Phase 5

PRODUCT DEMAND PREDICTION USING MACHINE LEARNING

SUBMITTED BY

J.S.Fenisha.

Problem Statement:

Create a machine learning model that forecasts product demand based on historical sales and external factors, helping businesses optimize inventory management and production planning to meet customer needs efficiently.

Problem Definition:

The problem is to create a machine learning model that forecasts product demand based on historical sales data and external factors. The goal is to help businesses optimize inventory management and production planning to efficiently meet customer needs. This project involves data collection, data preprocessing, feature engineering, model selection, training, and evaluation.

Design Thinking:

1. Data Collection: Collect historical sales data and external factors that influence demand, such as marketing campaigns, holidays, economic indicators, etc.

2. Data Preprocessing: Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations

3. Feature Engineering: Create additional features that capture seasonal patterns, trends, and external influences on product demand.

4. Model Selection: Choose suitable regression algorithms (e.g., Linear Regression, Random Forest, XGBoost) for demand forecasting.

5. Model Training: Train the selected model using the preprocessed data. Evaluation: Evaluate the model's performance using appropriate regression metrics (e.g., Mean Absolute Error, Root Mean Squared Error).

Product Demand Prediction :

A product company plans to offer discounts on its product during the upcoming holiday season. The company wants to find the price at which its product can be a better deal compared to its competitors. For this task, the company provided a dataset of past changes in sales based on price changes. You need to train a model that can predict the demand for the product in the market with different price segments.

The dataset that we have for this task contains data about:

1.The product id

2.Store id

3.Total price at which product was sold

4.Base price at which product was sold

5.Units sold (quantity demanded) Product Demand Prediction with Machine Learning

Product Demand Prediction using Python

Let’s start by importing the necessary Python libraries and the dataset we need for the task of product demand prediction:

```
# Import necessary libraries import pandas as
pd import numpy as np import
matplotlib.pyplot as plt

from sklearn.model_selection import
train_test_split from sklearn.ensemble import
RandomForestRegressor from sklearn.metrics
import mean_squared_error import
statsmodels.api as sm
```

```
# Load your dataset (assuming it has columns
like 'date', 'quantity_sold',

'price', 'promotion', etc.) data =
pd.read_csv('product_sales.csv')


# Data preprocessing and feature engineering

# Extract relevant date features data['date'] =
pd.to_datetime(data['date']) data['year'] =
data['date'].dt.year data['month'] =
data['date'].dt.month data['day_of_week'] =
data['date'].dt.dayofweek # Split data into
features (X) and target (y)

X = data[['year', 'month', 'day_of_week',
'price', 'promotion']] y = data['quantity_sold']


# 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)

# Initialize and train a Random Forest
regressor
rf_model =
RandomForestRegressor(n_estimators=100,
random_state=42) rf_model.fit(X_train,
y_train)

# Make predictions on the test set y_pred_rf =
rf_model.predict(X_test)

# Evaluate the Random Forest model mse_rf =
mean_squared_error(y_test, y_pred_rf)
```

```
print(f"Random Forest Mean Squared Error:
{mse_rf}")


# Optionally, you can visualize feature
importances from the Random Forest model

feature_importances =
rf_model.feature_importances_ features =
X.columns

plt.bar(features, feature_importances)
plt.xlabel('Features') plt.ylabel('Importance')
plt.title('Feature Importances') plt.show()


# If you want to explore time series analysis,
you can use the statsmodels library

# Fit a time series model (e.g., SARIMA) to the
data
```

```
model = sm.tsa.SARIMAX(y_train, order=(1, 1,
1), seasonal_order=(1, 1, 1, 12)) results =
model.fit()

# Make predictions using the time series
model
y_pred_ts = results.predict(start=len(y_train),
end=len(y_train) + len(y_test) - 1,
dynamic=False)

# Evaluate the time series model
mse_ts = mean_squared_error(y_test,
y_pred_ts) print(f"Time Series Mean Squared
Error: {mse_ts}")
```

# Use the models for demand forecasting as needed

# You can input future features to predict future demand

# Close any open figures from the Random Forest feature importances plot plt.close().

Data set

A data set (or dataset) is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question. The data set lists values for each of the variables, such as for example height and weight of an object, for each member of the data set. Data sets can also consist of a collection of documents or files.[2]

the open data discipline, data set is the unit to measure the information released in a public open data repository. The European data.europa.eu portal aggregates more than a million data sets.[3]

Properties

Several characteristics define a data set's structure and properties. These include the number and types of the attributes or variables, and various statistical measures

applicable to them, such as standard deviation and kurtosis.[4]

The values may be numbers, such as real numbers or integers, for example representing a person's height in centimeters, but may also be nominal data (i.e., not consisting of numerical values), for example representing a person's ethnicity. More generally, values may be of any of the kinds described as a level of measurement. For each variable, the values are normally all of the same kind. Missing values may exist, which must be indicated somehow.

In statistics, data sets usually come from actual observations obtained

by sampling a statistical population, and each row corresponds to the observations on one element of that population. Data sets may

further be generated by algorithms for the purpose of testing certain kinds of software. Some modern statistical analysis software such as SPSS still present their data in the classical data set fashion. If data is missing or suspicious an imputation method may be used to complete a data set.[5]

The dataset that we have for this task contains data about:

1. the product id;
2. store id;
3. total price at which product was sold;
4. base price at which product was sold;
5. Units sold (quantity demanded);

I hope you now understand what kind of problem statements you will get for the product demand prediction task. In the section below, I will walk you through predicting product demand with machine learning using Python.

```
import pandas as pd import numpy as np
import plotly.express as px import seaborn as
sns import matplotlib.pyplot as plt from
sklearn.model_selection import
train_test_split from sklearn.tree import
DecisionTreeRegressor


data =
pd.read_csv("https://raw.githubusercontent.c
om/amankharwal/Websitedata/master/dema
nd.csv") data.head()
```

```
   ID  Store ID  Total Price  Base Price  Units Sold
0   1    8091      99.0375    111.8625        20
1   2    8091      99.0375     99.0375       28
```

```
2    3     8091     133.9500    133.9500          19
3    4     8091     133.9500    133.9500          44
4    5     8091     141.0750    141.0750          52
```

Now let’s have a look at whether this dataset contains any null values or not:

```
1
data.isnull().sum()
```

```
ID            0
Store ID      0
Total Price    1
Base Price     0 Units Sold     0 dtype: int64
```

So the dataset has only one missing value in the Total Price column, I will remove that entire row for now:

```
1
```

```
data = data.dropna()
```

Let us now analyze the relationship between the price and the demand for the product. Here I will use a scatter plot to see how the demand for the product varies with the price change:

```
fig = px.scatter(data, x="Units Sold", y="Total Price",
                 size='Units Sold')
fig.show()
```

We can see that most of the data points show the sales of the product is increasing as the price is decreasing with some exceptions. Now let’s have a look at the correlation between the features of the dataset:

print(data.corr())ID Store ID Total Price Base Price Units Sold ID   1.000000 0.007464 0.008473   0.018932   -0.010616

Store ID   0.007464 1.000000   -0.038315 -0.038848   -0.004372

Total Price 0.008473 -0.038315   1.000000 0.958885   -0.235625

Base Price   0.018932 -0.038848   0.958885 1.000000   -0.140032 Units Sold -0.010616 -0.004372   -0.235625   -0.140032   1.000000 correlations = data.corr(method='pearson') plt.figure(figsize=(15, 12)) sns.heatmap(correlations, cmap="coolwarm", annot=True) plt.show()

Product Demand Prediction Model

Now let's move to the task of training a machine learning model to predict the demand for the product at different prices. I will choose the Total Price and the Base Price column as the features to train the model, and the Units Sold column as labels for the model

```
xtrain, xtest, ytrain, ytest = train_test_split(x,
y,
                                    test_size=0.2,
random_state=42)
from sklearn.tree import
DecisionTreeRegressor
model = DecisionTreeRegressor()
```

model.fit(xtrain, ytrain) Now let’s input the features (Total Price, Base Price) into the model and predict how much quantity can be demanded based on those values:Now let’s input the features (Total Price, Base Price) into the model and predict how much quantity can be demanded based on those values:

```
#features = [["Total Price", "Base Price"]]
features = np.array([[133.00, 140.00]])
model.predict(features)
```

```
array([27.])
```

Summary

So this is how you can train a machine learning model for the task of product demand prediction using Python. Price is one of the major factors that affect the demand for the product. If a product is not a necessity, only a few people buy the product even if the price increases. I hope you liked this article on product demand prediction with machine learning using Python.

DEFINITION:

In demand forecasting, machine learning algorithms can analyze historical sales patterns and predict future trends. The first step is collecting data about past sales, such as product type, quantity sold, purchase frequency, seasonality, discounts, and more.

FEATURES ENGINEERING:

Predicting product demand using a machine learning model through feature engineering involves creating relevant input features that capture the factors influencing demand. Here's a high-level approach:

1. Data Collection: Gather historical data on product demand. This dataset should include information on products, time periods, and any relevant influencing factors (e.g., price, promotions, weather, seasonality).

2. Data Preprocessing:

- Handle missing data.

- Encode categorical features (e.g., product categories, location).

- Create time-related features (e.g., day of the week, month).

- Normalize or scale numerical features (e.g., price).

3. Feature Engineering:

- Create lag features: Include past demand as features to capture trends.

- Generate rolling statistics (e.g., rolling averages or moving sums) to capture seasonality.
- Create interaction features (e.g., product-category interaction).
- Include external factors (e.g., holidays, economic indicators) that might affect demand.

4. Data Splitting: Split the dataset into training and testing sets for model evaluation.

5. Model Selection::
   - Choose an appropriate machine learning algorithm for regression or time series forecasting (e.g., linear regression, decision trees, Random Forest, ARIMA, LSTM).
   - Train multiple models and evaluate their performance using metrics like Mean Absolute

Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

6. Model Training:

- Train the selected model on the training data using the engineered features.

7. Model Evaluation:

- Evaluate the model on the testing data to assess its predictive accuracy.
- Consider using cross-validation for more robust model evaluation.

8. Hyperparameter Tuning:

- Optimize model hyperparameters to improve performance.

9. Deployment:

- Deploy the trained model into a production environment for making real-time predictions.

- Implement a monitoring system to detect model performance degradation.

10. Continuous Improvement::

- Continuously update the model with new data to adapt to changing demand patterns.

- Refine feature engineering based on feedback and changing business conditions.

The success of your demand prediction model depends on the quality of your data, the relevance of your engineered features, and the choice of an appropriate machine learning algorithm. It's an iterative process that may require fine-tuning and constant monitoring for accuracy.

MODEL TRAINING:

Feature Engineering:

Create relevant features that may affect demand, such as seasonality, trends, or lag variables (e.g., sales from previous months).

Data Split:

Divide your dataset into training and testing sets. Typically, you would use a large portion for training and a smaller portion for testing (e.g., 80/20 split).

Model Selection:

Choose a machine learning model suitable for time-series forecasting. Popular choices include Linear Regression, Decision Trees, Random Forests, XGBoost, and ARIMA for time-series data.

Model Training:

Train the selected model on the training data, using the historical data to learn patterns and relationships that influence demand.Model Evaluation:

Evaluate the model's performance using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) on the test data.

Hyperparameter Tuning:

Fine-tune the model's hyperparameters to improve its performance.Model Validation: Validate the model's performance on a holdout dataset or through cross-validation to ensure it's robust.

Prediction:

Once the model performs well, you can use it to make demand predictions for future periods.

Deployment:

Integrate the trained model into your business processes, so it can provide real-time or periodic demand forecasts.Monitoring and Maintenance:

Continuously monitor the model's performance and retrain it periodically to adapt to changing demand patterns.

EVALUATION:

```
 import pandas as pd

import numpy as np

import plotly.express as px

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import
train_test_split

from sklearn.tree import DecisionTreeRegressor
```

data =
pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/demand.csv")

data.head()

DATA SET:

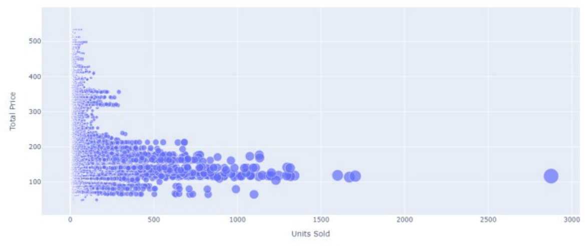
```
ID  Store ID  Total Price  Base Price  Units Sold
ID          1.000000  0.007464     0.008473    0.018932
-0.010616
Store ID    0.007464  1.000000    -0.038315
-0.038848   -0.004372
Total Price  0.008473 -0.038315     1.000000
0.958885   -0.235625
Base Price   0.018932 -0.038848     0.958885
1.000000   -0.140032
Units Sold  -0.010616 -0.004372    -0.235625
-0.140032    1.000000
```

SCATTER PLOT:

Here I will use a scatter plot to see how the demand for the product varies with the price change:

```
fig = px.scatter(data, x="Units Sold", y="Total
Price",  size='Units Sold')

fig.show()
```



The correlation between the features of the dataset:

```
print(data.corr())

ID  Store ID  Total Price  Base Price  Units Sold ID

1.000000  0.007464     0.008473    0.018932
-0.010616

Store ID     0.007464  1.000000    -0.038315
-0.038848   -0.004372
```

```
Total Price  0.008473 -0.038315     1.000000
0.958885   -0.235625
Base Price   0.018932 -0.038848     0.958885
1.000000   -0.140032
Units Sold  -0.010616 -0.004372    -0.235625
-0.140032    1.000000
```

CONCLUSION:

The conclusion of a product demand prediction using machine learning would depend on the specific analysis and results of the project.