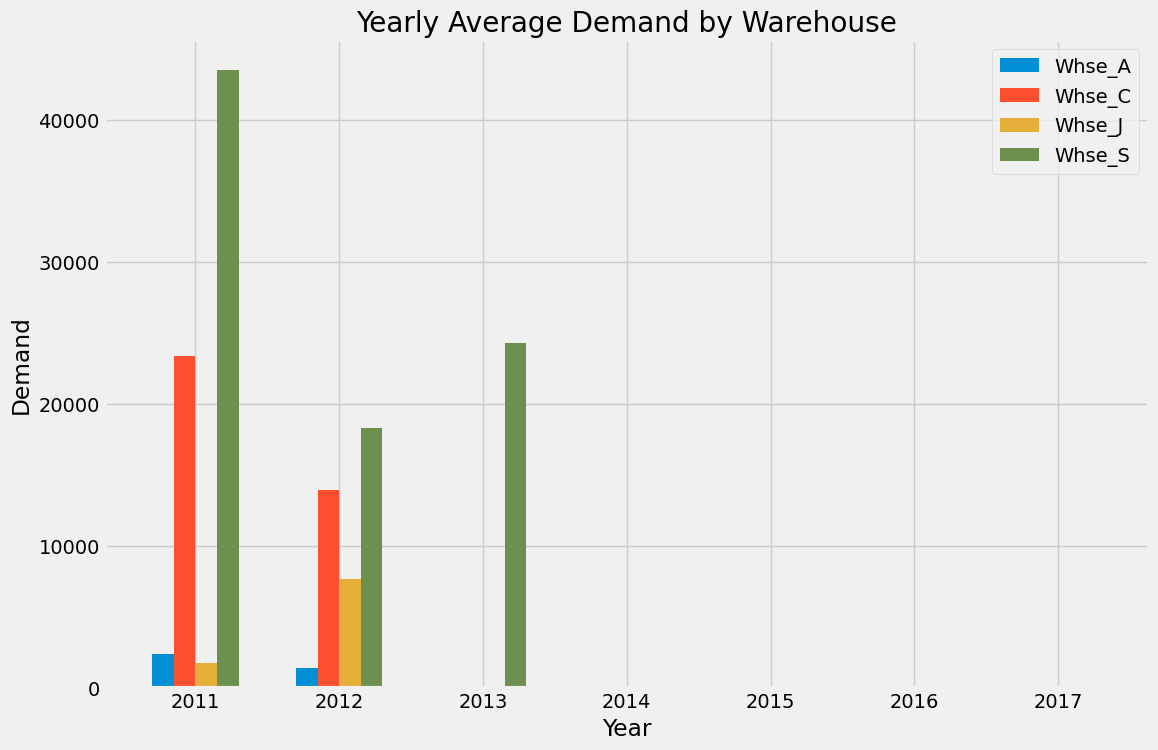


Figure 4: Yearly average demand by warehouse

**MODEL & EVALUATION**

*# Rename the columns*  
df = df.rename(columns = {'Product\_Code': 'Code',  
 'Product\_Category':'Category',  
 'Order\_Demand':'Demand'})  
df.head()

Code Warehouse Category Demand  
Date   
2011-01-08 Product\_0965 Whse\_A Category\_006 2.0  
2011-05-31 Product\_1724 Whse\_A Category\_003 108.0  
2011-06-24 Product\_1521 Whse\_S Category\_019 85000.0  
2011-06-24 Product\_1521 Whse\_S Category\_019 7000.0  
2011-09-02 Product\_1507 Whse\_C Category\_019 1250.0

df\_new = df.loc[df.index >= '01-01-2012']  
df\_new.plot(kind = 'line',figsize=(15, 5),color = color\_pal[0], title = 'Order Demand' )  
plt.show()

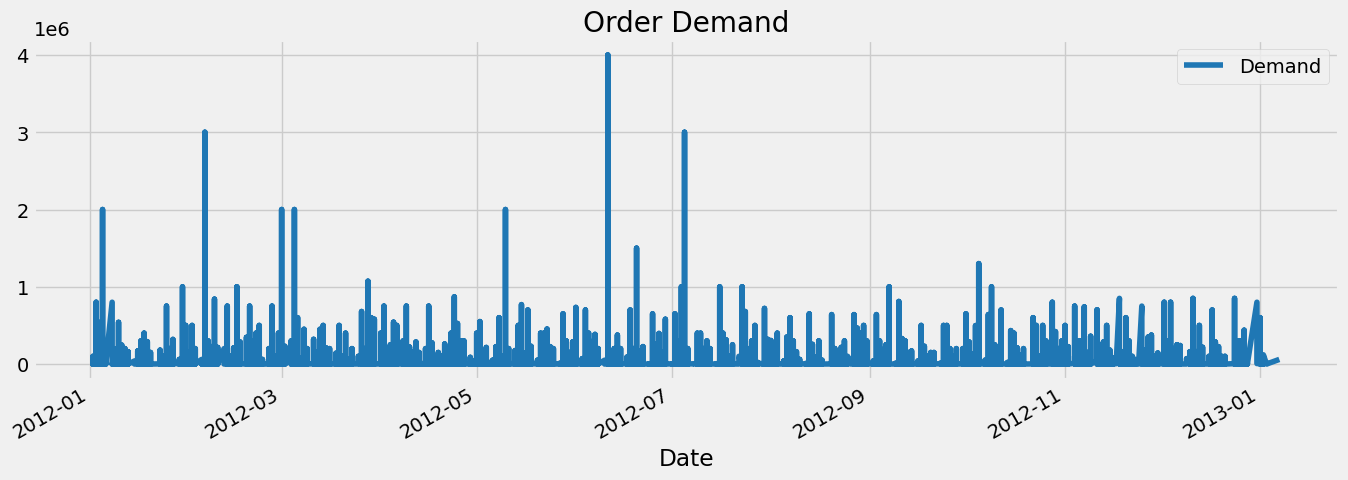


Figure 5: Order Demand

*" Function that create time series features using the index columns"*  
**def** create\_feature(dataframe):  
 dataframe = dataframe.copy()  
 dataframe['day\_of\_the\_week'] = dataframe.index.dayofweek  
 dataframe['Quarter'] = dataframe.index.quarter  
 dataframe['Month'] = dataframe.index.month  
 dataframe['Year'] = dataframe.index.year  
 dataframe['Week'] = dataframe.index.isocalendar().week.astype(int)  
 **return** dataframe  
  
df = create\_feature(df\_new)  
df.dtypes

Code object  
Warehouse object  
Category object  
Demand float64  
day\_of\_the\_week int64  
Quarter int64  
Month int64  
Year int64  
Week int64  
dtype: object

*# features, Target variable*  
Features = ['day\_of\_the\_week', 'Quarter','Month', 'Year', 'Week']  
target = ['Demand']

*# Split the data in Train/ Test set*  
df\_train = df.loc[df.index <= '2012-11-01'].copy()  
df\_test = df.loc[df.index > '2012-11-01'].copy()  
  
*#plot train/test*  
fig, ax = plt.subplots(figsize = (15,5))  
df\_train.Demand.plot(ax = ax, label = 'train set',legend = 'train set', title = 'Train/Test set')  
df\_test.Demand.plot(ax = ax, legend = 'train set', label = 'test set')  
plt.show()

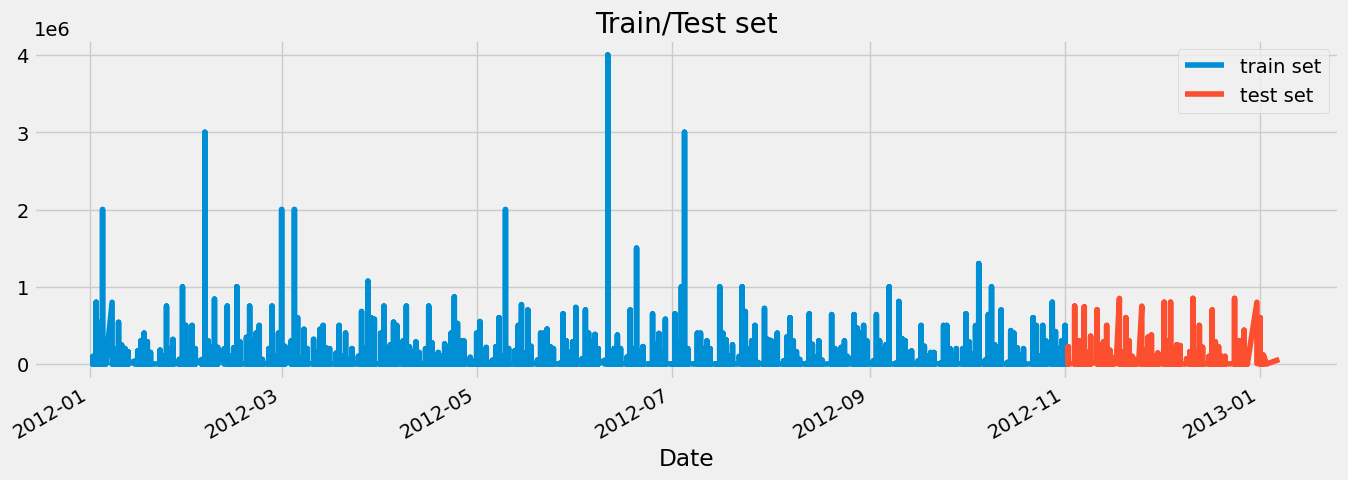


Figure 6: Train/Test Set

X\_train = df\_train[Features]  
X\_test = df\_test[Features]  
  
y\_train = df\_train[target]  
y\_test = df\_test[target]

**from** xgboost **import** XGBRegressor  
model = XGBRegressor(n\_estimators = 1000,  
 early\_stopping\_rounds = 50,  
 learning\_rate = 0.01).fit(X\_train, y\_train,  
 eval\_set = [(X\_train,y\_train), (X\_test,y\_test)],  
 verbose = 50)

[0] validation\_0-rmse:43755.20432 validation\_1-rmse:33950.47798  
[50] validation\_0-rmse:43697.64467 validation\_1-rmse:33839.96202  
[100] validation\_0-rmse:43670.79389 validation\_1-rmse:33806.24955  
[150] validation\_0-rmse:43656.35298 validation\_1-rmse:33809.45442  
[174] validation\_0-rmse:43651.85164 validation\_1-rmse:33815.04443

*# Features Importance*  
fea\_Imp = pd.DataFrame(model.feature\_importances\_, index = Features, columns = ['feature\_importance'])  
*#plot the features importance*  
fea\_Imp.feature\_importance.sort\_values().plot(kind = 'barh', figsize =(5,3))  
plt.show()

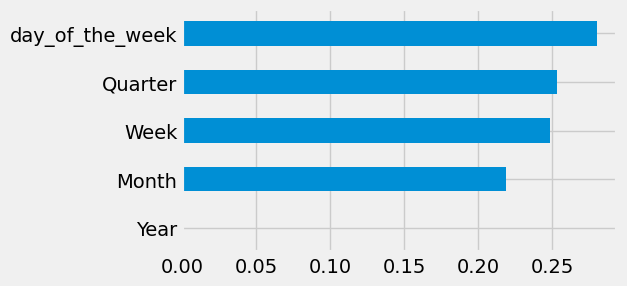
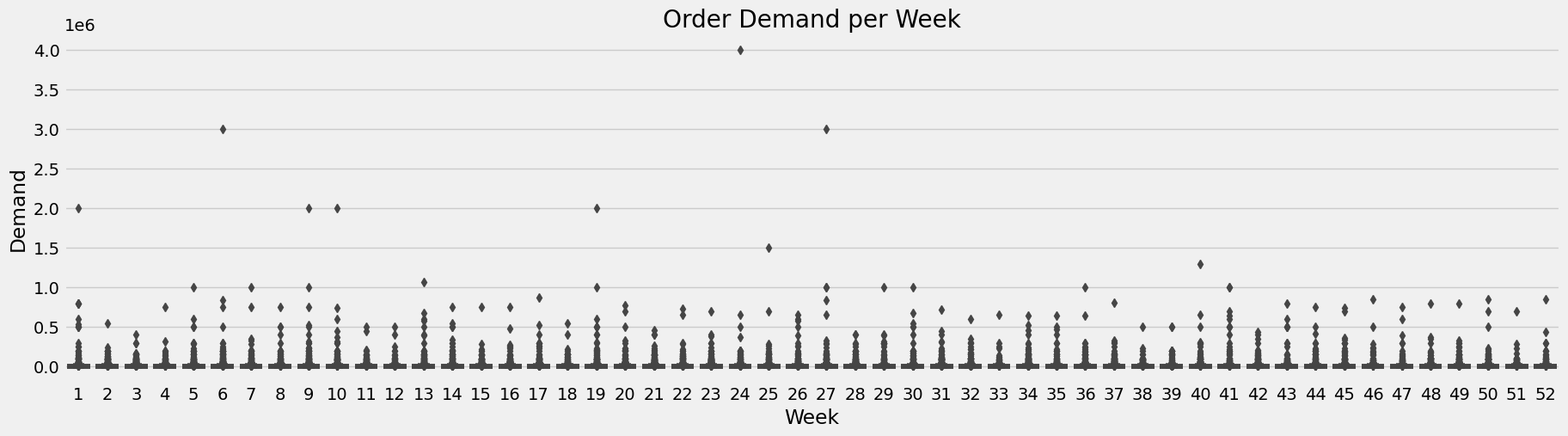


Figure 7: Feature extraction

**TIME SERIES ANALYSIS**

fig, ax = plt.subplots(figsize=(20, 5))  
sns.boxplot(data=df, x='Week', y='Demand')  
ax.set\_title('Order Demand per Week')  
plt.show()

  
Figure 8: Order Demand Per Week

df\_week = df.resample('W').mean()  
df\_week['Demand'].plot(figsize = (15,4), title = 'Weekly Order demand')  
plt.show()

<ipython-input-28-ff46de005cc5>:1: FutureWarning: The default value of numeric\_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric\_only will default to False. Either specify numeric\_only or select only columns which should be valid for the function.  
 df\_week = df.resample('W').mean()

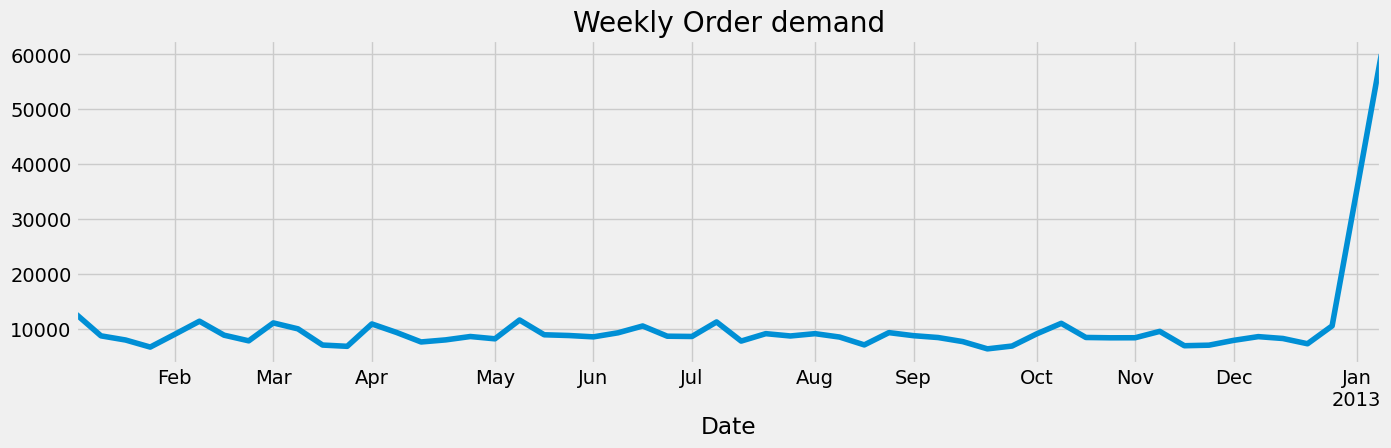


Figure 9: Weekly Order Demand

df\_month = df.resample('MS').mean()  
df\_month.Demand.plot(figsize = (25,5), title = 'Monthly Order Demand', color = color\_pal[2])  
plt.ylabel('Demand')  
plt.show()

<ipython-input-29-f9bbe684cd85>:1: FutureWarning: The default value of numeric\_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric\_only will default to False. Either specify numeric\_only or select only columns which should be valid for the function.  
 df\_month = df.resample('MS').mean()



Figure 10: Monthly order Demand

**ARIMA MODEL**

**from** statsmodels.graphics.tsaplots **import** plot\_acf, plot\_pacf

df\_diff = df\_month.Demand.diff()[1:]  
df\_diff.plot(figsize=(12,6), color = color\_pal[1])  
plt.axhline(0, linestyle='--', color='k', alpha=0.3)

<matplotlib.lines.Line2D at 0x780e3468f5e0>

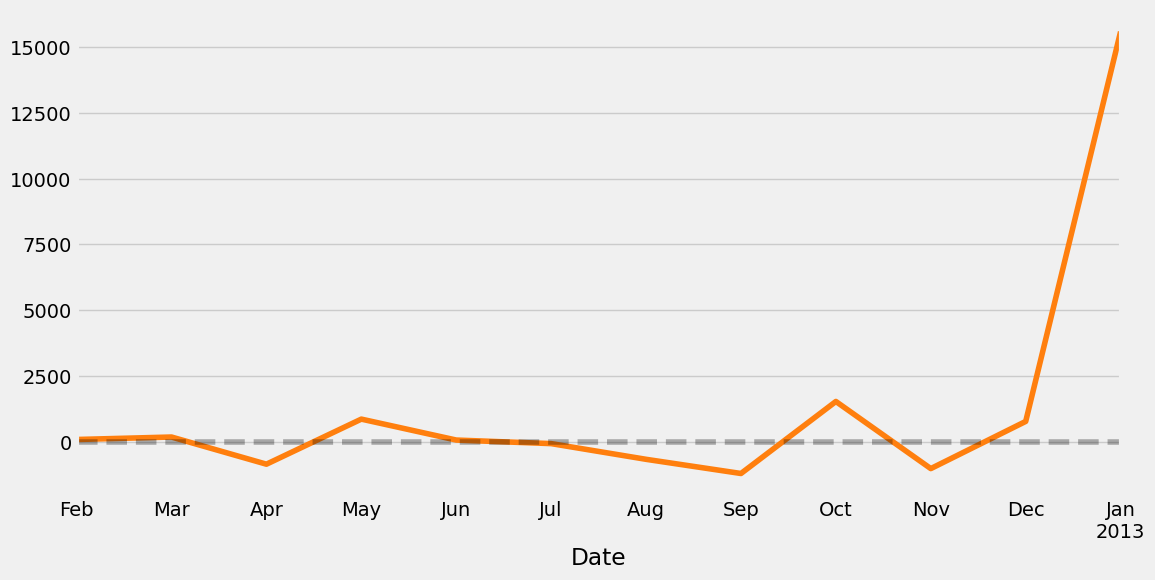


Figure 11: Monthly order demand

**TRAIN & TEST SPLIT**

**from** datetime **import** datetime  
**from** datetime **import** timedelta  
train\_end = datetime(2012,11,1)  
test\_end = datetime(2013,1,1)  
  
df\_train = df\_diff[:train\_end]  
df\_test = df\_diff[train\_end:test\_end]  
  
*# plot train and test dataset on the same graph*  
fig, ax = plt.subplots(figsize=(15, 5))  
df\_train.plot(ax=ax, label='Training Set', title='Data Train/Test Split')  
df\_test.plot(ax=ax, label='Test Set')  
ax.axvline('01-01-2016', color='black', ls='--')  
ax.legend(['Training Set', 'Test Set','Split Point'])  
plt.show()

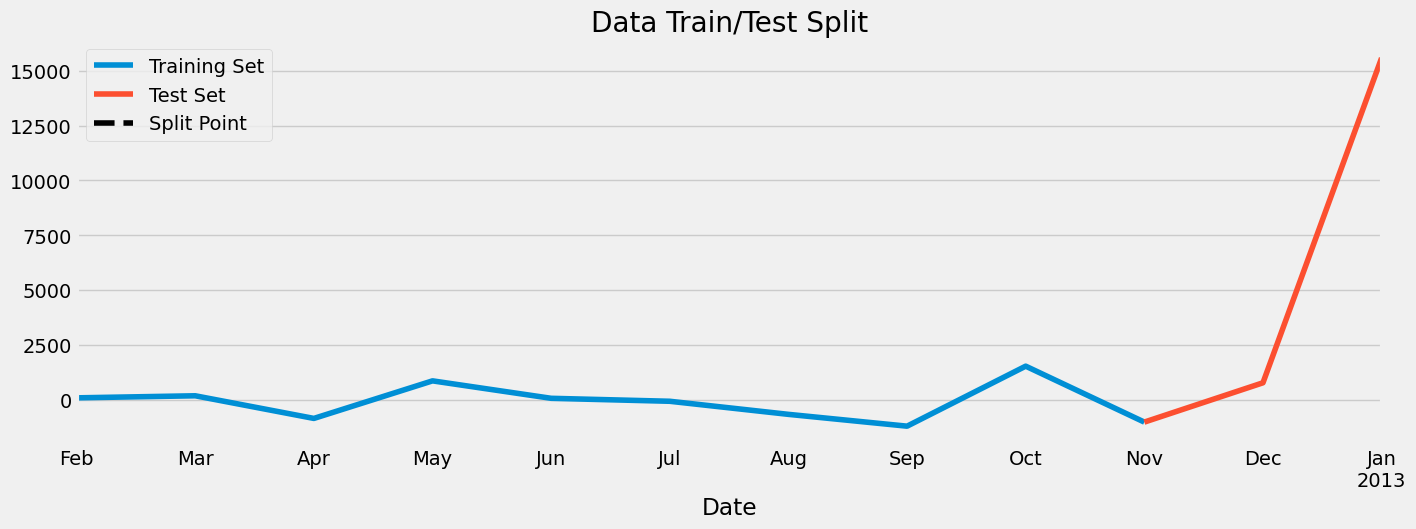


Figure 12: Data Train/Test Split

**from** statsmodels.tsa.arima.model **import** ARIMA  
model\_arima = ARIMA(df\_train, order = (2,1,1)).fit()

/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle\_retvals  
 warnings.warn("Maximum Likelihood optimization failed to "

print(model\_arima.summary())

SARIMAX Results   
==============================================================================  
Dep. Variable: Demand No. Observations: 10  
Model: ARIMA(2, 1, 1) Log Likelihood -73.174  
Date: Sun, 29 Oct 2023 AIC 154.349  
Time: 13:57:39 BIC 155.138  
Sample: 02-01-2012 HQIC 152.646  
 - 11-01-2012   
Covariance Type: opg   
==============================================================================  
 coef std err z P>|z| [0.025 0.975]  
------------------------------------------------------------------------------  
ar.L1 -0.6329 0.455 -1.391 0.164 -1.525 0.259  
ar.L2 -0.2624 0.853 -0.308 0.758 -1.934 1.409  
ma.L1 -0.9997 0.644 -1.553 0.120 -2.261 0.262  
sigma2 4.899e+05 1.31e-06 3.73e+11 0.000 4.9e+05 4.9e+05  
===================================================================================  
Ljung-Box (L1) (Q): 0.35 Jarque-Bera (JB): 0.63  
Prob(Q): 0.55 Prob(JB): 0.73  
Heteroskedasticity (H): 4.17 Skew: -0.58  
Prob(H) (two-sided): 0.27 Kurtosis: 2.45  
===================================================================================  
  
Warnings:  
[1] Covariance matrix calculated using the outer product of gradients (complex-step).  
[2] Covariance matrix is singular or near-singular, with condition number 2.57e+27. Standard errors may be unstable.

*# predict Demand for the next year*  
future = model\_arima.predict(start = '2013-01-01', end = '2013-12-01')  
*# Getting only the value of zeroth index since the diff() operation looses first value.*  
future.iloc[0] = df\_month.query('index == "2013-01-01"')['Demand']  
*#cumsum Return cumulative sum over a DataFrame or Series axis*  
future = future.cumsum()

**Predicting Next Year Demand**

fig, ax = plt.subplots(figsize=(15, 5))  
df\_month.Demand.plot(ax=ax, label='Product Demand', title='Product Demand/ Future prediction')  
future.plot(ax=ax, label='Future')  
ax.legend(['Product demand', 'Future prediction'])  
plt.show()

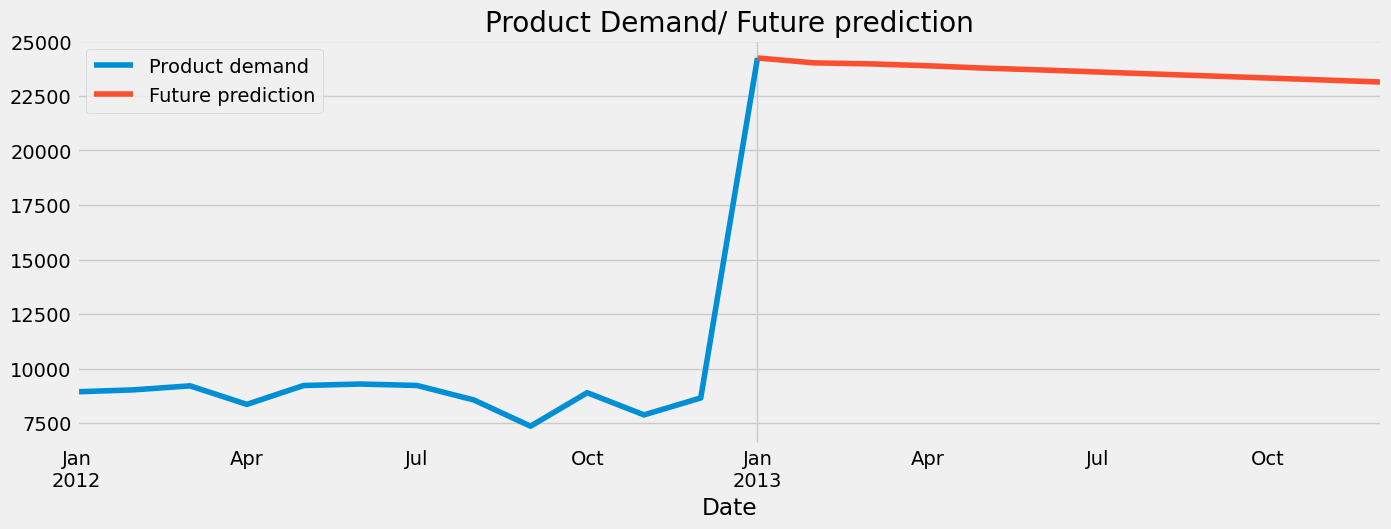


Figure 12: Future Prediction

**8. Prediction Insights:**

Once a machine learning model has been trained and evaluated, it can be used to generate predictions about future demand. These predictions can be used to inform a variety of business decisions, such as:

* How much inventory to stock
* How to allocate marketing resource
* When to launch new products
* How to price products

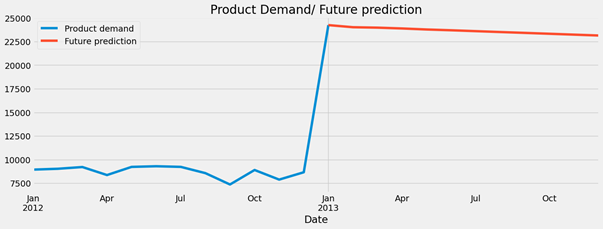


Figure 13: Product Demand/ Future Prediction

**9. Conclusion:**

In conclusion, the product demand documentation provides a comprehensive overview of the factors influencing the demand for a particular product. Through careful analysis, we have identified key drivers such as market trends, consumer preferences, and external influences. The documentation not only outlines the current demand landscape but also anticipates potential future shifts.

By understanding the market dynamics, we can make informed decisions regarding production, marketing strategies, and resource allocation. The insights gathered from this documentation serve as a valuable foundation for product development, helping us tailor our offerings to meet the evolving needs of consumers.

Moreover, the documentation serves as a communication tool, facilitating collaboration among various teams within the organization. It provides a shared understanding of the market forces at play and fosters alignment in decision-making processes.

As we move forward, it is crucial to regularly update and refine this documentation to stay agile in response to market changes. Continual monitoring of demand indicators and feedback mechanisms ensures that our product offerings remain relevant and competitive in a dynamic business environment.

**10.Reference:**

1. Forecasting new product demand using ML
2. Demand forecasting model for time-series pharmaceutical data using shallow and deep neural network model
3. Evaluation of market demand for a specific product
4. Machine learning in prediction demand for Fast-moving consumer goods: An Exploratory research
5. Impact of product platform and market demand on manufacturing systems performance and production cost
6. Deep learning for demand forecasting in the fashion and apparel Retail industry
7. Predictive big data analytics for supply chain Demand forecasting: methods, applications, And research opportunities
8. Demand Forecasting Model using Deep Learning Methods for Supply Chain Management
9. Fashion Retail: Forecasting Demand for New Items
10. Demand Prediction using Machine Learning Methods and Stacked Generalization