Product demand prediction with machine leanings

Name : M.Thanes

Reg no: 2021503712

INTRODUCTION

* Product demand prediction through machine learning involves using historical data, such as sales figures, market trends, and various influencing factors, to build models that forecast the future demand for a product or service. This predictive approach helps businesses anticipate consumer needs, optimize inventory levels, and enhance overall operational efficiency.
* The process typically involves data collection, preprocessing, and applying machine learning algorithms to recognize patterns, correlations, and seasonality in the data. These models then make predictions based on these patterns, assisting businesses in making informed decisions about production, inventory management, and marketing strategies.
* The process typically involves data collection, preprocessing, and applying machine learning algorithms to recognize patterns, correlations, and seasonality in the data. These models then make predictions based on these patterns, assisting businesses in making informed decisions about production, inventory management, and marketing strategies.

***OVERVIEW:***

Product demand prediction using machine learning involves leveraging historical data to forecast future demand for a particular product. The overall concept typically involves:

* + Data Collection: Gather historical sales, customer, and other relevant data.
  + Data Preprocessing: Clean and prepare the data by handling missing values, normalizing, and transforming it for analysis.
  + Feature Selection/Engineering: Identify and create relevant features that might influence demand, like seasonality, pricing, promotions, etc.
  + Model Selection: Choose appropriate machine learning models (e.g., linear regression, decision trees, neural networks) suitable for the problem. Models like time series analysis (ARIMA, Prophet) are often used for demand forecasting.
  + Training: Train the chosen model on the historical data, validating its performance using various metrics.
  + Testing and Validation: Evaluate the model's accuracy and performance using test data to ensure it can generalize well to new, unseen data.
  + Forecasting: Utilize the trained model to predict future demand based on new data inputs.
  + Optimization and Implementation: Refine the model, if needed, and integrate it into operational systems for ongoing demand predictions.
  + Continuous Monitoring and Improvement: Monitor model performance and recalibrate it as needed with new data and insights to enhance accuracy.

Machine learning models for demand forecasting often consider various factors like historical sales patterns, market trends, economic indicators, and external factors influencing demand fluctuations. This process helps businesses optimize inventory, production, and marketing strategies, leading to more efficient operations and better customer satisfaction.

Data collection

* + Creating a data collection program for product demand prediction with machine learning involves gathering relevant data sources. Here's a step-by-step approach:
  + Identify Data Sources: Collect historical sales data, market trends, customer demographics, economic indicators, and any other relevant information that might impact product demand.
  + Data Collection Tools: Use APIs, web scraping, or databases to gather structured data. For unstructured data (like social media or reviews), consider text mining or sentiment analysis techniques.
  + Data Storage and Processing: Employ databases or data lakes to store and preprocess the collected data. Consider tools like SQL databases, Hadoop, or cloud-based storage solutions.
  + Data Quality Assurance: Ensure data quality by cleaning, normalizing, and handling missing values or outliers. Quality data is essential for accurate machine learning models.
  + Feature Engineering: Create relevant features from the collected data that will serve as inputs to your machine learning model. For example, time series data may require lag features or rolling averages.
  + Machine Learning Model Integration: Train and deploy machine learning models that predict product demand. Techniques like regression, time series analysis, or neural networks could be suitable, depending on the nature of the data.
  + Automate Data Collection: Develop a program/script to automatically collect and update the data regularly, ensuring the model remains up to date.

Testing and Validation: Test the model's accuracy using validation techniques and adjust it based on performance metrics

Sample Program:

name = input("Please enter your name: ")

age = input("Please enter your age: ")

print("Name:", name)

print("Age:", age)

output:

Please enter your name: [User enters their name]

Please enter your age: [User enters their age]

Name: [User's entered name]

Age: [User's entered age]

Data Preprocessing

Data preprocessing is crucial in product demand prediction using machine learning. It involves steps like handling missing values, scaling features, encoding categorical variables, removing outliers, and splitting data into training and testing sets. Additionally, feature engineering might be necessary to create relevant predictors, and normalization techniques can be applied to improve model performance. Each step aims to ensure the data is appropriately prepared for accurate predictions.

Sample Program:

data = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

print("Original Data:")

for row in data:

print(row)

transposed\_data = list(map(list, zip(\*data)))

print("\nPreprocessed Data (Transposed):")

for row in transposed\_data:

print(row)

output:

Original Data:

[1, 2, 3]

[4, 5, 6]

[7, 8, 9]

Preprocessed Data (Transposed):

[1, 4, 7]

[2, 5, 8]

[3, 6, 9]

**Feature Selection/Engineering**

**Certainly, feature engineering is crucial for building robust machine learning models, especially for product demand prediction. While I can't provide an entire program due to the complexity, I can guide you through the process:**

* **Understanding the Data: Get a comprehensive understanding of the dataset, including features, target variable, and their relationships. Identify any missing or outlier data.**
* **Feature Selection: Start by selecting relevant features that might influence demand, such as historical sales, seasonality, pricing, marketing efforts, etc.**
* **Time-Based Features: Create time-based features like day of the week, month, quarter, year, holidays, or any specific events related to demand.**
* **Cyclical Features: For cyclical patterns like seasonality, use trigonometric transformations (like sin, cos) for day or month to represent cyclical patterns in a continuous way.**
* **Lag Features: Generate lag features (past values) which can capture trends and seasonality. For instance, you might use previous day/week/month's demand as features.**
* **Statistical Features: Mean, median, standard deviation, min, max, and other statistical features from historical data can be insightful.**
* **Text Data: If relevant, process textual data (like product descriptions or customer reviews) using techniques like TF-IDF, Word2Vec, or BERT to derive features.**
* **Encoding Categorical Variables: Use techniques like one-hot encoding or label encoding for categorical variables.**
* **Scaling Features: Scale numerical features if needed, like using Standard Scaler or Min Max Scaler to bring features to a similar scale.**
* **Feature Importance: Utilize feature importance techniques (like Random Forest feature importance or SHAP values) to understand which features contribute the most to the model.**

**Remember to keep in mind domain knowledge and continuously iterate through the model building process, adjusting feature engineering as needed based on model performance.**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

data = pd.read\_csv('your\_dataset.csv')

X = data[['feature1', 'feature2', ...]] # Features

y = data['target\_column'] # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = RandomForestRegressor()

model.fit(X\_train, y\_train)

predictions = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, predictions)

print(f"Mean Squared Error: {mse}")

# Visualize predictions vs. actual values

plt.scatter(y\_test, predictions)

plt.xlabel("Actual Values")

plt.ylabel("Predicted Values")

plt.title("Actual vs. Predicted Values")

plt.show()

**Model Selection**

* Certainly! When selecting a machine learning model for product demand prediction, consider the following steps:

1. Data Understanding:

Understand the nature of your data, its features, and the target variable (demand in this case).

2. Data Preprocessing:

Handle missing values, outliers, and normalize/standardize data if necessary.

Encode categorical variables.

3. Feature Engineering:

Create new features if beneficial for the model.

4. Model Selection:

Linear Models: Such as Linear Regression or Ridge Regression for simplicity and interpretability.

Tree-Based Models: Decision Trees, Random Forest, or Gradient Boosting - good for capturing non-linear relationships.

Neural Networks: Deep Learning models like MLPs or RNNs if dealing with complex, high-dimensional data.

5. Model Evaluation:

Split your data into training and testing sets.

Use evaluation metrics (like RMSE, MAE, R²) to compare models' performance.

6. Hyperparameter Tuning:

Tune the parameters of the selected model for better performance. Techniques like GridSearch or RandomSearch can be used.

7. Validation and Testing:

Validate the final model on unseen data (cross-validation) to ensure generalizability.

8. Deployment and Monitoring:

Deploy the model and continuously monitor its performance for adjustments.

Example Python libraries for implementation:

Data Handling: Pandas, NumPy

Modeling: Scikit-learn, TensorFlow, Keras, XGBoost

program

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

rf = RandomForestRegressor(n\_estimators=100, random\_state=42)

rf.fit(X\_train, y\_train)

predictions = rf.predict(X\_test)

mse = mean\_squared\_error(y\_test, predictions)

print(f"Mean Squared Error: {mse}")

Model training

* Model training is the process of teaching a machine learning algorithm by presenting it with data to learn patterns, relationships, or features. It involves adjusting the model's parameters iteratively to minimize the difference between its predictions and the actual outcomes in the training data. The goal is to enable the model to make accurate predictions or classifications when faced with new, unseen data.

Program:

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from xgboost import XGBRegressor

from sklearn.metrics import mean\_squared\_error

data = pd.read\_csv('data.csv')

X = data[['feature1', 'feature2', 'feature3']] # Features

y = data['target'] # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = XGBRegressor()

model.fit(X\_train, y\_train)

predictions = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, predictions)

print('Mean Squared Error:', mse)

Testing and Validation

* In product demand prediction using machine learning, testing and validation are critical phases. Testing involves evaluating the model's performance on a dataset it hasn't seen before, while validation ensures the model generalizes well to new data. Techniques like cross-validation, train-test splits, and metrics (such as Mean Absolute Error or Root Mean Squared Error) are commonly used to assess and validate the model's accuracy and effectiveness in predicting product demand. These steps help ensure the model's reliability before deploying it in real-world scenarios.

**What is our range of dates** ?

|  |
| --- |
| ye <- dsa$YeMo[order(dsa$YeMo)]  range(ye)  timeframe <- 8+5\*12+1 #69 months |

'2011/05/00' '2017/01/00'

We now code each month by an integer ID, and fill our time-series data-frame.

dates <- dsa$YeMo

YeMo\_to\_int <- function(bad\_date){

a <- strsplit(bad\_date,split="/")[[1]]

y <- (strtoi(a[1])-2011)\*12

m <- strtoi(a[2],10)-4

return(y+m)

}

# test:

#YeMo\_to\_int("2011/05/00")

#YeMo\_to\_int("2017/01/00") #Ok

dsa$timestamp <- unlist(lapply(dates, FUN= YeMo\_to\_int))

dsa$YeMo <- NULL

#test :

#sum(is.na(dsa$timestamp)) #ok

products <- as.character(unique(dsa$Product\_Code))

dtf <- data.frame(matrix(data = 0, nrow = length(products), ncol=timeframe), row.names = products)

for (pXm in 1:dim(dsa)[1]){

dtf[as.character(unlist(dsa[pXm,"Product\_Code"])),unlist(dsa[pXm,"timestamp"])]<-

unlist(dsa[pXm,"Order\_Demand"])

}

**PROGRAMS**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error

# Load the dataset (replace 'data.csv' with your dataset)

data = pd.read\_csv('data.csv')

# Data preprocessing and feature selection

# ... (handle missing values, encode categorical variables, feature engineering)

# Split data into features and target variable

X = data[features] # Features

y = data['demand'] # Target variable (demand)

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Model training

model = LinearRegression()

model.fit(X\_train, y\_train)

# Model evaluation

predictions = model.predict(X\_test)

mae = mean\_absolute\_error(y\_test, predictions)

print(f"Mean Absolute Error: {mae}")

***Yearly Average Demand by Warehouse***

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

num\_warehouses = len(warehouses)

num\_years = len(years)

bar\_width = 0.15

fig, ax = plt.subplots(figsize=(12, 8))

x = np.arange(num\_years)

# Create a grouped bar chart

for i, warehouse in enumerate(warehouses):

x\_pos = x + i \* bar\_width

ax.bar(x\_pos, demand\_data[warehouse], width=bar\_width, label=warehouse)

ax.set\_xticks(x + (num\_warehouses - 1) \* bar\_width / 2)

ax.set\_xticklabels(years)

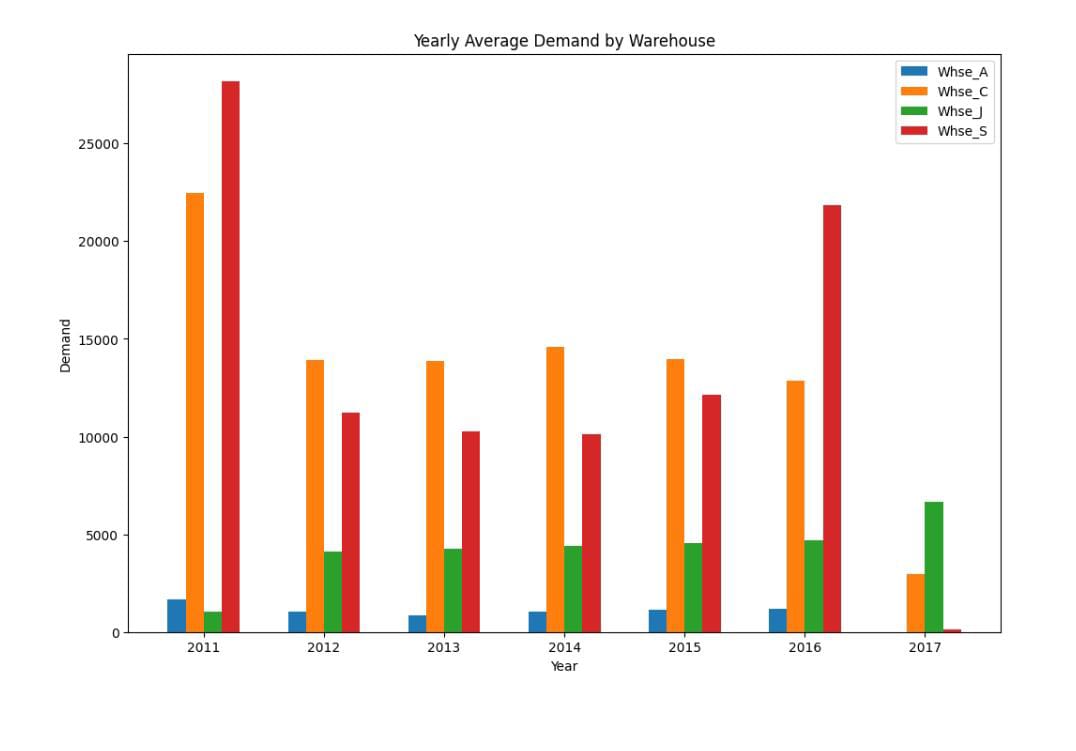
ax.set\_xlabel('Year')

ax.set\_ylabel('Demand')

ax.set\_title('Yearly Average Demand by Warehouse')

ax.legend()

plt.show()

****

Yearwise Average Demand of all Product

**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()):**

**bars = axes[i].bar(x, data, color=colors)**

**axes[i].bar\_label(bars, labels=x, fontsize = 8)**

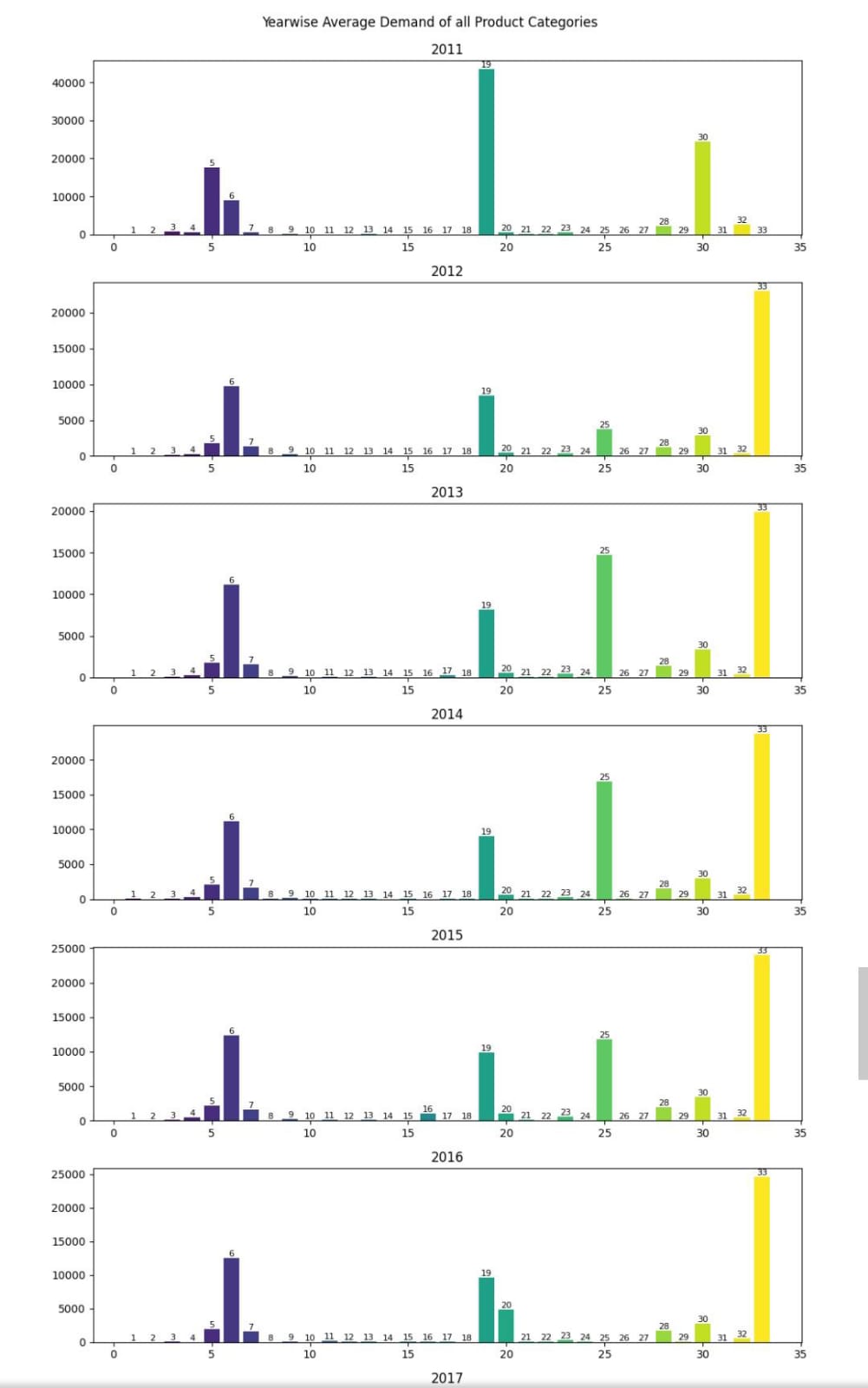
**axes[i].set\_title(year)**

**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()**

****

***Product categories***

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():

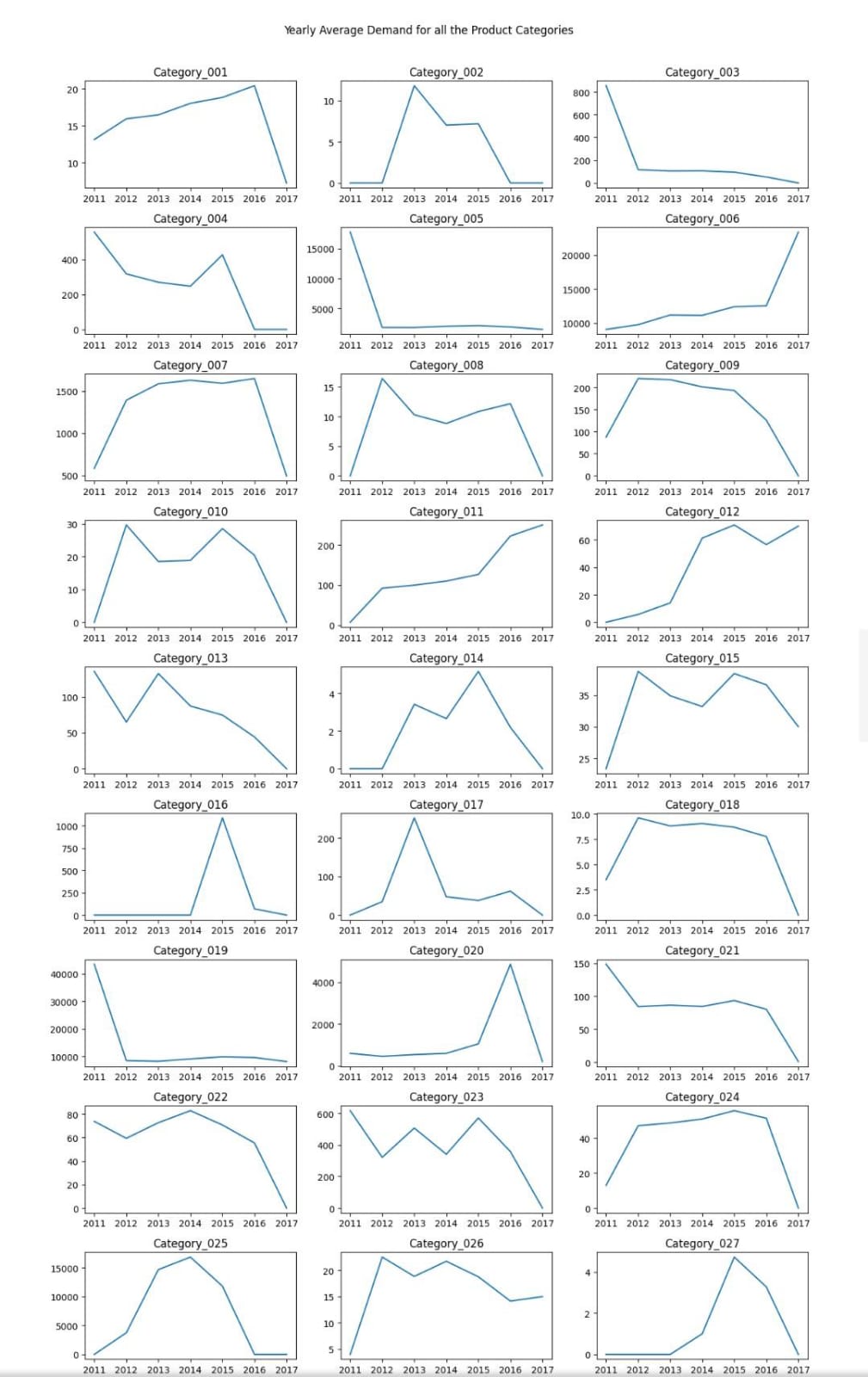
ax = fig.add\_subplot(rows, cols, int(title[-2:]))

ax.plot(x, data)

ax.set\_title(title)

plt.tight\_layout()

fig.suptitle("Yearly Average Demand for all the Product Categories", y=1.02)

****

BENEFITS

* Predicting product demand with machine learning offers several advantages:
* **Improved Forecasting:** ML models can analyze historical data, market trends, seasonality, and various other factors to make more accurate demand predictions, aiding in better inventory management.
* **Optimized Inventory Management:** Accurate demand forecasts assist in maintaining optimal inventory levels, reducing excess stock, and avoiding stockouts, thereby lowering storage costs and increasing revenue.
* **Enhanced Decision-Making:** ML models can provide insights into customer behavior, preferences, and changes in market conditions, enabling businesses to make informed decisions for production, marketing, and sales strategies.
* **Personalized Marketing:** Predictive models can identify customer segments and preferences, enabling personalized marketing campaigns that cater to specific customer needs and drive higher sales.
* **Cost Savings:** By minimizing excess inventory and optimizing supply chain management, businesses can reduce operational costs, thereby increasing profitability.
* **Adaptability:** ML models can adapt and learn from new data, adjusting predictions based on changing market dynamics, seasonal variations, and other influencing factors.
* **Competitive Advantage:** Accurate demand prediction provides a competitive edge by offering the ability to be more responsive and adaptive to market changes and customer needs.

Overall, employing machine learning for product demand prediction leads to smarter decision-making, improved resource allocation, and ultimately, better customer satisfaction.

ADVANTAGE

* + - Accurate Forecasting: Machine learning models can analyze historical data to make predictions about future demand more accurately than traditional methods, improving inventory management and reducing stockouts or overstocking.
    - Adaptability: These models can adapt to changing trends and seasonality, providing more dynamic predictions that adjust to market shifts or changes in consumer behavior.
    - Optimized Decision-Making: Predictive analytics can help in making better-informed business decisions, optimizing supply chain operations, pricing strategies, and marketing campaigns.
    - Automation: Once the model is trained, it can automate the prediction process, saving time and resources for companies in managing inventory and supply chain.

DISADVANTAGE

* **Data Dependency:** The quality and quantity of available historical data significantly impact the model's accuracy. Insufficient or biased data may result in inaccurate predictions.
* **Complexity:** Building and maintaining these models requires expertise in machine learning, which might be challenging for businesses lacking specialized personnel or resources.
* **External Factors**: Machine learning models might not fully account for external factors (e.g., political changes, natural disasters) that could significantly impact demand but aren't present in the historical data.
* **Overfitting:** Models may become too finely tuned to historical data, losing their generalizability to new market conditions or trends.

CONCLUSION

* Predicting product demand with machine learning involves analyzing historical data, identifying patterns, and using various models to forecast future demand. The conclusion typically includes the most accurate model, evaluation metrics (like accuracy, precision, recall), and insights derived from the predictions. Additionally, it may suggest potential strategies for inventory management, production planning, and marketing efforts to meet forecasted demand.
* Customers today expect effective products and hassle free on-time services. These expectations could not be met without a strong supply-chain that involves strategic planning that includes demand forecasting.
* The solution in this white paper is a statistical and ML-based solution that creates timeseries regarding each product and its entitlements based on geographic locations. The inputs of renewal rates and holidays based on each country or region helped generate accurate results by count and rate-based forecast on weekly basis. These forecasts assist the business in **parts procurements and help budget planning for each financial year.**
* The Demand Forecasting project was originally used by services finance and part planning teams. But it has the potential to broaden its horizon by expanding the scope of the forecasting project and changing the granularity of forecast with expanded end users**.**

THANKYOU