Project 10: Product Demand Prediction with Machine Learnings

SHORT EXPLANATION(Production demand prediction with machine learning)

Production demand prediction with machine learning refers to the use of machine learning algorithms and techniques to forecast the future demand for a product or service. This is a crucial aspect of supply chain and inventory management for businesses. Here's a short explanation of the concept:

Objective: The primary goal is to accurately estimate how much of a product or service will be needed in the future. This prediction helps businesses optimize their production processes, manage inventory efficiently, and ensure they meet customer demand without overstocking or understocking.

Data Input: Machine learning models for demand prediction require historical data, such as sales records, customer orders, and relevant external factors like seasonality, economic indicators, or marketing campaigns. The quality and quantity of data are critical for model accuracy.

Model Selection: Various machine learning algorithms can be used, including regression models, time series forecasting, neural networks, and more. The choice of the model depends on the nature of the data and the complexity of the demand patterns.

Feature Engineering: Data preprocessing involves feature engineering, which may include data normalization, encoding categorical variables, handling missing data, and creating relevant features like lag variables or rolling averages.

Training and Testing: The model is trained on historical data, where it learns the patterns and relationships between input variables and demand. It is then tested on a separate dataset to evaluate its performance and ensure it can make accurate predictions.

Forecasting: Once the model is trained and validated, it can be used to make demand predictions for future time periods. These predictions can be at different levels, such as daily, weekly, or monthly, depending on the business needs.

Optimization: Businesses can use the demand predictions to optimize their production schedules, inventory levels, and procurement strategies. This can lead to cost savings, improved customer service, and better overall operational efficiency.

Monitoring and Iteration: Demand patterns can change over time due to various factors, so it's essential to monitor the model's performance regularly. If the accuracy decreases, the model may need to be retrained with updated data or modified to account for changing conditions.

In summary, production demand prediction with machine learning leverages historical data and advanced algorithms to forecast future demand accurately. This enables businesses to make informed decisions, reduce costs, and ensure they meet customer needs effectively.

DATASET

A 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.

Dataset based on Product Demand Prediction with Machine Learning are collected in www.kaggle.com/data as detail in following columns:

- ID
- Store ID
- Total Price
- Base Price
- Units Sold

Dataset link

https://www.kaggle.com/datasets/chakradharmattapalli/product-demand-prediction-with-machine-learning

Details about column in the dataset

In future demand prediction using machine learning, the dataset we mentioned appears to be a tabular dataset with several columns that likely represent various attributes of products or items for which we want to predict future demand. Let me provide a detailed explanation of each column:

1. **ID (Identifier):**

This column typically contains a unique identifier for each item or product in your dataset. It's useful for tracking and referencing specific items but usually not used as a feature for prediction.

2. Store ID:

This column represents the unique identifier of the store or location where the product is sold or stored. This can be crucial if you want to predict demand variations based on different stores.

3. Total Price:

This column likely contains the total price of the product, which is the actual amount a customer pays for it. Total price can be influenced by various factors like discounts, promotions, and taxes.

4. Base Price:

This column represents the base price of the product, which is the price before any discounts or promotions are applied. It provides information about the inherent value of the product.

5. Units Sold (or Sold Quantity):

This column contains the number of units or quantity of the product sold during a specific period. This is typically the target variable you want to predict in future demand prediction models. It's the value you aim to forecast based on the other features.

In a machine learning context, we might use these columns in the following way:

ID and Store Id:

These columns are usually not used for prediction directly but can be useful for grouping and aggregation if we want to analyze sales patterns for specific products or stores.

Total Price and Base Price:

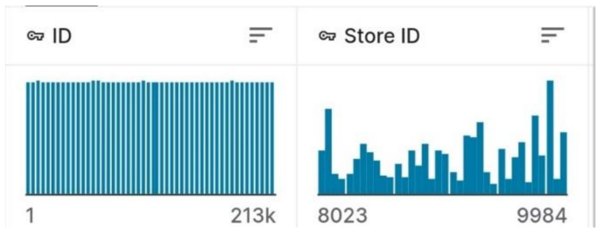
These columns can be used as features in our model if we want to examine the impact of pricing on future demand. For example, we can calculate price elasticity.

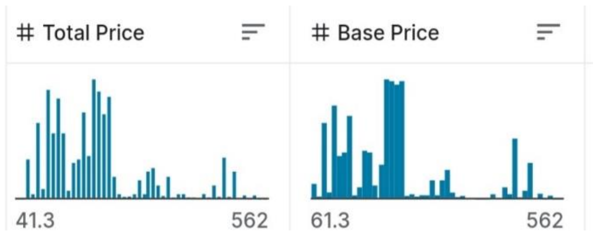
Units Sold:

This is your target variable. You'll build a predictive model using machine learning techniques where you'll use the other columns (Total Price, Base Price, Store ID, etc.) as input features to predict the future demand, which is the number of units sold.

In our machine learning model will learn patterns and relationships between these input features and the target variable (Units Sold) from historical data. Once the model is trained, we can use it to make predictions for future demand based on the values of the Input features.

Data set flowchart:





DETAILS OF LIBRARIES TO BE USER AND WAY TO DOWNLOAD

DETAILS OF LIBRARIES:

1. PANDAS:

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- Pandas is one of the tools in Machine Learning which is used for data cleaning and analysis. It has features which are used for exploring, cleaning, transforming and visualizing from data.

2. NUMPY:

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices

3. PLOTPY.EXPRESS :

- The plotly.express module (usually imported as px) contains functions that can create entire figures at once, and is referred to as Plotly Express or PX.
- Plotly Express is a built-in part of the plotly library, and is the recommended starting point for creating most common figures.

4. SEABORN:

- Seaborn is a library for making statistical graphics in Python.
- It builds on top of matplotlib and integrates closely with pandas data structures.
- Seaborn helps you explore and understand your data.

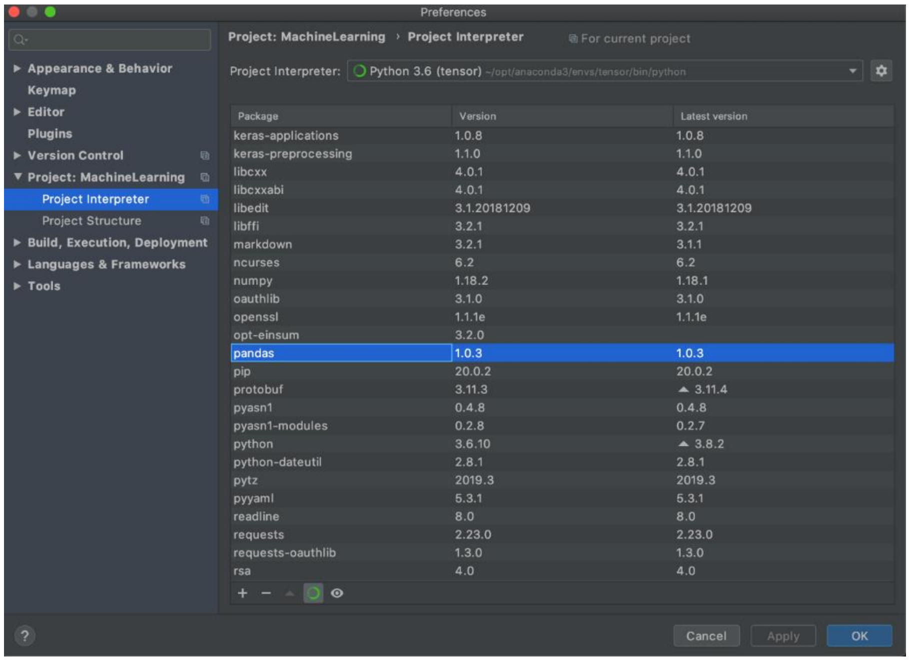
5. MATPLOTLIB:

- Matplotlib is a low-level library of Python which is used for data visualization. It is easy to use and emulates
- MATLAB like graphs and visualization

<u>How to import packages in pycharm:</u>

To install Pandas on PyCharm,

1. click on File
2. Go to the Settings.
3. Under Settings, choose your Python project and select Python Interpreter.
4. Then, search for the Pandas package and
5. click Install Package



Project Interpreter: Python 3.6 (tensor) ~/opt/anaconda3/envs/tensor/bin/python

Package	Version	Latest version
keras-applications	1.0.8	1.0.8
keras-preprocessing	1.1.0	1.1.0
libcxx	4.0.1	4.0.1
libcxxabi	4.0.1	4.0.1
libedit	3.1.20181209	3.1.20181209
libffi	3.2.1	3.2.1
markdown	3.2.1	3.1.1
ncurses	6.2	6.2
numpy	1.18.2	1.18.1
oauthlib	3.1.0	3.1.0
openssl	1.1.1e	1.1.1e
opt-einsum	3.2.0	
pandas	1.0.3	1.0.3
pip	20.0.2	20.0.2
protobuf	3.11.3	▲ 3.11.4
pyasn1	0.4.8	0.4.8
pyasn1-modules	0.2.8	0.2.7
python	3.6.10	▲ 3.8.2
python-dateutil	2.8.1	2.8.1
pytz	2019.3	2019.3
pyyaml	5.3.1	5.3.1
readline	8.0	8.0
requests	2.23.0	2.23.0
requests-oauthlib	1.3.0	1.3.0
rsa	4.0	4.0

NOTE:

Simlarly the other packages are Downloaded in Pycharm

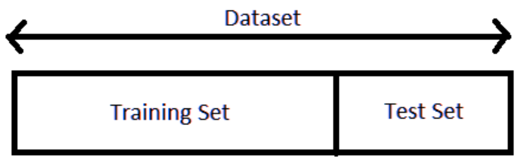
How to import packages?

Demo code:

```
 import pandas as pd

# Replace 'your_dataset.csv' with the actual path or URL to your dataset
dataset_path = 'your_dataset.csv'

# Read the dataset into a pandas DataFrame
df = pd.read_csv(dataset_path)

# Display the first few rows of the dataset to verify that it was read correctly
print(df.head())
```

HOW TO TRAIN AND TEST

A machine learning algorithm works in two stages. We usually split the data around 20%-80% between testing and training stages. Under supervised learning, we split a dataset into a training data and test data in Python ML.



Prerequisite for Train and Test Data

We will need the following Python libraries for this tutorial- pandas and sklearn.
We can install these with pip-

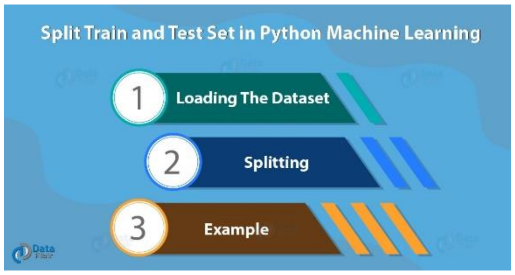
```
pip install pandas

pip install sklearn
```

We use pandas to import the dataset and sklearn to perform the splitting. You can import these packages as-

```
>>> import pandas as pd

>>> from sklearn.model_selection import train_test_split
```

How to Split Train and Test Set in Python Machine Learning?

Following are the process of Train and Test set in Python ML. So, let's take a dataset first.



a. Loading the Dataset

Let's load the forestfires dataset using pandas.

```
>>> data=pd.read_csv('productdemand.csv')
>>> data.head()
```

b. Splitting

Let's split this data into labels and features. Now, what's that? Using features, we predict labels. I mean using features (the data we use to predict labels), we predict labels (the data we want to predict).

```
>>> y=data.temp
>>> x=data.drop('temp',axis=1)
```

Temp is a label to predict temperatures in y; we use the drop() function to take all other data in x. Then, we split the data.

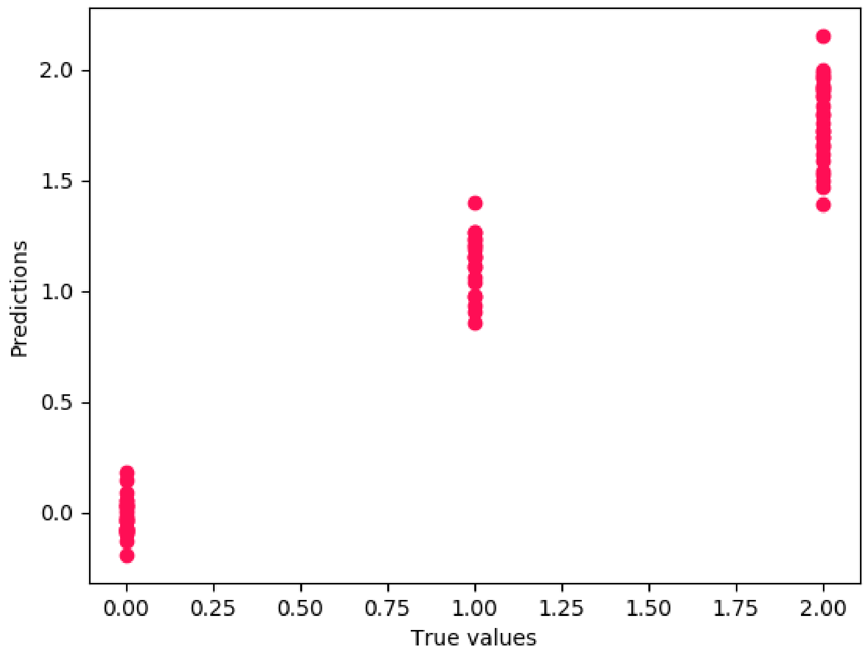
```
>>> x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

The line test_size=0.2 suggests that the test data should be 20% of the dataset and the rest should be train data. With the outputs of the shape() functions, you can see that we have 104 rows in the test data and 413 in the training data.

c. Plotting of Train and Test Set in Python

We fit our model on the train data to make predictions on it. Let's import the linear_model from sklearn, apply linear regression to the dataset, and plot the results.

```
>>> from sklearn.linear_model import LinearRegression as lm
>>> model=lm().fit(x_train,y_train)
>>> predictions=model.predict(x_test)
>>> import matplotlib.pyplot as plt
>>> plt.scatter(y_test,predictions)
>>> plt.xlabel('True values')
>>> plt.ylabel('Predictions')
>>> plt.show()
```



0.9396299518034936
So, this was all about Train and Test Set in Python Machine Learning.

REST OF EXPLANATION

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

```
>>>data.isnull().sum()
```

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

```
>>>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:

```
>>>correlations = data.corr(method='pearson')
>>>plt.figure(figsize=(15, 12))
>>>sns.heatmap(correlations, cmap="coolwarm", annot=True)
>>>plt.show()
```

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:

```
>>>x = data[["Total Price", "Base Price"]]
>>>y = data["Units Sold"]
```

Now let's split the data into training and test sets and use the decision tree regression algorithm to train our 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:

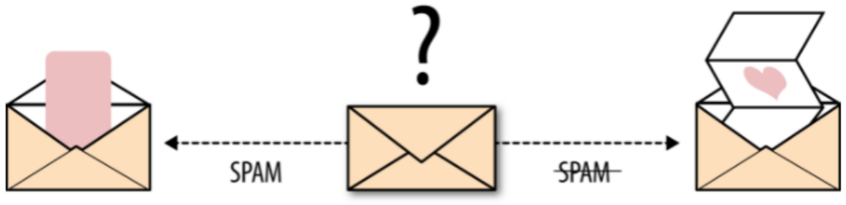
```
>>>#features = [["Total Price", "Base Price"]]
>>>features = np.array([[133.00, 140.00]])
>>>model.predict(features)
```

METRICES USED FOR THE ACCURACY CHECK:

Classification Metrics in Machine Learning:

- Classification is about predicting the class labels given input data.
- In binary classification, there are only two possible output classes(i.e., Dichotomy).
- In multiclass classification, more than two possible classes can be present. I'll focus only on binary classification.

A very common example of binary classification is spam detection, where the input data could include the email text and metadata (sender, sending time), and the output label is either *"spam" or "not spam."* (*See Figure*) Sometimes, people use some other names also for the two classes: "positive" and "negative," or "class 1" and "class 0."



Email spam detection is a binary classification problem (source: From Book—Evaluating Machine Learning Model—O'Reilly)

There are many ways for measuring classification performance. Accuracy, confusion matrix, log-loss, and AUC-ROC are some of the most popular metrics. Precision-recall is a widely used metrics for classification problems.

Accuracy:

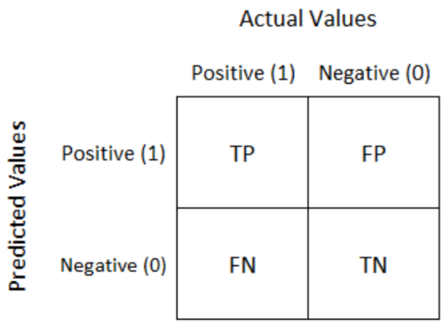
- Accuracy simply measures how often the classifier correctly predicts.
- We can define accuracy as the ratio of the number of correct predictions and the total number of predictions.

$$\textbf{\textit{Accuracy}} = \frac{TP + TN}{TP + TN + FP + FN}$$

- When any model gives an accuracy rate of 99%, you might think that model is performing very good but this is not always true and can be misleading in some situations.

Confusion Matrix:

- Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values.
- A confusion matrix is defined as thetable that is often used to describe the performance of a classification model on a set of the test data for which the true values are known.



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

- It is extremely useful for measuring the Recall, Precision, Accuracy, and AUC-ROC curves.

Recall (Sensitivity):

It explains how many of the actual positive cases we were able to predict correctly with our model. Recall is a useful metric in cases where False Negative is of higher concern than False Positive. It *is important in medical cases where it doesn't matter whether we raise a false alarm but the actual positive cases should not go undetected!*

Recall for a label is defined as the number of true positives divided by the total number of actual positives.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

F1 Score:

It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall.

F1 Score is the harmonic mean of precision and recall**.**

$$F1 = 2.\,\frac{Precision\ \times Recall}{Precision + Recall}$$

The F1 score punishes extreme values more. F1 Score could be an effective evaluation metric in the following cases:

- When FP and FN are equally costly.
- Adding more data doesn’t effectively change the outcome
- True Negative is high

AUC-ROC

The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values and separates the ‘signal’ from the ‘noise’.